Application of Computer Vision Techniques for Monitoring Steel Manufacturing Processes

Callum O'Donovan

November 2024

Swansea University

Faculty of Science and Engineering

This thesis has been submitted in fulfilment of the requirements for the degree of Doctor of Engineering

Abstract

Computer vision (CV) is a branch of artificial intelligence (AI) that enables machines to understand visual input. The recent rise of deep learning (DL) has empowered CV significantly, leading to well-established applications such as autonomous vehicles, medical diagnosis and facial recognition. These new capabilities extend to the manufacturing sector, however they have not been widely adopted to monitor processes in the steel industry due to challenges related to harsh environmental conditions such as poor lighting, heat distortion, dust particles and vibrations. As a result, existing datasets are limited and advances have predominantly been evaluated within research settings but not real-world settings. Therefore, this project investigates the application of CV for monitoring steel production processes and how integration impacts state-of-the-art technology. This work aims to produce CV systems capable of monitoring different processes and utilise them to draw valuable real-world insights for industry. Also, it aims to investigate how these systems, and CV as a whole, can enhance the efficiency, quality and sustainability of steel manufacturing. This research involves the development of CV models tailored to three processes: ladle pouring, galvanising and gas stirring. In each case study, DL and traditional methods are used to monitor real or simulated production environments and extract useful information. Primary outcomes of this research include a foundation for monitoring ladle pouring to reduce emissions, a deployed system for quantifying zinc splatter occurring during galvanisation in real-time, and a tool for comparing the wear rate and stirring efficiency of different gas stirring approaches. Outcomes of this work highlight the revolutionary benefits of applying CV in production environments for process monitoring and control. By developing CV models for monitoring processes, overcoming harsh conditions typical in production environments, and drawing valuable insights from CV application, this work establishes a strong foundation for real-world implementation of CV in manufacturing.

Declarations and Statements

Contents

Acknowledgements

This research was carried out at Swansea University under the Materials and Manufacturing Academy (M2A) which is funded by the European Social Fund via the Welsh Government. I would like to thank all members of the M2A team that made this possible with their continuous support throughout the project.

I would like to acknowledge Mohammed Kamran, Ivan Popov and Tata Steel, who provided stimulating challenges and discussions, as well as the data required to conduct my research.

I would like to express my deepest gratitude to Professor Cinzia Giannetti for all of her inspiration, guidance and insights over the years, it has been a privilege. I am also grateful for the valuable contributions provided by Dr. Grazia Todeschini and Professor Cameron Pleydell-Pearce which have made this achievement possible.

I would like to thank Barbara for her patience and encouragement, for helping me to sustain long working hours, and for listening to me talk endlessly about computer vision. I could not have done it without you.

I would like to thank my sister for being an excellent role model who has inspired me to pursue great things. Your perseverance and boldness have always been a source of strength for me.

Last but not least, I would like to thank my parents for never giving up on me throughout all of the emotionally difficult times that led to this moment. Your unwavering support has enabled me to achieve more than I ever imagined.

This thesis is dedicated to my grandparents, Yvonne and Andrew, who have guided my journey through spirit and grace. To Yvonne, who I have the utmost love and respect for. To Andrew, if you could see me today, I hope you would be proud of who I have become.

List of Figures

List of Tables

Acronyms

AHE Adaptive Histogram Equalisation

AI Artificial Intelligence

AL Auto-Labelling

AMBE Absolute Mean Brightness Error

ANN Artificial Neural Network

AP Average Precision

API Application Programming Interface

ASGI Asynchronous Server Gateway Interface

ASR Automatic Speech Recognition

AUC Area Under Curve

AutoNAC Automatic Neural Architecture Construction

BG Background

BGS Background Subtraction

BOF Basic-Oxygen Furnace

BPBHE Brightness Preserving Bi-Histogram Equalisation

BPDFHE Brightness Preserving Dynamic Fuzzy Histogram Equalisation

CBAM Convolutional Block Attention Module

CCTV Closed-Circuit Television

CDF Cumulative Distribution Function

CLAHE Contrast-Limited Adaptive Histogram Equalisation

CNN Convolutional Neural Network

CNT	Counting
COCO	Common Objects in Context
CPU	Central Processing Unit
CRNN	Convolutional Recurrent Neural Network
CSPNet	Cross Stage Partial Network
CT	Computed Tomography
CUDA	Compute Unified Device Architecture
CUDnn	CUDA Deep Neural Network library
CV	Computer Vision
DAGM	Deutsche Arbeitsgemeinschaft für Mustererkennung
DCNN	Deep Convolutional Neural Network
DeepSORT	Deep Simple Online and Real-Time Tracking
DFEM	Dilated Feature Enhancement Model
DFL	Distribution Focal Loss
DIAL	Differential Absorption LiDAR
DL	Deep Learning
DoE	Design of Experiments
DPM	Deformable Part Model
EGS	Efficient Grid Search
EKF	Extended Kalman Filter
FastSAM	Fast Segment Anything Model
FCIS	Fully Convolutional Instance-Aware Semantic Segmentation
FCN	Fully Convolutional Network

FG Foreground

FN False Negative

FP False Positive

FPN Feature Pyramid Network

GAN Generative Adversarial Network

GC Gas Chromatography

GCN Gradient Clip Normalisation

GDDR6 Graphics Double Data Rate 6

GMG Gaussian Mixture-based Background Foreground Segmentation

GMM Gaussian Mixture Model

GPU Graphical Processing Unit

GSOC Google Summer of Code

HE Histogram Equalisation

HSL Hue-Saturation-Luminance

HSV Hue-Saturation-Value

ID Identification

IoT Internet of Things

IoU Intersection over Union

IP Internet Protocol

JSON JavaScript Object Notation

KD Knowledge Distillation

KNN K-Nearest Neighbors

KolektorSDD Kolektor Surface Defect Dataset

LBP Local Binary Pattern

LM Learning Momentum

LOM Light Optical Microscopy

LPBF Laser Powder Bed Fusion

LPDDR5 Low Power Double Data Rate 5

LR Learning Rate

LSBP Local SVD Binary Pattern

LSTM Long Short-Term Memory

MA Moving Average

MAE Mean Average Error

mAP Mean Average Precision

Mask R-CNN Mask Region-based Convolutional Neural Network

ML Machine Learning

MLP Multi-Layer Perceptron

MNC Multi-Task Network Cascades

MOG Mixture of Gaussians

MOG2 Mixture of Gaussians 2

MOT Multi-Object Tracking

MOTA Multiple Object Tracking Accuracy

MRI Magnetic Resonance Imaging

MS Mass Spectrometry

MSE Mean Squared Error

NAS Neural Architecture Search

NEU-DET Northeastern University Detection Dataset

NIN Network-in-Network

NLP Natural Language Processing

NMS Non-Maximum Suppression

NSLNet Neural Structured Learning Network

NUS National University of Singapore

ONNX Open Neural Network Exchange

OOM Out of Memory

P Precision

PANet Path Aggregation Network

PASCAL Pattern Analysis Statistical Modelling and Computational Learning

PCA Principal Component Analysis

PCB Printed Circuit Board

PMF Probability Mass Function

PO1 Pot Operator 1

PO2 Pot Operator 2

PSNR Peak Signal-to-Noise Ratio

PT Production Testing

R Recall

R-CNN Region-based Convolutional Neural Network

RELM Regularised Extreme Learning Machine

ResNet Residual Network

RGB Red-Green-Blue

RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RoI	Region of Interest
RPN	Region Proposal Network
RUL	Remaining Useful Life
SAM	Segment Anything Model
SBC	Single Board Computer
SDI	Slag Distribution Image
SDM	Saliency Detection Model
SDS	Slag Detection System
SE	Stirring Efficiency
SF	Scale Factor
SGD	Stochastic Gradient Descent
SIN	Slab Identification Number
SME	Small Medium-Sized Enterprises
SOLO	Segmenting Objects by Locations
SORT	Simple Online and Real-Time Tracking
SOTA	State of the Art
SPP	Spatial Pyramid Pooling
SPRENet	Slag Removal Path Estimation Network
SSD	Single-Shot Detector
SVD	Singular Value Decomposition
SVM	Support Vector Machine

TAS2-Net	Triple-Attention Semantic Segmentation Network
TLU-Net	Transfer Learning-based U-Net
TNN	Thresholding Neural Network
TP	True Positive
TRT	TensorRT
UIQI	Universal Image Quality Index
UV	Ultra-Violet
VGG	Visual Geometry Group
VIA	VGG Image Annotator
VOC	Visual Object Classes
WF	Wear Rate Factor
WS	Washing Severity
XML	Extensible Markup Language
YOLACT	You Only Look At Coefficients
YOLO	You Only Look Once
YOLO-NAS	You Only Look Once - Neural Architecture Search
ZF	Zeiler and Fergus

Chapter 1: Introduction

1.1 Background

Artificial intelligence (AI) is a rapidly evolving area of computer science concerned with incorporating human-like behavior and intelligence into machines or systems [1]. There are various subfields of AI which include, but are not limited to, natural language processing (NLP), automated speech recognition (ASR) and computer vision (CV) [1]. Whilst NLP and ASR are closely related and focus on text and sound-based understanding, CV focuses on using visual data to enable computational models and systems to acquire an understanding of different environments [1].

Traditionally, computer vision has aimed to mimic the visual capabilities of humans through tasks such as object recognition, image segmentation and motion detection [2]. This has relied heavily on explicit programming and simple pattern recognition techniques, and has therefore been greatly limited in terms of handling the complexity and variability inherently present in visual data [2]. However, in recent years, computer vision has been revolutionised through advances in machine learning (ML), particularly the type of ML referred to as deep learning (DL) [2]. Whilst ML a subfield of AI that is based on building analytical models that can learn from historical data [1], DL is a subfield of ML that is centred around the use of multi-layered neural networks for pattern recognition [1]. In broad terms, neural networks are algorithms that have been modelled based on the architecture of the human brain [2]. They are comprised of layers of neurons through which data is passed, typically in series, so that each layer can extract useful features from the data [2].

With the dramatic improvement of computational capabilities over the last two decades, DL has been greatly empowered due to increased data storage capacity as well as neural network training resource availability [2]. Big Data, characterised by high volume, high velocity or high variety, is the major driving force that empowers deep learning models to learn complex patterns [3]. Consequentially, the state of DL-based CV has made leaps and bounds in terms of recognising complex patterns and making intelligent decisions with a level of precision that was previously unattainable. These enhanced capabilities have opened the door to a vast number of breakthroughs in tasks such as image classification, object detection and object segmentation [2]. The integration of CV, DL and Big Data has laid the foundation for generations of innovative applications across various sectors including healthcare, agriculture and particularly manufacturing.

Through insights gained from visual data, steel manufacturers can optimise the operational efficiency of their processes, improve product quality, ensure safer working conditions, and much more. Areas of steel production that are already experiencing major advancements due to CV include process monitoring and automation, quality control via defect detection systems and microstructural analysis (see Section 2.7 of Chapter 2 for more examples of applications) [4, 5, 6].

1.2 Research Objectives

The overall aim of this project is to enhance the efficiency and quality of steel production through the application of advanced computer vision technologies. This is approached by addressing several key challenges that expose novel opportunities in the steel industry. In order to address some of the challenges and opportunities in steel production, this thesis focuses on three primary research objectives:

1. *Conduct a comprehensive literature review of modern advances in computer vision techniques with emphasis on industrial and manufacturing scenarios. This includes establishing a clear understanding of the state-of-the-art, critically analysing existing literature, and identifying research gaps.*

2. *Design novel computer vision systems tailored for different processes within steel production (hot metal ladle pouring, galvanisation, and gas stirring) to monitor and analyse key process variables that give insights for enhancing operational efficiency and product quality.*

3. *Investigate the deployment of computer vision technologies into steel production environments as both post-processing and real-time monitoring applications. This includes the consideration of robustness and scalability of systems to ensure they can withstand the complexities and variations of deploying to real industrial settings.*

1.3 Thesis Structure

Chapter 2: Provides an extensive review of existing computer vision techniques and their applications in manufacturing, highlights advances and identifies gaps in the literature. This chapter sets the stage for the thesis firstly by establishing the significance of computer vision in manufacturing, particularly in the steel industry. By identifying gaps, it also highlights the need for innovative solutions and therefore justifies the purpose of this research.

Chapter 3: Details common techniques and tools that were utilised across multiple case studies. The fundamental principles of computer vision methods used in this research are covered, as well as their configurations, justifications for their selection, and potential limitations. Additional resources that were critical to the success of this work such as software, hardware and the Common Objects in Context (COCO) dataset are also detailed. This ensures a clear methodological framework is defined from the outset.

Chapter 4: Presents Case Study 1, a novel method for tracking the motion of hot metal ladles during the pouring process in harsh environmental conditions. The chapter details the application of Contrast-Limited Adaptive Histogram Equalisation (CLAHE) for contrast enhancement of poorly-lit footage, and Mask R-CNN (Mask Region-based Convolutional Neural Network) for ladle segmentation. It demonstrates how these techniques can be used to monitor ladle pouring height, rotation angle, and furnace flame severity. The insights gained pave the way for enhancing ladle process monitoring and control.

Chapter 5: Presents Case Study 2, an innovative approach for quantifying the severity of zinc splatter during the galvanisation process in real-time. It describes the use of Counting (CNT) background subtraction (BGS) for splatter segmentation and YOLOv5 (You Only Look Once) for air knife detection. The chapter also investigates the deployment of the model onto hardware suitable for a production environment, proving its robustness and practicality. The developed system can be implemented to reduce equipment downtime and improve product quality.

Chapter 6: Presents Case Study 3, which focuses on the analysis of gaseous plume dynamics, refractory wear, and stirring efficiency in gas stirring processes within a basic-oxygen furnace (BOF). The chapter introduces a novel approach using computer vision to monitor various plume characteristics from experiments simulating gas stirring. It details the use YOLOv5 for plume detection, DeepSORT (Deep Simple Online and Real-Time Tracking) for tracking, and CNT background subtraction for segmentation. It demonstrates how application of these techniques was used to estimate the impact of different stirring configurations on refractory wear and stirring efficiency.

Chapter 7: Provides a comparative analysis of the three case studies presented in Chapters 4, 5 and 6 in terms of methodologies and results. This chapter also discusses the common challenges encountered, such as dealing with harsh environmental conditions and overcoming resource constraints with regards to data availability, computational expense and time. By synthesising insights gained from each case study, this emphasises the broader impact of the solutions presented.

Chapter 8: Concludes the thesis with a summary of the key findings and significant contributions to the field of computer vision in manufacturing. It reflects on the original research objectives and how they were achieved, and also summarises key findings and implications of this research on the state of manufacturing technology. Additionally, this chapter summarises the main limitations experienced in this project, before offering recommendations for future research.

1.4 Key Research Outcomes

The research presented in this thesis has several significant outcomes that contribute to both the academic body of knowledge and the practical applications in the steel production industry. The key outcomes are listed below:

- Developed a computer vision model capable of estimating ladle pouring height, angle of rotation and resulting flame severity in ladle pouring processes which sets a foundation for future real-world applications.
- Developed and deployed a model for quantifying splatter severity on a galvanising line on a NVIDIA Jetson Orin Nano which exhibits real-time capability over a Wi-Fi network.
- Created a model for detecting, tracking, and segmenting gaseous plumes in furnace gas stirring simulations to obtain stirring efficiency rating and wear rate factor values for assessment of different stirring setups.

These advancements highlight the potential of computer vision technologies to enhance operational efficiency and process quality in steel manufacturing. Further details on each case study are discussed in the subsequent chapters of this thesis.

1.5 Research Outputs

The outputs of this project are stated below:

- Publication of two journal papers (items 1 and 2 below), based on Case Study 1 (Chapter 4) and Case Study 2 (Chapter 5) respectively.
- Submission of one journal paper (item 3 below), based on Case Study 3 (Chapter 6). This is currently under review.
- Publication of two conference papers (items 4 and 5 below), based on the wider implications of computer vision application in manufacturing with regards to sustainability and Industry 4.0.
- Deployment of a real-time, Wi-Fi enabled, splatter severity quantification model on an Internet of Things (IoT) device suitable for implementation into the galvanising line. If process technologists at the galvanising site decided to implement this it would aid in reducing equipment degradation and surface defects. This emphasises the practical applicability of this research.
- Contribution of a model for gaseous plume analysis to a collaborative project, which has been utilised by another researcher to gain understanding of gas dynamics in steel production, and could be used to aid optimisation of the real furnace gas stirring process to reduce equipment degradation and improve stirring efficiency.

Implications of these contributions for future research and industry application are provided in the conclusion chapter which finalises findings and proposes directions for future work. The publications mentioned are listed below:

1. [7]: This article details the development and evaluation of a computer vision model designed to estimate ladle pouring height, ladle rotation angle and flame severity during the ladle pouring process.
2. [8]: This article focuses on the development, evaluation, and deployment of a model designed to quantify zinc splatter severity occurring on the galvanising line.

3. [9]: This article describes the development, evaluation and application of a model designed to monitor gaseous plumes during gas stirring simulations and give insight on the resulting refractory wear rate and stirring efficiency.

4. [10]: This conference paper, presented at the International Conference on Industry 4.0 and Smart Manufacturing in 2023, investigates the integration of computer vision technology into steel factories to improve the sustainability of the industry.

5. [11]: This conference paper, presented at the International Conference on Manufacturing Research in 2023, investigates modern advances in computer vision applications for steel production and their implications on the sustainability of the industry.

Chapter 2: Literature Review

This chapter will give an overview of a typical computer vision (CV) project workflow followed in industry, before presenting state-of-the-art (SOTA) methods shaping the field. Techniques will be discussed and critically analysed in terms of suitability for application. Additionally, applications of traditional techniques (denoising, tracking and background subtraction), whilst more modern techniques utilising deep learning (object detection and instance segmentation), will initially only focus on theoretical performance evaluations. Finally, applications that use deep learning-based techniques will be presented and critically analysed in terms of their suitability for deployment. Through analysis of both SOTA methods and specific manufacturing applications, gaps in the field will be identified. Some gaps will be directly addressed throughout this project, and others will remain as potential avenues for future research.

2.1 Introduction

Chapter 1 already provided an introduction that explains how CV is a branch of artificial intelligence (AI) and has advanced significantly in recent years which has enabled the development of intelligent systems that interpret visual data with precision. This literature review will build upon the foundational understanding of CV that was laid out in Chapter 1 by presenting advancements in CV technology and applications relevant to the manufacturing sector, with a particular focus on steel production where possible. The objective of this review is to provide a comprehensive background of existing CV techniques and evaluate their potential for real manufacturing applications.

The project workflow of CV is fundamental to understanding how models are integrated into systems as real practical tools that enhance manufacturing processes. Typically, a CV project begins with data collection and pre-processing which ensure datasets are sufficiently large and a suitable format for model development. Denoising methods are considered a crucial part of pre-processing. With the right datasets, model development is conducted which involves training, validation and testing. All object detection and instance segmentation networks discussed in this review undergo this process. For tracking methods, this varies depending on whether deep learning is involved, and for background subtraction methods, training is not required. Following development, models are normally deployed in a real-world scenario on a suitable device which could be a conventional computer, a cloud server, a mobile device, or a single-board computer (SBC). The datasets introduced in this review, with the exception of those related to surface defect detection, are summarised in Table 1. In Section 2.7.4 where surface defect detection is discussed, a separate table is included.

Table 1: Object detection datasets discussed in this review

Dataset	Images	Classes	Description	Source
VOC 2007	9963	20	"Visual Object Classes" - Dataset of common objects for benchmarking object detection models.	[12]
VOC 2010	10103	20	Dataset of common objects for benchmarking object detection models.	[12]
VOC 2012	11530	20	Dataset of common objects for benchmarking object detection models.	[12]
COCO 2015	205000	80+	"Common Objects in Context" - Dataset of common objects in natural environments for benchmarking models on various CV tasks, primarily object detection, segmentation and image captioning.	[13]
COCO 2016	205000	80+	Extension of COCO 2015 with keypoints for human pose estimation.	[14]
COCO 2017	164000	80+	Updated train/val/test splits with continued support for detection, segmentation, captioning, and keypoints.	[13]
Roboflow-100	224714	829	Crowd-sourced dataset for benchmarking models on object detection tasks across a wider variety of domains than VOC and COCO.	[15]

2.2 Denoising Methods

The pre-processing stage of developing CV models can involve techniques with a variety of functions, including but not limited to, data compression for computational efficiency, data normalisation for consistent input formats, data augmentation for maximising the use of limited datasets, and denoising for model accuracy. This section will give an overview of denoising techniques relevant to this project, critical analysis of their application in real-world settings and identification of research gaps.

2.2.1 Histogram Equalisation Methods

Histogram Equalisation (HE) is a commonly used contrast enhancement technique that lends itself well to computer vision tasks [16]. HE works by using the intensity values of all pixels within an image to produce a histogram, and then adjusting intensity values of pixels with more frequent intensities to produce a more evenly distributed histogram [17]. This results in areas of lower local contrast changing to a higher contrast, making them more visible [17]. In greyscale images "intensity" refers to how dark or light a pixel is. In HSV colour space (hue-saturation-value), HE can be performed on value channels whereas in HSL colour space (hue-saturation-luminance), HE can be performed on luminance values. In RGB images the intensity or brightness is more complex [17]. Standard HE is prone to over-enhancing images with highly variable contrast which is common in industrial environments such as steelworks. In Figure 1 an example image with poor contrast is shown on the left, and its corresponding histogram is shown on the right [18]. In Figure 2 the same image in Figure 1 after being processed with HE is shown on the left, and its corresponding histogram is shown on the right [18].

Adaptive Histogram Equalisation (AHE) is a variant of HE that takes local spatial information into consideration by dividing images into tiles and performing HE on each tile [17]. This localised approach is more advantageous in images with more diverse lighting conditions such as industrial environments. In [19], a variety of medical images including X-ray, computed tomography (CT) and magnetic resonance imaging (MRI) scans, were used to evaluate the method against HE and another, more applicable variant discussed next called Contrast-Limited Adaptive Histogram Equalisation (CLAHE). The absolute mean brightness error (AMBE) and peak signal-to-noise ratio (PSNR) metrics were used for evaluation. AMBE measures the quality of brightness preservation after HE processing and a lower number indicates better performance [20], whereas PSNR refers to the extent of distortion and a

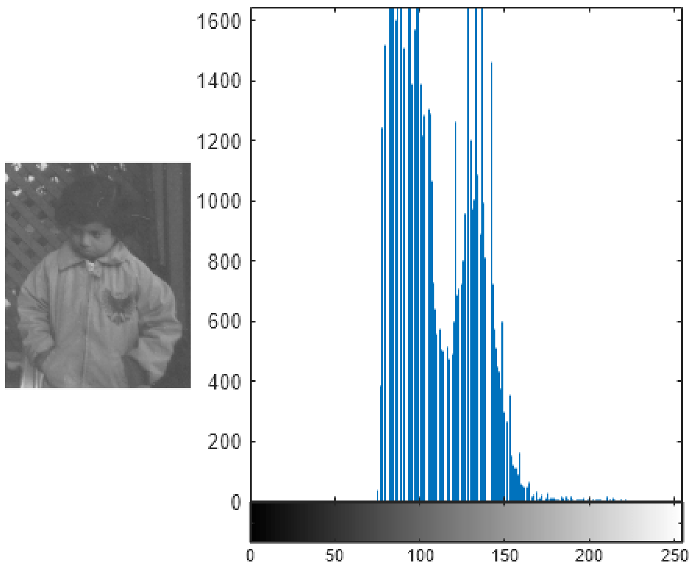


Figure 1: Example of an image with poor contrast and its corresponding histogram

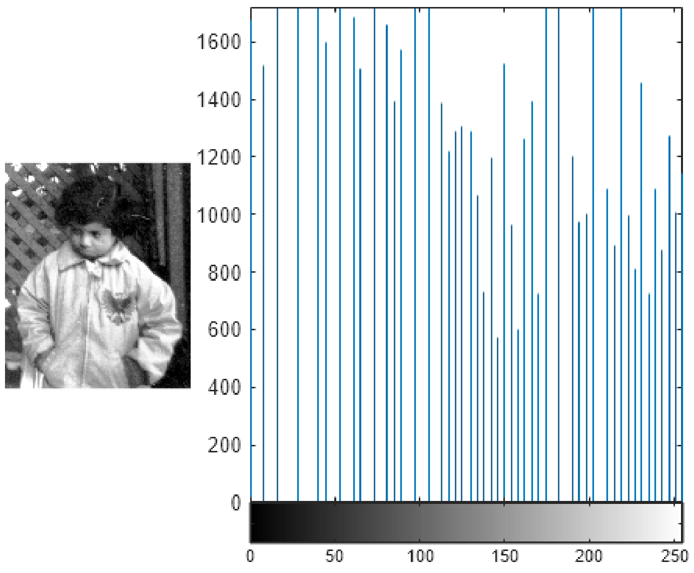


Figure 2: Example of an image processed by histogram equalisation and its corresponding histogram

higher number indicates better performance [21]. Despite the concept of AHE being sound, using AHE in [19] resulted in AMBE being higher and PSNR being lower than when HE and CLAHE were used. This suggests application in manufacturing environments is unsuitable due to poor brightness preservation and image distortion [19].

CLAHE is a further modified variant of AHE that operates by clipping the contrast of each equalised tile and redistributing clipped intensity over histogram bins [17]. This method is more sophisticated than HE and AHE as it avoids overemphasis of any noise present and prevents edge-shadowing [17]. When enhancing MRI images, CLAHE was reported in [19] to achieve an average AMBE value of approximately 20% that of HE and 16% of AHE, whilst achieving an average PSNR value of approximate double that of HE and two and a half times that of AHE, demonstrating its effectiveness in real-world applications. This was further demonstrated in [22] which evaluated CLAHE against HE, contrast stretching and a newly proposed method developed by the authors at enhancing medical images (X-ray, CT, MRI and mammogram images). For PSNR, CLAHE performed better on almost every image, whilst for AMBE it outperformed HE in every case [22]. For PSNR, CLAHE was very close to the performance of the proposed method and for AMBE it was the closest by a significant amount [22]. The results of these studies highlight the capability of CLAHE to enhance image contrast whilst preserving image quality, which is crucial in CV applications for steel processes where lighting conditions may be harsh and high image quality will be necessary for precision. However, it would be beneficial to evaluate the performance of CLAHE on a wide range of steel production environments to identify the strengths and weaknesses of the technique more specifically.

Brightness Preserving Bi-Histogram Equalisation (BPBHE) is another variant that uses two sub-images which capture pixels at the lower and upper halves of the intensity distribution [23]. Both sub-images are processed using HE and then combined by using the pixel values from the dark sub-image for original pixel intensities below the mean intensity, and the pixel values from the bright sub-image for original pixel intensities above the mean intensity [23]. The success of this method has been demonstrated in various use-cases such as with images containing a large amount of snow and clouds (high intensity images), dark rooms (low intensity images) and poorly-lit objects [23]. In comparison to other HE variants, it has shown success in applications such as emphasis of pathological features within brain MRI and lung CT scans [20]. For the CT scans, BPBHE was combined with a Gaussian probability function and resulted in the lowest AMBE for the focused area of the pathological material, as well as for the whole image [20]. Whilst this approach excels in a variety of scenarios, including where there are extreme variations in contrast, the dual-image aspect has the potential to impact computational expense which might be limiting in real-time applications.

The final HE variant that will be discussed is Brightness Preserving Dynamic Fuzzy Histogram Equalisation (BPDFHE). This variant processes images in the fuzzy domain using a fuzzy histogram rather than the standard histogram and Gaussian probability function [24]. Fuzzy logic is a problem solving approach that allows for degrees of truth as opposed to binary logic that only allows true or false, which is useful when dealing with approximations which are typically necessary to deal with in real-world scenarios [25]. The effectiveness of BPDFHE has been demonstrated in [26] where it performed better than CLAHE at enhancing leaf images for disease detection for multiple metrics including mean squared error (MSE) and PSNR, as well as in enhancing high-resolution aerial images in [27] where it performed better than HE and BPBHE in terms of PSNR, UIQI (universal image quality index - a quality retention metric) and AMBE. The leaves and aerial images contained more detail than medical images previously discussed, suggesting BPDFHE can be particularly effective at dealing with more complex images in comparison to other HE variants. However, the fuzzy logic aspect could come with computational expense. In the cases of both BPBHE and BPDFHE, the literature and following analyses suggest these two HE variants could be useful for industrial application but only for highly specific scenarios.

2.2.2 Morphological Operations

Whilst HE variants are beneficial for improving the contrast of images, morphological operations are beneficial for eliminating small, irrelevant entities which reduces noise and can aid in improving the shape of desired structures. Morphological operations discussed in this thesis are erosion and dilation. Erosion involves passing a kernel of a pre-defined size and structure over each frame, computing the local minimum over the kernel area and replacing the image pixel under the kernel anchor point with the minimum value [28]. This process effectively removes the edges of contours so that they are smaller or non-existent and is demonstrated in Figure 3 [29]. While this operation is similar to convolution in that it involves moving a kernel across an image, it differs in that it computes a local minimum rather than performing a weighted sum of the kernel and the pixel values. Oppositely, dilation slides a kernel over images and calculates the local maximum which it uses to replace the image pixel under the kernel anchor point, which effectively thickens contour edges so that contours are larger [28]. This is demonstrated in Figure 4 [30]. In practice, erosion can be used to “shrink away” small noise contours which are unwanted features that may appear in due to dust particles, heat distortion, moving machinery, or other sources of inaccuracies such as generally poor image quality. A follow-up dilation operation ensures helpful contours are reverted to their original state that existed before noise removal occurred.

(a) Before erosion (b) After erosion

Figure 3: Figure showing the effects of erosion

(a) Before dilation (b) After dilation

Figure 4: Figure showing the effects of dilation

This approach has been used in applications such as steel billet defect detection [31] where an 87.5% detection rate was achieved which highlights the effectiveness of it in quality control scenarios and shows efficacy when dealing with varying surface textures. Also, morphological operations aided real-time defect detection of high-speed steel bar in coil [32] where a 96.7% detection rate was achieved by the proposed model. This reinforces the applicability of this approach in steel production environments since real-time capabilities were demonstrated. Additionally, morphological operations were used for size distribution estimation of iron-ore pellets [33] where the approach proposed in the study achieved over 90% accuracy in diameter estimation. The fact that this is based on material handling rather than defect detection emphasises the suitability of morphological approaches for a variety of steel production tasks.

Despite promising outcomes in the literature, applying morphological operations still pose challenges such as requiring optimal sensitivity settings to adequately remove noise without reducing the quality of important features. Similarly to contrast enhancement, it could be beneficial to evaluate the performance of morphological operations on a wide range of steel production environments to build a better understanding of which scenarios could benefit most from its use.

2.3 Object Detection Methods

Object detection is typically a core element of the model development process and encompasses a wide range of techniques that, fundamentally, perform the tasks of object localisation (predicting an object's location using a bounding box) and object classification (predicting an object's class) together. This section will provide an overview of some of the key detection networks that constitute modern CV. This will include SOTA as well as some predecessors of the SOTA to build a comprehensive picture of how the field has evolved rapidly year-on-year. Model evaluation results based on standard evaluation datasets in the field will provide some insight into conventional evaluation methods, as well as demonstrate the evolution of CV since deep learning burgeoned. Additionally, there will be critical analysis of model application to real-world settings and identification of research gaps.

2.3.1 R-CNN

R-CNN (Region-based Convolutional Neural Network) is a basic object detection model that consists of three modules shown in Figure 5 [34]. The first module generates region proposals that define all possible detections available to the detector, the second module is a large convolutional neural network (CNN) that performs feature extraction on each region, and the third module is a set of class-specific linear support vector machines (SVM) that classify the objects within the regions [34].

The authors of the R-CNN paper evaluated the model on the Pattern Analysis, Statistical Modeling and Computational Learning Visual Object Classes (PASCAL VOC) 2007 and 2010 datasets which are described quantitatively in Table 1 [12]. For the 2007 set, several R-CNN variants were compared along with DPM (deformable part model) methods for benchmarking [34, 35]. Results showed that for every category, R-CNN was superior and achieved a mean average precision (mAP) of 0.585 [34]. mAP is the mean of the average precisions (APs) achieved for each class. AP is the area under a plotted precision-recall curve and is a metric commonly used to describe the accuracy of object detection algorithms (see Section 3.4.3 of

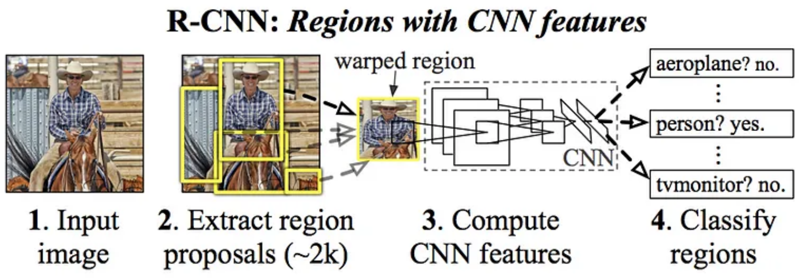


Figure 5: Schematic representing R-CNN object detection

Chapter 3). For the 2010 set, R-CNN was benchmarked with a DPM model [35], Selective Search [36], Regionlets [37] and SegDPM [34, 38]. Results showed that the R-CNN was again superior in all categories with an mAP of 0.537 [34].

Despite success, R-CNN is relatively inefficient, resulting in high computational cost and slow inference times, making it unsuitable for real-time applications [39]. This is because R-CNN proposes each region individually using Selective Search which is an algorithm that firstly generates a set of initial segments through segmentation, and then iteratively merges them based on similarity metrics, such as color and texture, to create region proposals [36]. This process is time-consuming. Furthermore, R-CNN also has a modular pipeline (region proposal generation, feature extraction, classification), that requires individual optimisation for each module rather than end-to-end that could improve speed and accuracy [36, 39]. This has led to the development of more efficient architectures such as Fast R-CNN and Faster R-CNN.

2.3.2 Fast R-CNN

Fast R-CNN is a development from R-CNN that is advantageous due to its improved training and testing speed, combined with its increased accuracy [39]. An overview is shown in Figure 6 [39]. Fast R-CNN is a more streamlined than the modular build of R-CNN [39]. Unlike R-CNN, Fast R-CNN takes region proposals (from a method such as Selective Search) as input and processes a single image at once rather than a single proposal, through use of a pre-trained fully convolutional network (FCN) [39]. A region of interest (RoI) pooling layer is used to extract vectors from the feature map for each proposal which are then used to perform classification and localisation simultaneously [39]. Also, Fast R-CNN eliminates the hundreds of gigabytes storage requirement of R-CNN as it does not cache features [39]. These changes make it much more suitable for speed-critical applications.

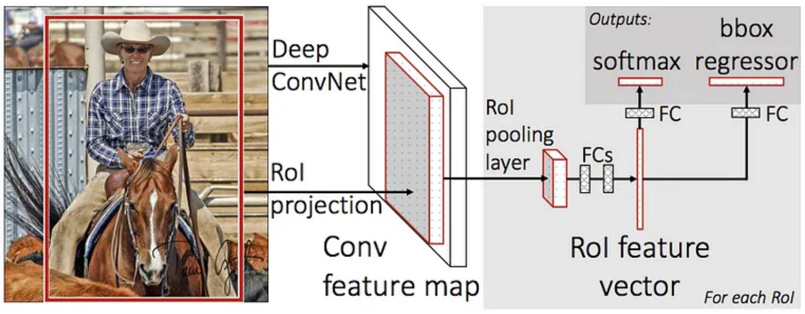


Figure 6: Schematic representing Fast R-CNN object detection

Fast R-CNN was evaluated by the authors by using pre-trained ImageNet [40] models named (in order of smallest to largest network size) CaffeNet (an implementation using Caffe) [41] [34], VGG_CNN_M_1024 [42] and VGG16 (Visual Geometry Group) [43] for feature extraction [39]. Using these networks, from smallest to largest, Fast R-CNN training speed was benchmarked against R-CNN and SPPNet (Spatial Pyramid Pooling Network) [44] and was 18.3, 14 and 8.8 times faster than R-CNN respectively, and significantly passed SPPNet speed also [39]. SPPNet is a CNN that pools each feature map at multiple scales and concatenates pooled feature maps into a single fixed-sized output, enabling multi-scale feature extraction without resizing images or adding more network layers, both of which can introduce inefficiencies [44].

Also, depending on the choice of feature extraction network, Fast R-CNN was 80 to 146 times faster at testing than R-CNN and SPPNet [39]. In terms of inference speed, 0.10, to 0.32 images per second was achieved depending on the choice of feature extractor [39]. The significant reduction in training and testing times of Fast R-CNN are much more aligned with industrial challenges that demand rapid model development and implementation. Using CaffeNet and VGG_CNN_M_1024, Fast R-CNN achieved only slightly lower mAPs than R-CNN which was impressive considering the huge increase in training and testing speeds [39]. The slight mAP reduction highlights the trade-off between speed and precision which is crucial to consider when implementing object detection into a steel production environment. When detail is critical, speed may have to be sacrificed and vice-versa. When using VGG16, Fast R-CNN achieved the highest of all three algorithms with 0.669, whilst R-CNN and SPPnet achieved mAPs of 0.660 and 0.631 respectively [39]. This suggests that Fast R-CNN performance is dependent on the network size and therefore this is a factor that should also be considered when implementing it into an industrial application.

Fast R-CNN precision was evaluated on the VOC 2007, VOC 2010, VOC 2012 and

Microsoft Common Objects in Context (COCO) 2015 [45] datasets which are detailed in Table 1 [39]. For the VOC 2007 dataset, variants of Fast R-CNN were compared to SPPNet and R-CNN, both with bounding box regression [39]. The first variant of Fast R-CNN was trained on VOC 2007, the second was the same except without difficult examples and the third was trained on VOC 2007 and VOC 2012 [39]. Results showed that for 15 out of 20 object categories, the Fast R-CNN trained on two VOC datasets achieved the highest AP (and therefore the highest mAP overall – 0.700) [39]. This demonstrates the robustness of Fast R-CNN across diverse and complex data, which is promising for industry.

Fast R-CNN trained using the VOC 2012 dataset and Fast R-CNN trained using both the VOC 2007 and VOC 2012 datasets, were compared to BabyLearning [46], R-CNN with bounding box regression and SegDeepM [47] for detection performance on the VOC 2010 test set [39]. Results showed that the Fast R-CNN model trained on both datasets achieved the highest AP for 15 of the 20 objects and an mAP of 0.688. The 15 categories Fast R-CNN performed best in for VOC 2010 set were not the exact same as for the VOC 2007 set [39]. These results further emphasise the ability of Fast R-CNN to generalise well on new data since different datasets reveal different strengths and weaknesses of the same model. This is crucial in industrial applications where the environment may change significantly.

The same two Fast R-CNN variants were also evaluated on the VOC 2012 test set against BabyLearning, NUS_NIN_c2000 (NUS referring to the National University of Singapore and NIN referring to "Network-in-Network" architecture) [46] and R-CNN with bounding box regression [39]. The Fast R-CNN variant trained on two datasets achieved the highest AP in 18 of 20 categories and achieved an mAP of 0.684 [39]. Again, the consistently high performance of Fast R-CNN across various datasets shows its effectiveness and adaptability which is desirable in industry.

2.3.3 Faster R-CNN

Fast R-CNN was developed further into Faster R-CNN by adding a new module to the beginning of the architecture [48]. An overview of Faster R-CNN is shown below in Figure 7 [48]. The new module is a deep FCN used as a Region Proposal Network (RPN), in contrast to Selective Search discussed previously [48]. The RPN is trainable end-to-end and regresses region bounds and objectness (probability of a region containing an object) scores at the same time [48]. FCNs use convolutional layers followed by more convolutional layers instead of the more common fully-connected layers. Faster R-CNN works by training the region proposal network (RPN) and Fast R-CNN independently, whilst allowing them to share features through their shared convolutional layers using alternating training [48]. Sharing features means the computational expense and time required to train and test the Faster R-CNN network is greatly reduced in comparison to Fast R-CNN [48].

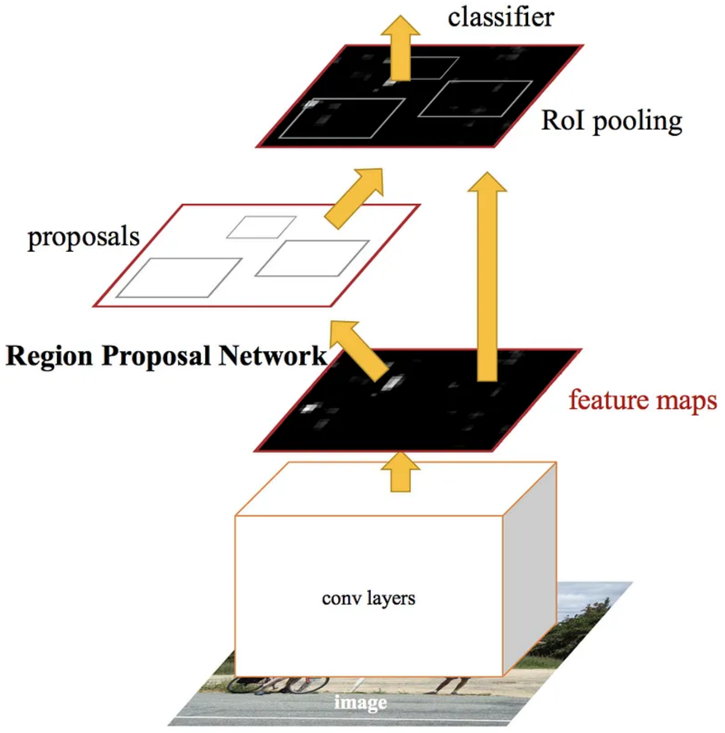


Figure 7: Schematic representing Faster R-CNN object detection

Faster R-CNN performance was evaluated on the PASCAL VOC 2007 and 2012 datasets [48]. Two models were tested (Zeiler and Fergus (ZF) [49] and VGG-16) which used five and 13 shared convolutional layers respectively [48]. Faster R-CNN using the ZF network ran at 17fps [48], and using the VGG network ran at 5fps [48]. Fast R-CNN that used the VGG network ran at 0.5fps [48]. These results show that Faster R-CNN can be ten to 34 times faster than Fast R-CNN depending on the CNN used [48]. Also, the introduction of an RPN significantly improves speed, which is due to a reduction in the number of proposals requiring processing. This is important when considering real-time industrial application due to the high operating speeds found in production sites.

For the VOC 2007 test set, variants of Faster R-CNN were compared with variants of Fast R-CNN [48]. Results showed that Faster R-CNN trained on the COCO 2015, VOC 2007 and VOC 2012 datasets achieved the highest mAP of 0.788, followed by the Faster R-CNN trained on just the VOC datasets (0.732 mAP), followed by the Fast R-CNN trained on just the VOC datasets (0.700 mAP) which was closely followed by Faster R-CNN trained on only the VOC 2007 dataset (0.699 mAP) [48]. Whilst a model trained on more data performs better on unseen data, overall, Faster R-CNN is more accurate than Fast R-CNN on this dataset. These results show the capability of Faster R-CNN to generalise well across different datasets. This in particular shows robustness since COCO and VOC are unrelated (as opposed to VOC variants which are), which is promising for real-world application where unpredictable scenarios are common.

Regarding the VOC 2012 test set, Faster R-CNN trained on all three datasets achieved the highest mAP again (0.759) and this was followed by Faster R-CNN trained on just the VOC datasets (0.704 mAP) [48]. This trend continued as with the VOC 2007 evaluation results [48]. Results also demonstrate that Faster R-CNN performed better than Fast R-CNN (0.684 mAP) on the VOC 2012 test set [39, 48]. These results reinforce the real-world applicability of Faster R-CNN and the benefits of the added RPN module. However, despite the benefits, the complexity of configuring the RPN could be considered a limitation, since it often requires fine-tuning which is balanced with the detector component for optimal performance and efficient use of computational resources. This can be particularly challenging for less experienced engineers and could be detrimental in resource-constrained scenarios.

2.3.4 YOLO

YOLO (“You Only Look Once”) is part of a family of networks that are an alternative option to the R-CNN family. An overview of YOLO is shown in Figure 8 [50]. YOLO works by treating object detection as a single regression problem, hence the name [50]. Unlike multi-stage methods such as R-CNN and its derivatives, YOLO uses a CNN to predict object bounding boxes and object class probabilities simultaneously, resulting in fast inferences [50]. It firstly divides an image into a grid and each grid cell predicts bounding boxes, confidence scores for those boxes (indicating accuracy and object presence), and class probabilities on those cells containing an object [50]. YOLO uses Darknet-19 as an image classifier, which is a fast open source neural network that allows YOLO to process images at high speed [50]. Due to the good accuracy and high speed of YOLO, it is suitable for applications requiring real-time processing [50].

When evaluated on the PASCAL VOC 2007 dataset, YOLO achieved a speed of 45fps on a Titan X graphical processing unit (GPU) with no batch processing required (meaning data can be streamed) [50]. An evolved version of YOLO named Fast-YOLO improves this for a speed of 155fps at the cost of reduced accuracy [50]. YOLO and Fast-YOLO achieved mAPs of 0.634 and 0.527 respectively [50]. These results suggest that YOLO is

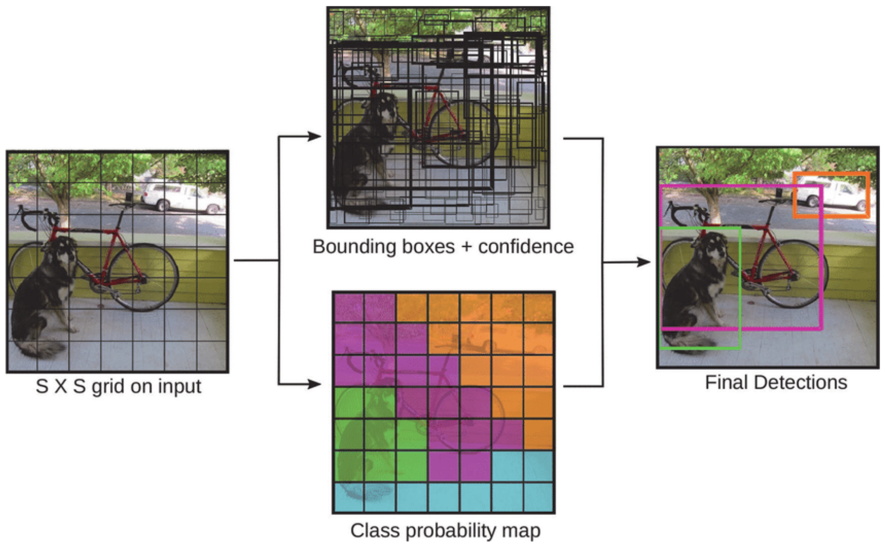


Figure 8: Schematic representing YOLO object detection

highly suitable for real-time applications but at the cost of precision. In deployment, the trade-off between speed and precision must be carefully considered based on the demands of the specific challenge.

2.3.5 YOLOv2

YOLOv2 is an evolution of the original YOLO model which addressed localisation errors and lower recall compared to Fast R-CNN, whilst maintaining the high speed of the original model [51]. Improvements included the addition of batch normalisation to all convolutional layers which removed the need for other regularisation methods and dropout layers, training the classifier on 448x448 images rather than 224x224 which improved precision, replacing direct bounding box prediction of the fully-connected network with pre-defined anchor boxes which enhanced recall, as well as using k-means clustering to optimise anchor box dimensions which improved accuracy and stability [51]. A diagram representing this process is shown in Figure 9. Other improvements included a passthrough layer that takes a higher resolution feature map from an earlier layer to improve the detection of smaller objects, and a multi-scale training strategy where the model dynamically adjusts input image size during training, improving its ability to deal image size variations [51].

YOLOv2 was evaluated against a variety of models, SSD512 (Single-Shot Detector), Fast R-CNN and Faster R-CNN most notably, on the VOC 2007, VOC 2012 and COCO 2015 datasets (shown in Table 1) [51]. On the VOC 2007 dataset, YOLOv2 was able to achieve the highest mAP and the the highest frame rate [51]. On the VOC 2012 dataset,

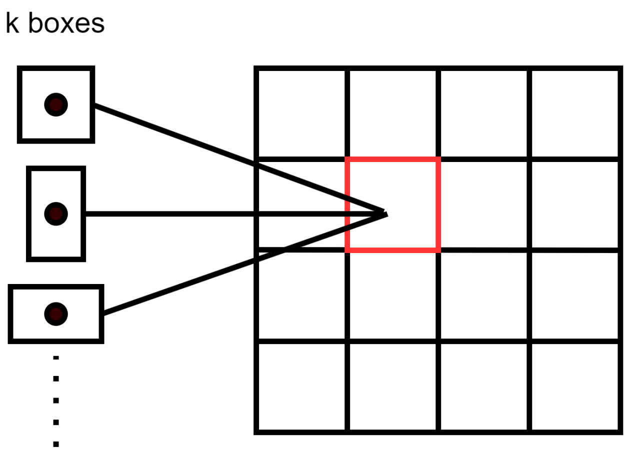
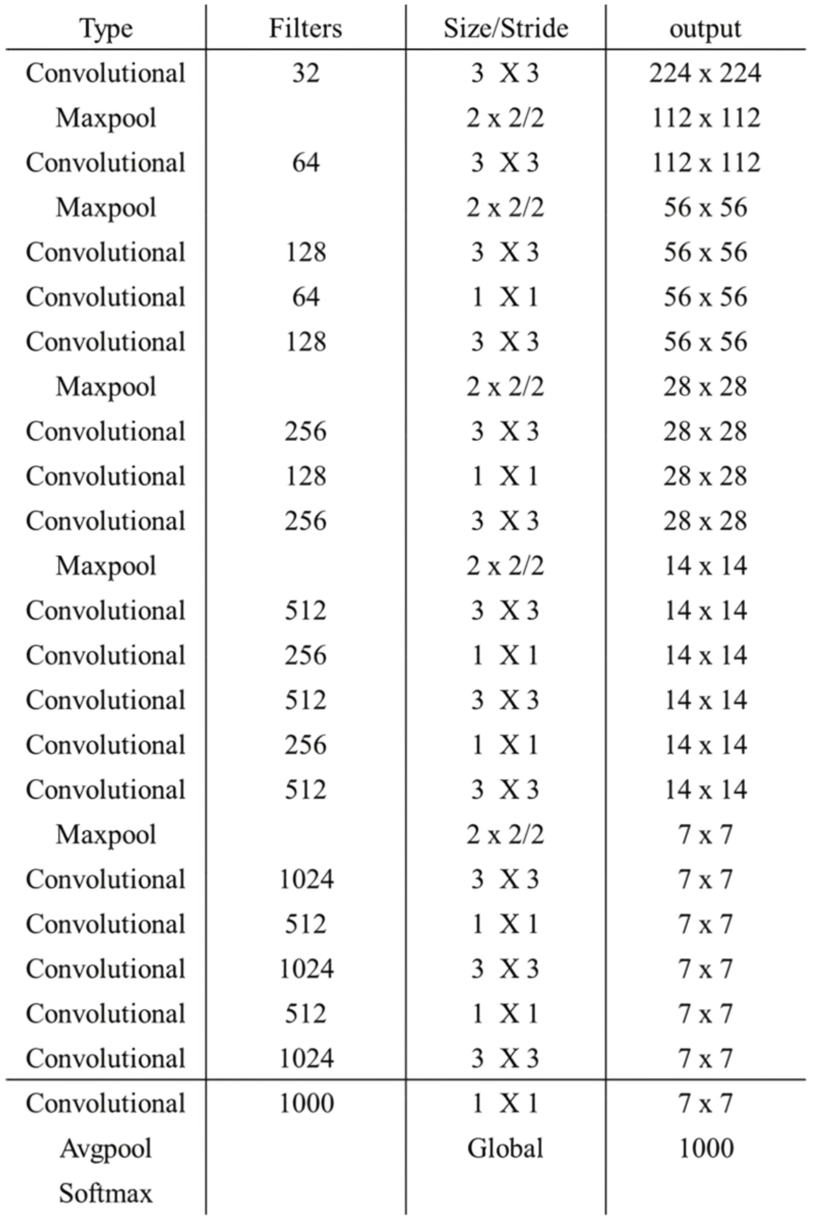


Figure 9: Schematic representing bounding box prediction with k pre-defined anchor boxes

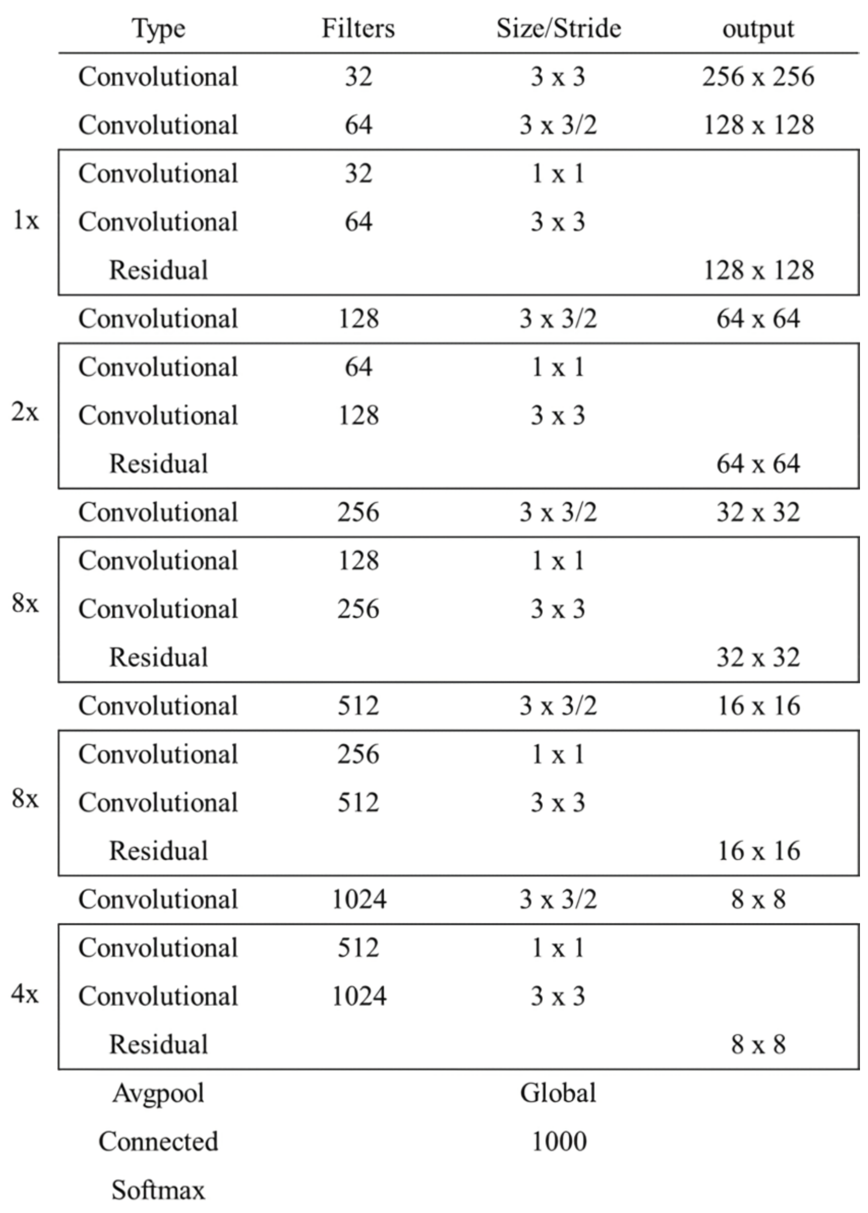
YOLOv2 achieved a competitive mAP that was marginally surpassed by SSD512 [52] and ResNet (Residual Neural Network) [53], but was over twice as fast [51]. Finally, on the COCO 2015 dataset, YOLOv2 was competitive at the 0.50 IoU (intersection-over-union) threshold, but using the COCO mAP metric it was surpassed by Faster R-CNN, SSD300 and SSD512 [51, 52]. IoU measures the overlap between the predicted and actual object boundaries. It is calculated as the area of overlap divided by the area of union between predicted and ground-truth bounding boxes. A prediction with a higher IoU means the model is more accurate and the prediction is more likely to be considered a true positive (depending on if the IoU surpasses the pre-defined threshold). The COCO mAP metric accounts for multiple IoU thresholds and is therefore more informative than just using one. More details on this are provided in Section 3.4.1. It is worth noting that whilst YOLOv2 was surpassed by a maximum of 0.052 mAP ($\sim$20% increase), it made inferences at least twice as fast as all other models evaluated [51]. Despite the inferior precision of YOLOv2, the advantage it demonstrated with regards to speed was significant enough to make it a strong choice for real-time applications. Similarly to the original YOLO model, YOLOv2 also poses a trade-off between speed and precision which should be considered if implementing it for industrial application. Whilst an R-CNN variant may be more suitable for precision-critical applications such as crack detection, YOLOv2 may be more suitable for time-critical applications such as monitoring high-speed production lines.

2.3.6 YOLOv3

YOLOv3 is the successor of YOLOv2 and incorporates several improvements. Firstly, for class prediction YOLOv2 uses a softmax layer which assumes that each box has one class which is not always true (i.e. objects overlap), so YOLOv3 uses a multi-label approach instead [54]. This allows the model to deal with overlapping objects which is highly beneficial in industrial environments. Secondly, boxes are predicted across three different scales similarly to feature pyramid networks (FPNs) which are discussed in the following subsection in relation to RetinaNet [54, 55]. This is achieved through the new Darknet-53 in place of Darknet-19, which enriches feature extraction with more convolutional layers (19 to 53) and uses ResNet-type skip connections [54]. Skip connections allow training data to bypass certain layers to facilitate gradient flow during backpropagation to combat vanishing gradients, a common problem in deep neural networks [53]. Darknet-19 and Darknet-53 architectures are shown in Figure 10 [56]. Feature maps from different layers are then concatenated which aids detection of smaller objects through a mixture of fine-grained and semantic information, before the last layer predicts the bounding box, objectness and class predictions [54].



Type	Filters	Size/Stride	output
Convolutional	32	3 X 3	224 x 224
Maxpool		2 x 2/2	112 x 112
Convolutional	64	3 X 3	112 x 112
Maxpool		2 x 2/2	56 x 56
Convolutional	128	3 X 3	56 x 56
Convolutional	64	1 X 1	56 x 56
Convolutional	128	3 X 3	56 x 56
Maxpool		2 x 2/2	28 x 28
Convolutional	256	3 X 3	28 x 28
Convolutional	128	1 X 1	28 x 28
Convolutional	256	3 X 3	28 x 28
Maxpool		2 x 2/2	14 x 14
Convolutional	512	3 X 3	14 x 14
Convolutional	256	1 X 1	14 x 14
Convolutional	512	3 X 3	14 x 14
Convolutional	256	1 X 1	14 x 14
Convolutional	512	3 X 3	14 x 14
Maxpool		2 x 2/2	7 x 7
Convolutional	1024	3 X 3	7 x 7
Convolutional	512	1 X 1	7 x 7
Convolutional	1024	3 X 3	7 x 7
Convolutional	512	1 X 1	7 x 7
Convolutional	1024	3 X 3	7 x 7
Convolutional	1000	1 X 1	7 x 7
Avgpool		Global	1000
Softmax			



	Type	Filters	Size/Stride	output
	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3/2	128 x 128
1x	Convolutional	32	1 x 1	
	Convolutional	64	3 x 3	
	Residual			128 x 128
	Convolutional	128	3 x 3/2	64 x 64
2x	Convolutional	64	1 x 1	
	Convolutional	128	3 x 3	
	Residual			64 x 64
	Convolutional	256	3 x 3/2	32 x 32
8x	Convolutional	128	1 x 1	
	Convolutional	256	3 x 3	
	Residual			32 x 32
	Convolutional	512	3 x 3/2	16 x 16
8x	Convolutional	256	1 x 1	
	Convolutional	512	3 x 3	
	Residual			16 x 16
	Convolutional	1024	3 x 3/2	8 x 8
4x	Convolutional	512	1 x 1	
	Convolutional	1024	3 x 3	
	Residual			8 x 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 10: The architectures of Darknet-19 (left) and Darknet-53 (right)

YOLOv3 was evaluated using the COCO 2017 dataset (shown in Table 1) and the mAP-50 metric (mAP based on a 0.5 IoU threshold). YOLOv3-320, YOLOv3-416 and YOLOv3-608 achieved mAP-50 values of 0.515, 0.553 and 0.579 respectively, whilst achieving inference times of 22ms, 29ms and 51ms respectively [54]. Compared to other object detection models such as SSD and RetinaNet variants, YOLOv3 performed the best in terms of inference time and competitively in terms of mAP, with YOLOv3-608 had the second highest mAP-50 [54]. These results highlight the improved balance between speed and precision compared to YOLOv2 since speed is still superior whilst precision is more competitive. This makes YOLOv3 more promising for diverse industrial application than YOLOv2, meaning it may be suitable for both real-time applications and those which require more precision. However, the precision is not sufficient for precision-critical applications. YOLOv3-608 was also evaluated using the COCO 2017 dataset with the COCO mAP metric, where it was marginally surpassed by Faster R-CNN variants and SSD variants (which it greatly surpassed in speed) [54]. There was a significant gap of 0.078 mAP between YOLOv3 and RetinaNet performance, however RetinaNet was reported to take 3.8 times longer when making inferences [54]. These results show the trade-off between speed and precision in object detection tasks. If precision is critical to an application, RetinaNet would be a better choice, whereas YOLOv3 would be superior for time-critical applications.

2.3.7 RetinaNet

Naturally, discussion now leads to RetinaNet before continuing with the most recent YOLO models. RetinaNet is a one-stage object detection network that was released in 2017, between the release of YOLOv2 and YOLOv3 [55]. The model addresses the problem of class imbalances using a specialised focal loss function that, during training, reduces the weight of easy samples and increases the weight of difficult samples [55]. Similarly to YOLO models, RetinaNet does not separate region proposal and classification into two stages [55]. These two key characteristics of RetinaNet enable it to perform with fast inference times whilst maintaining high accuracy [55]. The architecture is constituted by a CNN backbone network for feature extraction which is typically a ResNet variant combined with an FPN to aid detection at various scales [55]. A diagram of an FPN can be seen in Figure 11 [57]. FPNs combine high-level semantic features with low-level detailed features in order to capture information from multiple scales. This results in robust performance across different scales using a single network.

The authors of the RetinaNet paper benchmarked it on the COCO 2017 dataset against

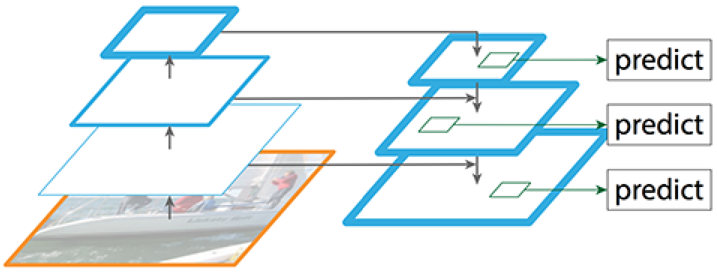


Figure 11: Schematic representing the architecture of an FPN

Faster R-CNN variants, SSD variants and YOLOv2 using mAP [55]. The results showed that RetinaNet achieved significantly higher mAP values (0.391 and 0.408 for two different variants) than other models, with the next best mAP being from a Faster R-CNN variant at 0.368 [55]. Another evaluation involving other RetinaNet variants, SSD variants and YOLOv2, showed RetinaNet variants achieved the highest COCO mAP overall and made inferences at a competitive speed, with different variants offering a different trade-off between mAP and inference speed [55]. These results indicate the superior precision of RetinaNet whilst being competitive with regards to speed. Similarly but opposite to YOLOv3, RetinaNet would be well-suited to precision-critical applications such as defect detection, and may even be used at moderately high speeds such as intermittently checking alignment of steel sheets, but not in applications where high speed is critical such as real-time monitoring of conveyor belts.

2.3.8 YOLOv5

YOLOv5 is the fifth iteration of YOLO which features new improvements. Firstly, CSP-Darknet-53 is used as the CNN backbone which incorporates a Cross Stage Partial Network (CSPNet) which improves computational efficiency by dividing the base layer feature map into two parts, and then re-combining them after one part has passed through subsequent layers and one has skipped them [58]. Secondly, spatial pyramid pooling (via SPPNet) enhances feature extraction at various scales and a Path Aggregation Network (PANet) [59] refines feature aggregation, which together improve localisation accuracy. An overview of YOLOv5 is provided in Figure 12 which shows how data flows from CSP blocks for feature extraction, to SPP and PANet for multi-scale feature extraction and aggregation, before reaching the output layers.

YOLOv5 is currently one of the most suitable choices for applications due to its speed, flexibility, active open source community, user-friendly implementation and general ease of

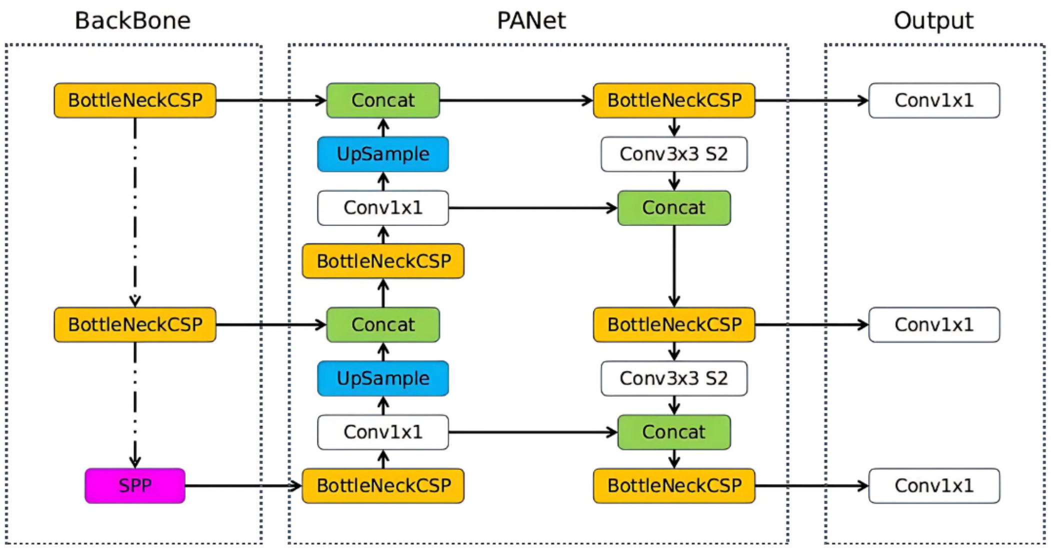


Figure 12: Schematic representing the architecture of YOLOv5

deployment. Whilst it is recognised that YOLOv7 [60], YOLOv8 [61], YOLO-NAS (YOLO Neural Architecture Search) [62] and others are now available and suitable for application with good support, they were not released by the time any case study presented in this thesis had began. As well as this, YOLOv5 has more information built up by community contributions over time. Despite all of these positive points, there is no official paper for YOLOv5 and most available information on it is located at the Ultralytics Github repository, managed by Glenn Jocher, the developer of YOLOv5 [63].

There is no official paper benchmarking the model, however a study exists evaluating YOLOv5 performance [64], which reports YOLOv5 as superior to EfficientDet [65] in terms of COCO mAP and inference speed on the COCO 2017 dataset, whilst the EfficientDet paper [65] reports variants of its own that are superior to YOLOv3, RetinaNet and multiple other competitive models in both speed and accuracy on the COCO 2017 dataset using the COCO mAP metric. This is not a direct comparison made by one author, however it is the best available to date and does show the effectiveness of the model since the same dataset and same metric was used for comparison. It implies that YOLOv5 outperforms older models in terms of speed and precision, which makes it a very promising choice for a variety of steel production applications that are both time-critical and precision-critical. However, the absence of a direct comparison in peer-reviewed literature means YOLOv5 should undergo rigorous performance evaluation if chosen for application, to ensure it meets the requirements of the task at hand.

2.3.9 YOLOv8

At the time of writing this thesis, there is currently no official paper released on YOLOv8, similarly to YOLOv5. Some of the key improvements of YOLOv8 firstly include a C2f module (cross stage partial bottleneck with two convolutions) to the CSP backbone which combines high-level features with contextual information to improve detection accuracy [58, 61]. Secondly, YOLOv8 is anchor-free which reduces box predictions, speeding up non-maximum suppression (NMS) and reducing computational expense [58, 61]. YOLOv8 can also perform other tasks such as instance segmentation, pose estimation and classification [61]. A full diagram of the YOLOv8 architecture is excluded from this review for conciseness, however it can be found at [66].

YOLOv8 has been benchmarked against previous versions using the COCO 2017 dataset and the COCO mAP [61]. Whilst YOLOv5n (n for "nano" - the smallest, fastest variant) achieved approximately 0.275 mAP at an inference time of approximately 1ms per image, YOLOv8n achieved approximately 0.375 mAP with virtually the same inference speed [61]. This is a significant improvement in precision without sacrificing speed, which makes YOLOv8 even more diverse in terms of industrial application than YOLOv5. Moving across from the smallest to largest variants, the performance gap between YOLOv8 and YOLOv7 became obvious only at the large and extra large network sizes, where YOLOv8l achieved approximately 0.02 more mAP than YOLOv7l at the cost of approximately 0.10ms per image, and YOLOv8X achieved about 0.01 more mAP than YOLOv7X at the cost of approximately 0.25ms per image [61]. These results highlight the ongoing theme of a speed-precision trade-off if the maximum performance for one of them is desired. However, it is clear that as YOLO variants develop over time, the lower and upper bounds of speed and precision improve which makes successive models more suitable for a wider range of industrial applications. One limitation of the literature surrounding YOLOv8 is that there is a lack of it, particularly in industrial application and even more specifically, for steel production processes. This gap will close over time as more applications are developed.

2.3.10 YOLO-NAS

YOLO-NAS (YOLO Neural Architecture Search), developed by Deci, is one of the most recent YOLO variants released [62]. It uses a technique called Neural Architecture Search (NAS) to improve upon YOLOv8 in terms of speed, accuracy and efficiency [62]. Also, it is quantised using a hybrid approach which balances between accuracy and latency [62]. Quantisation is a technique that reduces computational costs for more rapid, efficient deployment by converting high-precision data types into lower precision data types [67]. Additionally, it uses Deci's proprietary Automated Neural Architecture Construction technology (AutoNAC) to fine-tune the model structure for optimal performance [62]. Finally, YOLO-NAS uses methods such as Knowledge Distillation (KD) and Distribution Focal Loss (DFL) during training to enhance its ability to classify and localise objects [62]. A full diagram of the YOLO-NAS architecture is excluded from this review for conciseness, however it can be found at [62].

Upon evaluation using the COCO 2017 dataset, YOLO-NAS variants outperformed all other YOLO variants, from version five to eight, significantly on both mAP and inference speed [62]. Upon evaluation using the the Roboflow-100 dataset which is in Table 1, both the small and medium variants of YOLO-NAS outperformed YOLOv5, YOLOv7 and YOLOv8 on mAP (extrapolation suggests other YOLO-NAS variants also being superior) [62]. These results emphasise the rapid advancements in object detection with each successive model. This means that new models are becoming increasingly adaptable to a range of industrial applications, but also it exposes a gap in the research for extensive evaluation of newly released models with regards to industrial, and in particular steel industry-based applications. Whilst initial evaluations are useful and give a reliable indication to which models generally show superior performance in terms of speed and precision, there is a lack of datasets and evaluation approaches that indicate how they perform in harsh industrial environments. Similarly to YOLOv8, YOLO-NAS suffers from a lack of peer-reviewed publications that report comparable performance metrics.

2.4 Tracking Methods

Object tracking is continuously identifying objects and following them as they move through time and space. This requires precision and speed, but is most challenging when it requires robustness to dynamic environments such as those found in steel industry. Some challenges here include occlusions, similar looking objects, and objects moving in and out of the field of view. Whilst challenging, object tracking adds an additional dimension to visual understanding on top of detection which can be highly beneficial. In this section, an overview of traditional techniques such as Kalman filtering and the SORT (Simple Online and Real-Time Tracking) algorithm is provided, as well as more modern approaches that utilise deep learning such as LSTM (long short-term memory) networks and the DeepSORT (Deep Simple Online and Real-Time Tracking) algorithm. They will be critically analysed in terms of their suitability for real application where appropriate, and research gaps will be identified.

2.4.1 Kalman Filtering

A Kalman filter is an algorithm that uses an initial estimation, combined with measured values over a series of timesteps, to predict the future state of a system [68]. Whilst the initial estimation might not be precise due to limited information, the algorithm improves its accuracy through a two-step process [68]. Firstly, it makes a prediction and secondly, it updates that prediction based on the most recent measurements it is provided [68]. Using this approach iteratively leads to precise tracking of a desired variable, such as the position of an object [68]. A diagram representing the Kalman filtering process can be seen in Figure 13, and specific equations used in Kalman filtering are presented in Section 4.2.4.

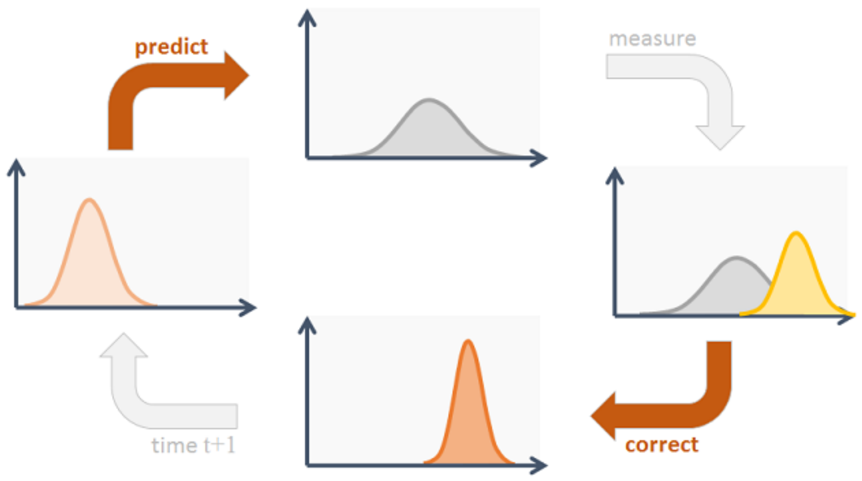


Figure 13: Schematic representing the Kalman filtering algorithm

Kalman filters have demonstrated their practical capabilities both as standalone tools and in combination with machine learning (ML) techniques. One notable application involved integrating Kalman filters with a fuzzy expert system for tracking tool tips for fastening [69]. The Kalman filter aided estimation of tool orientation and center of mass location, and the system reduced tool position error during an experiment where eight bolts were fastened [69]. Whilst the operator approach gave a final position error of 93mm, the model achieved 6mm which demonstrates a substantial improvement in operational precision [69]. Kalman filtering was also used in welding applications, where it was used to enhance the robustness of weld position detection and seam tracking [70]. In this case, Kalman filtering led to a significant reduction in weld position covariance error from 0.0084mm to 0.0010mm, whilst the seam tracking error decreased from 0.33mm to 0.11mm [70]. Again, these results show that Kalman filtering led to a significant reduction in error which demonstrates the effectiveness of it in high-precision applications. Kalman filtering was also used in cutting of gamma-prime strengthened alloys, where it was used for tool flank wear estimation [71]. In one experiment, compared to manual measurements the root mean square error (RMSE) was reduced by 41%, in a repeat experiment it increased error by 8%, and then reduced it by 25% in a third experiment [71]. These results show how Kalman filters can be sensitive to environmental changes which could be detrimental for application, depending on the scenario.

Examples of applications using Kalman filter variants include [72] where an extended Kalman Filter (EKF) was used to estimate flank wear area during wet turning of Inconel 718, where it increased the accuracy of estimation by 60%. This suggests that Kalman filtering can significantly improve operational efficiency and product quality by mitigating inaccuracies that lead to defects. Damage detection in composite beams via tracking of the neutral axis under different loading conditions was achieved using multi-rate Kalman filtering in [73]. The traditional approach of directly estimating damage resulted in false negatives due to a high standard deviation in performance, however using Kalman filtering, the standard deviation was much smaller and therefore avoided false negatives [73]. This demonstrates the ability of Kalman filtering to improve the reliability of structural health monitoring systems which could transfer to steel production equipment health monitoring applications. Also, contact force and torque sensing for robotic manipulators in manufacturing tasks was achieved using adaptive Kalman filtering, where force estimation RMSE was between 0.78N and 1.35N, whilst torque estimation RMSE was between 0.12Nm to 0.18Nm [74]. This application emphasises the suitability of Kalman filtering for enhancing the precision of robotic systems, which is becoming increasingly applicable to manufacturing production lines, and could also transfer to processes involving other moving parts.

There are a range of examples of Kalman filters being combined with machine learning for manufacturing applications. Chatter detection during milling was achieved using AlexNet (a deep CNN) [75] with Kalman filtering in [76] with 98.9% accuracy. This high level of accuracy demonstrates the potential of combining machine learning models with Kalman filtering for dynamic process monitoring and suggests it could play a critical role in minimising defects and equipment downtime. Steel sheet coils were tracked at 8fps during transport to the uncoiler using a CNN with Kalman filtering, where a standard deviation of below 15 pixels was achieved [77]. These results expose the high precision capabilities of Kalman filters, as well as their real-time capabilities, which makes them a valuable asset in real-time monitoring systems in manufacturing environments. Furthermore, an EKF was used in conjunction with a vanilla artificial neural network (ANN) to monitor tool wear condition with a classification accuracy of 89.2% [78]. This reinforces the suitability of Kalman filtering in maintenance applications. Finally, deformation force monitoring of aero-engine casing during machining was achieved by combining a deep autoregressive network with Kalman filtering, which resulted in a monitoring success rate 30% higher than the traditional approach [79]. Again, this result emphasises the benefits of using Kalman filtering with machine learning to improve reliability of monitoring systems.

2.4.2 LSTMs

Long short-term memory networks (LSTMs) are a more sophisticated type of recurrent neural network (RNN) that are designed to retain information over multiple timesteps, making them useful when dealing with sequential data such as video frames [80]. Whilst traditional RNNs can be effective at remembering information from recent timesteps using their hidden state, LSTMs have a unique gating architecture shown in Figure 14 [81]. This includes an input gate (middle σ and $tanh$ operations), a forget gate (left-side σ operaton) and an output gate (right-side σ and $tanh$ operations) [80]. The input gate controls the flow of new information into the cell state, the forget gate decides what information is discarded from the cell state, and the output gate controls which data is output from the cell state to the hidden state [80]. This means LSTMs can retain information over longer sequences and retain the most relevant data, which is a key aspect of object tracking where the environment is dynamic [80]. Whilst LSTMs are excellent at dealing with sequential data, CNNs are well-suited to dealing with spatial data, making the combination of the two a powerful force for dealing with spatio-temporal scenarios such as when tracking objects through videos [80].

In the context of manufacturing, LSTMs have been used to prevent cyber-physical attacks compromising mechanical properties of additively manufactured products within 0.85ms, with precision and recall values of 0.950 and 0.980 respectively [82]. This suggests LSTMs can be used in applications where precision is critical, and also that they may be useful in real-time applications. A bi-directional LSTM was also proposed for tracking and predicting remaining useful life (RUL) of manufacturing machines with an RMSE of 15.42 cycles, which was superior compared to a deep CNN, an RNN (recurrent neural network), an MLP (multi-layer perceptron - a type of classic neural network) and an LSTM, which achieved RMSE values ranging from roughly 18 to 21 cycles [83]. The superior performance highlights the effectiveness of LSTMs at dealing with sequential data and in particular, equipment health monitoring data, which is useful in steel production. LSTMs were also used as part of a closed-loop system to track the trajectory of piezoelectric actuators and reduced the maxi-

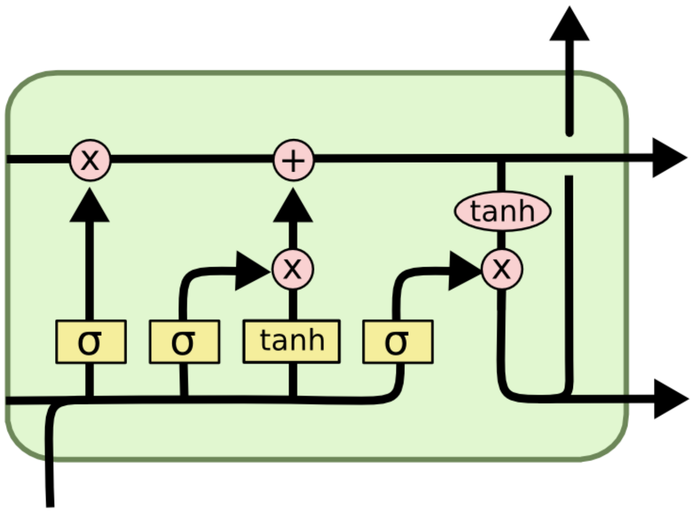


Figure 14: Schematic representing LSTM architecture

mum tracking error from 1.59µm to 0.15µm (90.4% reduction) [84]. This level of precision reinforces the idea that LSTMs are suitable for high-precision engineering tasks. Furthermore, a bi-directional LSTM network was used in a smart manufacturing system involving throwing and catching robots for enhanced part transportation, where the LSTM was used for trajectory tracking and prediction of thrown objects and speeds of up to 10ms^{-1} over distances of up to 3m were achieved, with a maximum error achieved of no more than 2mm [85]. This shows that LSTMs are precise but also that they are appropriate for real-time applications in manufacturing environments. However, since LSTMs are a type of neural network, they typically require large datasets for training and can be computationally expensive to train, which could be limiting in resource-constrained environments.

CNN-LSTMs have been used for electro-mechanical equipment health condition recognition and prediction where 98.6% test accuracy was achieved [86], weld penetration monitoring using dynamic weld pool images with 0.3mm mean square error (MSE) [87] and power data pattern detection and tracking for manufacturing sites with a test loss of 0.1197 [88]. Note that loss is a measure of the difference between predictions and real values. These applications demonstrate the robust model performance that can be achieved across a variety of scenarios when integrating LSTMs with CNNs. This is because LSTMs exceed at dealing with sequential data, whilst CNNs exceed at dealing with spatial data.

2.4.3 DeepSORT

DeepSORT (Deep Simple Online and Real-Time Tracking) is an advancement of the original SORT (Simple Online and Real-Time Tracking) algorithm [89] and is a key player in many object tracking tasks. SORT was developed as a straightforward approach for tracking objects in real-time using motion information, and uses Kalman filtering discussed in a previous subsection [89]. Whilst SORT is fast and efficient, it has issues with handling occlusions and creates too many "identity switches" when objects are blocked [90]. DeepSORT integrates SORT with YOLO (version is optional) to enable sophisticated feature extraction of objects. This, as well as data association techniques, greatly improve Kalman filter tracking performance across sequential frames, particularly in cases of occlusion and objects with similar appearances [90]. This is further supported through a track management module [90]. A diagram providing an overview of DeepSORT is shown in Figure 15 [91]. DeepSORT allows for more robust object tracking, therefore lending itself well to various real-world applications such as tracking moving machinery, humans and goods [90, 92, 93]. Due to deep learning integration, the computational demands of DeepSORT are increased compared to SORT which may need to be considered in industrial application, depending on the available resources and project goals.

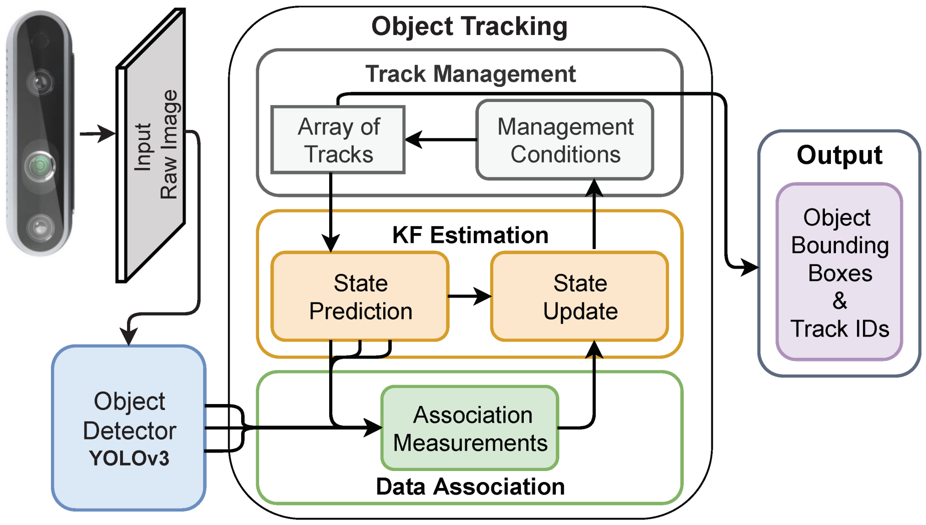


Figure 15: Schematic representing the DeepSORT algorithm

DeepSORT was combined with YOLOv3 to track crane movement in a manufacturing site as presented in [92] where no evaluation was performed, which highlights a gap in quantitatively measuring tracking performance. This is because even though conventional metrics exist for object tracking such as multi-object tracking accuracy (MOTA), reports often lack thorough evaluation using these metrics and typically focus more on evaluating detection performance, since it is often the foundation of tracking and high performance detection often leads to high performance tracking (see Section 3.4.5). Another article reported that DeepSORT and ScaledYOLOv4 were combined for defect tracking for a real-time manufacturing system which reduced double counting of defects by over 50% which demonstrates the real-time capabilities of DeepSORT as well as its suitability for quality control applications [94]. Additionally, DeepSORT and YOLOv5 were combined to help distinguish humans from goods in warehouse scenarios in [93] where, using the MOTA tracking metric, a performance of 75.8% was achieved. The quantitative assessment using MOTA in [93], and the lack of it in [92] suggests there is inconsistency in the literature with regards to measuring tracking performance. MOTA is a widely used metric for evaluating the performance of multi-object tracking algorithms and takes into account false positives, false negatives and identity switches (see Section 3.4.5) [95].

2.5 Instance Segmentation Methods

Instance segmentation methods are an advancement upon object detection methods in that they combine the tasks of detection and semantic segmentation [96]. In semantic segmentation, all pixels within an image are classified into a category according to which object it represents, resulting in full image segmentation into different regions. However, this does not include differentiation between individual instances of the same class. In contrast, instance segmentation distinguishes between instances of the same class, allowing for objects to be identified as separate entities [97]. In an industrial context, distinguishing between multiple objects of the same type is an incredibly useful ability. Instance segmentation is typically performed using deep learning models that constitute of a combination of CNNs and other techniques. Segmentation, and especially this type, offers insights into visual data that is much more specific than object detection, since it not only localises and classified objects, but also identifies the precise boundaries of them at pixel-level. This section will give an overview of some of the most prominent approaches to this task to date, critically analyse their suitability for real application where appropriate, and identify research gaps.

2.5.1 Mask R-CNN

Mask R-CNN (Mask Region-based Convolutional Neural Network) enhances Faster R-CNN by enabling pixel-level segmentation capabilities [98]. After feature extraction is performed by the CNN backbone, an RoIAlign layer is used to refine feature maps, which is an improvement upon the RoIPool method used in previous models [98]. RoIPool divides RoIs into quantised spatial bins which can lead to misalignment between the RoI and its features [98]. RoIAlign however, uses bi-linear interpolation on each bin prior to aggregation which ensures alignment and therefore more accurate segmentation [98]. Mask R-CNN then incorporates a segmentation mask generation branch parallel to the bounding box branch at the end of the network [98]. An FCN generates pixel-level segmentation masks for each RoI which results in greater accuracy and efficiency compared to the fully-connected layers used in earlier R-CNN models [98]. Figure 16 shows the flow of data in Mask R-CNN from input to RoIAlign to the final segmented output.

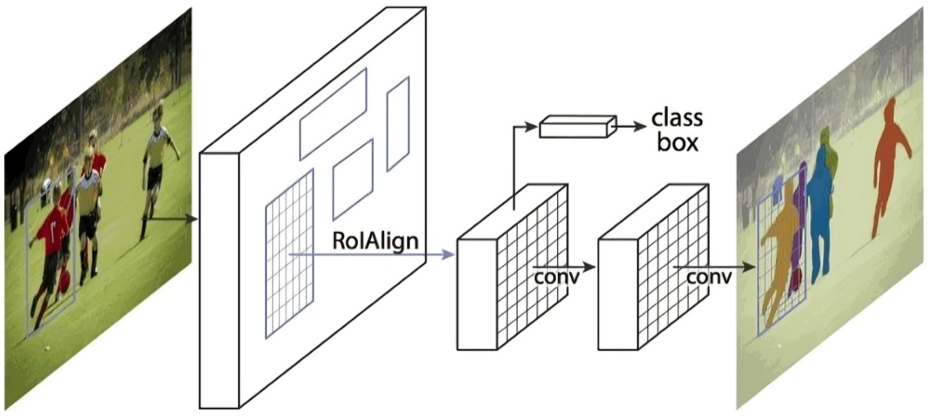


Figure 16: Schematic representing Mask R-CNN instance segmentation

Using a NVIDIA Tesla M40 GPU, the authors of the Mask R-CNN paper evaluated its instance segmentation performance on the COCO 2017 dataset (which includes mask labels as well as detection labels), and compared it to the COCO 2015 and COCO 2016 (shown in Table 1) segmentation challenge winners, MNC (Multi-Task Network Cascade) and FCIS (Fully Convolutional Instance-Aware Semantic Segmentation) respectively [98]. Mask R-CNN outperformed both models with a mask AP of 0.371 (calculated the same way as for object detection except mask IoU is used instead of bounding box IoU - see Section 3.4.1 of Chapter 3), whilst MNC [99], FCIS [100] and FCIS+++ [100] achieved APs of 0.246, 0.292 and 0.336 respectively [98]. This suggests that at the time, Mask R-CNN was superior to other instance segmentation models in terms of precision. Mask R-CNN processed images at a frame rate of 5fps, which advanced the field at the time it was released. However, this may not suffice for real-time applications and as CV has progressed over the years, more modern algorithms are capable of higher speeds. As discussed later in Section 3.4.6 of Chapter 3, the meaning of "real-time" is ambiguous. One common definition of a real-time system is one that processes frames fast enough for practical application [101], whereas another popular definition is that a minimum frame rate of 25fps should be achieved [102]. Mask R-CNN is unlikely to meet the stricter definition of real-time, which highlights a potential limitation in terms of deployment in real-time scenarios [98].

2.5.2 YOLACT

YOLACT ("You Only Look At Coefficients") is a one-stage instance segmentation network that works by generating prototype masks and computing mask coefficients for every instance and then combining them to produce instance masks [103]. Firstly, a deep backbone network is used for feature extraction and features are refined through an FPN to enhance multi-scale detection [103]. Deep backbone features enable more robust mask generation and higher resolution prototypes improve mask quality, especially for smaller objects [103]. Next, prototype masks are generated using a branch named Protonet (a fully convolutional network) whilst a YOLOv3 prediction head predicts class confidences, bounding boxes and mask coefficients [103]. The final masks are produced by merging prototypes with coefficients which results in detailed segmentation of individual objects [103]. A diagram of the YOLACT architecture can be seen in Figure 17 [103].

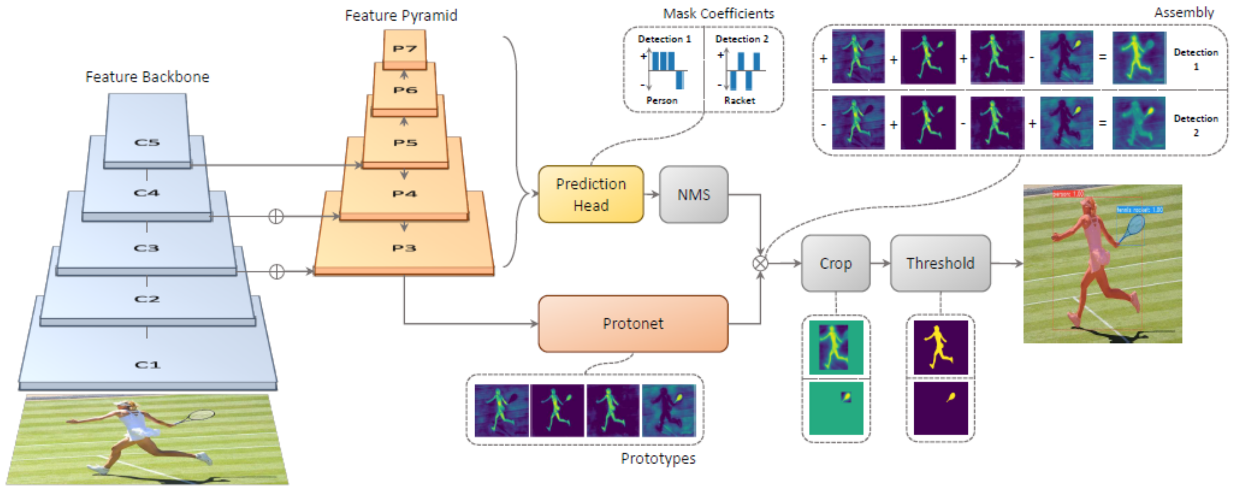


Figure 17: Schematic representing the architecture of YOLACT

YOLACT performance was evaluated on the COCO 2017 dataset against several models including FCIS, Mask R-CNN and others. YOLACT achieved a mask AP of 0.298 which was slightly higher than FCIS and 0.059 below Mask R-CNN [103]. However, YOLACT ran at 33.5fps as opposed to Mask R-CNN that ran at 8.6fps [103]. These results suggest that at the cost of precision, YOLACT is much more adapted to handling real-time applications than similar models. The speed-precision trade-off between YOLACT and Mask R-CNN is comparable to that of R-CNN and YOLO object detecton variants, and must be considered when attempting to implement these models into industrial systems. YOLACT was evaluated using a Titan XP GPU [103].

2.5.3 YOLACT++

YOLACT++ is a more precise version of YOLACT whilst attempting to retain real-time performance [104]. Key improvements include a fast mask re-scoring network which can be seen in Figure 18 and re-ranks mask predictions according to their quality [104]. Also, the backbone uses deformable convolutions which help align feature sampling with instances of different scales, rotations and aspect ratios, resulting in better detections and more precise prototype masks [104]. Furthermore, the prediction head was optimised by changing multi-scale anchors per FPN level which improved the speed-accuracy trade-off [104].

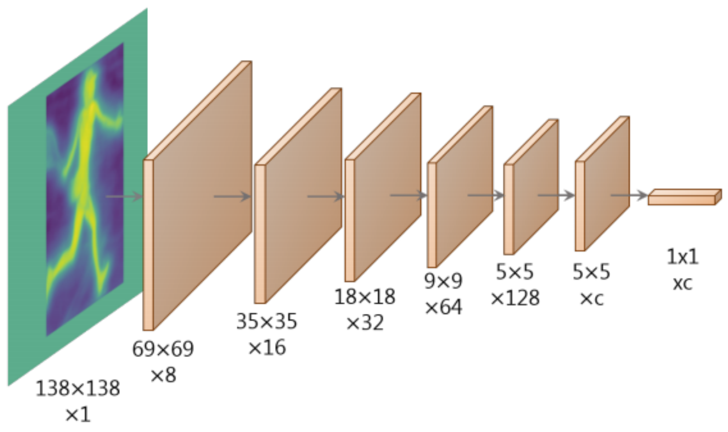


Figure 18: Schematic representing the fast mask re-scoring network in YOLACT++

Upon evaluation of YOLACT++ on the COCO 2017 dataset, it achieved a mask AP of 0.341 at a speed of 33.5fps which matched the already superior speed of YOLACT, whilst significantly improving the mAP closer towards the likes of Mask R-CNN [104]. This improvement demonstrates the effectiveness of the re-scoring and deformable convolutions. At the time the YOLACT++ paper was authored, it was the first instance segmentation method to achieve over 30fps and over 0.300 mAP on the COCO dataset, demonstrating its potential effectiveness for real-world applications requiring real-time processing and high quality instance segmentation [104]. YOLACT++ was evaluated using a Titan XP GPU [104].

2.5.4 SOLO

SOLO ("Segmenting Objects by Locations") is a streamlined, single-shot method for segmentation that simplifies the process of mask prediction by classifying each pixel within an instance based on its location and size [105]. It does this by segmenting the image into a grid of S x S cells which are used to predict classes of objects with center coordinates that fall within a given cell [105]. Individual pixels are then classified based on the relative area of the cell they reside in, which ensures accurate mask generation [105]. Additionally, SOLO uses an FPN which utilises multi-level feature maps to differentiate objects of varying sizes [105]. Figure 19 provides an overview of the the flow of data in SOLO [105].

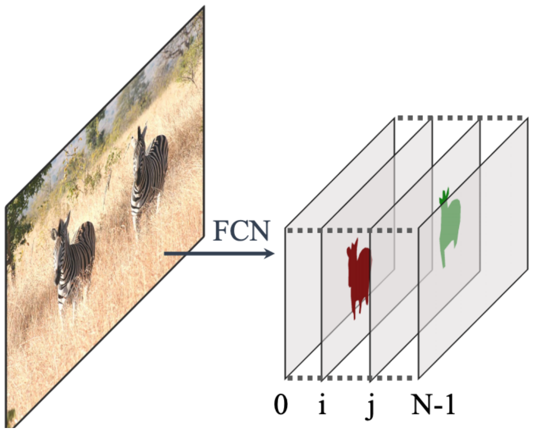


Figure 19: Schematic representing SOLO instance segmentation

SOLO instance segmentation performance was evaluated on the COCO 2017 dataset against MCN, FCIS, YOLACT, Mask R-CNN and others in [105], and achieved a mask AP of 0.371 which matches what was reported for Mask R-CNN in [98]. SOLO ran at 10.4fps and one variant even ran at 22.5fps at the cost of an AP reduction to 0.342 [105]. These results suggest SOLO is competitive with other instance segmentation models in terms of both speed and precision, and like other models such as YOLACT, has a trade-off between the two metrics which should be considered if implemented in real-world applications. The results indicate that SOLO balances speed and precision effectively, however it must be noted that SOLO was evaluated using a Tesla V100 GPU which is significantly more resource-intensive than the Tesla M40 used in Mask R-CNN evaluation and Titan XP used in YOLACT and YOLACT++ evaluation [105]. This difference can significantly affect performance and should be considered when comparing models.

2.5.5 SOLOv2

SOLOv2 is the successor of SOLO and is enhanced with regards to efficiency and accuracy [106]. SOLOv2 introduces a dynamic instance segmentation strategy which instead of using a fixed grid, it dynamically predicts pixel categories on a pixel-by-pixel basis [106]. SOLOv2 decouples mask generation into dynamic mask kernel prediction with a kernel branch and high-resolution mask feature learning with a feature branch to ensure different instances of the same class are distinguished [106]. Additionally, the FPN is more sophisticated in a variety of ways, including improved integration of multi-level features which aids computational efficiency and enhances feature extraction of objects of varying sizes [106]. Figure 20 provides an overview of the the flow of data in SOLOv2 [106].

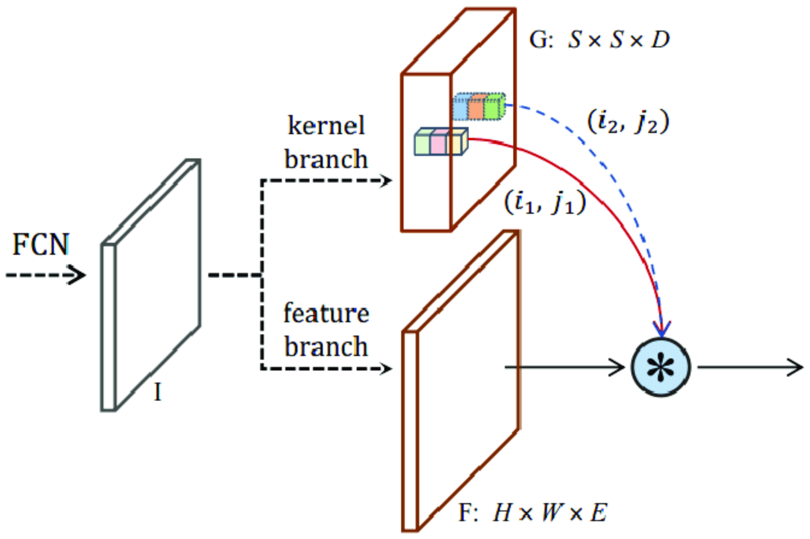


Figure 20: Schematic representing SOLOv2 instance segmentation

SOLOv2 performance was evaluated on the COCO 2017 dataset against SOLO, Mask R-CNN, YOLACT and others, and achieved the highest mask AP of 0.417, whilst a lightweight variant achieved a mask AP of 0.371 with a speed of 31.3fps which is convincingly meets real-time performance standards [106]. These results surpass similar models in terms of speed and precision which suggests it is a strong choice for real-world instance segmentation applications. However, like SOLO, SOLOv2 was also evaluated using a Tesla V100 GPU which potentially gives it a significant advantage with regards to performance metrics, at the cost of significant computational expense [106]. In industry, deployment is often resource-constrained, particularly on production lines where real-time Internet of Things (IoT) applications are common. Therefore, whilst SOLOv2 shows promising results, it would need to be evaluated on less powerful GPUs for realistic assessment of its practical applicability.

2.5.6 YOLOv5-seg and YOLOv8-seg

A Protonet module, similar to the one mentioned as part of YOLACT and YOLACT++, was incorporated into the YOLOv5 (two years after its original release) and YOLOv8 (as part of its release) architectures as an instance segmentation branch, in parallel to the original detection head [107, 108]. This was a complimentary addition to both models and made them more versatile in terms of application. Figure 21 shows the architecture of Protonet [103].

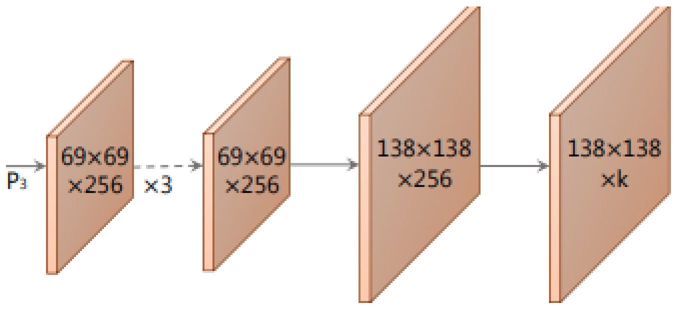


Figure 21: Schematic representing the architecture of Protonet

When evaluated on the COCO 2017 test set, YOLOv5-seg variants ranged from 0.276 to 0.507 mAP and approximately 222fps to 833fps, where increasing model size resulted in slower but more precise predictions [107]. Also, YOLOv8-seg variants ranged from 0.367 to 0.534 mAP whilst running at 240fps to 846fps [108]. In comparison to previously discussed models, these results show that medium to extra large YOLO segmentation models generally surpass previous instance segmentation networks in terms of mAP, whilst greatly surpassing them in terms of inference speed. This makes them much more suitable for real-time applications in industry. The “seg” variants are relatively new (even YOLOv5-seg) and therefore documented applications are limited, however they are likely to become key players in many future industrial scenarios.

Both models were evaluated on a NVIDIA A100 GPU, which must be taken into account since it is significantly more computationally intensive than previously mentioned GPUs and therefore is likely to have provided a significant advantage during evaluation [107, 108].

A more realistic assessment would need to be undertaken before deploying these models, however even with the GPU advantage removed, these models are likely to retain a significant portion of their speed.

2.5.7 FastSAM

FastSAM (Fast Segment Anything Model) is a cutting-edge segmentation model that is different to all the models discussed so far, since it performs segmentation based on user input prompts [109]. Whilst this may mean it will have different use-cases than some of the previous models, it is difficult to predict how the forefront of CV will evolve in terms of industry applications. FastSAM is a faster, more lightweight version of SAM (Segment Anything Model) released by Meta mid-2023 that is currently at the forefront of CV [110]. SAM is a promptable visual transformer network trained on a dataset containing over 1 billion masks (SA-1B) within over 11 million images, making it highly adaptable [110]. It is described as a zero-shot model, meaning it can classify objects that are not in its training set, which is impressive [110]. FastSAM utilises previously discussed components such as a CNN backbone and an FPN for multi-scale feature extraction, integrated with a Protonet module for mask generation [109]. These are used in combination with user prompts which can be point-based, box-based or text-based and guide the segmentation process [109]. Figure 22 shows the architecture of FastSAM [109].

FastSAM achieved 0.465 mask AP on the COCO 2017 dataset without being trained on it [110]. However, SAM is reported to run very slowly, far more slowly than 1fps, meaning it is currently unsuitable for many industrial applications [110]. FastSAM is a first attempt to combat this by using a YOLOv8-seg backbone and only 2% of the original SA-1B dataset, which allows it to make inferences 50x faster than the original SAM [109]. At the cost of a reduction to 0.379 mask AP on the COCO dataset, FastSAM can run at 25fps using a NVIDIA GeForce RTX 3090 (and over 83fps when optimised using TensorRT - see Section 5.4.2 of Chapter 5 for more details on TensorRT) [109]. This is much faster and more precise than Mask R-CNN, and places it near YOLACT++ in terms of speed, which is very promising for real-time future developments. However, it still requires high-end hardware and the prompt requirement is generally undesirable for many production line applications which typically work fully automatically in resource-constrained environments.

Current IoT hardware is typically not as powerful as an RTX 3090, however this is becoming less true each year. Also, prompt-based applications in manufacturing could be beneficial provided they are used where appropriate. For example, a fully automated system

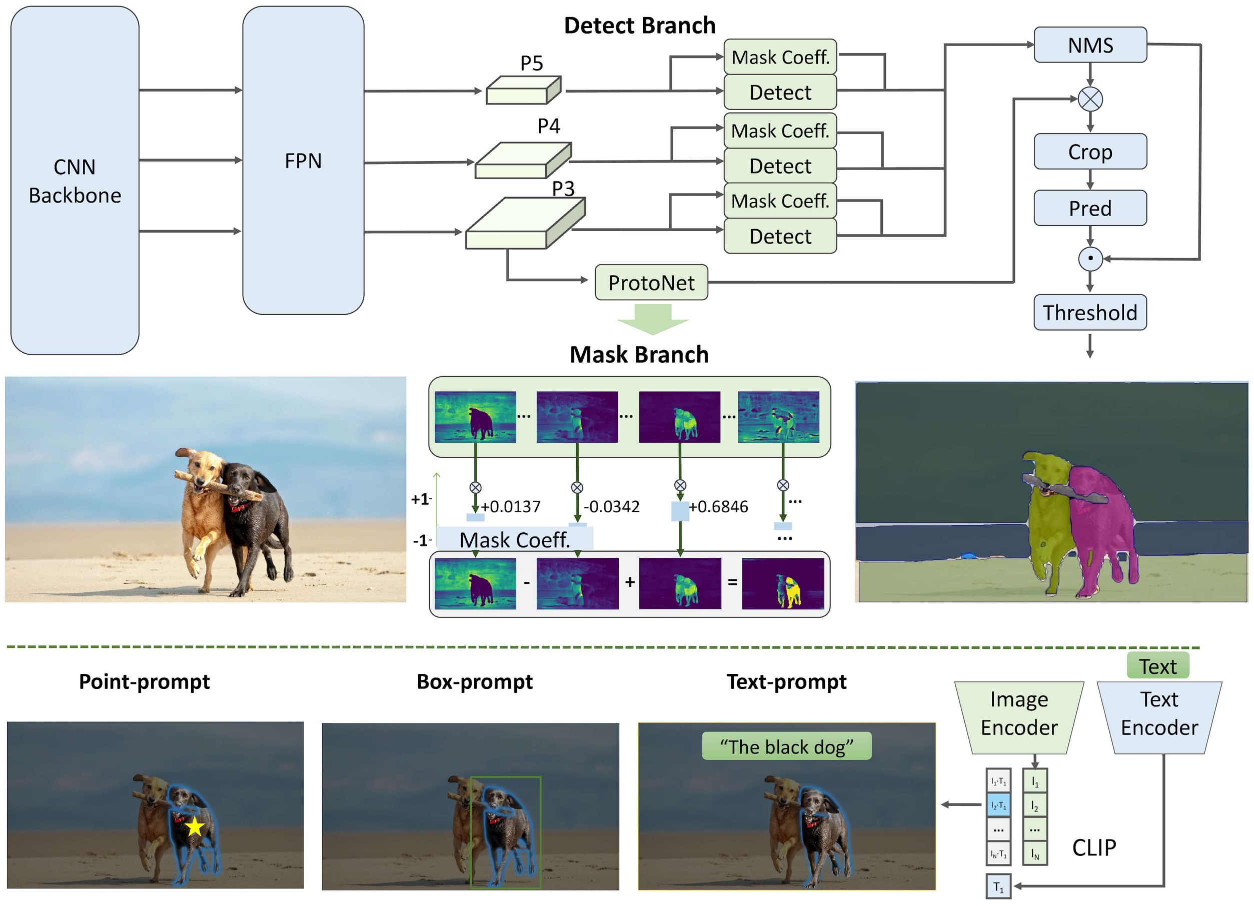


Figure 22: Schematic representing the architecture of FastSAM

would be better for steel surface defect detection, whereas a prompt-based system might be better for adjusting inspection specifications on-the-fly for different customers. More investigation into prompt-based computer vision applications in the steel industry would certainly address a gap in the research.

2.6 Background Subtraction Methods

As mentioned, instance segmentation networks can be excellent at extracting and interpreting complex features, however they are resource-intensive in terms of time, data and hardware. Alternatively, traditional background subtraction (BGS) algorithms, which are often (but not always) based on Gaussian mixture models (GMMs), are less time-consuming, less computationally intensive and easier to implement [111]. Figure 23 shows a simplistic overview of how background subtraction algorithms typically operate [112]. The basis of background subtraction methods is to segment moving foreground objects from static background pixels across a sequence of images using various thresholding techniques, as opposed to instance segmentation networks that classify each pixel and create masks that distinguish each object instance. This section will discuss examples in literature where BGS techniques have been used for CV applications, including critical analysis of their suitability for application in real-world settings and identification of research gaps.

GMMs in particular use a set of Gaussian distributions to represent parts of the background using Gaussian probability densities, which collectively, are able to represent the entire background [113]. Both BGS and segmentation have their limitations. For example, BGS approaches are less accurate when dealing with complex scenes, especially when they involve dynamic backgrounds [114]. Meanwhile, instance segmentation typically requires labelled datasets and time-consuming model training in comparison to BGS algorithms that are capable of operating effectively almost instantly.

Whilst BGS techniques have shown potential in various areas, their value is yet to be capitalised on in the steel industry. In [115], the "Mixture of Gaussians" (MOG) algorithm is used in combination with a timed motion history image method (motion segmentation), as well as Kalman filtering, to achieve real-time vehicle traffic tracking. Four-phase BGS which combined MOG, Shi-Tomasi feature extraction [116], optical flow vehicle tracking [117] and centroid estimation was applied for traffic surveillance and reported in [118], where the model estimated the speed of three cars with below 0.5km per hour error [118]. These studies demonstrate the capabilities of BGS algorithms to not only operate in real-time, but also integrate with other techniques such as Kalman filtering and optical flow estimation,

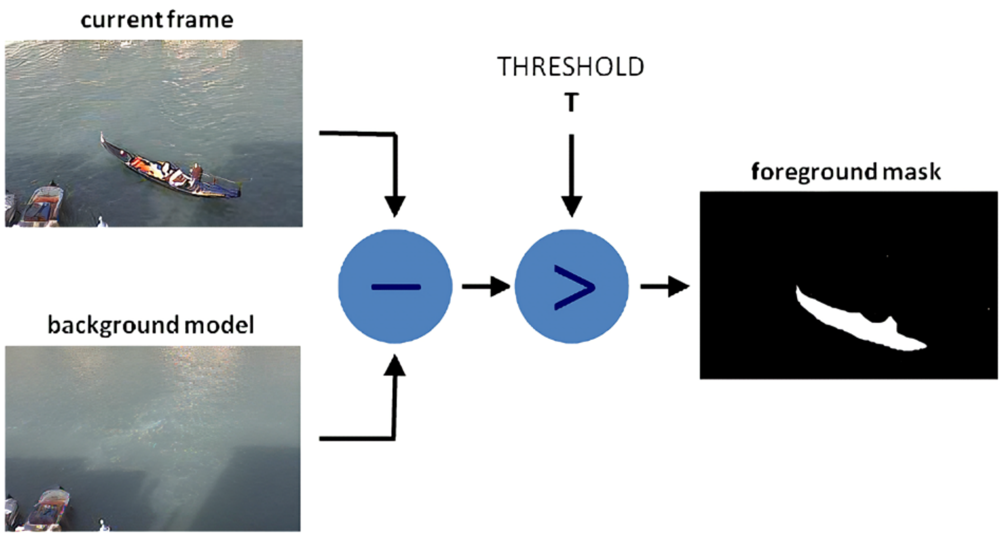


Figure 23: Schematic representing the background subtraction process

which is promising for real-time measurement systems in production lines such as monitoring the flow of materials.

BGS algorithms available in OpenCV [119] were evaluated on ship detection on inland waters, which found that the Google Summer of Code (GSOC) and Counting (CNT) algorithms performed the best in terms of quality [120]. Meanwhile, BGS algorithms were evaluated on animal detection using near infrared spectrum images of moving wild mammals, which included evaluation of Gaussian Mixture-based Background Foreground Segmentation (GMG), KNN (K-Nearest Neighbours), MOG and MOG2 [121]. The KNN algorithm produced a mask most similar to the handcrafted labels used for validation, which was followed by MOG2 [121]. However, MOG2 was faster, making it more suitable for real-time application. These studies show how the effectiveness of different BGS algorithms can vary depending on the specific environment they are being used in, as well as the objects they are used to track. If used in any real-world application, it would be important to experiment with a variety of BGS algorithms to ensure the best one is chosen.

Additional examples of applications include [122] where a GMM was combined with KNN (in a different way to [121]) to recognise suspicious activity in a gated region with 97% accuracy, [123] where MOG BGS was coupled with principal component analysis (PCA) for fall detection of vulnerable people with a detection rate of 86.21%, and [124] where MOG BGS was paired with Kalman appearance tracking to track patients with Parkinson's disease for fall detection and to inform caregivers of severity and evolution of the disease with average precision and average recall values of 0.871 and 0.875 respectively. These studies show the high performance of BGS in terms of both precision and speed, as well as reinforcing their compatibility with other techniques and practical applicability to a diverse range of scenarios. However, with all of the applications discussed here, there is a clear gap in the research for steel production-based applications using BGS which is important to address due to the low resource-intensiveness yet high effectiveness of the approach. Furthermore, deep learning has been combined with BGS in the past such as in [125, 126, 127], however existing literature is by no means extensive. Using these two sets of techniques in combination is an exciting area of future research that has potential to improve the current state of manufacturing technology.

2.7 Computer Vision Applications for Manufacturing

CV opens the door to a multitude of potential benefits including efficiency, quality, and safety. So far in this literature review, different types of CV techniques have been presented and discussed in terms of their theoretical performance and potential suitability for application. Traditional CV approaches have been described and their application in real-world scenarios has been demonstrated, whilst modern approaches that utilise deep learning have only been described and evaluated against each other on benchmark datasets such as the COCO dataset. This was explained in the introduction of this chapter and is deliberate in order to draw a clear line between the theoretical performance of DL models, and the practical impact of them.

This section will focus on a wide range of CV applications, stretching from gas monitoring, to monitoring of blast furnaces and ladles, to surface defection inspection, and even applications specific to health and safety. Each subsection will briefly provide some background knowledge on the topic before critically analysing recent advances driven by CV and identifying research gaps where possible. Whilst focus is primarily on steel manufacturing, in some cases other kinds of applications are discussed (where appropriate and still relevant to steelmaking in some way), such as Section 2.7.8 which covers other subfields of manufacturing.

2.7.1 Gas Monitoring

Historically, the monitoring of gases relied on a variety of analytical methods that describe chemical properties and compositions. These include gas chromatography (GC) [128, 129], mass spectrometry (MS) [130, 131], infrared spectroscopy [132, 133], ultraviolet (UV) [134, 135] and differential absorption LiDAR (DIAL) [136, 137]. Before the advent of deep learning, traditional CV approaches such as Hough transforms, Canny edge detection and stereo vision were also used for gas analysis [138, 139, 140].

With the rapid advancement in hardware and data processing capabilities, the field of deep learning has experienced significant growth, leading to revolutionary changes in CV techniques. This has notably enhanced gas monitoring systems which is evident in literature such as [141] which presented BubCNN, a bubble detection network built upon Faster R-CNN and a shape regression network. The model achieved an mAP ranging from 0.740 to 1.000 depending on the amount of space between bubbles [141]. The variability of model performance depending on the space between bubbles highlights the challenging nature of dealing with dynamic scenes in real-world applications where conditions such as lighting and camera position can significantly affect results. Developing more standardised approaches for managing these factors in steel industry has not been addressed and would be challenging but highly beneficial for making model deployment for industrial applications more efficient.

In another study, U-Net and Mask R-CNN were utilised to detect and segment bubbles during the boiling of water at different sub-atmospheric pressures to measure dynamic boiling characteristics [142]. Mask R-CNN outperformed U-Net at counting bubbles despite inferior segmentation accuracy, and under varying conditions deviated by less than 10% from the real number of bubbles. However, this was achieved in a laboratory setting where the environmental conditions were carefully controlled. For applicability of this approach to large-scale industrial environments, such as across multiple steel production sites where conditions are more variable, model performance may suffer significantly.

A smart gas stove system element that was capable of gas leak detection and prevention was reported in [143]. The model combined Haar Cascading with a CNN for the task of gas detection as well as age detection to restrict prevent usage by young children, however no evaluation results were reported which raises concerns about the reliability and effectiveness of the element [143].

Finally, flames and smoke detection using a CNN was achieved with 93% accuracy in [144], where SSD [52] and Faster R-CNN [48] were also evaluated and achieved 85% and 89% accuracy respectively [144]. This demonstrates the potential of CV to enhance safety systems. However, the variability in performance shown in both this study and previous studies suggests rigorous testing would be required before deploying CV models in safety-critical scenarios.

2.7.2 Blast Furnace Process Monitoring

The blast furnace plays a crucial role in the steel production process, where raw materials such as iron ore, coke, and limestone react under extremely high temperatures, resulting in the production of pig iron, a raw material for steel products [145]. In this process, these raw materials are added in precise quantities before being added to the furnace. Hot air is then heated to a temperature range of 900 to 1300℃ and blown into the furnace, leading to a reaction with the coke, which ultimately forms molten iron through various chemical reactions [145]. At the same time, limestone reacts with ore impurities to create molten slag which not only helps remove impurities from the iron, but also forms a layer that protects against oxidation [145]. The heavier molten iron settles at the bottom of the furnace, from where it is periodically extracted through a “tap hole” into a crucible. Additionally, the floating slag is also intermittently skimmed off the top for disposal [146]. These steps are crucial to maintain the high purity of the molten iron necessary for high quality steel production.

A temperature measurement and compensation method using infrared imaging with deep learning was presented in [147], where a series of steps were described to determine the temperature of molten iron whilst compensating for measurement error caused by dust [147]. Canny edge detection was used in combination with morphological operations such as coarsening, skeletonisation and deburring to identify molten iron boundaries in captured thermal images [147]. To compensate for measurement errors, dust features were extracted and used as inputs to an ensemble network comprising of a vanilla ANN and support vector regression [147]. This model enhanced measurement accuracy by predicting the measurement error caused by dust [147]. The final model achieved an RMSE of 0.087°C for temperature measurement [147] which demonstrates the potential of CV both for refining measurements and dealing with alternative forms of imaging data such as infrared.

In [148], a blast furnace chute wear characterisation method that was developed using a CNN-RELM (Regularised Extreme Learning Machine) [149] to detect holes, metallic screws, metal sheet, rusted metal and weld marks on six different chute images was proposed, which achieved 80% accuracy [148]. This approach is innovative and the accuracy is fairly high, however for actual industrial application, this approach would need to be tested on a much wider variety of data samples which would likely expose the need for further development. The underlying concept of this approach could be applied to a range of equipment typically seen in steelworks.

Fast R-CNN and a variational mode decomposition algorithm were coupled to detect blast furnace bearing faults with a precision of 0.970 and a recall of 0.910 in [150]. These performance values are impressive and indicate the effectiveness of CV in fault detection. However, Fast R-CNN has been surpassed by other detection networks in terms of both speed and precision, so this model could be improved with little effort. Upgrading to a superior network such as Faster R-CNN or YOLOv8 would improve the likelihood of the approach remaining robust to more variable environmental conditions during deployment and could also provide real-time capabilities.

Also, YOLOv5 was applied to infrared images for blast furnace charging state recognition with an accuracy of over 95.5% in [151], which emphasises the potential of CV models to be used in a range of blast furnace-related applications.

Finally, a dynamic attention-wise deep network that used attention to self-learn relationships between process parameters and prediction targets was proposed in [152]. The model was reportedly used for molten iron quality indices and achieved RMSE values of 0.0644, 0.0026 and 0.0071 on silicon, sulphur and phosphorus percentages respectively. These results indicate the precision of CV techniques when predicting quality control factors in steel manufacturing.

2.7.3 Slag Process Monitoring

Slag, the primary by-product in steel manufacturing, emerges from a variety of processes and typically contains a mix of silicon, aluminum, calcium, and magnesium oxides [153, 154]. Traditionally, slag removal has been a manual task. The manual process of separating slag from molten metal, known as de-slagging, poses significant risks of severe injury or even fatality to workers. In addition, inadequate removal of slag can lead to a notable decrease in the quality of the steel produced, resulting in the wastage of materials and energy [154, 155].

A CNN model was used to estimate a path to automate the de-slagging process from ladles according to [153], and the proposed model achieved 91.96% accuracy. A similar study was published using a slag distribution image (SDI) and a slag removal path estimation network (SPRENet) [146]. SRPENet uses the SDI to train the network, then operates by estimating removal path control points before estimating a goodness score of the points estimated [146]. The model was compared to the performance of a human operator and was reported as being 19.83% more effective, whilst having real-time capabilities [146]. The results of these studies indicate that CV approaches are promising for automating hazardous tasks which is beneficial for both safety and efficiency. However, the robustness and scalability of these approaches require further investigation before implementation.

A convolutional recurrent neural network (CRNN) was utilised to determine whether or not a slag dart had plugged the exit hole of a basic-oxygen furnace (BOF) or not during tapping, according to [4]. The model presented was constituted by a CNN and an LSTM and used real-time closed-circuit television (CCTV) as input [4] to achieve a reported accuracy of 99.45%, which was 10% better than operator judgement, whilst reducing workload by 30% [4]. These results show a significant improvement over the traditional approach and therefore emphasises the revolutionary potential of deep learning in the steel industry.

Slag dart input success during tapping was estimated to prevent slag carry-over by combining pre-processing of slag-detection system (SDS) image data, followed by use of an LSTM to deal with sequential frames in [154]. A classification accuracy of 99.61% was achieved when the model was evaluated on the test set [154]. This reinforces the idea that CV models can perform tasks that are traditionally performed manually, whilst also highlighting their benefits in improving product quality. However, a common theme with these applications (both slag-related and others), is that they require extensive testing, development and integration with current systems before being production-ready.

2.7.4 Surface Defect Detection

Ensuring quality of steel products is crucial for steelmaking. Surface defects indicate imperfections that not only compromise structural integrity and overall performance, but also ruin the aesthetic quality. Defects lead to wasted resources such as energy and materials. Also, if defective steel reaches the customer it could harm manufacturer reputation and even customer health. Historically, manual methods of steel defect detection were primarily used, which resulted in a high false detection rate [156]. For the most experienced workers the detection rate of defects has been reported as 80%, leaving one in five defects overlooked [156]. The datasets introduced in this section are summarised in Table 2.

U-Net [164] was pre-trained on the ImageNet dataset before being utilised as the foundation of a network called TLU-Net (Transfer Learning-based U-Net) [5]. An accuracy of 97% was achieved on the Severstal steel surface defect dataset, which was promising for industrial application and testing the model in a real-world environment with more defect types would ensure deployment is practical [157]. Real-world environments introduce challenges such as noise, lighting changes, distortion from heat and movement, as well as many others. Ensuring that models can perform well whilst these challenges are present is a key difference between proper deployment and simply developing and evaluating a model.

Various types of surface defect detection were performed using a variant of the Swin

Table 2: Surface defect detection datasets discussed in this review

Dataset	Images	Classes	Description	Source
Severstal	18106	4	Steel surface defect dataset for object detection, including pitted surface, crazing, scratches and patches.	[157]
DAGM	16100	10	"Deutsche Arbeitsgemeinschaft für Mustererkennung" - Synthetic dataset for defect detection on textured surfaces.	[158]
KolektorSDD	399	2	"Kolektor Surface Defect Dataset" - Real-world detection dataset of defective and non-defective production items collected by Kolektor Group.	[159, 160]
NEU-DET	1800	6	"Northeastern University Detection Dataset" - Real-world detection dataset of six common surface defects on hot-rolled steel strips, including rolled-in scale, patches, crazing, pitted surfaces, inclusions and scratches.	[161]
Magnetic-Tile	2688	6	Dataset of surface defects on magnetic tiles including blowholes, cracks, breaks, fraying, unevenness and free.	[162, 163]

Transformer model (Cas-VSwin Transformer) according to [165], where detection was performed on four surface defect datasets. The model achieved APs of 0.999, 0.997, 0.994 and 0.805 on the DAGM dataset which consists of synthetic textured surface defects [158], the KolektorSDD dataset which contains production item defects [159], the Severstal Steel dataset [157], and the NEU-DET datset which contains steel surface defects [161], respectively [165]. These results are impressive however there is quite a significant decrease in performance on the NEU-DET dataset compared to others, which is most similar to what is expected in industry since it is steel-based and includes more defect types than the Severstal dataset. Whilst promising, further investigation with more steel-based datasets would be required before attempting to deploy this model as part of a steel defect detection system.

In [166], YOLOv5 was modified by embedding the backbone with a Convolutional Block Attention Module (CBAM) [167] to enhance feature extraction. Four defects from NEU-DET were evaluated and the model achieved 0.854 mAP, which was marginally better than a CBAM version of YOLOv4 and standard YOLOv4 [166]. Whilst the mAP highlights the effectiveness of the model and potential applicability for industry, the evaluation is questionable since YOLOv4-CBAM and YOLOv5-CBAM showed barely any difference in performance and standard YOLOv5 was not included in the evaluation. However, there was a significant enough increase in performance from YOLOv4 to YOLOv4-CBAM to imply that the attention module was beneficial for defect detection. Also worth noting is that only four defects from NEU-DET were used which means further investigation is required to assess the ability of the model to deal with more diverse data.

All six defects of the NEU-DET dataset were used to evaluate a CNN classification model with 93% accuracy in [168]. Whilst this is a good result, more extensive testing would be required in real-world environments to confirm the practical applicability of this model. Furthermore, the lack of localisation could be limiting in scenarios where multiple defects are close together.

Also, in [169], a model named NSLNet (Neural Structured Learning Network) was reported as achieving over 99% classification performance with limited training data. NSLNet consisted of Neural Structured Learning, Histogram Equalisation and a vanilla classifier [169]. NSLNet was developed and evaluated using the NEU-DET dataset as well as an extended version of NEU-DET that includes noise, blurring, lighting changes and other perturbations that are more similar to what would be experienced in a real production line. Therefore, this model could be a great foundation for developing a real surface defect detection system. However, it purely classifies defects with no localisation, which like [168], could be limiting if multiple defects occur close together.

CenterNet [170] was combined with a dilated feature enhancement model (DFEM) and a prediction head to result in a model called DCC-CenterNet (meaning of "DCC" is not stated), which was evaluated on the NEU-DET dataset, where it was report to achieve an mAP of 0.794 and a speed of 71.37fps [171]. The mAP is high enough to consider this model for further development and eventual deployment, however the result that is more unique to this study is the speed. Some defect detecton studies do not report speed and those that do, do not report speeds this high. For real-time surface defect detection at high strip speeds, this model should be considered.

Similarly, MSFT-YOLO (meaning of "MSFT" is not stated), is a model comprised of YOLOv5, a transformer and a bi-directional FPN, and was tested on the NEU-DET dataset where it achieved 0.757 mAP and 29.10fps [172]. The precision of this model comes close to that of DCC-CenterNet whilst speed is less than half. However, both results are still promising for industrial application in real-time systems and should therefore be investigated more deeply in future work to see how they compare on more diverse data and computational expense.

A segmentation network named TAS2-Net (Triple-Attention Semantic Segmentation Network), comprised of GAN-based augmentation (where "GAN" is the abbreviation of "generative adversarial networks") and various attention modules, was evaluated on the NEU-DET, DAGM and Magnetic-Tile [162] datasets against SDM (Saliency Detection Model) [173], RefineNet [174], U-Net and other models and performed the best in every metric on every dataset, with precisions of 0.981, 0.972 and 0.963 on the DAGM, NEU-DET and Magnetic-Tile datasets respectively [175]. On the NEU-DET dataset, TAS2-Net achieved 41fps also. These results suggest the model is effective on a range of datasets in terms of both precision and speed which indicates its potential for industrial application. However, there appears to be no evaluation in terms of mAP which is typically the most common and informative metric used for detection tasks and should therefore be included so that it can be more easily compared to the likes of DCC-CenterNet and MSFT-YOLO. Additionally, this study implies that use of GANs and attention networks could be highly beneficial to defect detection tasks, but this may be at the cost of significantly increased computational expense.

2.7.5 Microstructural Analysis

Microstructural analysis is important for material property assessment both physically and chemically. Historically, analysis has been performed manually which is labor-intensive and allows for significant potential of human error due to the intricate nature of microstructures [176]. However, technology is progressing due to the successful application of classification and segmentation methods. Since segmentation is at pixel-level, it is particularly suitable for microstructural analysis tasks due to the intricacies of the imaging involved.

Microstructural segmentation and analysis of ultra-high carbon steel was performed using a PixelNet variant [6] as reported in [177]. The model was able to distinguish proeutectoid cementite network, fields of spheroidite particles, ferritic matrix within the particle-free denuded zone near the network, and Widmanstätten laths, with an impressive 92.6% accuracy [177]. This segmentation built the foundation for describing cementite particle size and denuded zone width distributions [177]. This application is innovative and indicates great potential for the approach used. However, the dataset only contained 24 samples. Whilst this highlights how well the model performed with minimal training samples (six-fold cross-validation was used so 20 training samples per experiment), it also highlights the lack of evaluation on a wide range of diverse data samples to ensure robustness across different alloys and environmental conditions [177].

A deep CNN was also used for classification of eight different types of steel microstructure captured using light optical microscopy (LOM), on which it achieved 96.5% accuracy [178]. The accuracy is impressive and demonstrates the effectiveness of CNNs when dealing with microstructural images. However, the test set of 20283 samples was pre-processed to remove certain samples that were highly dissimilar to more typical examples, which likely made the test set less challenging and the accuracy higher [178]. Practical application of this model would require development using a dataset that is more representative of what is found in real-world scenarios.

2.7.6 Health and Safety

Health and safety is a key component of any industry, but particularly in steel production. Given the nature of the equipment involved such has heavy machinery carrying molten metal, other hot materials and hazardous chemicals such as those involved in coating, it is vital that practices remain closely aligned with health and safety regulations.

In regards to this, several existing CV studies have been conducted. Faster R-CNN was trained on 4500 images labelled with and without helmets in order to develop a safety helmet recognition system for steel factories, where a reported mAP of 0.712 was achieved [179]. Whilst the mAP is reasonably high, full implementation would require more development. However, this study lays the foundation for a variety of safety compliance checking systems which could be highly beneficial to hazardous industries such as steel. Safety helmet recognition is a relatively simple example, but with more research and development, future advances could involve full safety gear checks involving gloves or high-visibility clothing. Additionally, CV could be used to ensure that operational procedures are being followed prior to equipment access. Overall, these types of applications could significantly lower the risk of accidents.

Another study presented a model that used Mask R-CNN to validate the correct alignment of crane hooks and ladle trunnions to prevent accidents, which achieved a segmentation accuracy of 92% and a safety judgement accuracy (whether the alignment is correct or not) of 96% [180]. Varying from [179], this study highlights the suitability of CV approaches to increasing safety standards for tasks directly involving large, heavy, moving machinery. Whilst the evaluation in [180] is not particularly extensive, the application is innovative and lays a foundation for future applications involving other moving equipment such as furnace doors or industrial presses. These could improve workplace safety, but also improve operational efficiency and equipment lifespan.

2.7.7 Identification Number Recognition

Automatic recognition of identification (ID) numbers printed on steel products facilitates efficient manufacturing. Traditionally, this task is conducted using the vision and judgement of workers, often whilst they are affected by performance influencing factors such as high-temperatures, as well as dusty and dark environments, with ID characters being a fairly large distance away. This is detrimental for operational efficiency and the reliability of workers correctly reading the ID [181]. It also adds unnecessary responsibility on the workers and could impact their productivity in other areas [181]. CV approaches to this task typically involve pre-processing of samples to distinguish ID characters from the background, followed by character detection. Automating this process with CV could lead to significant improvements in operational efficiency, as well as improved reliability due to elimination of human error.

An early attempt at this task used steel billets for experimentation and used a combination of image pre-processing, image segmentation (not using deep learning) and a back-propagation neural network (this part is deep learning) for character recognition [181]. For the characters from zero to nine, the model accuracy ranged from 95.29% to 97.53% with inference times ranging from 55ms to 62ms [181]. The accuracy range reported is promising for applications based on character recognition which could be developed for a variety of logistical scenarios involving any kind of resource or part that requires an identification number. Furthermore, the inference times imply that this approach has potential for real-time application which is desirable if many objects are being identified in quick succession such as on a production line.

Another study applied a deep CNN (DCNN) with transfer learning to localise and classify steel slab identification numbers (SINs) and for localisation, recall and precision values of 0.927 and 0.924 were achieved respectively, whilst 99.6% classification accuracy was achieved [182]. This study used images collected from a real industrial site which meant challenges such as changes in lighting conditions, heat distortion, camera distance and slab length were included in the dataset. Furthermore, both letters and numbers were included in the samples. Therefore, this approach is very promising for real industrial application. Applying this model to other ID recognition scenarios would be beneficial to the field in terms of research and industrial advancement.

Automatic slab identification number recognition was also performed using ground-truth data "weakly" annotated using a one-click system, to train variations of an FCN [183]. The proposed method achieved a recall of 0.996 and a precision of 0.995, which indicates the effectiveness of the models used even with low-detail annotations [183]. This approach is highly efficient and results in high performance despite minimal annotation effort which effectively addresses a common bottleneck in deep learning projects. Additionally, the dataset was collected from two different places of a real steelworks which implies there were environmental challenges overcome such as lighting and noise. Whilst the annotation approach is innovative and efficient, since the annotator only clicks the centroid of each character to label it, there are certainly limitations to this approach. For example, this assumes all characters can be annotated to a high standard by using the same size bounding boxes, which is only realistic if the distance between the camera and slab remains constant. Also, in terms of scalability, this may become problematic since slightly different slab setups would require different pre-set bounding box sizes.

Finally, a more advanced task of recognising arched hot-rolled steel coil identification numbers was achieved by combining image processing, followed by segmentation (without deep learning), alignment to account for the arched labels, followed by character classification using a CNN and a final post-processing stage [184]. The model achieved an accuracy of 98.49% with a speed of 0.37s per image [184]. This study highlights the extended capability of CV to deal with arched text, whilst also demonstrating high accuracy and inference speeds significantly faster than human speed. This uncovers more potential applications for ID recognition than other studies, since it proves that text does not have to be linear for successful recognition. Other variants could include recognition tasks where text is vertical or circular. This added flexibility of ID recognition could be crucial for developing technology that can recognise ID characters on every labelled part in an entire steelworks site.

2.7.8 Other Manufacturing Applications

CV techniques are widely applied across various manufacturing sectors, not just in steel production. The rapid development of AI presents opportunities for applications in the broader field of manufacturing such as automotive and additive manufacturing. This section will explore the integration of CV into such areas.

In the automotive industry, CV has been applied to the inspection of surface quality. A notable example is a study that developed a system for assessing the quality of painted car bodies [185]. This system comprised a two-step process, defect detection using TinyDefectRNet (based on the previously released TinyDefectNet [186]), a model based on YOLOv3, followed by an evaluation of appearance quality [185]. The system recall and precision varied between approximately 0.919 to 0.953 and 0.882 to 0.907, respectively, depending on the car body side under examination, with average analysis times ranging from 20.30s for the hood to around 64.70s and 64.20s for the left and right sides respectively [185]. This application is innovative and if deployed, could be beneficial to the automotive industry. However, the appearance of different vehicles can vary significantly and therefore, for this model to perform consistently across different cars it would need to be thoroughly trained on a wide variety of samples from every body side of many different car models. The diversity of the dataset created for the study is unclear.

Additionally, YOLOv3 was used to localise and classify three types of solder joints on automotive door panels (rectangle, semi-circle, and circle) [187], achieving an mAP of 0.852 and a detection time of 0.18s per panel image, which meets the real-time requirements of the production line. The mAP is good but shows room for improvement which may be possible through a simple change from YOLOv3 to a more recent model such as YOLOv8. This would also improve the detection speed.

In electronics manufacturing, CV technology has shown significant benefits. For instance, [188] describes automated surface inspection of copper clad laminate images for defect detection using an efficient CNN architecture. This architecture included convolutional layers, squeeze-and-excitation blocks, and squeeze-and-expand blocks [188]. The model achieved precision and recall values of 0.991, outperforming other models such as MobileNet-v2 [189] and ResNet-50 [53]. The proposed model also ran at 20ms per image, which was faster than other models, except MobileNet-v2, which it was more accurate than [188]. These metrics are impressive and suggests the model is suitable for integration into production lines, which was achieved, as reported by the authors [188]. Also, according to the authors, there is room to improve the computational efficiency of the model, which is a potential area for future research [188].

Another study used an improved version of YOLOv3 for detecting electronic components on printed circuit boards (PCBs) using both real and synthetic data [190]. The model recognised 29 different component types such as resistors and capacitors and achieved an mAP of 0.931 [190]. This was highly successful and promising for real-world application, especially considering the high number of different categories. However, it was noticed that the AP of the resistor, capacitor and component instances were 0.480, 0.720, and 0.560 respectively, which was because resistors and capacitors were visually similar, and instances within the component category varied significantly [190]. It is likely that if a more recent version of YOLO such as YOLOv8 was used for this task, low-scoring category AP values could be improved. Note that the “component” category included components that did not fit neatly into any other category [190].

CV applications in additive manufacturing include process monitoring and defect detection. One research paper presented a hybrid CNN model for quality-level classification in laser powder bed fusion (LPBF) processes [191]. When tested under overheating, normal, irregularity and balling conditions, it achieved detection accuracy values of 0.995, 0.996, 0.998 and 0.996 respectively [191]. These high-performance results across a variety of defects demonstrate the potential of CV technology to ensure high product quality in additive manufacturing [191]. Further investigation using different additive manufacturing processes, different machines and different materials would be valuable for advancing the field.

Another study in LPBF utilised a parallel model combining a CNN and a thresholding neural network (TNN) to segment spatter signatures [192]. This model, tested across different laser powers, achieved precision and recall values of 0.777 and 0.805, respectively [192]. This application is innovative and demonstrates reasonable effectiveness whilst suggesting there is significant room for improvement. Spatter signatures have a complex and variable appearance in terms of shape, intensity and image coverage. Therefore, this kind of segmentation could be transferable to other applications involving “spatter-like” or “splatter-like” attributes such as splattering liquid.

2.8 Conclusions

This literature review has investigated modern advancements of CV technology with particular focus on their integration into manufacturing industry and a deeper emphasis on steel production. The review discussed several key groups of techniques that are believed to be beneficial towards industry, detailing their relative performances and suitability for industrial application. Additionally, discussion has included CV applications that have begun to revolutionise the technological state of manufacturing. Current and potential future benefits from this integration include, but are not limited to, improved operational efficiency, reduced defects, process automation and increased workplace safety, which are all of high value to the industry. Through examination of relevant techniques, as well as promising applications, this review has highlighted the applicability of CV to a diverse range of scenarios across all areas of manufacturing, and especially steel production. Also, this review has identified several research gaps and opportunities for future research and development.

The most significant research gaps identified include the need for real-time processing capabilities in edge-based CV systems without using large, powerful GPUs that lack portability and low computational expense. Improving models to not just perform well on standard metrics, but to perform efficiently on edge devices at real-time speeds is invaluable to successful implementation into high-pace, space-constrained scenarios such as those found on steel production lines.

Additionally, there is a lack of diverse datasets that properly represent the full complexity of industrial environments, which is critical for model development to ensure robustness to harsh conditions such as those found in steel production. Dataset creation is a substantial challenge separate from model development and deployment, and there are many different steel production scenarios that models could be developed for. Therefore, developing datasets that are large enough, diverse enough and also incorporate the visually harsh environments in steel industry, is a considerable challenge. If a dataset was built to target key emerging applications of CV in steel industry, such as a steel-based version of the COCO dataset, this could be incredibly valuable for the industry.

The literature also highlights the lack of extensive real-world application, testing and validation of these systems, since many models do not predict real-world characteristics and are evaluated using highly controlled conditions that are optimal for producing high performance results. Real-world application, testing and validation is essential for ensuring suitability for real application, which includes reliability, robustness and scalability. This does not simply mean adapting models to deal with harsh environmental conditions, but also to ensure aspects such as computational efficiency, hardware compatibility and ease of integration with existing industrial systems, are suitable for deployment. This ties in with the fact that most of the literature focuses on achieving good scores in metrics such as accuracy and mAP, whereas details on model deployment and the capability to perform accurate real-world measurements would provide better insights into the real-world applicability of research undertaken.

Furthermore, the integration of deep learning with traditional techniques such as background subtraction is an innovative research direction that requires further investigation, particularly in terms of computational efficiency and deployment on resource-constrained devices. Use of background subtraction algorithms for steel production applications is a gap in the research by itself, but incorporating the modern capabilities of deep learning opens up many possibilities for what is relatively unexplored in the field.

Another significant gap is the need for more standardised evaluation approaches for fair comparison of CV models in industrial environments. Whilst most studies use the same metrics, they are not inclusive of aspects such as dataset size, diversity or difficulty. This could be addressed by establishing metrics that account for the complexity of a given environment such as the level of heat distortion, the abundance of dust or the quality of lighting. This would help to distinguish between studies that use datasets that are easier to achieve good performance values on, and datasets that are more representative of what is found in industrial environments.

Analysis of the cost-effectiveness and potential return on investment of implementing CV techniques in manufacturing settings would be highly beneficial to the field and is not addressed in any reviewed literature. This would give more insight into which advances in CV are most likely to translate to financially viable solutions, and therefore help stakeholders in making more well-informed decisions in a field that is largely misunderstood by the wider public.

Also closely related, is consideration of human factors to ensure human interaction with CV systems is as easy and as effective as possible, which has not been addressed in literature. By focusing on user interaction, implementation of CV is more likely to occur quickly with less resistance from operators, and less technical issues. This includes aspects such as user-friendly interfaces, operator training strategies and ergonomic design.

This thesis will address several of the research gaps identified. Firstly, by deploying a model that not only performs well on typical CV metrics, but also operates efficiently in real-time on the edge without the need for heavy-duty hardware, meaning it is suitable for the high-pace and space-constrained environments of production lines.

Secondly, novel datasets will be developed for specific aspects of processes that CV technology is developed for. This contributes to reducing the data scarcity issue in the field. These datasets will incorporate the harsh conditions of steel production environments such as heat, dust, vibrations and poor lighting conditions, and will be used to develop CV systems that can be utilised in industry settings.

Thirdly, real-world application of CV models will be addressed by measuring real-world characteristics in every case study. In most cases, these measurements will be validated by data acquired through more traditional methods such as visual inspection and manual measurement. Robustness to environmental challenges is a key aspect of every case study, and scalability is ensured in two of three case studies. Real-world testing and validation will also be addressed through the development of a fully functioning real-time edge system functioning over a Wi-Fi network. Whilst this has not been physically integrated into the production line, it has been extensively tested on specialised hardware and optimised thoroughly for deployment, which has thrown up many of the challenges that would be experienced in the implementation stages as opposed to just the model development stages.

Fourthly, this thesis will pursue the integration of deep learning with traditional CV techniques and draw conclusions on both the current effectiveness and the future potential of these kinds of approaches.

Lastly, whilst standardised approaches have not necessarily been developed here, models have been evaluated against manually recorded data obtained from other researchers and site operators. This promotes the importance of comparing model outputs to real-world measurements, which gives a more industry-specific evaluation of model performances.

By addressing these gaps, this thesis aims to advance the field of computer vision within the context of steel production through demonstration of practical solutions that are novel and can be utilised by industry to enhance operational efficiency, sustainability and overall technological capabilities. This thesis also aims to bridge the gap between theoretical research and real-world application through consideration of model deployment, as well as model development. Furthermore, completion of this research provides a baseline for future studies involving CV technology that is not only technically sound but also aligned with the demands of industry.

Chapter 3: Methods and Materials

This chapter covers the primary methodologies and tools that form the basis of this research, with focus on those which have been utilised in multiple case studies. Components such as the Mask R-CNN (Mask Region-based Convolutional Neural Network) segmentation network and DeepSORT (Deep Simple Online and Real-Time Tracking) network are unique to specific case studies and are therefore excluded to streamline the thesis structure. Key aspects of this research are presented including computer vision (CV) tasks, algorithms and evaluation metrics, as well as the Microsoft Common Objects in Context (COCO) dataset which was the only public dataset utilised in this research. Also provided, are the software and hardware tools that were crucial for performing certain tasks.

3.1 Introduction

The integration of CV technology into industry has significantly advanced in recent years, which is directly applicable to steel production. As discussed in Section 2.7 of Chapter 2, these technologies have proven to be invaluable when addressing complex challenges such as surface defect detection, microstructural analysis and slag monitoring [4, 5, 6]. This chapter outlines the resources utilised for this body of work whilst highlighting their relevance and value for current industry challenges.

The objective of this chapter is to clearly lay out the key resources used for this thesis to ensure the reader can understand the core components used to achieve presented outcomes. Furthermore, this chapter aims to aid future projects that are replicating or re-thinking this work.

This chapter contributes to the field by detailing a set of valuable methodologies and tools that are well-suited to projects involving the application of computer vision algorithms to steel process monitoring. Algorithms such as YOLOv5 (You Only Look Once) and Counting (CNT) background subtraction (BGS) are noted for their strengths in monitoring applications, whilst the COCO dataset is identified as a dataset beneficial when pre-training a model for any computer vision application.

3.2 Tasks

CV encompasses a variety of tasks that allow computers to gain an understanding of digital images and videos. In the context of this project, the CV tasks performed are classification, object detection, instance segmentation and multi-object tracking. Background subtraction is also a CV task relevant to this work, but is traditional and does not use deep learning. Therefore, it will be introduced as part of its own section (Section 3.3).

3.2.1 Classification

Classification refers to the task of identifying the class of an object. Typically, a set of class labels are pre-defined and during inference, a label is assigned to either an image, or specific objects within an image [193]. This is one of the most basic CV tasks and allows computers to gain a basic understanding of visual data that is presented to it.

3.2.2 Object Detection

Object detection goes a step further than classification by identifying object class, whilst also identifying object location (known as object localisation) [193]. Locations are indicated by drawing bounding boxes around detected objects. Object detection networks can be two-stage detectors which generate bounding box proposals, and then refine boxes whilst classifying objects within them [193], or they can be one-stage detectors which simultaneously perform full classification and localisation [193]. YOLOv5 (You Only Look Once) used in this project is one-stage, hence the name. Mask R-CNN that is used in Case Study 1 (Chapter 4), is based on the Faster R-CNN detector which is two-stage [63, 98]. Typically, two-stage detectors are more accurate and one-stage detectors are faster [193]. Detection is useful for applications where the presence and position of objects is of interest.

3.2.3 Instance Segmentation

Instance segmentation advances further from the capabilities of object detection by predicting precise pixel-level object boundaries whilst distinguishing between different instances of the same class [194]. This task is useful for applications that require detailed analysis of the exact shape, size, and location of objects, and enables CV applications to build a more complex understanding of different scenes. The model used in Chapter 4, Mask R-CNN, is an instance segmentation network.

3.2.4 Multi-Object Tracking

Multi-object tracking (MOT) refers to the identification (ID) and tracking of multiple objects as their position changes across a sequence of video frames [195]. Typically, tracking is performed through a "tracking-by-detection" approach which starts with object detection and then uses a re-identification model to associate object IDs across frames [195]. MOT is useful for analysing scenes where understanding patterns in object movement over time is of interest. MOT has been used in Case Study 3 (Chapter 6) to track multiple plumes simultaneously.

3.3 Algorithms

In CV, algorithms are arguably the most important part of developing an application since it is essentially them which allow computers to gain an understanding of visual data. In the context of this project, common algorithms utilised include YOLOv5 for object detection, and the Counting algorithm for background subtraction.

3.3.1 YOLOv5

YOLOv5 was a core component of Case Study 2 (Chapter 5) and Case Study 3. Whilst an overview of YOLOv5 was provided in Chapter 2, this section will go into more depth on application details both generally and in the context of this research, the rationale for selecting this model over alternatives, as well as some known limitations of the model.

YOLOv5 has already been commended for being computationally efficient, fast and adaptable in Section 2.3.8 of Chapter 2. Additionally, it has a wide variety of customisable options for optimal application. Firstly, there are several variants that differ in size, offering different degrees of speed, precision and computational cost. Figure 24 shows the variants that have been used in this research and their results when tested on the COCO dataset [196].

In addition to model size, YOLOv5 has many different hyperparameters that can be adjusted depending on the scenario for optimal performance. Examples of key hyperparameters include learning rate, which dictates how quickly the model adapts to data during training, and weight decay, which is a regularisation technique that reduces overfitting by penalising larger model weights to help maintain small weights, therefore improving resistance to noise [197, 198]. To clarify, overfitting is when a neural network overly adapts to a set of training data which results in poorer performance on unseen data. There is also the option to modify

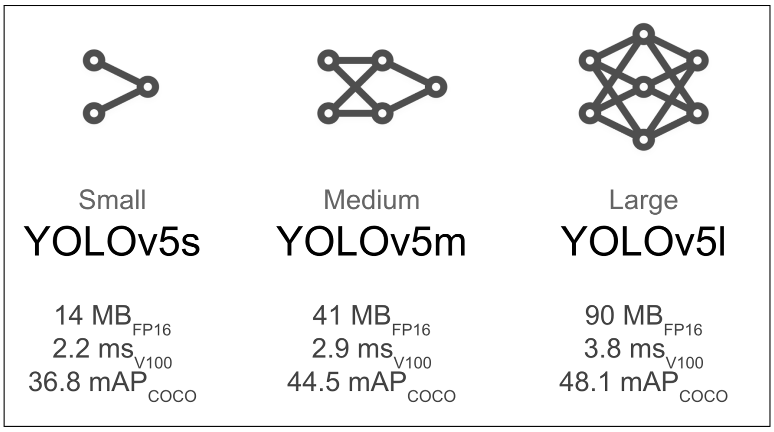


Figure 24: Variants of YOLOv5 differing by network size

thresholds for non-maximum suppression (NMS) and detection confidence which control the number of false positives, and therefore offer a trade-off between precision and recall (introduced in Section 3.4).

Finally, Ultralytics have incorporated a feature to export trained models in .onnx format for optimal deployment performance [199]. ONNX (Open Neural Network Exchange) is different to the typical PyTorch format and enables portability of models across different devices, which is particularly useful for real-world application of CV [200]. Additionally, ONNX models can be optimised for high speed and high precision, solidifying their practicality for real applications. This feature was used prior to converting the Chapter 5 model into .engine format for TensorRT optimisation and inference, which is explained more in Chapter 5 [201].

In Chapter 5, zinc splatter severity was quantified during the galvanising process in real-time. YOLOv5 was used to dynamically adjust the splatter measurement region in response to air knife movement for accurate monitoring. YOLOv5 was also used to ensure consistent model performance across varying camera positions, by scaling severity boundaries based on bounding box sizes. In Chapter 6, gaseous plumes formed during a gas stirring simulation were monitored for design optimisation. YOLOv5 was used to classify plume state (jetting, forming, collapsing) and bounding box dimensions were used to calculate plume geometries.

At the time of initiating the studies, YOLOv5 was the true state-of-the-art object detection network. More improved versions of YOLO have been released since then, but YOLOv5 still stands as a well-established, proven and effective technique with substantial community support. Due to being released several years ago, it is arguable that YOLOv5 is still a more appropriate choice of detection network for industry challenges that require robustness, than a more evolved version such as YOLOv8. This is evident from the literature that has been published exhibiting either direct applications of YOLOv5, or applications using models that have been built using YOLOv5 as a foundation. Several examples of this are covered in Chapter 2 [93, 151, 172], which show a range of successful cases such as blast furnace charging state recognition and defect detection with high mean average precision (mAP) results and over 29fps inference speed. In this research, YOLOv5 has been combined with background subtraction to reap the benefits from both approaches. Through this novel combination of techniques, it has been possible to monitor distinct steel production processes and extract valuable insights. This has not been attempted in any existing literature.

Whilst YOLOv5 is an excellent tool for solving real-world problems with object detection, it is also important to acknowledge some known limitations of the model that were considered before utilising it. It is known that there is challenge in balancing a trade-off between speed and accuracy, and therefore it was expected that this would have to be accounted for by sacrificing one or the other for each case study depending on the aims. Inference speed was critical in Chapter 5, and therefore it used YOLOv5s. In Chapter 6, precision was more critical, (however YOLOv5s slightly outperformed YOLOv5l anyway due to overfitting). Another limitation is that YOLOv5 is a 2D object detector and therefore, unless great model developments are performed, the model is limited to learning and making predictions from 2D features. In engineering scenarios such as those described in this thesis, lack of a third dimension constrains model capabilities. It is acknowledged that there are ways to overcome this, such as using multiple cameras, but it still remains a limitation.

3.3.2 CNT Background Subtraction

CNT (Counting) background subtraction which was also a key component of this project, was utilised in all three case studies and successful in two. Whilst an overview of different BGS algorithms was given in Section 2.6 of Chapter 2, this section will go more in-depth on the CNT algorithm in particular, application details (both general and in the context of this research), the rationale for selecting this model over alternatives, as well as some known limitations of the model.

The CNT algorithm works by counting the number of frames each pixel value stays constant for (called pixel stability) [202]. It uses pixel stability thresholds to dictate whether the stability value of each pixel should class it as foreground or background [202]. The algorithm is also based on colour similarity which uses a colour similarity threshold to define what colour ranges are considered the "same" colour [202]. This algorithm is more procedural than based on statistical distribution models (like other BGS algorithms), so it is not represented by any formula. Therefore, Algorithm 1 presents the operation of the CNT algorithm.

Algorithm 1 CNT Algorithm

```
for each pixel in the frame do
    if pixelColour == previousPixelColour then
        pixelStability += 1
    else
        pixelStability -= 1
    end if
    if pixelStability $\geq$ minPixelStability then
        classify pixel as background
    else
        classify pixel as foreground
    end if
end for
```

In Chapter 5, the CNT algorithm was utilised for segmenting zinc splatter from the background. The speed of the CNT algorithm was crucial for monitoring splatter in real-time, and the ability to adapt to environmental changes ensured precise segmentation despite the highly variable nature of the splatter. In Chapter 6, the CNT algorithm was used to segment plumes from the background. The adaptability of CNT background subtraction to the complex fluid dynamics of plumes was critical to the success of the model.

CNT background subtraction is highly effective at motion detection which makes it useful for industrial applications, particularly those with static backgrounds [203]. The CNT algorithm is faster than all other OpenCV BGS algorithms and literature has reported it as the highest quality algorithm also [120, 203]. Furthermore, the literature (see Section 2.6 of Chapter 2) shows that OpenCV BGS algorithms have been successfully applied in a variety of cases such as ship detection, vehicle detection, suspicious activity recognition, fall detection, wild mammal detection and Parkinson's patient tracking [115, 118, 120, 121, 122, 123, 124]. Despite these examples, there are no known cases of these algorithms being successfully applied to steel processes, which highlights the novelty and contribution of this thesis. Furthermore, there are a few known cases of BGS being combined with deep learning which are mentioned in Section 2.6 of Chapter 2, however they are again, not for steel production processes and do not use an object detector as efficient for real-time application as YOLOv5 [125, 126, 127].

Despite the strengths of background subtraction and particularly the CNT algorithm, considering limitations is still important. The limitations of background subtraction have already been discussed in Section 2.6 of Chapter 2, which mentioned that dynamic backgrounds may significantly affect BGS performance [114]. Industrial environments such as those related to this project, are typically harsh with multiple sources of noise and moving objects. Whilst BGS can deal with noise effectively (provided it is applied correctly), moving objects that are not being tracked can be problematic and should therefore be overcome using another approach, such as object detection. The air knife movement in Chapter 5 is a great example of this kind of scenario and exemplifies overcoming limitations of one technique by capitalising on the benefits of another.

3.4 Metrics

In CV, metrics are essential for evaluating model performance against previous and future iterations of development, as well as comparison against similar models performing the same or similar tasks. Metrics utilised in this work include precision, recall, average precision (AP), mean average precision (mAP) and inference speed. These are conventional metrics as evidenced in evaluation reports throughout Chapter 2. Multi-object tracking accuracy (MOTA) is also included here to be comprehensive, however is not used in the case studies for reasons explained in Section 3.4.5.

3.4.1 Intersection-over-Union

Intersection-over-union (IoU) is a simple way to indicate the accuracy of an object detection (or instance segmentation) network. Essentially, it is the amount of overlap (intersection) between predicted bounding boxes or segmentation masks, and the ground-truth boxes or masks, compared to the total area of the two boxes or masks (union) [204]. This is demonstrated in Equation (1).

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{1}$$

When evaluating models, depending on the approach, one or more IoU thresholds are set to distinguish between predictions considered accurate and predictions considered inaccurate. Predictions with an IoU above the IoU threshold are considered true positives (TP), and predictions below the threshold are considered false positives (FP). Therefore, predictions are categorised as:

- TP if $IoU >$ threshold.
- FP if $IoU <$ threshold.

3.4.2 Precision and Recall

The total numbers of true positives and true negatives can be used for deeper evaluation insights by calculating precision and recall for a given experiment. These are calculated as shown in Equation (2) and Equation (3) [205]. Precision indicates how many correct guesses the model made out of all guesses made, and recall indicates how many ground-truth boxes or masks were correctly predicted compared to the entire set. Note that in Equation (3) false negatives (FN) are also used, which are ground-truth instances that were unsuccessfully predicted. In this context, the distinction between a false positive and a false negative is that false positives are based on unsuccessful predictions that were made by the model, whereas false negatives are based on unsuccessfully predicted ground-truth instances in the data.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\#\text{predictions}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\#\text{ground-truths}} \tag{3}$$

3.4.3 Average Precision

To gain even more insight on model performance, precision and recall can be used to plot a precision-recall curve such as the yellow one shown in Figure 25 [206]. The values for all TP predictions on all test images are ordered based on increasing recall, and then the curve is smoothed by replacing the precision at every data point with the maximum precision value with the same or higher recall value, resulting in the green line in Figure 25 [206]. Smoothing involves replacing each precision value with the maximum precision found at

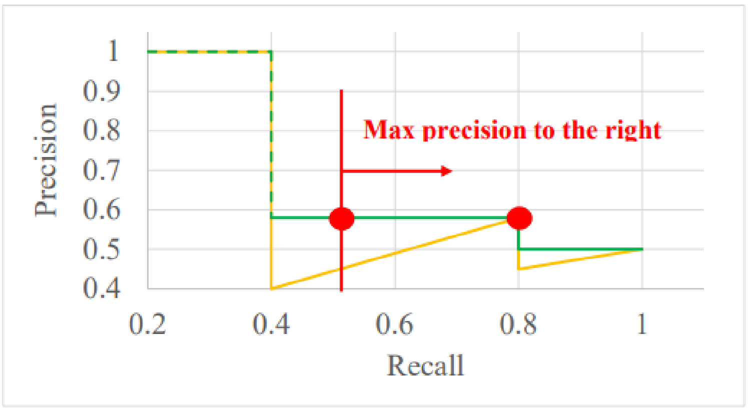


Figure 25: Example showing smoothing of a precision-recall curve

that recall level or higher. This is based on the principle that precision should not decrease as recall increases, which is an optimistic approach that ensures anomalies do not distort the calculation of average precision (AP), and is a standard approach in computer vision [207]. After smoothing the precision-recall curve, the AP is calculated using the area under the curve (AUC) [207].

The COCO AP is a standard evaluation metric in computer vision and is a variant of AP that is designed to be more comprehensive when measuring the performance of a model. For COCO evaluation, AP is calculated by dividing the curve at 101 points along the recall axis, summing the 101 corresponding precision values and then taking the mean average of these values, AP_r [208]. This is shown in Equation (4) [206, 208].

$$\begin{aligned} AP &= \frac{1}{101} \sum_{r \in (0.0, \ldots, 1.0)} AP_r \\ &= \frac{1}{101} \sum_{r \in (0.0, \ldots, 1.0)} p_{interp}(r) \end{aligned} \tag{4}$$

The $p_{interp}(r)$ in Equation (4) is the interpolated precision at each selected recall point $\tilde{r}$, which is the maximum precision at any given recall point, as shown in Equation (5) [206, 208].

$$p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r}) \tag{5}$$

3.4.4 Mean Average Precision

In multi-class problems, the AP is calculated for every class and the mean average of the AP values is calculated to obtain an mAP value, which is a common indicator of the performance of a model used in CV [207]. For the COCO mAP, which was designed to reduce any bias towards certain IoU thresholds, the mAP is calculated for threshold values ranging from 0.5 to 0.95 in increments of 0.05. After calculating all mAP values, a mean average of them is taken to give a final COCO mAP value. This is shown in Equation (6) [207].

$$mAP_{COCO} = \frac{mAP_{0.50} + mAP_{0.55} + ...mAP_{0.95}}{10} \tag{6}$$

It is worth noting that whilst the real definition of the COCO mAP has been demonstrated here, the term "mAP" can be ambiguous due to the different approaches researchers use for model evaluation. For example, some researchers do not clarify whether they use mAP at a specific IoU threshold or the COCO mAP. Also, some researchers calculate it using an 11-point interpolation rather than a 101-point interpolation, and sometimes the way it is averaged across a set of object instances or video frames can vary [209, 210, 211].

3.4.5 Multi-Object Tracking Accuracy

Multi-object tracking accuracy (MOTA) is used to evaluate the performance of multi-object tracking algorithms. It is calculated using the number of instances in the ground-truth that the algorithm did not detect (misses), the number of instances predicted by the algorithm that were not in the ground-truth data (false positives) and the number of instances where the identity of an object was changed incorrectly (identity switches) [95]. Equation (7) shows the MOTA calculation [95].

$$MOTA = 1 - \frac{(\text{Misses} + \text{False Positives} + \text{Mismatches})}{(\text{Total Ground} - \text{Truth Objects})} \tag{7}$$

Whilst MOTA is a strong indicator of tracking performance, researchers typically focus more on detection and segmentation performance, since they are the foundation of high performance tracking (as mentioned in Section 2.4 of Chapter 2 [195]. Additionally, it is difficult to obtain tracking annotations since good detection and segmentation performance typically leads to good tracking performance, and the benefits of good detection and segmentation are more obvious and immediate [212]. During this research, MOTA was not used due to these reasons. However, as the field develops and more tracking datasets and annotation methods become available, MOTA will give deeper insight into model performance. By using MOTA, a more holistic view of tracking algorithm performance will be available which will be particularly useful in complex tracking scenarios where objects interact closely and occlude each other, since in these scenarios, maintaining consistent identities will be crucial.

3.4.6 Inference Speed

Inference speed is a critical metric for many real-world applications of CV, particularly in the case of industrial process monitoring. In all three case studies, it has been preferable for models to operate at the fastest speed possible. However, the need for high speed performance varied. In Chapter 4, if possible, the model would be more beneficial if it operated in real-time, since this would enable it as a foundation for building a closed-loop control system. The definition of "real-time" in CV is ambiguous. One common definition of a real-time system is one that processes frames fast enough for practical application [101], whereas another popular definition is that a minimum frame rate of 25fps should be achieved [102]. For this project, the former of these two definitions has been considered "real-time" since this is what is needed for industry. In Chapter 5, the model was required to run in real-time upon deployment, including the additional network overhead from the video source sending frames and the inference device returning model results. In Chapter 6, more speed was preferable, especially for any future developments, however it was not critical to the design since the model was used as a post-processing tool after collecting footage of experiments. In this thesis, "inference speed" has been used to describe the frame speed a model can run at (measured in frames per second), whilst "inference time" refers to the time it takes a model to make an inference (typically measured in seconds or milliseconds).

3.5 Datasets

Datasets are arguably the most important component when developing CV applications with deep learning (DL). Each case study used unique datasets that were developed specifically for this project, however, the COCO (Common Objects in Context) dataset was used to pre-train Mask R-CNN in the first case study, and YOLOv5 in the second and third case studies. This is common practice, as is use of the COCO dataset for benchmarking model performances, as shown throughout Chapter 2.

3.5.1 Microsoft COCO Dataset

The COCO dataset is a renowned resource in the field of CV that is used for tasks such as image classification, object detection and instance segmentation [45]. The dataset was initially released in two separate parts; the first in 2014 and the second in 2015, which cumulatively contained 165482 training samples, 81208 validation samples and 81434 testing samples [45]. When first released, COCO contained 91 object classes easily recognisable by a four year old, within their natural everyday scenes. Examples include dogs, cats, aeroplanes, boats, kites, plants and teddy bears. Overall there were 2.5 million labelled instances and some examples are shown in Figure 26 [45, 213].

By presenting a diverse range of objects in their natural environments that exhibited challenges such as varying scales, angles, lighting and overlapping, COCO helped in training models to be robust to complex, realistic scenes. This was different to previous datasets containing objects in unobstructed, centered and clear contexts which led them to being less robust in other scenarios [45]. At the time of release, COCO was also unique as it focused on providing detailed instance segmentation masks, as opposed to other datasets that only focused on classification, detection and semantic segmentation [45]. This meant it could be used for training models with more precise localisation and applications requiring more detailed visual understanding [45].

Since the initial release in 2014, newer versions have been released with the latest complete dataset to date being the COCO 2017 dataset, however other partial updates occurred up until 2020, such as the inclusion of keypoint and panoptic annotations [214]. Since being released, the COCO dataset has been at the core of the COCO challenge which was hosted annually until 2020, whilst also becoming established as a highly popular benchmarking dataset for state-of-the-art models performing tasks such as object detection and segmentation [214]. By providing a consistent format for evaluating algorithms, the COCO dataset has become a fundamental resource for many advancements in the field.

For the purpose of this research, the COCO dataset was used as the foundation for pre-training Mask R-CNN for the ladle study. It was also used to pre-train YOLOv5 in both the splatter severity measurement and plume monitoring studies. Despite the fact that steel production environments do not fall into “common” contexts, pre-training models on COCO is still beneficial as it provides models with foundational knowledge on how to understand general objects, before being trained to specialise in a more specific context.

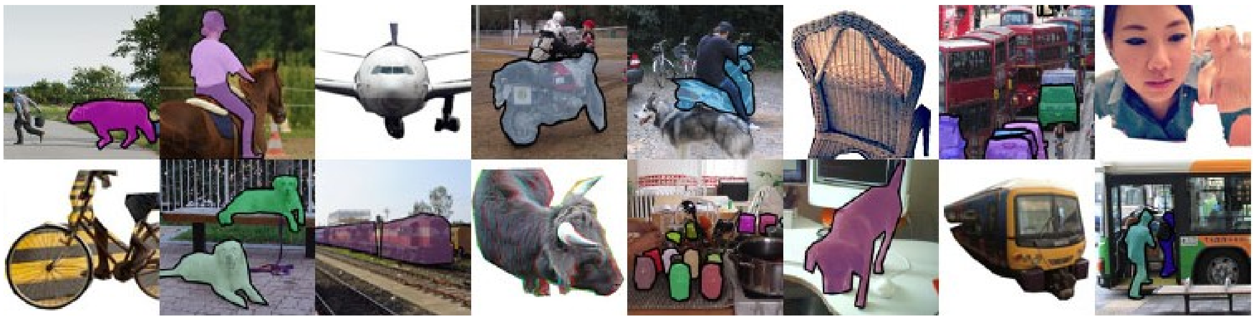


Figure 26: Annotated samples from the COCO dataset

3.6 Software and Hardware

Software and hardware were crucial to the success of this project since they provided the capabilities to perform all tasks. Key resources include Python, VGG (Visual Geometry Group) Image annotator (VIA), and graphical processing units (GPUs).

3.6.1 Python

Python served as the programming language for this research and was therefore crucial to the success of it. Python was suitable for a variety of reasons including its versatility, simplicity, well-established community and in particular its compatibility with deep learning projects. It was used for pre-processing data, model development, model evaluation, data visualisation and results analysis.

In terms of pre-processing data, Python libraries were used such as OpenCV for image processing, NumPy for numerical computations and data manipulation, Pandas for creating results tables which were typically exported in Microsoft Excel format, and JSON (Javascript object notation) for inspecting, editing and loading annotations. These libraries provided capabilities in data cleaning and data splitting into training, validation and testing sets, so that they were suitable for processing by the models.

Model development was driven by Python due to its seamless integration with deep learning frameworks which in this project, were TensorFlow and PyTorch. This was crucial in order to analyse, break down, modify and optimise models such as Mask R-CNN and YOLOv5 through actions such as custom layer addition, hyperparameter optimisation and standardising evaluation metrics. Furthermore, OpenCV played a large part in developing the elements of models that utilised more traditional CV techniques such as background segmentation and image denoising. Model evaluations which were, at times quite complex, were made possible through Python's relative ease of implementation of metrics such as precision, recall and COCO mAP.

Data visualisation was a key aspect for gaining insight on model behaviour and model performance, as well as actually presenting results to stakeholders. Using OpenCV, the effects of different techniques, such as image segmentation and image denoising, were visible along every stage of the overall process. Meanwhile, OpenCV was also used to produce video output from the models in various formats such as trimmed versions of raw data, binary mask videos, as well as original data with model predictions superimposed.

Finally, Python was not only used for visualising data but also analysing numerical results, such as severity levels of different process variables. Through Python, actions such as monitoring model performance, troubleshooting model issues and extracting insights from model outputs were possible.

In addition to the all of the aforementioned capabilities provided by Python, Microsoft Excel was also used for some aspects of data presentation and analysis. Due to being more familiar with stakeholders, it was useful for efficient exchange and demonstration of results, and therefore was used to compliment the primary workload conducted using Python.

3.6.2 VGG Image Annotator

VIA was the annotation tool of choice for every case study [215]. VIA is a simple open source manual annotation tool that enables users to annotate images with box or mask annotations and export them in multiple formats depending on their intended approach for using the annotations [215]. In this research, annotations were exported in the JSON format. This format is widely used and therefore relatively easy to integrate into model setups or convert to appropriate formats. For YOLOv5 applications (splatter severity measurement and plume monitoring studies), a conversion tool was used to convert the JSON annotations into .txt format suitable for YOLOv5 processing [216]. In the case of Mask R-CNN, the JSON files were suitable for processing with no conversion needed.

3.6.3 GPU

The NVIDIA GeForce RTX 2070 Super was the primary GPU used for this research. This device is a graphic card developed by NVIDIA that has 8GB of GDDR6 (graphics double data rate 6) memory and 2560 cores, meaning it has the ability to perform highly computationally intensive tasks [217]. The 2070 Super is built to support demanding activities such as video transcoding, physics simulation, ray tracing and more, making it well-suited for deep learning applications [217].

In the context of this research, this GPU significantly improved training and inference times, which was crucial not only for project efficiency, but also for model evaluation with regards to suitability for industrial application. By using this hardware, it was possible to perform tasks such as the hyperparameter optimisation in Chapter 4 within reasonable timescales. Additionally, it made a significant difference to the speed of the final splatter severity measurement model (pre-deployment), proving its ability to deal with real-time data processing particularly in the context of CV where computational demand is high due to the heavy-duty processing of large datasets containing high-resolution images. The compatibility of the chosen GPU with frameworks such as TensorFlow and Pytorch made this research much more streamlined, and was due to the CUDA (Compute Unified Device Architecture) and CUDnn (CUDA Deep Neural Network) libraries developed by NVIDIA, which enable optimised GPU-accelerated computations that are crucial for deep learning [218, 219].

3.7 Conclusion

This chapter has outlined the common computer vision tasks performed in this project, the algorithms used to perform them, the dataset used for pre-training of algorithms, the metrics used to evaluate algorithm performance, as well as the software and hardware resources used to conduct research.

Throughout this chapter, it has been made evident that decisions surrounding algorithm selection, evaluation metric selection and dataset selection are based on the literature reviewed in Chapter 2. This has ensured that techniques and resources used in this project align with what is typical in the field of computer vision.

Moving forward, the aspects discussed in this chapter form the foundations of the case studies in the following chapters. By establishing a robust methodological framework, this research paves the way for future innovations and applications in the steel industry.

Chapter 4: Hot Metal Ladle Pouring Process Parameter and Process Quality Estimation

In this chapter, a novel method is presented for tracking the motion of hot metal ladles during pouring in harsh environmental conditions. By utilising Contrast-Limited Adaptive Histogram Equalisation (CLAHE) for contrast enhancement, combined with Mask R-CNN (Mask Region-based Convolutional Neural Network) for ladle segmentation, this work is an innovative approach for monitoring ladle behaviour. As well as monitoring ladle pouring height and rotation angle, flame severity is estimated as a process quality indicator.

4.1 Introduction

Pouring of hot metal into basic-oxygen furnace (BOF) vessels is a critical step of the steel-making process. It is typically performed manually and is associated with challenges such as harmful emissions in the form of flames, smoke and metallic dust due to reactions occurring within the furnace, equipment degradation caused by reactions, and operational variability due to inconsistent human performance. This drives the need for minimising emissions to enhance workplace safety and extend the useful life of equipment, and standardising the process through automation. A schematic of the pouring setup is shown in Figure 27.

One aim of this case study was to segment a hot metal ladle in each frame of a video with a COCO (Common Objects in Context) mean average precision (mAP) of at least 0.5 (see Section 3.4.4 of Chapter 3 for more details on this metric). This was chosen as the acceptable minimum performance before attempting to monitor ladle behaviour, and was a pragmatic estimate based on the various mAP results reported throughout Chapter 2, as well as the highly challenging conditions of the task. Challenges included poor lighting conditions, the presence of smoke and dust obscuring the ladle, and resource-constraints with regards to time and primarily data availability since it was difficult to obtain. These constraints are akin to conditions present in industry. Additionally, another aim was to quantify the severity of emissions in the furnace based on what was observable in the footage. By monitoring ladle behaviour and the emission severity resulting from it, this research finally aimed to look at the relationship between the two types of measurements and provide insight on how pouring process parameters affect the quality of the pouring process.

This work demonstrates how computer vision (CV) can be used for monitoring and optimisation of the pouring process. Other potential benefits include full process automation in a closed-loop control system and gaining deeper insights for root-cause analysis and predictive maintenance strategies. These lead to reduced human error, improved health and safety, improved environmental impact and cost savings. This work was largely a learning experience that began during the infantile stages of the overall project, however it has laid a strong foundation in the field for ladle monitoring applications. Source code is available on GitHub [220].

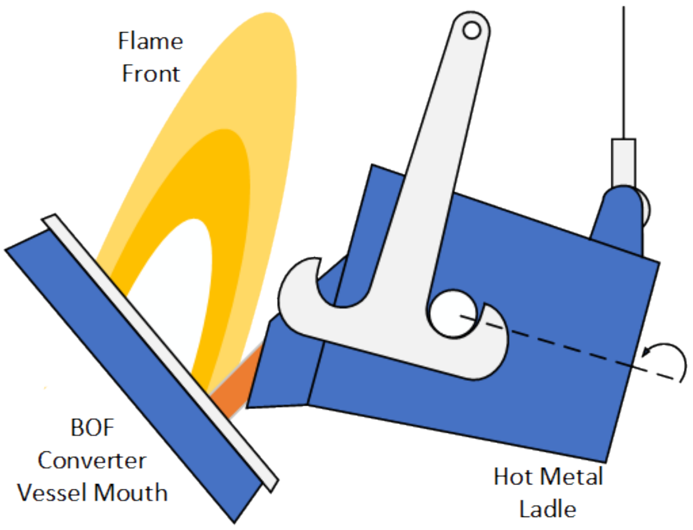


Figure 27: Schematic representing the ladle pouring setup

4.2 Methodology

The data for this case study was generated by another researcher recording different ladle pouring events. An overview of the methodology is given in Figure 28, which shows there were five main steps. The first step of the methodology involved data preparation, which included labelling frames, denoising them and then splitting them into training and validation sets. The data prepared in this case study addresses the gap in Chapter 2 regarding the lack of diverse datasets available for developing CV applications for steel production processes. This was followed by the initial training and testing of an instance segmentation network using the prepared ladle data. Mask R-CNN was selected for its state-of-the-art performance at the time this case study began, as shown in Section 2.5.1 of Chapter 2. Following training and testing, the hyperparameter optimisation was conducted which involved a large range of configurations to maximise model performance. Afterwards, Kalman filtering was applied for tracking of the predicted segmentation mask to improve model performance further. Finally, the model was adapted to measure real-world characteristics in order to draw useful insights for industry.

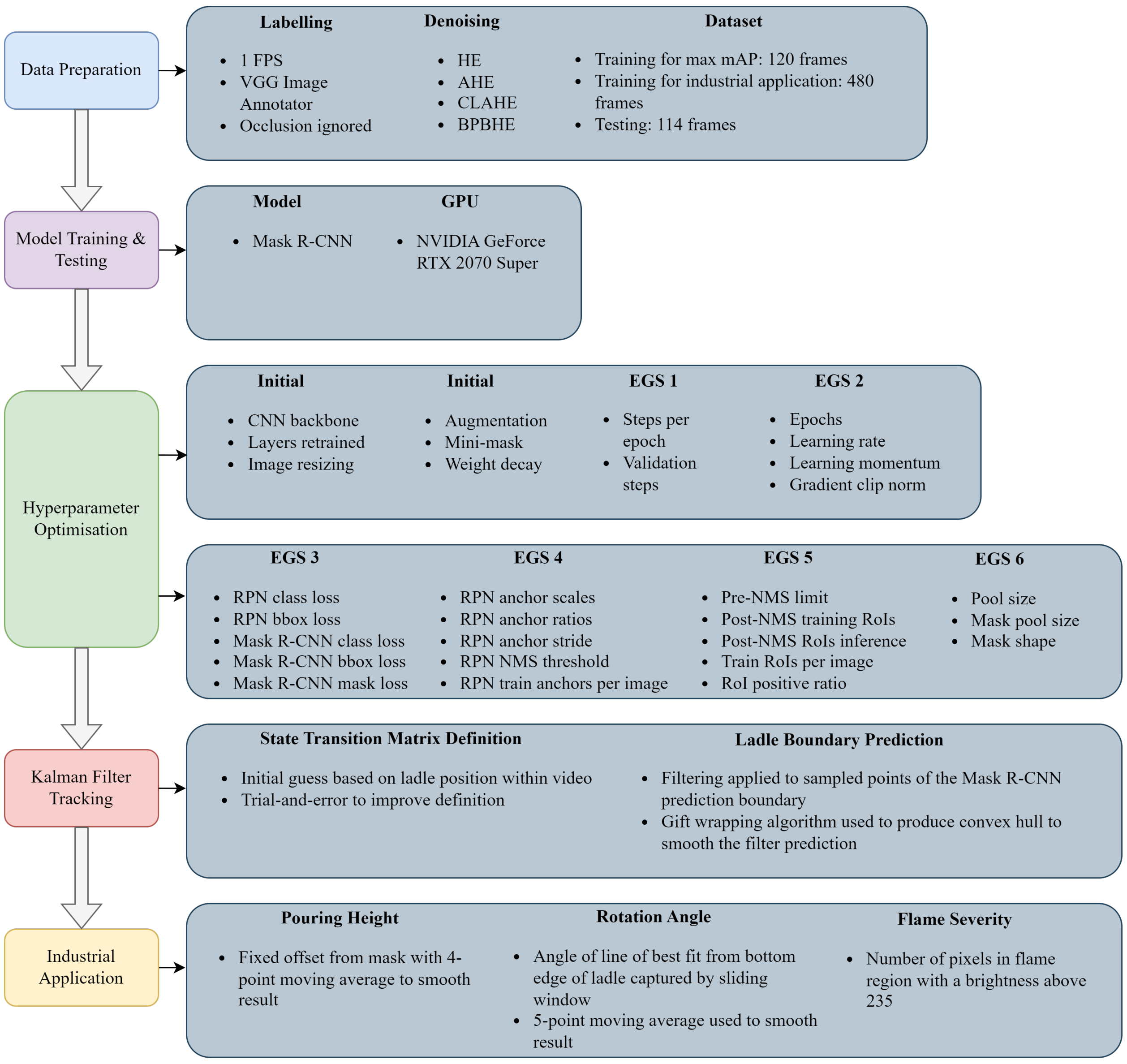


Figure 28: Overview of the methodology used for Case Study 1

4.2.1 Data Preparation

The first step of data preparation is labelling. Video frames were extracted at a rate of one frame per second using frame extraction code. This rate was chosen as a minimalistic approach, considering the original videos ran at over 30fps. This decision was made to lessen the computational demands and simplify the task whilst still providing a visually adequate output for analysis. The frames were labelled using the VGG (Visual Geometry Group) Image Annotator [215], involving the creation of a polygon (mask) around the ladle in each frame, which was then copied, pasted onto subsequent frames, and adjusted until all frames were labelled.

During the labelling process, there were frames where parts of the ladle were occluded by a metal structure that belonged to the furnace. To address this, three strategies were considered:

- Labelling the ladle as if there were no obstructions.
- Labelling only the prominently visible part of the ladle.
- Labelling all visible sections of the ladle, whether completely or partially visible.

Of the proposed approaches, the first was found to be the most practical choice. It provided the consistency required by Mask R-CNN whereas the other two methods led to significant variations in shape due to occlusions.

The second step of data preparation is frame denoising. Figure 29 illustrates that the video had sub-optimal lighting conditions, leading to frames where the ladle appeared distorted. This distortion negatively affected the performance of Mask R-CNN. To tackle this, efforts were made to enhance the visual clarity by improving lighting conditions in the frames. The approach to contrast enhancement involved initially converting the frames from RGB to greyscale, before applying different methods of Histogram Equalisation (HE) to enhance the lighting whilst preserving the overall quality of the frames. The specific variants of HE that were used are detailed below.

HE requires initially transforming the frames into greyscale to acquire the intensity values for each pixel. These values range from 0 to 255, indicating the degree of darkness (0) or lightness (255) of each pixel. Following this conversion, a histogram representing the frequency of each intensity level was constructed. Figure 30 displays the histogram corresponding to the greyscale image shown in Figure 29.

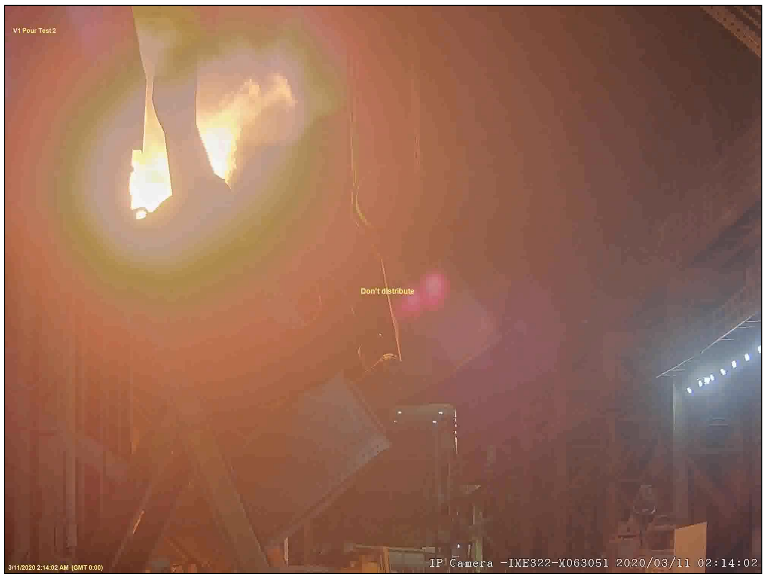

V1 Pour Test 2
Don't distribute
3/11/2020 2:14:02 AM (GMT 0:00)
IP Camera -IME322-M063051 2020/03/11 02:14:02

Figure 29: Frame with poor lighting, noticeable mostly around the right-side edge of the ladle

After generating a histogram like the one in Figure 30, it became feasible to improve the image contrast through equalisation, which involves utilising a broader range of intensity values more frequently. The first step in this process was calculating the probability mass function (PMF), which represents the likelihood of each intensity value appearing in the image. This calculation was achieved by dividing the frequency of each intensity value, as indicated in the histogram, by the total pixel count in the frame. This approach is detailed in Equation (8) where X generally represents any set of grey values in the image, $p(x_i)$ denotes the probability of any specific greyscale value occurring, s is the frequency of occurrence of that greyscale value, and S represents the total number of pixels in the frame [221].

$$p(x_i) = P(\{s \in S | X(s) = x_i\}) \tag{8}$$

It was possible to calculate the cumulative distribution function (CDF) using the PMF, since it is the cumulative sum of all PMF values. This is demonstrated in Equation (9) where $F(x)$ represents the CDF up to a specified intensity level x [221].

$$F(x) = P(X \leq x), \text{ for any } x \in \mathbb{R} \tag{9}$$

For each CDF value, which was determined by accumulating along the PMF values, a multi-

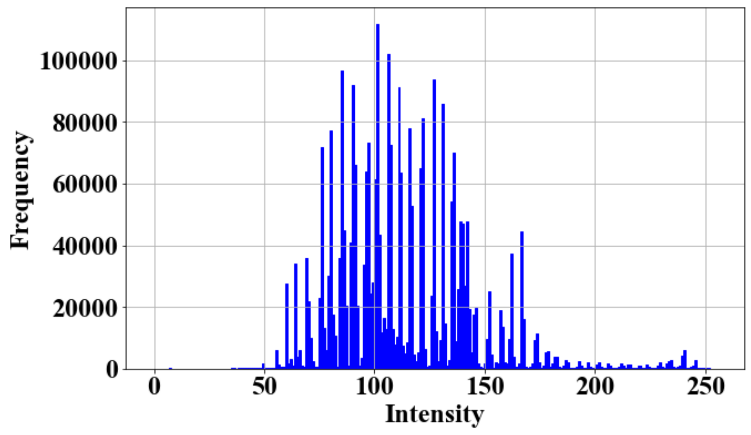


Figure 30: Greyscale histogram of the image in Figure 29, showing greyscale values along the x-axis and pixel frequency on the y-axis

plication was performed with the corresponding number of greyscale levels minus one. This calculation was done to obtain new intensity values for the image [221].

Adaptive Histogram Equalisation (AHE), an advanced version of the standard HE technique, starts by partitioning the image into uniformly sized square sections. Standard HE is then applied to each of these squares individually. This process results in a block-like appearance in the image's contrast, which is smoothed out using bilinear interpolation [222].

Due to the tendency of AHE to emphasise noise in areas where a square consists of pixels with mostly similar intensity values, contrast limiting can be applied before calculating the CDF [223]. This involves establishing a threshold for pixel frequency and clipping the histogram at this threshold before the CDF calculation, which helps prevent excessive enhancement of noise in uniform regions of the image. Contrast limiting is particularly effective for images that exhibit a relatively uniform distribution of intensity values [224].

The third step of data preparation is splitting the dataset. For the initial aim of achieving 0.5 minimum mAP, 120 frames were used for training and 114 frames were used for testing. These were essentially two different pouring videos. Whilst in many scenarios, it is typically ideal to use a split of about 70% for training, 10% to 20% for validation and 10% to 20% for testing, in this particular case the 120/114 split was a more logical approach. This was for two reasons.

Firstly, the data available was limited. The maximum amount of ladle video frames were 594 and labelling a few hundred was highly time-consuming due to the complex ladle shape, as well as the harsh environment the ladle existed in.

Secondly, rather than using random frames for training and testing, it was important that the model was trained on a full pour so that it was familiar with the whole process. Ideally, the training and validation sets would have contained several full pouring videos, and there would be a separate testing set for evaluating the model. To clarify, in this case study, the validation set was used for both intra-epoch validation for model feedback during training, as well as validation once it had been trained. This is not the same as using the same data for training and testing, but still has the potential to result in some bias that eventually leads to overfitting. This is fairly common practice in machine learning (ML) tasks where data is limited, and bias can be mitigated through approaches such as hyperparameter optimisation.

Once the model was proven to work, the remaining 360 frames available were labelled and added to the training set to improve performance for industrial application.

4.2.2 Model Training and Testing

Mask R-CNN was trained and tested using a NVIDIA GeForce RTX 2070 Super GPU (graphical processing unit). Network weights were pre-trained on the COCO dataset before being used for this case study. This reduced time and computational expense by ensuring the model was somewhat familiar with general objects, their features and their contexts. Steel production environments are different to the common objects in the COCO dataset, however pre-training is still beneficial for providing the model foundational knowledge of objects, boundaries, colours and other aspects of complex scenes. This is the first step before fine-tuning the model on the specialised area of steelmaking. A dataset that is similar to the COCO dataset but based on steelmaking environments would be highly valuable, since it could be used to accelerate the state of steelmaking technology significantly.

4.2.3 Mask R-CNN Hyperparameter Optimisation

The first part of the hyperparameter optimisation consisted of several grid searches using a variety of settings. These included the choice of backbone, the layers re-trained (after the COCO pre-training), the image resizing, augmentation and mini-mask configurations, as well as the weight decay. The second part of the hyperparameter optimisation consisted of six "efficient grid search" (EGS) setups. The experimental design of these searches were inspired by both grid search [225] and Taguchi's orthogonal arrays [226]. This approach used orthogonal arrays to dictate the style of grid searches by using levels and settings in the same combinations as they would be used in Design of Experiments (DoE). This way, grid searches could be performed on multiple hyperparameters simultaneously whilst minimising the number of experiments. Since there were a large number of hyperparameters tested (24), it was important to organise the experimental design into multiple groups. In this case study, EGS setups each contained anywhere from two to five hyperparameters. After each EGS, the best model was taken forwards which was decided primarily based on mAP, however at times, visual judgement did also come into the decision-making process. The order of initial grid searches, as well as all EGS setups, was chosen strategically in order to maximise benefits such as model performance and research findings. This was conducted by following an order that in most cases, aligned with the order of interaction of each hyperparameter during model training and testing. This was not the case for every single hyperparameter as Mask R-CNN is complex and sometimes has functions running in parallel. Additionally, since the case study was a learning process, some hyperparameters such as "pool size" were incorrectly ordered. The hyperparameters and their settings can be seen in Table 3 in order of experimentation.

Table 3: Hyperparameters and settings for optimisation where "CNN" is convolutional neural network, "ResNet" is Residual Neural Network, "RPN" is region proposal network, "Bbox" refers to "bounding box", "MRCNN" is Mask R-CNN, "NMS" is non-maximum suppression and "RoI" is region of interest

Hyperparameter	Settings
CNN backbone	ResNet50, ResNet101
Layers re-trained	heads, 3+, 4+, 5+, all
Image resizing	none, square1024, square2048, pad1024, pad2048
Augmentation	off, on
Mini-mask	off, on-56, on-112, on-224
Weight decay	0.01, 0.001, 0.0001
EGS 1	
Steps per epoch	250, 500, 1000
Validation steps	10, 25, 50
EGS 2	
Epochs	10, 20, 30
Learning rate	0.0001, 0.001, 0.02
Learning momentum	0.5, 0.9, 0.99
Gradient clip norm	2.5, 5, 10

Continued on next page

Table 3 – continued from previous page

Hyperparameter	Settings
EGS 3	
RPN class loss	1, 2, 3
RPN bbox loss	1, 2, 3
MRCNN class loss	1, 2, 3
MRCNN bbox loss	1, 2, 3
MRCNN mask loss	1, 2, 3
EGS 4	
RPN anchor scales	(16, 32, 64, 128, 256), (32, 64, 128, 256, 512), (64, 128, 256, 512, 1024)
RPN anchor ratios	(0.25, 0.5, 1), (0.5, 1, 2), (1, 2, 4)
RPN anchor stride	1, 2, 3
RPN NMS threshold	0.7, 0.8, 0.9
RPN train anchors per image	128, 256, 512
EGS 5	
Pre-NMS limit	3000, 6000, 12000
Post-NMS training RoIs	1000, 2000, 4000
Post-NMS RoIs inference	500, 1000, 2000
Train RoIs per image	100, 200, 400
RoI positive ratio	0.25, 0.33, 0.5
EGS 6	
Pool size	7, 14, 28
Mask pool size	14, 28, 56
Mask shape	28, 56, 112

4.2.3.1 CNN Backbone

The CNN backbone is a large network consisting of many layers, primarily convolutional layers, hence the name. The backbone is a fundamental part of Mask R-CNN and is used for feature extraction. ResNet50 and ResNet101 are typical choices due to having proven high performance and reasonable computational efficiency in many other tasks [227].

4.2.3.2 Layers Re-trained

The CNN backbone layers are divided up into stages. Since it is already pre-trained on the COCO dataset, the number of layers re-trained using custom data is optional, so the user dictates the depth of fine-tuning. Whilst "heads" only trains the top network layers, "3+" trains Stage 3 and up, "4+" trains Stage 4 and up, "5+" trains Stage 5 and up and "all" re-trains all layers [227].

4.2.3.3 Image Resizing

Image resizing mostly relates to processing efficiency and aspect ratio. Making images smaller reduces computational load, making them square can improve or worsen performance often depending on whether original image sizes are uniform or not, and pad64 ensures a certain image size is obtained whilst preserving image quality through padding [227].

4.2.3.4 Augmentation

Augmentation can help to improve model robustness when the dataset is limited, by processing data in various ways to increase the dataset diversity and number of samples. The augmentation in Mask R-CNN was limited to horizontal flipping by default. It was possible to perform custom augmentation however this was not attempted due to time constraints, and was unlikely to be of any benefit due to the complexity of this particular scenario.

4.2.3.5 Mini-Mask

The normal mask of an object is based on the bounding box of an object and so has edges that touch the bounding box. Alternatively, a mini-mask that wraps more tightly around the object can be used [228]. Whilst it can affect model performance by missing small details, it is more computationally efficient [228]. The default size was (56, 56) and since the ladle was likely to be no smaller than this, other settings chosen were (112, 112) and (224, 224) [227].

4.2.3.6 Weight Decay

Weight decay is also known as L2 regularisation and essentially adds a term to the loss function which is proportional to the sum of the squares of the weights [198]. It does this to prevent overfitting by lessening the effect of large weights [198]. The default setting chosen by the developers of Mask R-CNN was 0.0001 and generally the optimal value of this hyperparameter can vary largely between 0.1 and 0.00001 depending on how much overfitting needs to be counteracted [227]. Therefore, values of 0.0001, 0.001 and 0.01 were used here.

4.2.3.7 Steps Per Epoch

This is the number of batches of samples processed at once by the model during training. After each batch is processed, the model updates weights and biases for each neuron [227]. More steps per epoch takes longer but is less computationally intensive. The default was 1000 and this was time-consuming so it was halved once and twice for two more optimisation settings.

4.2.3.8 Validation Steps

Validation steps is the amount of validation steps performed at the end of each epoch. A higher number improves validation performance but slows down training [227]. The default was 50 steps which was time-consuming and so this was halved to 25 and then reduced further to ten since 12.5 steps were not possible.

4.2.3.9 Epochs

Epochs dictates the number of times the full training dataset is passed through the network and largely influences overall model performance. More epochs give the model a chance to learn more from the data, however too much of this can cause overfitting. The default was 20 epochs which was treated as a moderate setting so ten and 30 epochs were used in the optimisation.

4.2.3.10 Learning Rate

Learning rate (LR) controls the size of the steps the SGD (stochastic gradient descent) optimiser takes during training [227, 229]. A smaller learning rate allows for more granular learning but is generally slower, whilst a larger learning rate speeds up learning but may result in lower performance.

4.2.3.11 Learning Momentum

Learning momentum (LM) adds a fraction of the previous step's vector to the current step, which allows the optimiser to accelerate in directions where there is a consistent reduction in the loss function [230]. The default setting was 0.9 and typical ranges seemed to be from 0.5 to 0.99, and so these two extremities were used also for optimisation [227, 231, 232]

4.2.3.12 Gradient Clip Normalisation

Gradient clip normalisation (GCN) is a technique used to clip gradients to prevent the exploding gradient problem where large gradients cause the model to destabilise during training. The default value is five and a value of one has been known to work [233]. Not much information could be found on this hyperparameter and so 2.5 was used as the smallest experimental setting and ten was used as the largest, effectively halving and doubling the default.

4.2.3.13 Loss Weights

All of the loss weights dictate how much the learning process is optimised for different components of Mask R-CNN. By weighting one loss higher than other, the model will learn to improve in the higher weighted component relative to the lower weighted component. RPN class loss can be used to emphasise whether anchors contain objects or not, RPN bbox loss can emphasise the bounding box coordinates for each anchor, Mask R-CNN class loss can emphasise the accuracy of object categories during classification, Mask R-CNN bbox loss can emphasise the precision of bounding box coordinates of detected objects and Mask R-CNN mask loss can emphasise accuracy of pixel-level mask predictions [229]. The default for all losses was one, and experimental settings of two and three were also used, as this aligned with the number of settings in the orthogonal array without making any weights too extreme [227].

4.2.3.14 RPN Anchor Scales, Ratios and Stride

Scales, ratios and stride dictate the size, aspect ratio and density of anchors used for region proposal respectively [227]. Different values are appropriate depending on the scenario, and so the default settings were used as a moderate level and were halved and doubled to give the other experimental settings. Anchor stride was an exception to this since the default was one, so in this case two and three were used [227].

4.2.3.15 RPN NMS Threshold

This threshold controls the NMS process which effectively filters out overlapping bounding boxes by keeping the best boxes and filtering out boxes which overlap with them [234]. The threshold dictates how much overlap there has to be with a high confidence box for a lower confidence box to be removed [234]. The higher the threshold, the more boxes are kept.

4.2.3.16 RPN Train Anchors Per Image

The number of anchors used in each image to train the RPN are decided using this hyperparameter [227]. The higher this number, the more likely detection performance is to be improved, however it also increases computational load. The default was 256 and this was treated as moderate, giving half and double this as the other two settings for experimentation.

4.2.3.17 Pre-NMS Limit

The pre-NMS limit is the maximum number of proposals possible to use before applying NMS [227]. Depending on the objectness of proposals (likelihood to contain an object), some may be filtered out to leave only top scoring proposals for further processing. This helps improve computational load but if it is too small then it could affect performance. Similarly to other hyperparameters, the default value of 6000 proposals, as well as half and double this, were used for experimentation [227, 229].

4.2.3.18 Post-NMS Training and Inference RoIs

These hyperparameters are similar to the pre-NMS limit, except they are applied after NMS to further reduce proposal redundancy [227, 229]. Whilst training regions of interest (RoIs) are used during training, inference RoIs are used during inference [227, 229]. Both should be selected to achieve a balance of performance and computational efficiency, and so the default values, as well as half and double of them were used.

4.2.3.19 Train RoIs Per Image

This is the number of RoIs finally chosen for training on each image, similarly to post-NMS training RoIs, except this hyperparameter makes the final decision [227, 229]. The default was 200 and so this was halved and doubled.

4.2.3.20 RoI Positive Ratio

This determines the ratio of proposals that are considered objects compared to background. For example, a value of 0.25 assumes 25% of the sampled proposals are overlapping with the ladle and should therefore be used to train the network [227, 229]. The default value was 0.33, whilst 0.25 was used to force the model to focus more on distinguishing the ladle from the background, and 0.5 was used to give an equal balance to the ladle and the background to see if learning from both equally was beneficial [227].

4.2.3.21 Pool Size

Pool size is the size of the feature map extracted from each RoI for classification and bounding box regression [227, 229]. A larger pool size can contain more detailed information but carries a heavier computational load. Again, finding the right balance is the aim, and the default of seven was doubled once to 14, and twice to 28 since the ladle scenario was complex and was likely to benefit from more information [227].

4.2.3.22 Mask Pool Size

The mask pool size is the size of the feature map extracted from each RoI for mask prediction [227, 229]. It is similar to the pool size but is focused on generating masks for instance segmentation, rather than classification or object detection [227, 229]. The default of 14 was doubled to 28, and again to 56 since the ladle scenario was complex and was likely to benefit from more information [227].

4.2.3.23 Mask Shape

The mask shape is the resolution of the binary output mask for each object and defines the level of detail provided for final mask predictions [227, 229]. Again, the optimal size is a trade-off between performance and computational load. The default was (28,28) and since the mask pool size was doubled twice for experimentation, the mask shape was set to align with this [227].

4.2.4 Kalman Filter Tracking

Following Mask R-CNN optimisation, efforts were made to enhance the model's tracking capabilities over sequential frames by integrating edge refinement and stabilisation through

Kalman filtering. The Kalman filter is a computational method used for predicting the future trajectory of an object, and was used to associate masks between sequential frames to enhance the model's understanding of the ladle shape as it moved throughout videos (as opposed to individual frames). This is inspired by the likes of CNN-LSTM models (where "LSTM" abbreviates "long short-term memory" and DeepSORT (Deep Simple Online and Real-Time Tracking) reported in the literature and reviewed in Section 2.4 of Chapter 2 [86, 87, 88, 90]. Kalman filters utilise a set of five recursive equations to forecast unknown variables of a system.

The algorithm firstly involves an initial estimation or prediction of the system state, which is then refined based on feedback using data that may contain noise. The prediction phase uses Equation (10) and Equation (11) to derive the current state and error covariance estimates. This process results in what is known as the *a priori* estimate for the next time step. In the update phase, covered by Equations (12)-(14), a new measurement is included in this *a priori* estimate, leading to an updated, or *a posteriori*, estimate [68]. Further details of this process are given below.

In Equation (10), the first predict step element, which is the *a priori* next state estimation ($\hat{x}^-_{k+1}$), is calculated by adding the product of the state transition matrix (A_k), and the *a priori* estimate for the current step ($\hat{x}_k$), to the product of the control input matrix and the control vector (Bu_k) [68].

$$\hat{x}^-_{k+1} = A_k\hat{x}_k + Bu_k \tag{10}$$

Next, as shown in Equation (11), the error covariance *a priori* estimate for the next step (P^-_{k+1}), is calculated by the adding the product of the transition matrix and the current error covariance *a posteriori* estimation (P_k), to the transition matrix and process noise (Q_k) [68].

$$P^-_{k+1} = A_kP_k + A^T_k + Q_k \tag{11}$$

The first part of the update step is shown in Equation (12), where the optimal Kalman gain (K_k), is calculated using the current error covariance *a priori* estimate (P^-_k), the Jacobian matrix (H_k), and the measurement error covariance (R_k) [68].

$$K_k = P^-_k H^T_k(H_kP^-_kH^T_k + R_k)^{-1} \tag{12}$$

The *a posteriori* state estimate ($\hat{x}_k$), is calculated in Equation (13) using the *a priori* state estimate, the Kalman gain, and the actual measurement (z_k) [68].

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H_k\hat{x}_k^-) \tag{13}$$

The error covariance *a posteriori* estimate (P_k), is calculated in Equation (14) using an identity matrix (I), the Kalman gain, the Jacobian matrix, and the current step *a priori* error covariance [68].

$$P_k = (I - K_kH_k)P_k^- \tag{14}$$

The concept behind using Kalman filtering for this case study, was to assign a separate filter to each point in a set of points, which were sampled from the edges of the mask initially predicted by Mask R-CNN. The hypothesis was that each filter would process the pixel coordinates of its corresponding edge point from successive frames as input. The expectation was that the Kalman filter's updated predictions would refine the initial predictions made by Mask R-CNN. When these refined point predictions were used collectively, they were anticipated to enhance the overall mAP whilst stabilising the mask over sequential frames so that it became less erratic.

To implement this, edge pixels of the original Mask R-CNN mask were identified through image processing using OpenCV, and their coordinates were stored in a list. A fixed number of these points (32) were then sampled and input into the Kalman filters. Each filter was responsible for tracking an individual point, leading to a sequence of multi-point predictions that collectively comprised a refined mask.

To mitigate the jaggedness that naturally occurred from Kalman filter predictions, a convex hull (see Figure 31) was constructed using the gift wrapping algorithm on the predicted points [235]. The convex hull was designed to be the most efficient boundary that enclosed all points whilst maintaining all interior angles below 180° [236]. To ensure the convex hull was formed by a smooth set of points, K-Nearest Neighbours (KNN) was used to preserve the points closer together (which typically sat along the ladle edge) whilst excluding the surrounding points that were less accurate. The KNN background subtraction (BGS) algorithm discussed later is based on K-Nearest Neighbours but to be clear, it is not the exact same approach.

After establishing the convex hull, the state transition matrices of the Kalman filters were

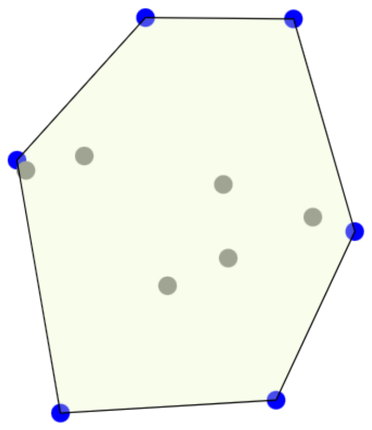


Figure 31: Example of a convex hull

refined through a series of experiments. The state transition matrix is a crucial component in Kalman filtering that connects the current state of the system, with its state at the next time step. It essentially guides the transition from one state to another [68].

In this particular application, the state transition matrix, with dimensions of 4x4, was used to model the movement of points across frames. The fact that the problem was in 2D meant that the matrix accounted for the x and y positions and velocities of the points, each represented by a separate row in the matrix. The elements within each row of this matrix correspond to variables in the 2D kinematic equations and so they can be adjusted based on available information on the system [237]. The essence of kinematics is to predict the future position and velocity of an object based on its current state and motion dynamics. In this case, acceleration is assumed to be zero and velocity is assumed to be constant.

Initially, the optimal values for the state transitions were unknown. Therefore, experimental adjustments were made, starting by incrementally altering the default value for x_{k-1} from one to ten, in attempt to align them with what was observable in the ladle footage. After each increment, the impact on the mAP was assessed, and the value yielding the highest mAP was retained. This methodology was then replicated for y_{k-1}, whilst keeping the previously optimised value for x_{k-1}. The same procedure was applied to the remaining two variables to refine the transition matrix further.

The default state transition matrix A_k is present in Equation (10) and written in full form in Equation (15) [237]. It is a matrix representation of the constant velocity model in kinematics. Elements that are equal to one are multipliers for terms corresponding in the *a priori* estimate for the current step also known as the state vector, $\hat{x}_k$ (see Equation (16)). Whilst elements in A_k carry over system information from one time step to the next, (including the Δt elements which account for the effect of velocity on position over the given time interval), the $\hat{x}_k$ elements represent the x position, y position, x velocity and y velocity in descending order of Equation (16). Therefore, the elements of A_k can be modified to align with what is observed in the ladle footage, to fine-tune Kalman filter performance.

$$A_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$$\hat{x}_k = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} \tag{16}$$

4.3 Results and Discussion

This section presents and discusses results obtained from experimentation described in Section 4.2, which begins with the results of denoising data via contrast enhancement, before moving onto Mask R-CNN application. Following the development and optimisation of Mask R-CNN, this section investigates the effectiveness of Kalman filtering in enhancing the accuracy of the predicted segmentation masks. Additionally, the results of experimenting with different background subtraction algorithms are included.

4.3.1 Denoising Data

In Figure 32 and Figure 33, the left column contains “good contrast” images, the middle column contains “moderate contrast” images and the right column “poor contrast”. Figure 32(a) is after converting to greyscale but before denoising, and the remaining rows each show the results of the various denoising techniques described in Section 4.2.1. The aim of denoising was to enhance image contrast to ensure there was a more distinct ladle visible within the data so the model could learn and predict with more ease. Important details to consider when evaluating the results in Figure 32 and Figure 33, are clarity of the ladle edges and the brightness of the furnace flame.

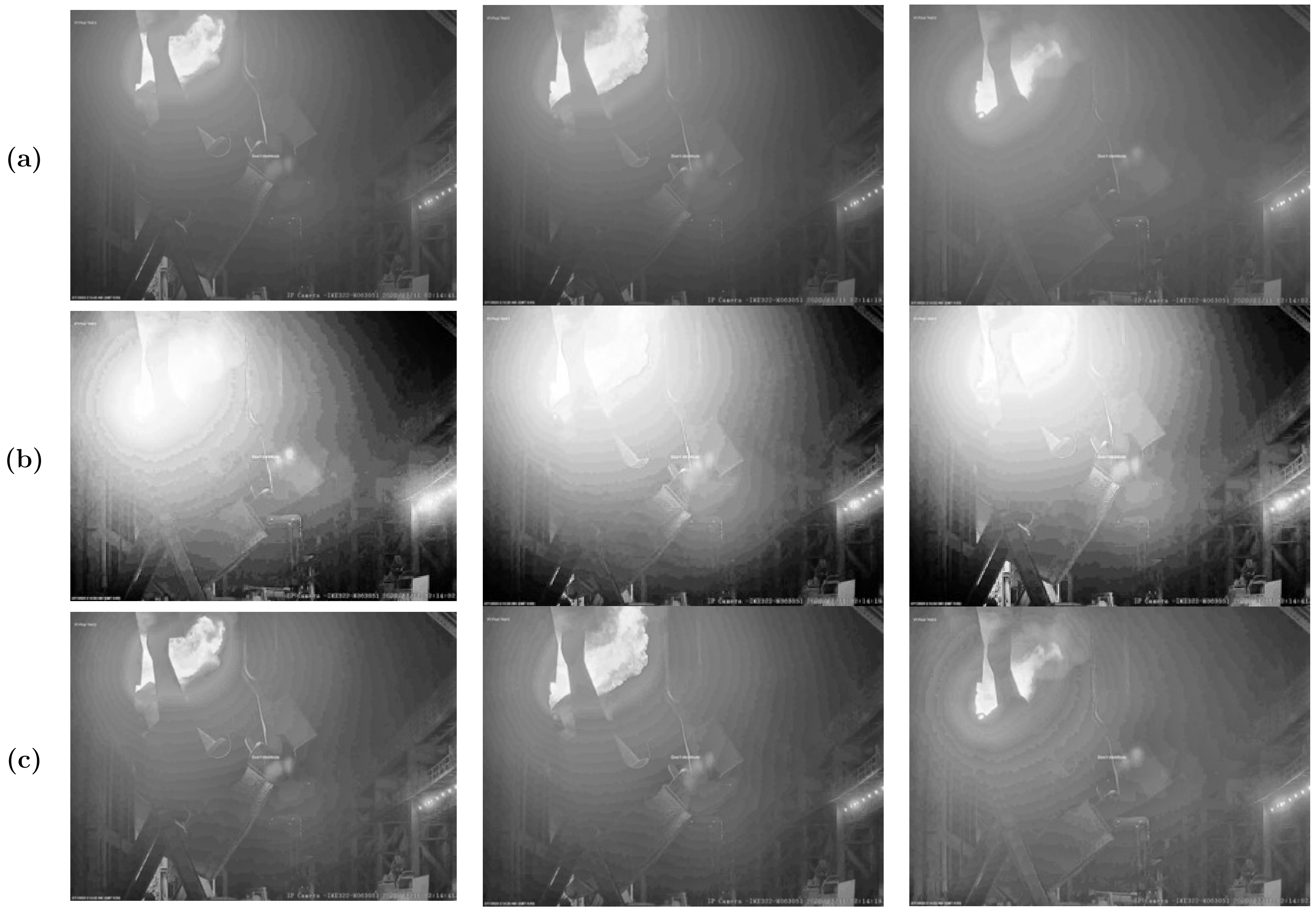


Figure 32: Good (left), moderate (center), and poor (right) quality contrast frames for (a) original greyscale, (b) HE, and (c) CLAHE

(a)

(b)

(c)

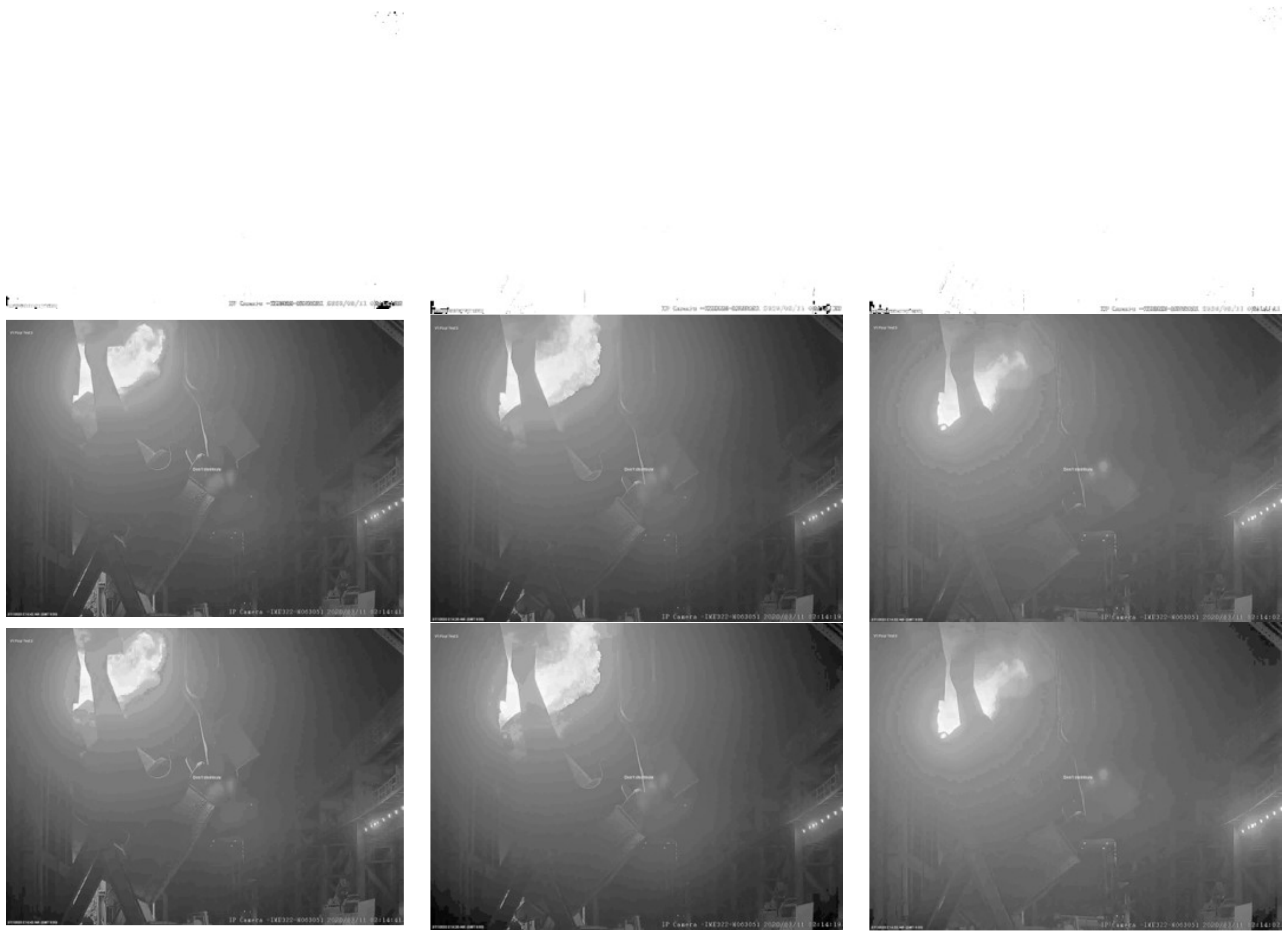


Figure 33: Good (left), moderate (center), and poor (right) quality contrast frames for (a) BPBHE (Note that these are supposed to be white), (b) Gaussian BPDFHE, and (c) Triangular BPDFHE

HE results in Figure 32(b) show a significant improvement with regards to ladle edge visibility compared to the greyscale images. This is especially evident in the poor contrast image, where the top-right quarter of the ladle changes from almost completely invisible to clearly lighter than the background. However, with regards to the furnace flame brightness, lens flare is significantly increased. Looking at the poor image, despite the overall change in image aesthetic, increased lens flare does not appear to be distorting any important aspects of the image. Contrarily, the good image could be considered worse than before HE application, as the ladle edges, particularly in the top-right quarter, are less distinct than before.

Standard HE smooths out the histogram of the intensity values for each pixel, effectively reducing extremities, and the bright furnace flame is a large extremity compared to the surrounding dark pixels. Therefore, since HE equalises over the whole image at once, the brightness of the furnace flame is distributed over the rest of the image. This is effective for the poor image because it is extremely dark everywhere except the flame location, so this is a good “source of light”. However, in the poor image, there is already some clarity present in the image before HE. Dispersing the extreme brightness from the flame to the already relatively bright pixels causes some extreme brightness to occur farther away from the flame, which includes pixels constituting the ladle edges.

CLAHE results in Figure 32(c) equalises multiple histograms representing divided “tiles” of the image to prevent over-enhancement of noise in homogeneous regions, which is the issue with the HE results. Therefore, a significantly improved and more balanced performance across varying contrast qualities is observable in the CLAHE results.

Brightness Preserving Bi-Histogram Equalisation (BPBHE) is designed to preserve the mean brightness of an image, and in this case, Figure 33(a) shows application has resulted in bright white images for all three contrast qualities. This indicates that the mean intensity value is high (higher intensity pixels are whiter), so the higher intensity sub-image must contain extremely bright pixels and upon equalisation, produces a white image. Meanwhile, the lower intensity sub-image contains some moderately dark pixels and some very bright pixels. When the sub-images are combined, overall intensity values are very high, which explains the extreme brightness that can be seen.

Results for the Brightness Preserving Dynamic Fuzzy Histogram Equalisation (BPDFHE) Gaussian variant are shown in Figure 33(b), and results for the triangular variant are shown in Figure 33(c). Both variants had an insignificant effect on the images. The brightness surrounding the flame was re-distributed, but no edges were made clearer. Despite this being a brightness-preserving variant like BPBHE, there is a great difference in the output

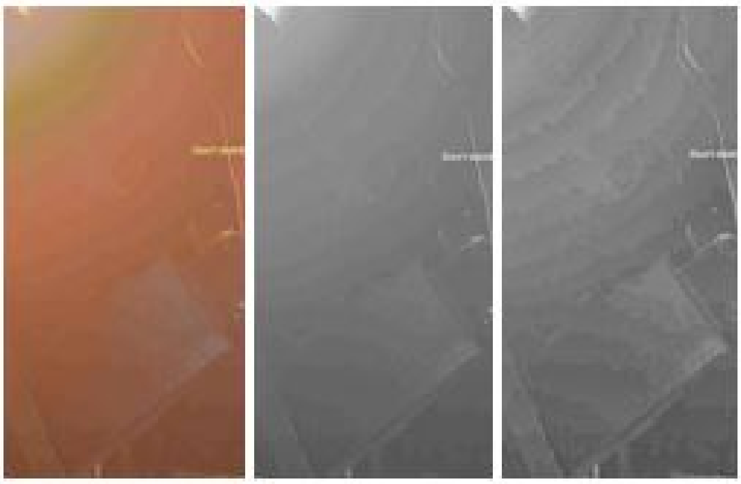


Figure 34: Figure showing the ladle edge in the original video (left), greyscale video (center) and denoised video (right)

of both methods. BPDFHE uses the fuzzy domain to process grey value inexactness better than in BPBHE, combined with fuzzy statistics that prevent random fluctuations in the histogram (such as the small black areas in BPBHE output).

The best technique appeared to be CLAHE, which aligned with the literature presented in Section 2.2 of Chapter 2. Figure 34 shows clearly the difference in edge visibility before and after CLAHE.

The performance of Mask R-CNN when trained and tested on denoised and original data was compared. This followed the format of the hyperparameter optimisation to analyse how CLAHE affected mAP with different model configurations, rather than just looking at one specific configuration. The first part of the hyperparameter optimisation was to test and evaluate the different in Mask R-CNN performance when different backbones (ResNet50 and ResNet101) were used, and when different numbers of re-trained layers were used. As will be discussed later, this led to five experiments for each backbone which included re-training of just the network heads, re-training of layers comprising Stage 3 and up, Stage 4 and up, Stage 5 and up, and finally re-training all layers of Mask R-CNN. To evaluate the effectiveness of applying CLAHE, ResNet101 with the five aforementioned re-training approaches was used. The results of this evaluation are shown in Table 4.

The results in Table 4 show that in four out of five cases, using CLAHE significantly improved mAP. Also, the highest mAP of all experiments by a significant amount, used CLAHE and model re-training from Stage 3 and up. However, when re-training all layers, using the original data with no contrast enhancement resulted in better performance, and

Table 4: mAP comparison pre-optimisation for original and denoised data

Layers Re-Trained	mAP_{COCO} (Original Frames)	mAP_{COCO} (CLAHE)
Heads	0.004	**0.112**
3+	0.008	**0.276**
4+	0.056	**0.131**
5+	0.027	**0.158**
All	**0.197**	0.113

this was the second highest of all experiments reported in Table 4. Therefore, results suggest CLAHE is effective at improving model performance in environments with poor lighting.

4.3.2 Mask R-CNN Initial Optimisation

Table 5 shows results from the first set of experiments based on the choice of CNN backbone (ResNet50 or ResNet101) and the amount of layers re-trained after pre-training on the COCO set (see Section 4.2.3.2 for explanation on stages and re-trained layers). In this chapter, both epoch and total training times are reported. Whilst epoch time provides insight into the efficiency of each training iteration, total train time offers a comprehensive view of the overall time consumption of each experiment. This highlights time and compute expense for each configuration.

The results show that firstly, ResNet101 takes longer to train than ResNet50 which is because it has approximately twice the number of layers. Also, across both ResNets, there was a clear trend of less re-training requiring less time, which makes sense since the data has to be passed through a smaller number of layers. In terms of losses, the training losses of ResNet101 are generally lower (better) than the ResNet50 training losses, which is likely due to the additional complexity that ResNet101 can handle over ResNet50. However, the validation losses for both models are high, which combined with the much lower training losses indicates that both models overfitted, which typically leads to poorer generalisation on unseen data. Since two different ResNets were re-trained to various extents, the models are likely not the issue here, but rather the training strategy or the data. The best epoch from each experiment was used and so any overtraining past the optimal point is not included in Table 5, which significantly reduces the likelihood of the low performance being due to the training strategy. Therefore, quality or quantity of the data are responsible. Data was labelled carefully but was limited in size, indicating that model results could be improved with more data. The large difference in training and validation losses also suggest that the validation set is unrepresentative of the training and testing sets. This could be due to including outliers in the validation set, or not having a large enough dataset to properly validate the model. Whilst there were no strong trends with regards to mAP, ResNet101 tended to have higher mAP values than ResNet50, which aligns with the better training losses and is due to ResNet101's better capabilities at learning complex data.

Overall the experimental results in Table 5 indicate the dataset was lacking in size. As shown throughout this chapter, the model was still developed to achieve reasonable performance through hyperparameter optimisation. In Table 5, using ResNet101 whilst re-training the third stage and up resulted in significantly better performance than other setups (0.276 mAP, shown in bold) and was therefore taken forward to the next stage.

Table 5: Backbone and re-trained layers experimental results

Backbone	Layers	Epoch Time (s)	Train Time (s)	Train Loss	Val. Loss	Inf. Time (s)	Precision	Recall	$\mathrm{mAP}_{\mathrm{COCO}}$
ResNet50	All	1698	33950	0.587	1.996	0.440	0.102	0.102	0.092
ResNet50	3+	1683	33657	0.586	2.440	0.434	0.153	0.153	0.149
ResNet50	4+	1656	33126	0.561	2.646	0.331	0.131	0.131	0.125
ResNet50	5+	1543	30855	0.339	3.078	0.392	0.109	0.109	0.107
ResNet50	Heads	1339	26781	0.606	2.177	0.349	0.129	0.129	0.128
ResNet101	All	1971	39416	0.480	2.909	0.365	0.128	0.128	0.113
ResNet101	3+	1834	36671	0.474	2.447	0.361	0.257	0.276	**0.276**
ResNet101	4+	1816	36319	0.479	2.547	0.447	0.127	0.133	0.131
ResNet101	5+	1622	32430	0.469	2.693	0.367	0.159	0.159	0.158
ResNet101	Heads	1407	28137	0.528	2.325	0.350	0.112	0.112	0.112

Table 6 shows results from the experiments based on image resizing settings. Style settings included none, square, pad64 and crop, and dimension settings allowed more customisation. These settings are described in Section 4.2.3.3. Results show that the pad64-1024 resizing yielded the best results in terms of precision, recall and mAP. This was followed by no resizing, suggesting that resizing the image to make it square was detrimental to performance. This is because the ladle shape is unique and quite complex, and therefore forcing the shape to ensure the overall image is square, especially in the harsh environment, compromises feature extraction. No image resizing was most efficient in terms of inference time, which is because it requires the least processing, whilst pad64-1024 was the second slowest after square-2048, which was not due to any obvious reason since it was more computationally lightweight than pad64-2048. The trend in inference times was similar to the trend in training times, suggesting the pad64-1024 configuration was less efficient than other configurations, however the difference is minimal. pad64-1024 had the lowest training loss and the highest mAP during testing, however it also had the highest validation loss and the largest difference between training and validation losses. As mentioned in Table 5 discussion, a larger validation loss often suggests poorer generalisation on new data, and a large difference between training and validation losses indicates overfitting, which also leads to poorer generalisation.

Data points collected from no image resizing are from the best experiment in Table 5 for efficiency, since the optimisation process was broad and configurations from Table 5 did not change at the beginning of the Table 6 experiments. This was continued throughout the optimisation process. In Table 6, using pad64-1024 image resizing resulted in the best performance (0.282 mAP, shown in bold) and was therefore taken forward to the next stage.

Table 6: Image resizing experimental results

Image Resizing	Epoch Time (s)	Train Time (s)	Train Loss	Val. Loss	Inf. Time (s)	Precision	Recall	mAP_{COCO}
None	1392	27843	0.472	0.985	0.361	0.257	0.276	0.276
Square-1024	1350	26997	0.479	2.046	0.368	0.116	0.116	0.105
Square-2048	1433	28668	0.453	1.094	0.516	0.208	0.208	0.204
Pad64-1024	1364	27284	0.443	1.620	0.410	0.304	0.304	**0.282**
Pad64-2048	1290	25804	0.482	0.912	0.375	0.178	0.178	0.173

Table 7 shows results from the third set of experiments which were based on the augmentation settings. Evidently, mAP, precision and recall were significantly higher when augmentation was not applied. Similarly to the image resizing discussion, this could be due to the ladle shape complexity and harsh lighting conditions observable in the footage, meaning only the original data was capable of containing sufficiently high quality features for the model to learn from. Interestingly, whilst the non-augmented data experiment had a large difference between training and validation losses as already discussed regarding image resizing, the augmented data experiment had a significantly smaller gap, smaller than any of the image resizing experiments and small enough to be more typical of successful model training. For augmentation, there was only about a 10% increase in loss from training to validation, suggesting the model found slightly more difficulty in predicting when it was being challenged rather than taught through labels. Since the performance was still poorer than when using no augmentation, this could just be by chance. However, the difference in loss gaps between the two experiments, and the nature of augmentation procedures, suggests that training and validation sets were originally too dissimilar, whilst training and testing sets were more similar. Therefore, when augmentation was applied to training and validation sets, it made them more similar resulting in a much smaller loss gap, however the training data became much less similar to the testing set, resulting in lower overall performance. These results emphasise the benefits of augmentation whilst also suggesting that the size and distribution of the dataset used in this optimisation were insufficient. In Table 7, using no augmentation resulted in the best performance (0.282 mAP, shown in bold) and was therefore taken forward to the next stage of the optimisation.

Table 7: Augmentation experimental results

Augmentation	Epoch Time (s)	Train Time (s)	Train Loss	Val. Loss	Inf. Time (s)	Precision	Recall	mAP_{COCO}
FALSE	1364	27284	0.443	1.620	0.410	0.304	0.304	**0.282**
TRUE	1302	26032	0.586	0.648	0.301	0.123	0.123	0.103

Table 8 shows results from mini-mask experiments, which show that model performance decreased significantly with both the application of a mini-mask, as well as increasing mini-mask size. Furthermore, the use of a mini-mask added a large computational strain, shown by the (112, 112) mini-mask training failing roughly half way through and the (224, 224) mini-mask failing completely, due to out-of-memory (OOM) errors. The purpose of mini-masks is to improve memory efficiency, however increasing mini-mask size still increases memory requirements. Therefore, these results suggest that somewhere between (56, 56) and (112, 112), the increased memory usage from the size of the mini-mask begins to outweigh memory saved. There were no trends in terms of training or inference times. In Table 8, using no mini-mask resulted in the best performance (0.282 mAP, shown in bold) and was therefore taken forward to the next stage.

Table 8: Mini-mask experimental results

Mini-Mask Size	Epoch Time (s)	Train Time (s)	Train Loss	Val. Loss	Inf. Time (s)	Precision	Recall	mAP_{COCO}
NA	1364	27284	0.443	1.620	0.410	0.304	0.304	**0.282**
56,56	1230	24605	0.479	1.362	0.330	0.187	0.187	0.156
112,112	1242	9933 (OOM)	0.011	3.206	0.485	0.043	0.043	0.041
224,224	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM

Table 9 shows results from weight decay experiments, which show mAP was much higher when weight decay was lower. In fact, when weight decay was 0.01 and 0.001, results were mostly the same. Meanwhile, when weight decay was 0.0001 performance metrics were much better, suggesting a lower weight decay was more effective and at some point between 0.0001 and 0.001, there was a cut-off where increasing weight decay no longer affected performance. Interestingly, the difference between training and validation losses increased with decreasing weight decay, suggesting a trade-off between overfitting risk and model performance, which is common in computer vision. There were no trends in terms of training or inference times. In Table 9, using a weight decay value of 0.0001 resulted in the best performance (0.282 mAP, shown in bold) and was therefore taken to the next stage.

Table 9: Weight decay experimental results

Weight Decay	Epoch Time (s)	Train Time (s)	Train Loss	Val. Loss	Inf. Time (s)	Precision	Recall	mAP_{COCO}
0.01	1255	25107	0.464	0.994	0.367	0.025	0.025	0.025
0.001	1209	24184	0.448	1.553	0.358	0.025	0.025	0.025
0.0001	1364	27284	0.443	1.620	0.410	0.304	0.304	**0.282**

4.3.3 Mask R-CNN Efficient Grid Search Optimisation

The initial optimisation included the backbone, the layers re-trained after pre-training, and settings for image resizing, augmentation, mini-masking, and weight decay. The second part of the optimisation consisted of the six "efficient grid search" (EGS) setups mentioned in Section 4.2.3. As previously mentioned, the design of these searches were inspired by both grid search [225] and Taguchi's orthogonal arrays [226]. For each search, discussion is supported by a table of results and an interaction plot which shows how changing different hyperparameters affected performance.

4.3.3.1 Efficient Grid Search 1

Table 10 shows experimental results from EGS 1 which included steps per epoch and validation steps (definitions are in Section 4.2). The letters "P" and "R" represent precision and recall, respectively. Training time generally increased with more steps per epoch and more validation steps. This is because more steps per epoch means more batches of data processed during one epoch, and more validation steps means more computations are performed at the end of each epoch. There were also a few outliers which were likely due to inherent variability in system performance. Factors such as background processes and small variations in hardware performance can cause these fluctuations.

The interaction plot in Figure 35 suggests the impact of the two hyperparameters on mAP. As shown, there was a tendency for mAP to be higher with more steps per epoch, however this effect was relatively small and could be due to random variability present in neural networks. Additionally, the validation steps appeared to have a more significant impact where using 50 over 25 or ten correlated with a significant increase in mAP.

The fact that the default values (1000 steps per epoch and 50 validation steps) gave the best performance overall, suggests that the developers of Mask R-CNN already optimised these settings either for another use-case or for general performance which seemingly aligns with this study. This was found with other configurations too.

Table 10: EGS 1 experimental results

Exp. No.	Steps Per Epoch	Val. Steps	Epoch Time (s)	Train Time (s)	Train Loss	Val. Loss	Inf. Time (s)	P	R	mAP$_{\text{COCO}}$
1	250	10	1244	21153	0.487	1.642	0.325	0.079	0.082	0.076
2	250	25	1271	24141	0.024	2.327	0.326	0.043	0.044	0.044
3	250	50	1382	27633	0.461	0.948	0.427	0.090	0.093	0.078
4	500	10	1264	25281	0.459	0.924	0.360	0.122	0.122	0.119
5	500	25	1327	26540	0.457	1.168	0.343	0.021	0.021	0.018
6	500	50	1368	27361	0.033	1.528	0.369	0.242	0.242	0.204
7	1000	10	1465	29294	0.147	1.292	0.457	0.064	0.064	0.048
8	1000	25	1283	25655	0.460	1.388	0.449	0.064	0.064	0.048
9	1000	50	1364	27284	0.443	1.629	0.410	0.304	0.304	**0.282**

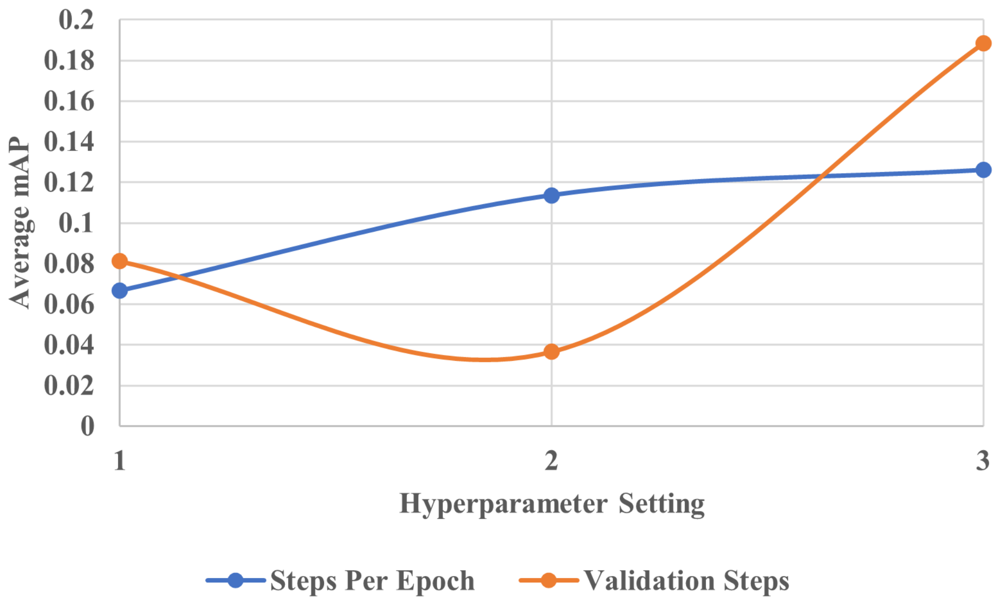


Figure 35: Interaction plot for EGS 1 which shows the average mAP score achieved for each setting of the “steps per epoch” and “validation steps” hyperparameters

4.3.3.2 Efficient Grid Search 2

Table 11 shows experimental results from EGS 2 which included epochs, learning rate, learning momentum and gradient clip normalisation. Differently to previous experiments, the best model was not taken forward from the previous set of experiments due to the Taguchi-inspired orthogonal arrays not naturally aligning all of the previous configurations for any one experiment. Unsurprisingly, the overall training time approximately double and tripled when increasing epochs from ten to 20 and 30 respectively. Meanwhile, this appeared to have no positive effect on overall performance, suggesting that ten epochs was enough to maximise model performance. This was further supported by the fact that for all sets of experiments up until this point, the best epoch was between the first and the tenth.

An interaction plot is shown in Figure 36 which visualises the impact of different hyperparameters on the mAP. The graph suggests that a higher learning momentum generally resulted in a higher mAP and the lowest learning rate resulted in higher mAP values. Meanwhile, the graph suggests that the gradient clip normalisation had the largest impact on mAP but with no obvious trend. A lower learning rate allows the model to learn more gradually, therefore learning more nuanced features, whilst a higher learning momentum helps to avoid local minima. However, all hyperparameters in EGS 2 would benefit from more thorough analysis, possibly in separate experiments for a more in-depth analysis on the effects of each one individually.

Based on this analysis, it is reasonable to state that these results are fairly inconclusive. Therefore, since the best model from EGS 1 used the default values of 0.001, 0.9 and five for LR, LM and GCN respectively, these settings were taken forward. This aligns with previous discussion stating that Mask R-CNN developers likely optimised many hyperparameters beforehand.

Table 11: EGS 2 experimental results

Exp. No.	Epochs	LR	LM	Grad. Clip Norm.	Epoch Time (s)	Train Time (s)	Train Loss	Val. Loss	Inf. Time (s)	P	R	mAP_{COCO}
1	10	0.02	0.5	2.5	1404	14038	0.467	1.055	0.397	0.076	0.076	0.075
2	10	0.001	0.9	5	1320	13198	0.463	1.300	0.367	0.036	0.036	0.028
3	10	0.0001	0.99	10	1411	14111	0.475	0.902	0.522	0.238	0.238	0.192
4	20	0.02	0.9	10	1399	27983	0.462	1.383	0.316	0.229	0.229	**0.198**
5	20	0.001	0.99	2.5	1336	26721	0.483	1.312	0.313	0.229	0.229	**0.198**
6	20	0.0001	0.5	10	1420	28393	0.482	1.368	0.435	0.048	0.048	0.040
7	30	0.02	0.99	5	1311	39329	0.494	1.034	0.345	0.060	0.060	0.052
8	30	0.001	0.5	10	1276	38286	0.444	1.703	0.334	0.096	0.096	0.075
9	30	0.0001	0.9	2.5	1278	38339	0.469	1.109	0.358	0.160	0.160	0.147

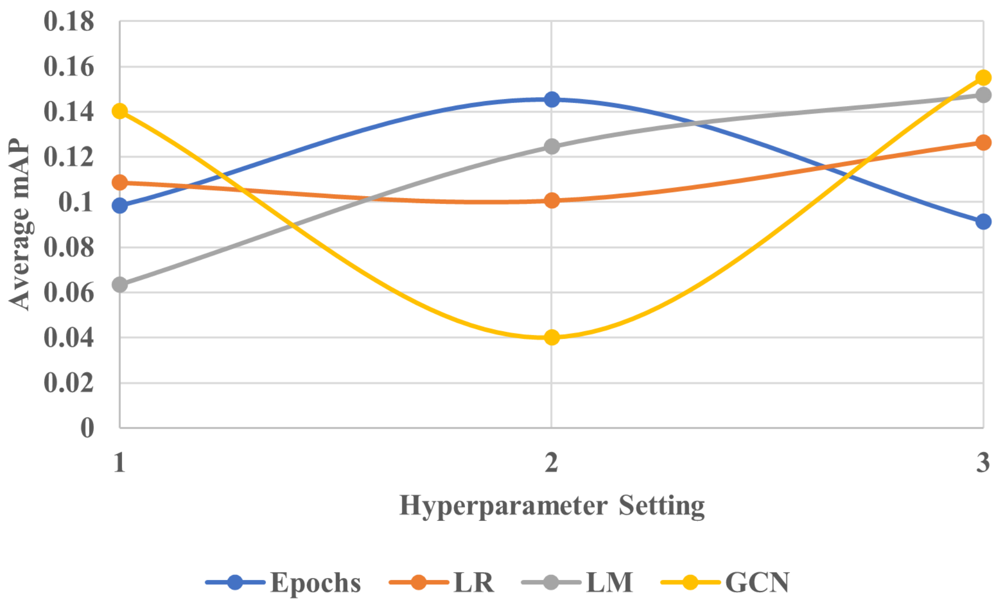


Figure 36: Interaction plot for EGS 2 which shows the average mAP score achieved for each setting of the “epochs”, “learning rate” (LR), “learning momentum” (LM) and “gradient clip normalisation” (GCN) hyperparameters

4.3.3.3 Efficient Grid Search 3

Table 12 shows experimental results from EGS 3 which included RPN class loss, RPN bounding box loss, Mask R-CNN class loss, Mask R-CNN bounding box loss and Mask R-CNN mask loss. There are a few particularly notable aspects of Table 12, the first being that the epoch and training times are a magnitude of order smaller than in previous experiments. Since the number of experiments in this grid search were much more than previous grid searches and each experiment was taking 20000-30000 seconds to train (approximately 5.5 to 8.3 hours), whilst EGS 1 results indicated that steps per epoch did not appear to have a significant impact on model performance, it was decided that the steps per epoch would be reduced to 100. In hindsight, this could have been attempted straight after EGS 1, however the need for drastically improving time consumption did not become apparent until larger sets of experiments were needed such as in EGS 3. The interaction plot for EGS 1 actually showed a slight favour to higher steps per epoch when considering mAP, and whilst this was not significant, it was decided that it was best to avoid decreasing it unless absolutely necessary. The second notable aspect is that varying loss weights had a significant impact on validation loss, as well as mAP. In terms of validation loss, it can be seen that the default settings in the first experiment (1, 1, 1, 1, 1) aligned with the best experiment from EGS 1. This experiment had significantly higher validation loss than any other experiment, indicating that for this case study, an equal loss weighting across classification, localisation and segmentation was undesirable. Whilst the first experiment had the highest validation loss by far, it lied somewhere in the middle of the distribution of mAP results. This meant that many other loss weight experiments performed much better.

An interaction plot is shown in Figure 37 which visualises the impact of different hyperparameters on the mAP. The graph suggests that lower RPN bbox loss and MRCNN class loss weights resulted in higher mAP performance. Lower RPN bbox loss being better could mean that the model performs better when it focuses less on perfecting initial region proposal boxes due to the later MRCNN localisation stage refining them anyway. Lower MRCNN class loss weighting improving mAP could be because the only class (other than the background) is the ladle, meaning there is not much need for the model to focus on improving this aspect of prediction. The MRCNN bbox seemed to have the largest effect on the mAP but with no obvious trend, so it would be useful to explore this hyperparameter on its own. It could have the largest effect because the refinement of bounding boxes is crucial for overall prediction, as opposed to RPN localisation which only needs to give initial approximations. MRCNN mask loss had the second largest effect, which is unsurprising since the mAP is directly based on the accuracy of the final mask prediction. These results actually emphasise just how much the final mask prediction relies on the underpinning localisation aspect (MRCNN bbox loss). Finally, according to Figure 37, the RPN class loss was the third most impactful weight closely following the mask loss in terms of impact. This could be because the initial RPN stage is crucial for distinguishing between the ladle and the background by learning which bounding boxes contain the ladle. Overall, the 24th experiment using loss weights of (3, 2, 1, 3, 3) produced the highest mAP (0.438) and was therefore chosen as the best configuration for these hyperparameters.

Table 12: EGS 3 experimental results

Exp. No.	RPN Class Loss	RPN Bbox Loss	MRCNN Class Loss	MRCNN Bbox Loss	MRCNN Mask Loss	Epoch Time (s)	Train Time (s)	Train Loss	Val. Loss	Inf. Time (s)	P	R	mAP_{COCO}
1	1	1	1	1	1	1364	27284	0.443	1.620	0.410	0.304	0.304	0.282
2	1	1	1	1	2	216	2161	0.399	0.669	0.503	0.382	0.424	0.424
3	1	1	1	1	3	218	2182	0.371	0.649	0.460	0.423	0.423	0.363
4	1	2	2	2	1	220	2201	0.454	0.696	0.451	0.323	0.328	0.273
5	1	2	2	2	2	212	2121	0.396	0.684	0.475	0.077	0.078	0.078
6	1	2	2	2	3	217	2166	0.386	0.679	0.471	0.284	0.291	0.291
7	1	3	3	3	1	216	2163	0.507	0.608	0.455	0.283	0.316	0.316
8	1	3	3	3	2	211	2108	0.369	0.646	0.472	0.364	0.364	0.354
9	1	3	3	3	3	216	2161	0.377	0.700	0.455	0.198	0.198	0.165
10	2	1	2	3	1	212	2121	0.425	0.781	0.466	0.317	0.326	0.286
11	2	1	2	3	2	208	2084	0.447	0.697	0.466	0.418	0.419	0.419
12	2	1	2	3	3	212	2123	0.391	0.849	0.460	0.261	0.290	0.290
13	2	2	3	1	1	216	2164	0.358	0.767	0.479	0.320	0.335	0.335
14	2	2	3	1	2	219	2188	0.516	0.668	0.476	0.131	0.156	0.156
15	2	2	3	1	3	216	2160	0.424	0.678	0.461	0.214	0.231	0.231
16	2	3	1	2	1	220	2203	0.380	0.796	0.449	0.111	0.111	0.101
17	2	3	1	2	2	222	2224	0.491	0.678	0.480	0.094	0.123	0.123
18	2	3	1	2	3	223	2229	0.413	0.711	0.477	0.229	0.264	0.264
19	3	1	3	2	1	227	2265	0.400	0.603	0.470	0.142	0.143	0.143
20	3	1	3	2	2	223	2228	0.387	0.705	0.462	0.145	0.150	0.150
21	3	1	3	2	3	221	2214	0.410	0.691	0.461	0.294	0.295	0.295
22	3	2	1	3	1	222	2217	0.491	0.678	0.441	0.442	0.446	0.394
23	3	2	1	3	2	225	2254	0.429	0.618	0.456	0.241	0.245	0.245
24	3	2	1	3	3	225	2253	0.408	0.812	0.453	0.443	0.446	**0.438**
25	3	3	2	1	1	218	2175	0.433	0.674	0.483	0.375	0.429	0.429
26	3	3	2	1	2	219	2187	0.606	0.733	0.460	0.163	0.189	0.189
27	3	3	2	1	3	219	2193	0.465	0.634	0.473	0.236	0.274	0.274

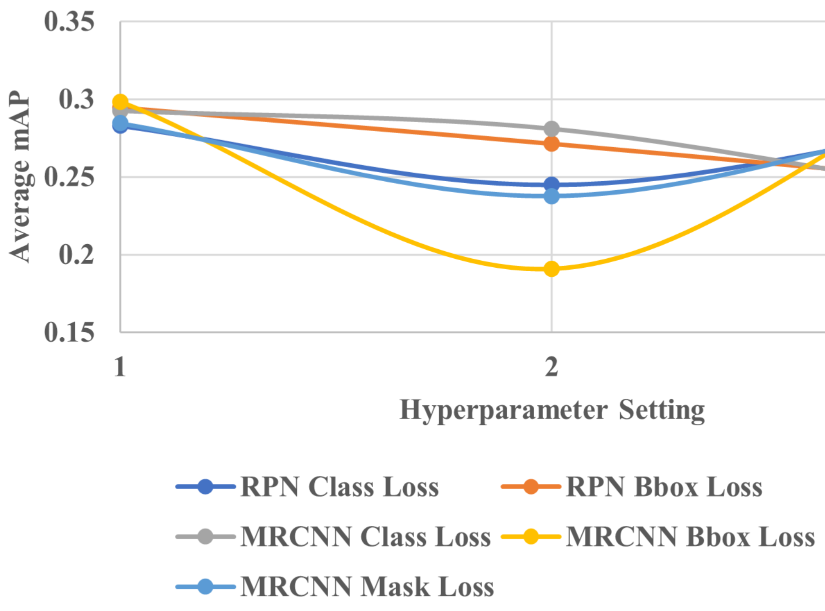


Figure 37: Interaction plot for EGS 3 which shows the average mAP score achieved for each setting of the “RPN class loss”, “RPN bbox loss”, “MRCNN class loss”, “MRCNN bbox loss” and “MRCNN mask loss” hyperparameters

4.3.3.4 Efficient Grid Search 4

Table 13 shows experimental results from EGS 4 which included RPN anchor scales, RPN anchor ratios, RPN anchor stride, RPN NMS threshold and RPN train anchors per image. Note that in this grid search, similarly to EGS 2, the default configurations for the relevant hyperparameters did not naturally align with any particular experiment and so every experiment is entirely new. The interaction plot in Figure 38 does not show any real trends and the most that can be drawn from the graph with regards to trends, is that excluding the very slight variation in effect between levels one and two, RPN train anchors per image generally had a more positive effect on mAP when it was lower. This is not strong evidence and would need to be experimented with more thoroughly to prove, however if true, it could be because less train anchors mean a lower likelihood of the model overfitting and somewhere between 256 and 512 anchors a threshold was crossed. Despite lack of trend insights, Figure 38 does suggest the possible extent of impact of each hyperparameter on the mAP. According to the plot, RPN anchor scales had the most significant impact on mAP, followed by RPN anchor stride. For anchor scales, this could be because they ensure the size of region proposals are similar to the size of the predicted objects, and since there is only one object (the ladle) that does not change size throughout the video, it is important for the anchors to be a specific size. This also aligns with the fact that medium-sized scales of (32, 64, 128, 256, 512) appeared to be superior to the larger and smaller settings. For anchor stride, this could be because it is crucial for the region proposals to be dense enough to capture the ladle whilst it is moving, whilst being sparse enough to ensure efficiency and perform robustly to noise. Additionally, the remaining three hyperparameters all appeared to impact mAP similarly. This could be because only one object is being tracked, so it is likely that train anchors per image, anchor ratios and NMS threshold are within the optimal range for every single setting, meaning they do no need to be changed. It is worth noting that due to the number of hyperparameters tested in one go, some of these suggestions may be based on results distorted by both the interaction between hyperparameters, as well as the factor of random variability always present in neural networks.

Overall, the 11th experiment was considered the best which used anchor scales of (32, 64, 128, 256, 512), anchor ratios of (0.25, 0.5, 1), anchor stride of two, NMS threshold of 0.9 and 256 train anchors per image. Whilst this experiment did not result in the highest mAP (0.396 compared to 0.438 in the first experiment), the mask appeared visually more accurate in terms of both shape and coverage of the ladle. It did produce higher precision than the highest mAP experiment, although a few other experiments did also. At this point of the model optimisation it became apparent that mAP was not as comprehensive to measuring the extent of desirable model performance as first thought. Whilst in most cases, experiments scoring a higher mAP visually appeared to segment the ladle better, in EGS 4, there was a discrepancy between mAP and actual best performance. The discrepancy was not large, but it was still significant. However, it is worth noting that whilst mAP does not include any subjectivity, human visual judgement does. Therefore, it is possible that the visual interpretation itself was inaccurate (although unlikely), and the mAP was actually a sufficient metric. One way to confirm or deny this, would be to use the judgement of several humans rather than just one.

Table 13: EGS 4 experimental results

Exp. No.	RPN Anchor Scales	RPN Anchor Ratios	RPN Anchor Stride	RPN NMS Threshold	RPN Train Anchors Per Image	Epoch Time (s)	Train Time (s)	Train Loss	Val. Loss	Inf. Time (s)	P	R	mAP_{COCO}
1	16, 32, 64, 128, 256	0.25, 0.5, 1	1	0.7	128	220	2204	0.416	0.6873	0.485	0.277	0.285	0.275
2	16, 32, 64, 128, 256	0.25, 0.5, 1	1	0.7	256	220	2201	0.511	0.8504	0.457	0.398	0.438	**0.438**
3	16, 32, 64, 128, 256	0.25, 0.5, 1	1	0.7	512	226	2255	0.502	0.693	0.470	0.131	0.155	0.155
4	16, 32, 64, 128, 256	0.5, 1, 2	2	0.8	128	221	2206	0.422	0.687	0.464	0.310	0.311	0.311
5	16, 32, 64, 128, 256	0.5, 1, 2	2	0.8	256	222	2224	0.408	0.605	0.462	0.248	0.252	0.250
6	16, 32, 64, 128, 256	0.5, 1, 2	2	0.8	512	223	2229	1.106	0.847	0.459	0.208	0.237	0.237
7	16, 32, 64, 128, 256	1, 2, 4	3	0.9	128	226	2257	0.508	0.822	0.457	0.286	0.288	0.283
8	16, 32, 64, 128, 256	1, 2, 4	3	0.9	256	219	2191	0.421	0.772	0.483	0.188	0.207	0.207
9	16, 32, 64, 128, 256	1, 2, 4	3	0.9	512	225	2247	0.409	0.796	0.455	0.294	0.296	0.288
10	32, 64, 128, 256, 512	0.25, 0.5, 1	2	0.9	128	225	2253	0.397	0.716	0.480	0.310	0.392	0.392
11	32, 64, 128, 256, 512	0.25, 0.5, 1	2	0.9	256	225	2254	0.366	0.707	0.448	0.404	0.404	**0.396**
12	32, 64, 128, 256, 512	0.25, 0.5, 1	2	0.9	512	225	2254	0.387	0.645	0.467	0.344	0.353	0.353
13	32, 64, 128, 256, 512	0.5, 1, 2	3	0.7	128	228	2277	0.431	0.714	0.486	0.319	0.393	0.393
14	32, 64, 128, 256, 512	0.5, 1, 2	3	0.7	256	223	2229	0.460	0.752	0.468	0.133	0.133	0.129
15	32, 64, 128, 256, 512	0.5, 1, 2	3	0.7	512	223	2226	0.484	0.751	0.451	0.354	0.354	0.342
16	32, 64, 128, 256, 512	1, 2, 4	1	0.8	128	226	2257	0.457	0.763	0.464	0.223	0.224	0.224
17	32, 64, 128, 256, 512	1, 2, 4	1	0.8	256	221	2214	0.367	0.885	0.478	0.414	0.426	0.426
18	32, 64, 128, 256, 512	1, 2, 4	1	0.8	512	216	2160	0.392	0.708	0.451	0.246	0.247	0.247
19	64, 128, 256, 512, 1024	0.25, 0.5, 1	3	0.8	128	222	2217	0.513	0.699	0.486	0.181	0.235	0.235
20	64, 128, 256, 512, 1024	0.25, 0.5, 1	3	0.8	256	226	2257	0.500	0.641	0.434	0.140	0.142	0.135
21	64, 128, 256, 512, 1024	0.25, 0.5, 1	3	0.8	512	221	2213	0.538	0.688	0.451	0.203	0.213	0.204
22	64, 128, 256, 512, 1024	0.5, 1, 2	1	0.9	128	224	2244	0.465	0.772	0.440	0.076	0.076	0.058
23	64, 128, 256, 512, 1024	0.5, 1, 2	1	0.9	256	225	2247	0.424	0.749	0.472	0.381	0.431	0.431
24	64, 128, 256, 512, 1024	0.5, 1, 2	1	0.9	512	228	2279	0.488	0.720	0.439	0.112	0.112	0.085
25	64, 128, 256, 512, 1024	1, 2, 4	2	0.7	128	223	2234	0.426	0.712	0.452	0.372	0.372	0.296
26	64, 128, 256, 512, 1024	1, 2, 4	2	0.7	256	228	2275	0.508	0.697	0.465	0.103	0.106	0.106
27	64, 128, 256, 512, 1024	1, 2, 4	2	0.7	512	232	2320	0.484	0.626	0.454	0.352	0.370	0.363

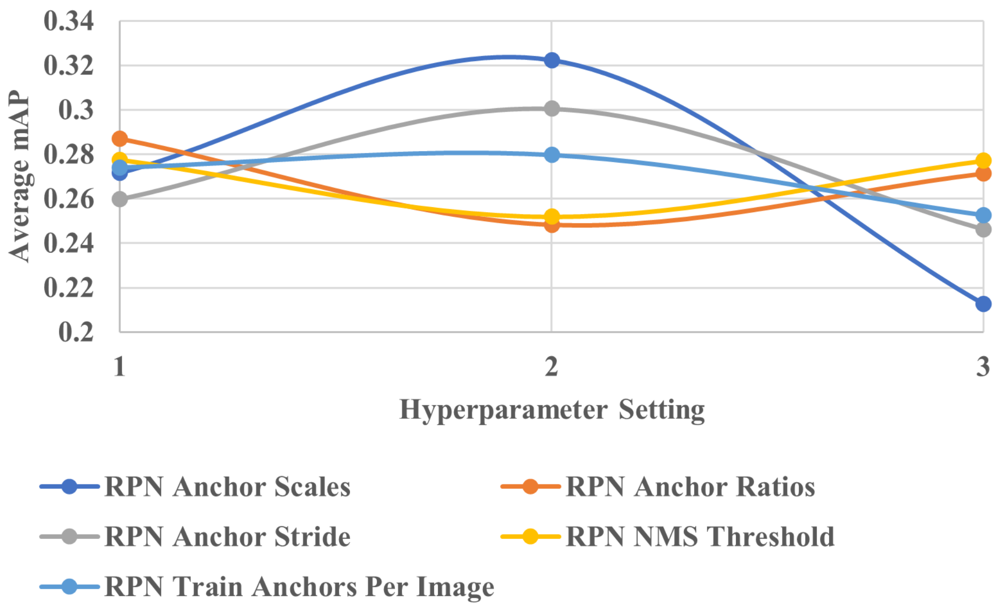


Figure 38: Interaction plot for EGS 4 which shows the average mAP score achieved for each setting of the "RPN anchor scales", "RPN anchor ratios", "RPN anchor stride", "RPN NMS threshold" and "RPN train anchors per image" hyperparameters

4.3.3.5 Efficient Grid Search 5

Table 14 shows experimental results from EGS 5 which included pre-NMS limit, post-NMS train RoIs, post-NMS inference RoIs, train RoIs per image and RoI positive ratio. Note that in this grid search, similarly to EGS 2 and EGS 4, the default configurations for the relevant hyperparameters did not naturally align with any particular experiment and so every experiment is new. The interaction plot in Figure 39 shows no strong trends, and similar but opposite to the train anchors in EGS 4, excluding the very slight variation in effect between levels one and two, post-NMS train RoIs generally had a more positive effect on mAP when it was lower. This is not strong evidence and would need to be experimented with more thoroughly to prove, however if true, it could be because below some threshold (between 2000 and 4000), the model is forced to pay more attention to better proposals. In terms of impact, Figure 39 shows that train RoIs per image and RoI positive ratio had the largest impacts on mAP. Train RoIs dictate how many samples the model has to learn from and therefore directly impacts the effectiveness of training, so it is unsurprising that it was shown to have a large effect. Also, RoI positive ratio controls the number of object and background samples during training which should ideally avoid false positives and false negatives, which directly impacts precision, recall, and resulting mAP. Meanwhile, post-NMS RoIs for both training and inference had similar impacts on mAP. These ensure that the correct amount of proposals are finally used for training and inference, and whilst they need to be sufficient, the most important aspect is that they are high quality. Finally, the pre-NMS proposal limit had almost no impact on the mAP, which is explainable by the fact that it is more relevant to computational efficiency, since post-NMS RoI values are what ultimately effect the number of RoIs actually used.

Overall, the 13th experiment was considered the best which used a pre-NMS limit of 6000, 2000 post-NMS training RoIs, 2000 post-NMS inference RoIs, 100 train RoIs per image, and a RoI positive ratio of 0.25. Similarly to EGS 4, whilst this experiment did not result in the highest mAP (0.426 compared to 0.466), the mask appeared visually more accurate in terms of both shape and coverage of the ladle. Again, this was subjective and could be confirmed through the use of more human input.

Table 14: EGS 5 experimental results

Exp. No.	Pre-NMS Limit	Post-NMS Train RoIs	Post-NMS Inf. RoIs	Train RoIs Per Image	RoI Positive Ratio	Epoch Time (s)	Train Time (s)	Train Loss	Val. Loss	Inf. Time (s)	P	R	mAP_{COCO}
1	3000	1000	500	100	0.25	224	2242	0.380	0.769	0.500	0.408	0.466	**0.466**
2	3000	1000	500	100	0.33	229	2285	0.418	0.781	0.453	0.245	0.246	0.246
3	3000	1000	500	100	0.5	227	2271	0.441	0.643	0.487	0.303	0.319	0.319
4	3000	2000	1000	200	0.25	227	2271	0.479	0.637	0.467	0.244	0.260	0.260
5	3000	2000	1000	200	0.33	221	2208	0.400	0.808	0.498	0.202	0.211	0.211
6	3000	2000	1000	200	0.5	227	2269	0.390	0.791	0.475	0.358	0.391	0.391
7	3000	4000	2000	400	0.25	239	2393	0.447	0.692	0.470	0.154	0.181	0.181
8	3000	4000	2000	400	0.33	228	2276	0.406	0.792	0.481	0.179	0.204	0.204
9	3000	4000	2000	400	0.5	237	2367	0.418	0.751	0.476	0.301	0.335	0.335
10	6000	1000	1000	400	0.25	237	2366	0.357	0.692	0.486	0.422	0.465	0.465
11	6000	1000	1000	400	0.33	235	2350	0.510	0.660	0.472	0.299	0.336	0.336
12	6000	1000	1000	400	0.5	232	2324	0.427	0.676	0.459	0.332	0.333	0.330
13	6000	2000	2000	100	0.25	236	2362	0.380	0.626	0.468	0.420	0.433	**0.426**
14	6000	2000	2000	100	0.33	237	2365	0.457	0.689	0.491	0.292	0.360	0.360
15	6000	2000	2000	100	0.5	236	2361	0.389	0.563	0.494	0.351	0.354	0.351
16	6000	4000	500	200	0.25	237	2371	0.409	0.687	0.492	0.150	0.165	0.165
17	6000	4000	500	200	0.33	238	2381	1.042	0.761	0.442	0.000	0.000	0.000
18	6000	4000	500	200	0.5	232	2316	0.454	0.796	0.471	0.088	0.099	0.099
19	12000	1000	2000	200	0.25	237	2371	0.366	0.730	0.479	0.328	0.342	0.336
20	12000	1000	2000	200	0.33	243	2430	0.472	0.681	0.451	0.076	0.077	0.076
21	12000	1000	2000	200	0.5	231	2312	0.406	0.691	0.484	0.194	0.226	0.226
22	12000	2000	500	400	0.25	233	2330	0.360	0.733	0.489	0.287	0.299	0.296
23	12000	2000	500	400	0.33	227	2265	0.521	0.814	0.490	0.150	0.156	0.155
24	12000	2000	500	400	0.5	237	2366	0.418	0.785	0.478	0.424	0.451	0.451
25	12000	4000	1000	100	0.25	226	2263	0.431	0.729	0.462	0.343	0.403	0.403
26	12000	4000	1000	100	0.33	226	2258	0.498	0.740	0.487	0.234	0.287	0.287
27	12000	4000	1000	100	0.5	224	2235	0.375	0.674	0.461	0.356	0.372	0.362

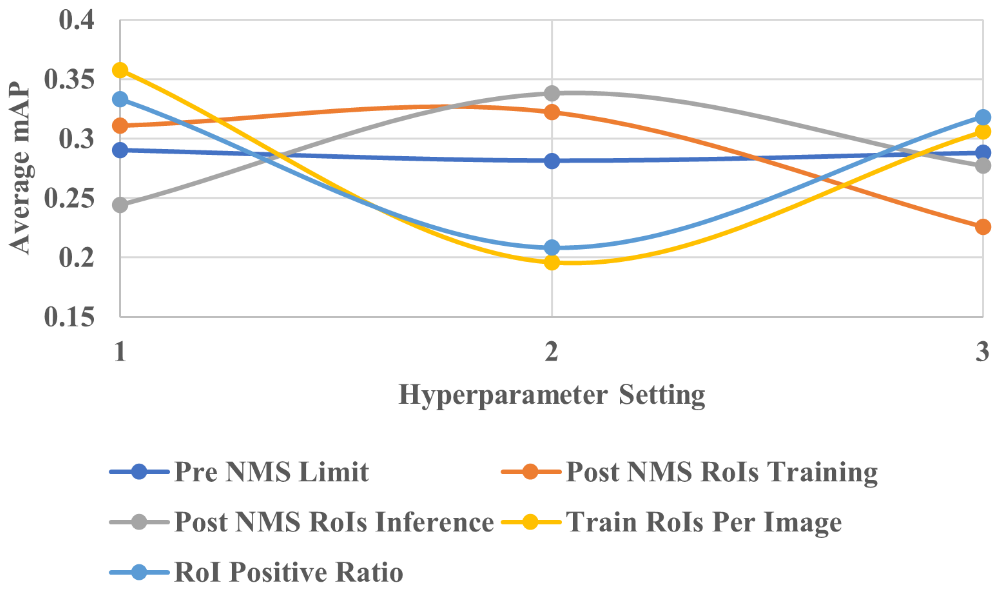


Figure 39: Interaction plot for EGS 5 which shows the average mAP score achieved for each setting of the "pre-NMS limit", "post-NMS RoIs training", "post-NMS RoIs inference", "train RoIs per image" and "RoI positive ratio" hyperparameters

4.3.3.6 Efficient Grid Search 6

Table 15 shows experimental results from EGS 6 which included pool size, mask pool size and mask shape. The interaction plot (Figure 40) suggests there were no trends, however pool size was shown to have almost no impact on mAP, whereas mask pool size and mask shape did. Pool size dictates the feature map size of each RoI after pooling which is essential for classification and localisation and therefore, in hindsight, would have been better off being grouped into an earlier grid search as opposed to being grouped with mask-related hyperparameters which are associated with the end of the network. Since the model optimisation was conducted in this manner, it is highly likely that due to other RPN-related hyperparameters were already optimised in previous grid search experimentation, the remaining effect of changing pool size was suppressed. Whilst this uncovers the fact that the organisation of the hyperparameter optimisation was sub-optimal in this case, it more importantly indicates that the optimisation up to this stage, at least in terms of RPN and RoI related hyperparameters, was successful. In other words, since the optimisation already led to high quality region proposals, there was no improvement that modifying pool size could make. Moving focus towards mask-related optimisation, the larger mask pool size impact makes sense, since it directly affects the resolution of feature maps for mask generation. This is the same for mask shape.

Overall the third experiment produced the best precision, recall and mAP, whilst also being judged visually as the best model. This experiment used a pool size of seven, a mask pool size of 56, and a mask shape of (112, 112) and achieved an mAP of 0.516.

Table 15: EGS 6 experimental results

Exp. No.	Pool Size	Mask Pool Size	Mask Shape	Epoch Time (s)	Train Time (s)	Train Loss	Val. Loss	Inf. Time (s)	P	R	mAP_{COCO}
1	7	14	28, 28	236	2362	0.380	0.626	0.468	0.420	0.430	0.426
2	7	28	56, 56	230	2302	0.996	0.849	0.469	0.002	0.003	0.003
3	7	56	112, 112	227	2270	0.416	0.636	0.455	0.477	0.516	**0.516**
4	14	14	56, 56	228	2281	0.397	0.558	0.457	0.242	0.242	0.242
5	14	28	112, 112	214	2140	0.418	0.593	0.472	0.358	0.358	0.358
6	14	56	28, 28	222	2215	0.445	0.766	0.444	0.348	0.348	0.296
7	28	14	112, 112	227	2269	0.362	0.767	0.466	0.366	0.367	0.364
8	28	28	28, 28	228	2282	0.437	0.707	0.451	0.193	0.213	0.213
9	28	56	56, 56	229	2288	0.404	0.696	0.490	0.239	0.283	0.283

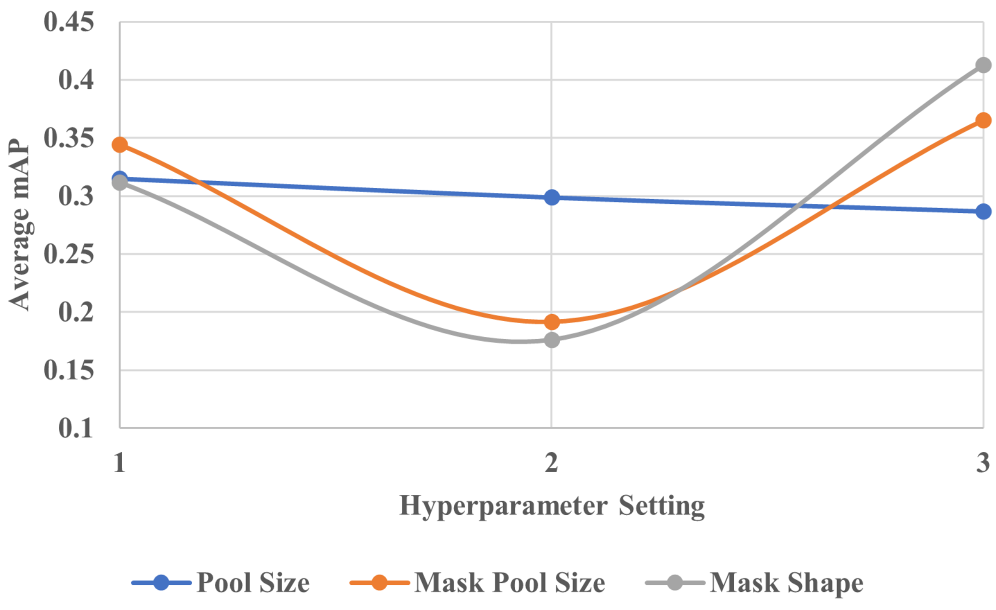


Figure 40: Interaction plot for EGS 6 which shows the average mAP score achieved for each setting of the “pool size”, “mask pool size” and “mask shape” hyperparameters

4.3.4 Analysis of Hyperparameter Optimisation Results

Table 16 presents the optimal settings for each hyperparameter after completing the hyperparameter optimisation. This model was trained for ten epochs which took just under four minutes each on average and just under 38 minutes in total. The mAP achieved was 0.516 and the inference time for each frame was 0.46s. This set of experiments exposed some interesting aspects to Mask R-CNN and its hyperparameters. For example, whilst increasing steps per epoch did correlate with a slight increase in mAP initially, decreasing this from 1000 to 100 appeared to have a noticeable impact on model performance but saved approximately 90% training time which was significant. Furthermore, since the best training epochs were between one and ten, reducing them from 20 to ten halved the required training time. These two findings were crucial to the efficiency of the optimisation at apparently no cost to model performance. In future work, a much more conservative approach will be taken to optimising hyperparameters, especially as many as the 24 optimised here.

EGS 3 was a key turning point in model performance. Before then, most hyperparameters appeared to have little effect on mAP and validation losses were typically about double that of the training losses indicating overfitting. This was likely caused by the dataset being too small. One pouring video was used for training and validation, and one was used for testing. However, for a task as complex as instance segmentation in a poorly-lit environment, this was insufficient and would possibly require ten or more videos for each subset to develop a high-performance model. In EGS 3 however, modifying loss weights closed the gap between training and validation losses, and validation losses decreased to more acceptable values. Loss weights had a large impact on performance because some operations of Mask R-CNN are much more important than others depending on the scenario. In this case study, only one object needed to be detected against the background, making the Mask R-CNN classification loss less important. Similarly, the purpose was to segment the ladle, meaning Mask R-CNN mask loss was crucial to the overall performance.

Also worth noting, is that the seemingly ineffective hyperparameters near the beginning of the optimisation were not necessarily unimportant, but were experimented at a stage where their impact could not be exposed. In future work it would be useful to incorporate optimisation of hyperparameters such as learning rate and learning momentum after optimising loss weights, which could result in an entirely different outcome to this one. The most extreme example of non-ideal optimisation organisation was with the pool size hyperparameter in EGS 6. The fact that it was grouped far later on in the optimisation than its most related hyperparameters (RoI and RPN based), meant that by the time it was tested, it was already redundant. This did uncover the success of some of the most crucial stages of the optimisation however.

Table 16: Optimal hyperparameter settings

Hyperparameter	Setting
CNN backbone	ResNet101
Layers re-trained	3+
Image resizing	pad1024
Augmentation	off
Mini-mask	off
Weight decay	0.0001
EGS 1	
Steps per epoch	100
Validation steps	50
EGS 2	
Epochs	Variable
Learning rate	0.001
Learning momentum	0.9
Gradient clip norm.	5
EGS 3	
RPN class loss	3
RPN bbox loss	2
MRCNN class loss	1
MRCNN bbox loss	3
MRCNN mask loss	3
EGS 4	
RPN anchor scales	(32, 64, 128, 256, 512)
RPN anchor ratios	(0.25, 0.5, 1)
RPN anchor stride	2
RPN NMS threshold	0.9
RPN train anchors per image	256

Continued on next page

Table 16 – continued from previous page

Hyperparameter	Settings
EGS 5	
Pre-NMS limit	6000
Post-NMS training RoIs	2000
Post-NMS RoIs inference	2000
Train RoIs per image	100
RoI positive ratio	0.25
EGS 6	
Pool size	7
Mask pool size	56
Mask shape	112

It is important to note that the inherent complexity of the hyperparameter optimisation played a large role in this task. By addressing 24 different hyperparameters, many involving complex mathematical concepts, strategising the approach to optimisation was incredibly challenging. Without already having expertise on Mask R-CNN, it would have been impossible to plan the optimisation in an optimal way within one iteration. In fact, there is no one optimal way, although some approaches could produce more useful conclusions than others. In this case study, an approach was taken that generally organised optimisation stages so that they aligned with sequential steps of the Mask R-CNN network. This approach turned out to be effective and evidently from discussion, there were many findings. However, by conducting this iteration, findings point to other potential optimisation strategies for further insight. Due to the complexity of Mask R-CNN, many iterations could be performed with many different strategies and novel approaches would still be uncovered.

Whilst the hyperparameter optimisation was a vast operation that was critical for many learnings related to Mask R-CNN, there were some constraints on resources. On one hand, these added to the challenge and made the learning process more tailored to industry where resource efficiency is critical, which aligns with the aims of this project. On the other hand, the results of this case study, whilst successful, do indicate that resources were limiting to the final model produced. To elaborate, the time constraints were useful in that they forced the pace of development and pushed some critical decisions such as reducing the steps per epoch which saved 90% of the training time from that point onwards. Oppositely, the data constraints were too constrictive when it came to model development. Whilst a few hundred labelled ladle frames were perceived as potentially being enough to develop a robust model initially, with the additional research experience gained throughout all case studies, a number of samples somewhere in the region of a few thousand would have been required at minimum to achieve a production level model. Not only the dataset size, but the diversity of the samples would need to be expanded also. Industrial application will be discussed in more depth in Section 4.4. It is worth noting that labelling 100 frames of the ladle mask while pouring was far more time-consuming than labelling 100 frames would have been for many other objects. This was not only due to the complex shape of the mask, but particularly because it was obstructed by emissions and was set in an environment with poor lighting. If an auto-labelling method was developed for this kind of challenge, it would be incredibly valuable to industry.

4.3.5 Kalman Filter Tracking

Following hyperparameter optimisation of Mask R-CNN, Kalman filtering experiments were conducted to further advance the model. As mentioned in Section 4.2.4, Kalman filtering was applied to enable the model to associate masks between sequential frames so that it would have an improved understanding of ladle shape and movement across the video rather than simply on a frame-by-frame basis. The expectation was that this would improve the robustness of segmentation by adding a sequential understanding as well as a spatial understanding. Kalman filters predict the future position of an object, and in this case, the task is multi-object tracking where the "objects" are points sampled from the predicted mask edges in each frame.

Equation (15) in Section 4.2.4 showed the default state transition matrix, where matrix values dictated how point positions and velocities changed (or stayed the same) across timesteps, whilst δt was the change in timestep. Since elements [3,3] and [4,4] were used for x and y velocity changes, and this model assumes constant velocity, they remained remain unchanged.

Element [1,1] was tested at values from one to ten incrementally, and six gave the best performance. Next, element [2,2] was tested in the same way whilst maintaining the value of six for [1,1], and a value of eight produced the best performance. The optimal matrix is shown in Equation (17).

$$\hat{A}_k = \begin{bmatrix} 6 & 0 & \Delta t & 0 \\ 0 & 8 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

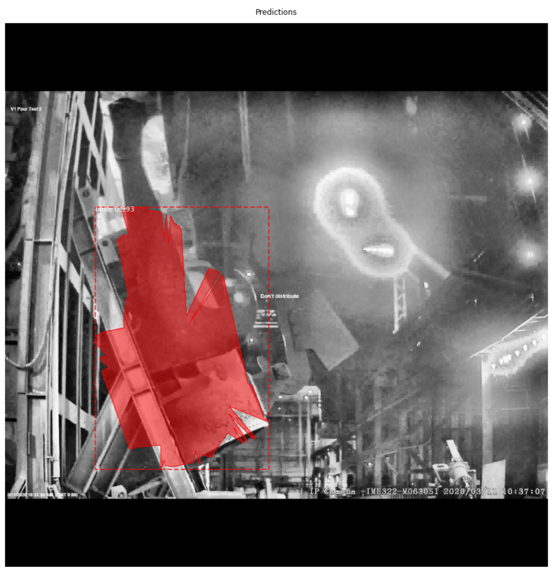
Table 17 shows how optimisation of the different elements improved the mAP. The original mAP after completing the hyperparameter optimisation was 0.516, using the hyperparameter settings summarised in Table 16 in Section 4.3.4. Therefore, optimising the first element gave an improvement of 0.058, and optimising the second gave an improvement of 0.032. The addition of the Kalman filtering increased the average inference time from 0.46s (2.17fps) to 0.59s (1.69fps).

Table 17: State transition matrix experimental results

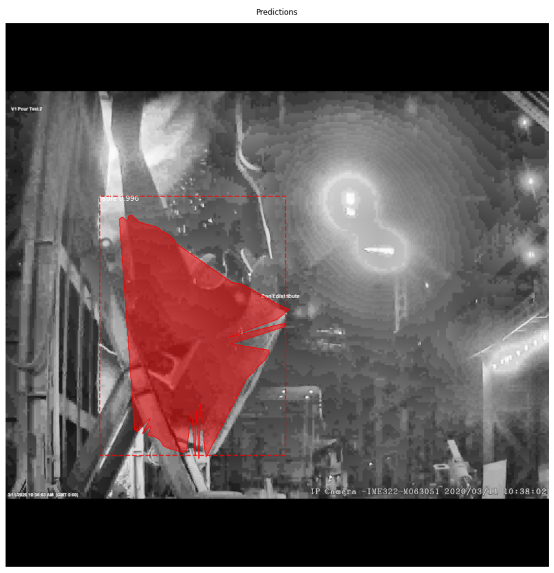
Variable Tested	mAP$_{\text{COCO}}$
x_{k-1}	0.574
y_{k-1}	0.606

In Table 17, x_{k-1} ([1,1] in Equation (17)) and y_{k-1} ([2,2] in Equation (17)) are x and y position values respectively. The improvement observed by changing [1,1] and [2,2] to six and eight respectively, is likely because they match what can be observed in the video. The ladle begins on the lower right-hand side of the video, and since Python reads images from top to bottom (therefore y-coordinate increases downwards), the six and eight in Equation (17) roughly align with the x and y positions respectively.

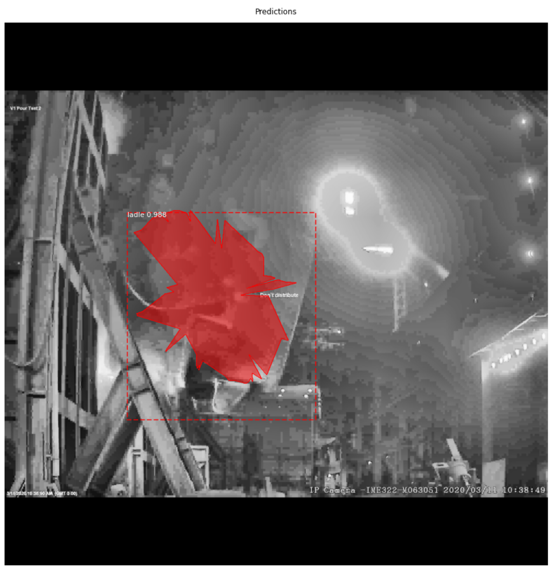
Figure 41 and Figure 42 shows examples of Kalman filtering effects on the predicted mask. From initial Kalman filter application, the mask result looked like Figure 41(a), (b) and (c). As shown, the masks reach the edges of the ladle quite well, however they are jagged and therefore incomplete, which is due to the point-to-point drawing of the edges. In Figure 42(a), (b) and (c), the same frames are shown but after the convex hull was applied. As shown, the jaggedness has been eliminated. In Figure 42(a), (b) and (c), there are also green, yellow and red points shown. The green points are the initial Kalman filter predictions, the yellow points are the measured mask edge points from Mask R-CNN, and the red points are the Kalman filter predictions after being updated based on the actual measurements. It may be noticeable that there are more yellow points than red points and the reason for this is the K-Nearest Neighbour clustering that was done to preserve points that were close to each other and exclude points that were surrounding the ladle edge but some distance away from it. K-Nearest Neighbours was applied with two neighbours, which was found to preserve points sitting on the ladle edge whilst eliminating unwanted point predictions.



(a) Frame 13 with Kalman filtering applied

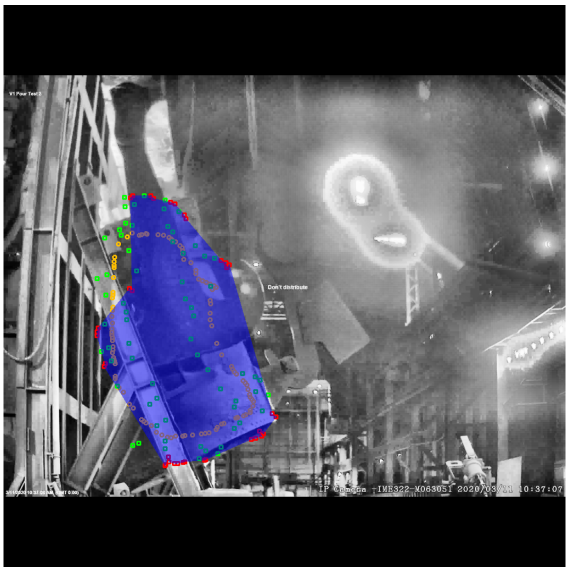


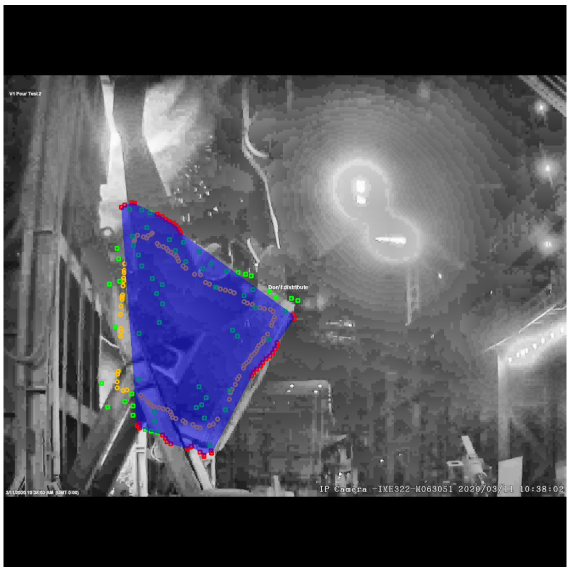
(b) Frame 66 with Kalman filtering applied



(c) Frame 112 with Kalman filtering applied

Figure 41: Comparison of mask outputs with Kalman filtering


(a) Frame 13 with Kalman filtering and convex hull applied


(b) Frame 66 with Kalman filtering and convex hull applied

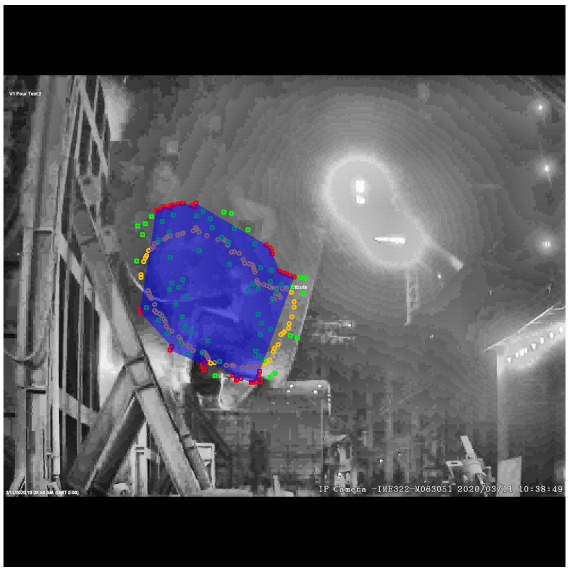

(c) Frame 112 with Kalman filtering and convex hull applied

Figure 42: Comparison of mask outputs after applying the convex hull operation

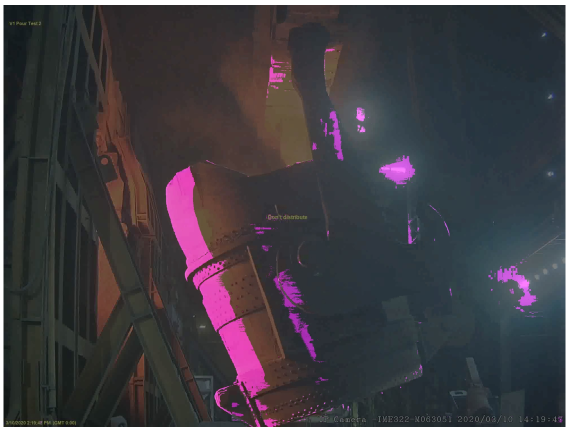
Figure 42(a), (b) and (c) can be used to compare the original mask prediction (yellow marks) to the Kalman filter mask. As shown, the Kalman filter does reach the ladle edges better than the original masks, however since the overall shape is based purely on positions of individual points, rather than the features of the ladle as a whole, there is less alignment with the natural shape of the ladle and the Kalman mask. Therefore, although Kalman filtering improved the mAP, it was less ideal for real-world measurements compared to the Mask R-CNN mask which had learned intricate features of the ladle and how it is supposed to look. This indicates avenues for future work. For example, the state transition matrix for every point was optimised in one go. In future work, it would likely enhance performance if each of the 32 filters were optimised individually in some way, possibly setting matrix elements as equations that take the pixel coordinates of a reference mask as input. Secondly, since the Kalman model used in this research assumes linear motion, it is limited particularly when the ladle rotates. Therefore, future work may benefit from using an extended Kalman filtering approach [68].

4.3.6 Background Subtraction

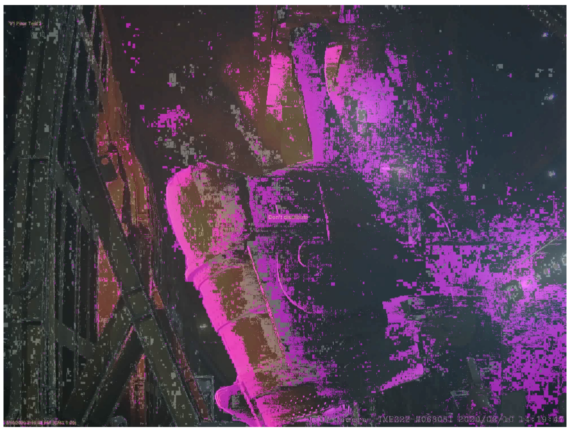
Based on the success found in applying background subtraction (BGS) algorithms in Chapter 5 and Chapter 6, the same algorithms were applied in this case study also, to see if there were any possible benefits to using this approach. The specific aspects of this case study that BGS were thought to be potentially useful for, were tracking the ladle as it moves since the background is "fixed" (which is not strictly true and is explained later), as well as segmenting out the furnace emissions during pouring.

4.3.6.1 Analysis of Ladle Movement Frame

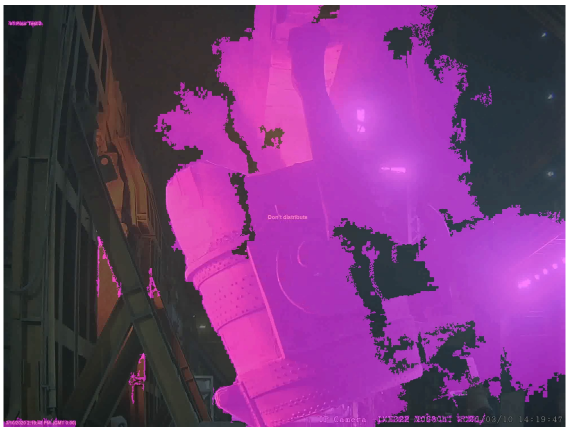
Figure 43 and Figure 44 show the results of an early frame where the ladle is approaching the furnace after being processed with different BGS algorithms including MOG (Mixture of Gaussians) [238], MOG2 [239], GSOC (Google Summer of Code) [240], GMG (Gaussian Mixture-based Background Foreground Segmentation) [241], KNN (K-Nearest Neighbours) [242] and CNT (Counting) algorithms [243]. At this stage of the footage, Figure 43(a) which uses the MOG (Mixture of Gaussians) algorithm shows some potential for useful application since it is capturing the the front edge of the ladle accurately. In all other cases (including the second MOG variant MOG2), the harsh conditions and poor lighting comprise an environment that is too complex for the algorithms to perform high quality segmentation on just the ladle.



(a) MOG

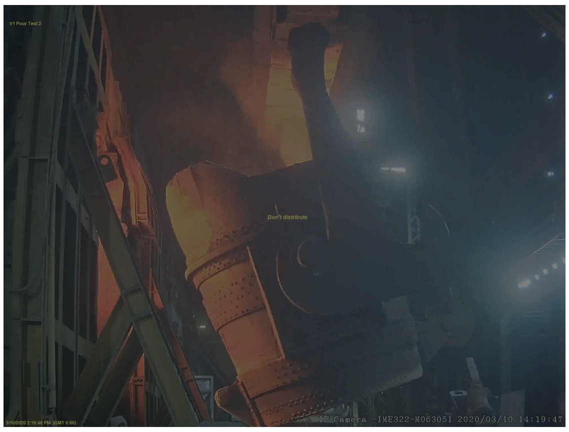


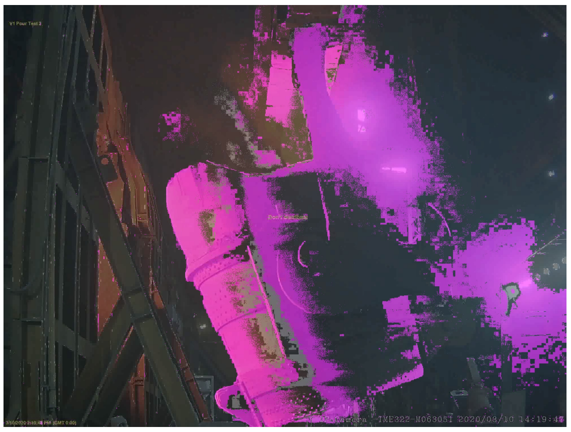
(b) MOG2



(c) GSOC

Figure 43: Figure showing the effects of different BGS algorithms (MOG, MOG2 and GSOC) on a frame showing the ladle moving towards the furnace


(a) GMG


(b) KNN

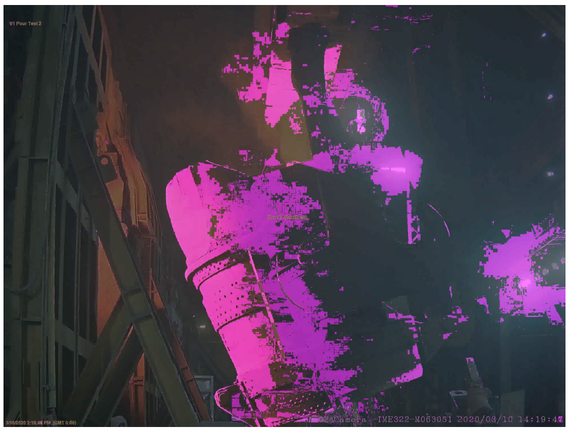
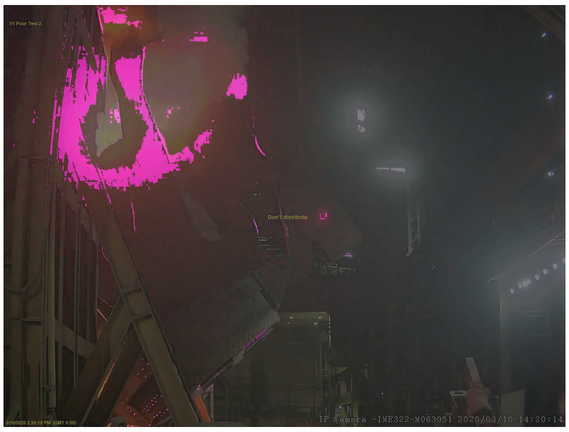
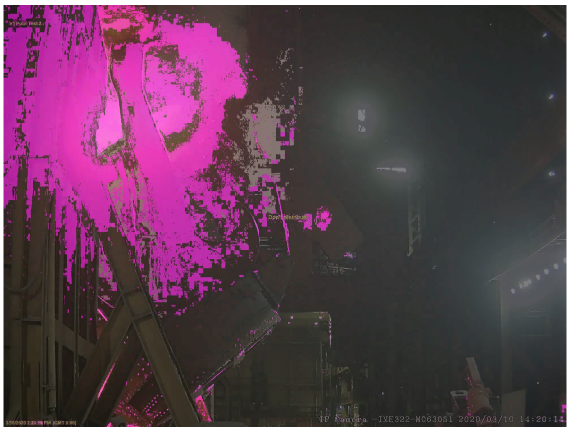

(c) CNT

Figure 44: Figure showing the effects of different BGS algorithms (GMG, KNN and CNT) on a frame showing the ladle moving towards the furnace

4.3.6.2 Analysis of Early Pour Frame

Figure 45 and Figure 46 shows the results at an early point of pouring after being processed with different BGS algorithms. This frame opens up discussion related more to the flame severity measurement than the ladle tracking. All algorithms are not segmenting the ladle, but instead, segmenting the flame region. Even though the ladle is moving, it does not stand out much from the background and features such as the three light sources on the right hand side, and particularly brightness changes in the flame region, are much more obvious from the perspective of the algorithms. Whilst all algorithms in Figure 45 and Figure 46 show potential use in flame severity measurement, they all have slightly different approaches as shown by the differing masks. More investigation is required to understand the effectiveness of these methods at measuring flame severity.


(a) MOG


(b) MOG2

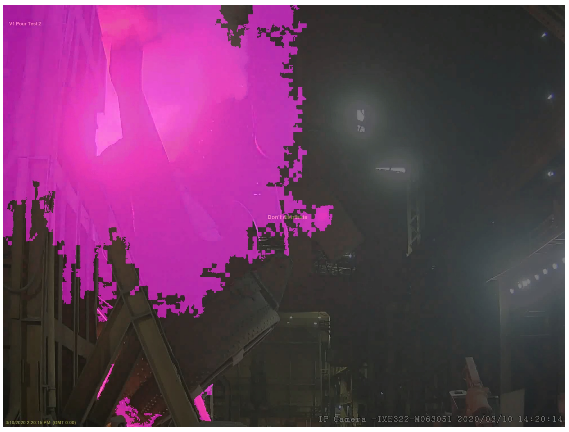
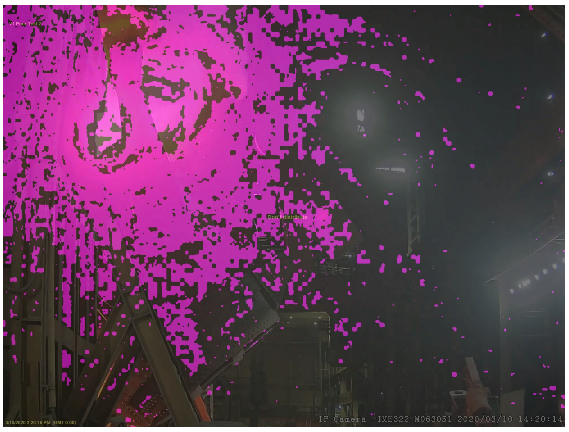
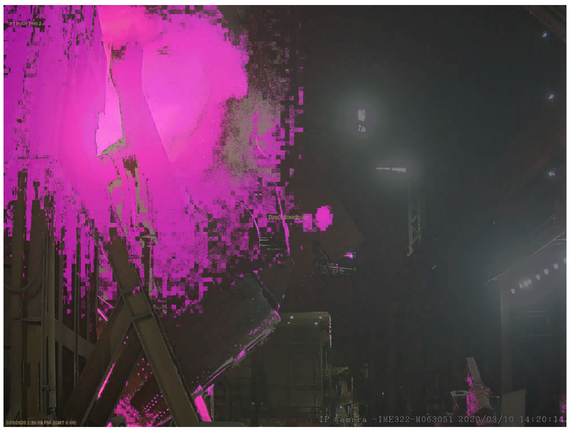

(c) GSOC

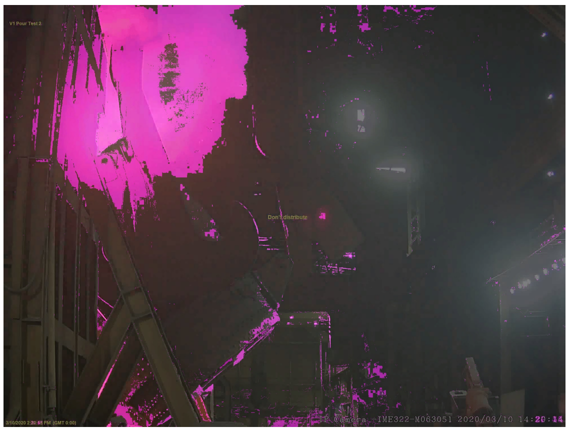
Figure 45: Figure showing the effects of different BGS algorithms (MOG, MOG2 and GSOC) on a frame showing the ladle starting to pour molten metal into the furnace



(a) GMG



(b) KNN

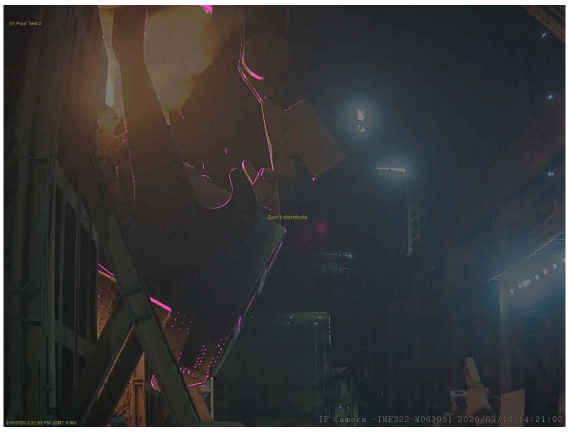


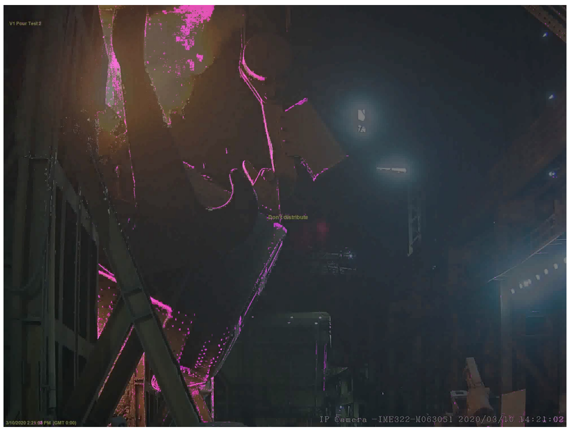
(c) CNT

Figure 46: Figure showing the effects of different BGS algorithms (GMG, KNN and CNT) on a frame showing the ladle starting to pour molten metal into the furnace

4.3.6.3 Analysis of Mid-Pour Video Frame

Finally, Figure 47 and Figure 48 show the results of a frame mid-pour after being processed with different BGS algorithms. These examples show some weak tracking of the ladle edges and in some cases such as (d) and (f) there is added noise. Furthermore, there is some mild emissions present in the flame region which are only captured by the CNT algorithm which captures the flame region as well as the added noise anyway, preventing it from being effective for real application. Whilst the results in Figure 45 and Figure 46 show potential for BGS algorithms for flame severity measurement, the results in Figure 47 and Figure 48 show that they may produce a lot of false negatives unless combined with other techniques or used in different lighting conditions.


(a) MOG


(b) MOG2

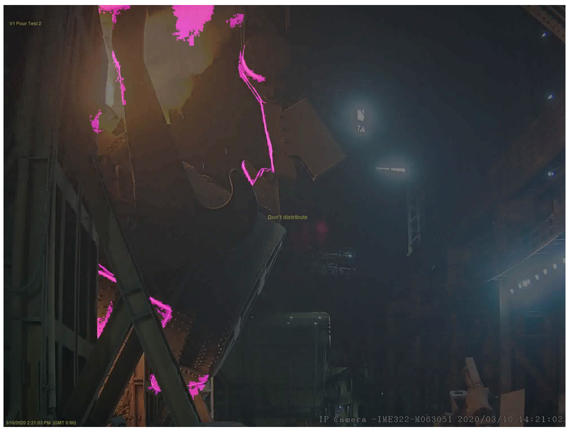
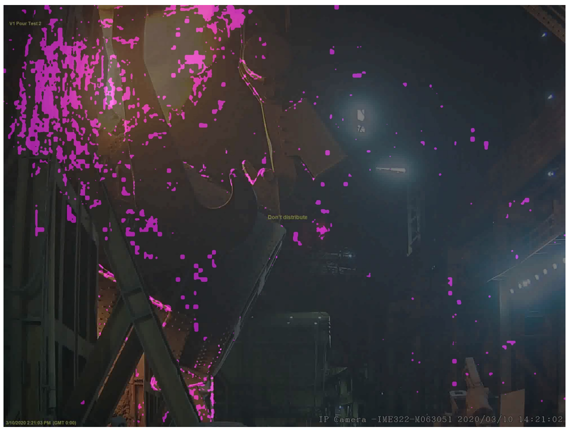
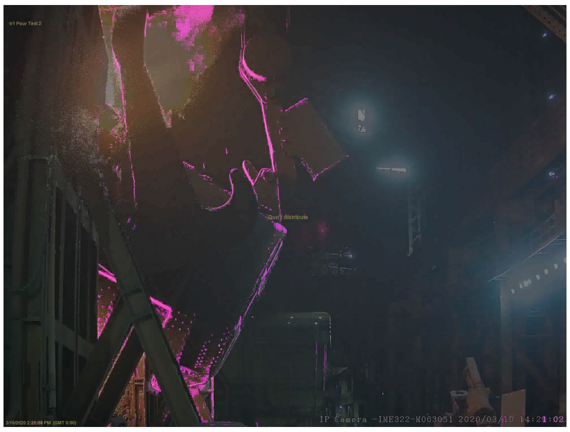

(c) GSOC

Figure 47: Figure showing the effects of different BGS algorithms (MOG, MOG2 and GSOC) on a frame showing the ladle pouring molten metal into the furnace


(a) GMG


(b) KNN

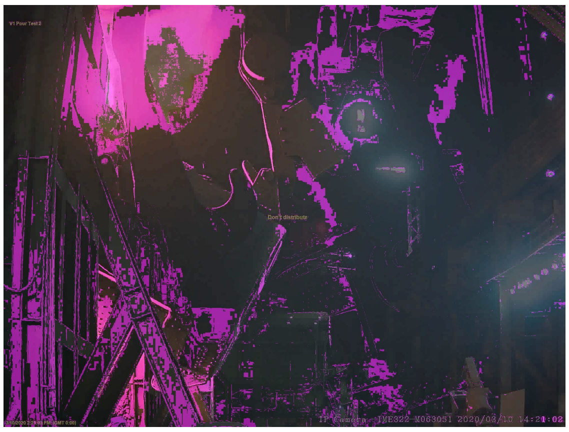

(c) CNT

Figure 48: Figure showing the effects of different BGS algorithms (GMG, KNN and CNT) on a frame showing the ladle pouring molten metal into the furnace

4.4 Industrial Application

This section investigates industrial application of the developed model for advancing upon traditional ladle technology. By monitoring measurements like ladle pouring height, ladle rotation angle, and furnace emission severity, this model provides real-world benefits. Firstly, it can be used to record data for optimising pouring for emission reduction, leading to increased safety and reduced equipment degradation. Secondly, it paves the way for automating the pouring process, which would standardise it and eliminate the need for manual labour. The following pages detail the approach used for real-world usage and the subsequent results obtained.

4.4.1 Measuring the Pouring Process

To demonstrate the potential of the model for industrial use, focus was placed on estimating three variables, namely pouring height, rotation angle, and flame or emission severity. These variables are key aspects when assessing pour quality since the pouring height and rotation angle majorly dictate ladle pouring behaviour, whilst the emission severity in the flame region is an indicator of the severity of harmful fumes and equipment degradation. Therefore, the concept was that ladle behaviour could be monitored simultaneously with emission severity, and the relationship between process parameters and process quality could be understood more intricately. This contributes towards addressing the gap identified in Chapter 2 regarding using CV models to measure real-world characteristics.

Pouring height was measured by taking the highest pixel of the segmentation mask and giving a fixed pixel offset, since there seemed to be a fairly constant-sized gap between the top of the mask and the top of the actual ladle. If the segmentation was very precise, this offset may be removed. A five-point moving average was also applied since the ladle height changed gradually whereas the mask changed every frame, and since it was unstable at times, this added unnecessary error to the model predictions.

Rotation angle was measured based on the bottom edge of the ladle as this was the most accurately predicted throughout the footage and therefore gave the most reliable estimate. The bottom edge was captured by a moving window that was scaled dynamically based on the location of the right-most point of the mask, since the ladle was always tilted slightly to the left. As the right-most point moved upwards (ladle pouring), the moving window became narrower and expanded in a downwards manner, effectively capturing the bottom ladle edge as it rotated from the bottom side to the right-hand side. A line of best fit was produced

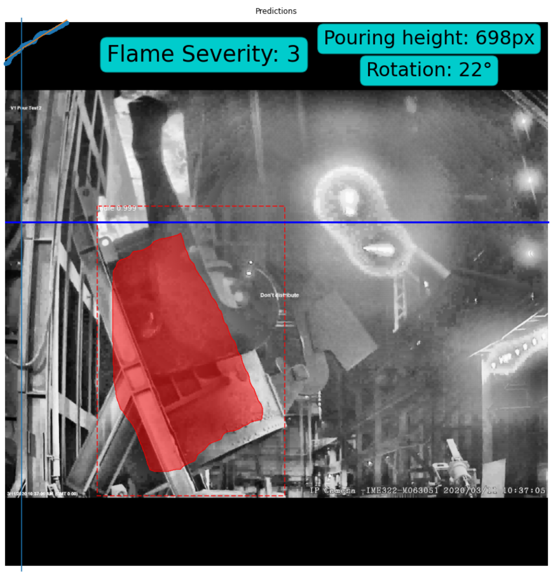
Table 18: Flame severity boundaries for the number of bright pixels in the flame region

Bright Pixels	Flame Severity
0-1000	0
1000-2000	1
2000-6000	2
6000-10000	3
10000-15000	4
15000+	5

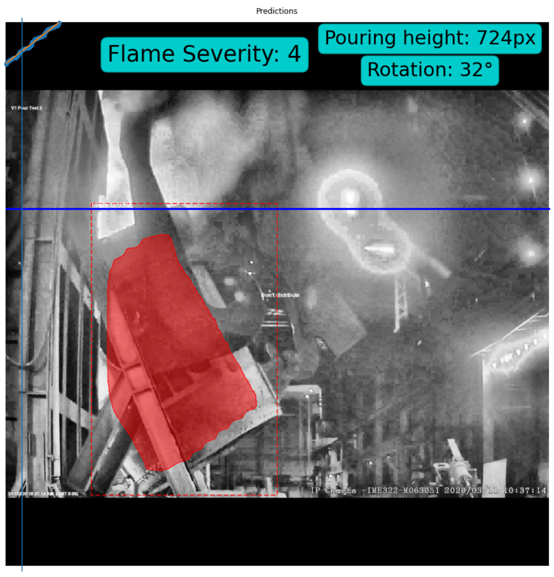
from the bottom ladle edge and used to calculate the angle of rotation. A four-point moving average was applied to this to eliminate small fluctuations in angle estimations to produce a smoother signal that was more like one that would be expected from the gradual and constant rotation of the ladle during pouring.

Finally, flame severity was measured by focusing on the area significantly affected by furnace flames. Pixels exceeding a brightness threshold of 235 were counted and depending on how this value fitted into several pre-defined ranges, a severity rating for each frame was established. Table 18 shows the boundaries that were used for each flame severity. These measurements (pouring height, rotation angle and flame severity) were captured over a whole pouring video and then used to gain insights on the pouring process through further analysis.

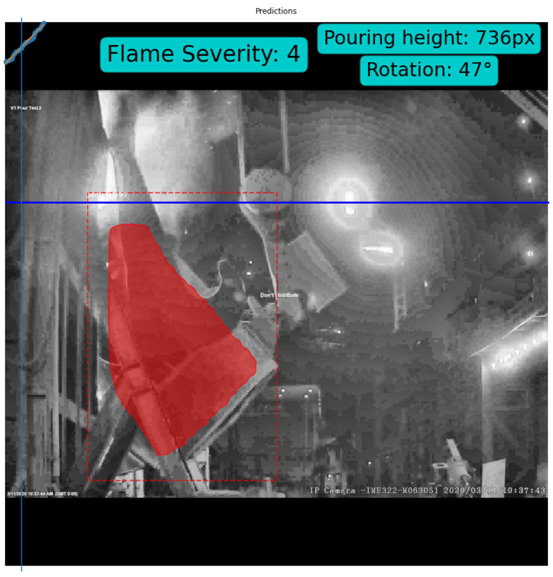
Examples of the final output of the model can be seen in Figure 49 and Figure 50. For this aspect of the research, the training set was increased from 120 to 480 images to maximise performance, and as previously discussed, Kalman filtering was disabled as it was found to make the segmentation mask more unstable which was detrimental to the pouring height and rotation angle estimations. The flame severity was on a scale of 0 to 5, the pouring height was measured in pixels and the rotation was measured in degrees. As shown in Figure 49 and Figure 50, the model performance was quite good across most frames which aligns with the mAP value of 0.516. From Figure 49(a) to (c), followed by Figure 50(a) to (c), the frames are in chronological order and show how the model dealt with the ladle rotating. The model appeared to capture the shape of the visible parts of the ladle quite well and even does quite a good job when the metal frame of the furnace obstructs part of it, demonstrating that the labelling choice discussed in Section 4.2.1 was effective. Despite the visible portion of the ladle changing whilst it also rotated, the mask was fairly consistent. However, it can also be seen that the edges of the mask do not reach the edges of the actual ladle, which seemed to be a noticeable limitation throughout the whole case study. As mentioned, this was counteracted by using Kalman filtering but at the cost of the quality of the mask shape which was crucial for real-world measurements. It is believed that if much more training data was provided to Mask R-CNN, then this limitation may be overcome, however considering the original quality and quantity of the data provided for this case study, the results shown in Figure 49 and Figure 50 demonstrate good performance. The frames shown in Figure 50(b) and (c) show a slight reduction in prediction mask quality and this was likely to be because Mask R-CNN was trained on less frames at this stage of the pouring process compared to say, Figure 49(b) and (c) which the pouring process looks like for the majority of each pour.



(a) Frame 11

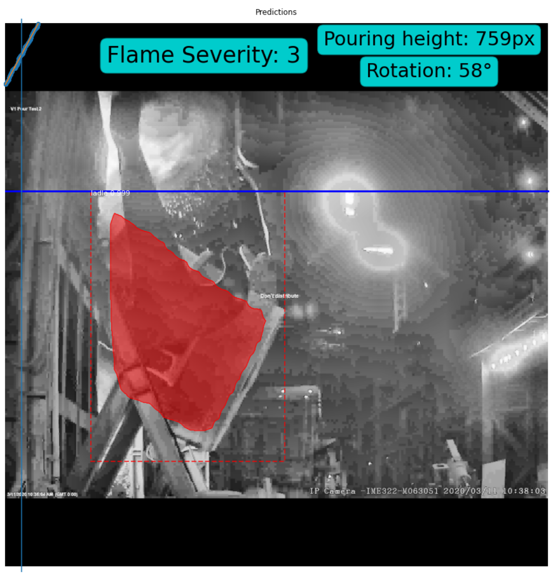


(b) Frame 19

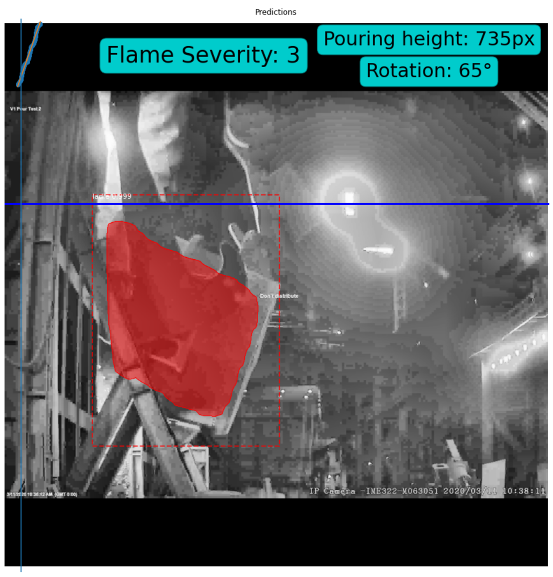


(c) Frame 48

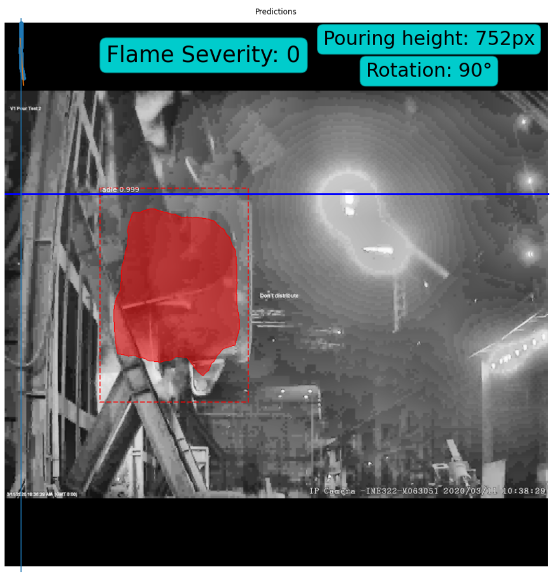
Figure 49: Frame results from final model output with measurements displayed (1 of 2)



(a) Frame 67



(b) Frame 75

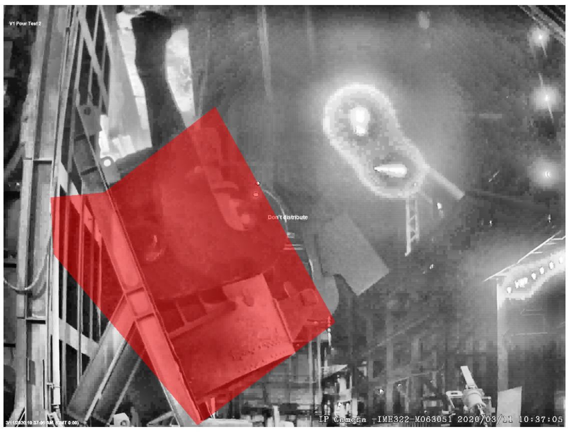


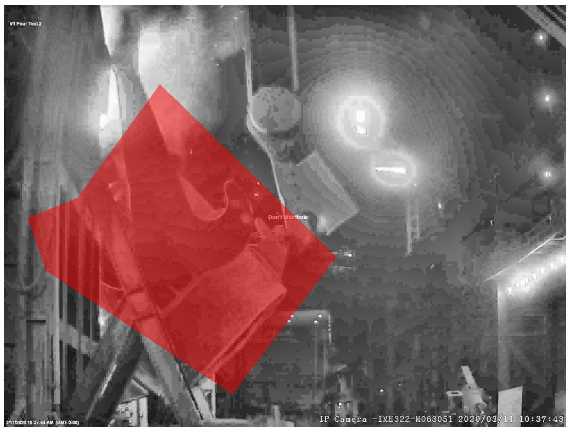
(c) Frame 92

Figure 50: Frame results from final model output with measurements displayed (2 of 2)

The pouring height is represented by the dark blue horizontal line in subfigures of Figure 49 and Figure 50. As shown, this is offset from the ladle mask. As previously discussed, this is because there was a consistent gap between the top of the mask and the actual top of the ladle and so an offset of 40 pixels was used. As shown in Figure 49 and Figure 50, this was partly due to the mask not reaching to the ladle edges. Whilst the Kalman filtering did a better job of this, it was less stable as mentioned previously.

In the top left of subfigures of Figure 49 and Figure 50, a light blue line can be seen which shows the perspective of the moving window. The line of best fit, produced from the bottom ladle edge and used to calculate the angle of rotation, is vaguely visible over the top of the light blue line in the top left of subfigures of Figure 49 and Figure 50. The light blue vertical line was used as a normal to calculate angles.


(a) Frame 11


(b) Frame 48

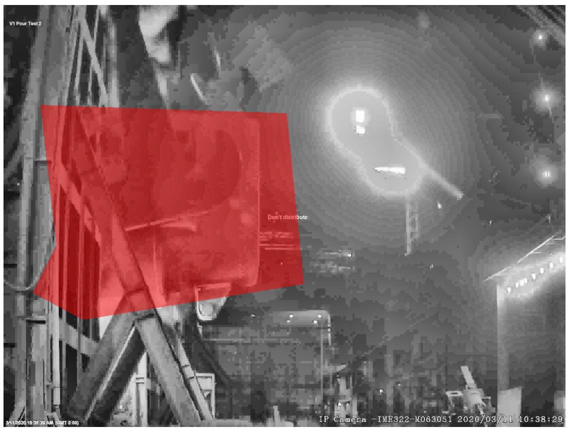

(c) Frame 92

Figure 51: Frame results from fixed mask output

Once the model had been developed to measure ladle behaviour and resulting emission severity, a fixed mask approach was attempted to try to refine the model performance further for industrial application. This was done by using the coordinates of one of the manual mask annotations and overlaying it onto each frame. The location was based on the centroid of the mask predicted by Mask R-CNN, and the rotation was based on the rotation angle already calculated from the mask's bottom edge. The concept of this is believed to be useful since the ladle shape did not change throughout the footage, however in this case, the predicted mask was not stable enough to produce an accurate fixed mask. If the predicted mask was more accurate, the fixed mask could have been used to refine the edges. The results of this can be seen in Figure 51. This is something to explore more in future work when a larger dataset is used to train the model.

4.4.2 Analysis of the Pouring Process

One of the primary purposes of the model developed is to use it to improve understanding of the pouring process. By improving understanding of process parameters, resulting process quality and the relationship between the two, valuable insights can be drawn which aid process optimisation for reducing emissions.

The measurements taken by the model were output in a table and then used to produce several graphs to draw insights from. Figure 52 shows how brightness in the flame region varied throughout the video. Note that pixels with a brightness of over 235 (on a scale of 0 to 255 representing black and white respectively), are considered bright pixels. The initially high value was due to the ladle being on the opposite side of the camera's view to the flame region. Once it moved into position to pour, the brightness in this area decreased rapidly. As pouring began, the brightness increased up until a peak at around 18 frames in, before gradually decreasing throughout the video. The initial peak was likely due to the initial contact of the liquid metal with the contents of the furnace causing a large spike in emissions. This suggests that this is a critical point of the pouring process with regards to minimising equipment degradation and emission severity.

Figure 53 shows the pouring height across frames. It increased quickly between Frame 15 and Frame 18 in preparation for pouring, and then steadily increased up to Frame 60 whilst pouring, which was likely done as the ladle became less full for a consistent pouring rate. There was a fluctuation in pouring height roughly between Frame 60 and Frame 100, which could be seen in the video. This could have been to keep the pouring process as smooth as possible whilst the ladle was almost empty and needed to be tilted more.

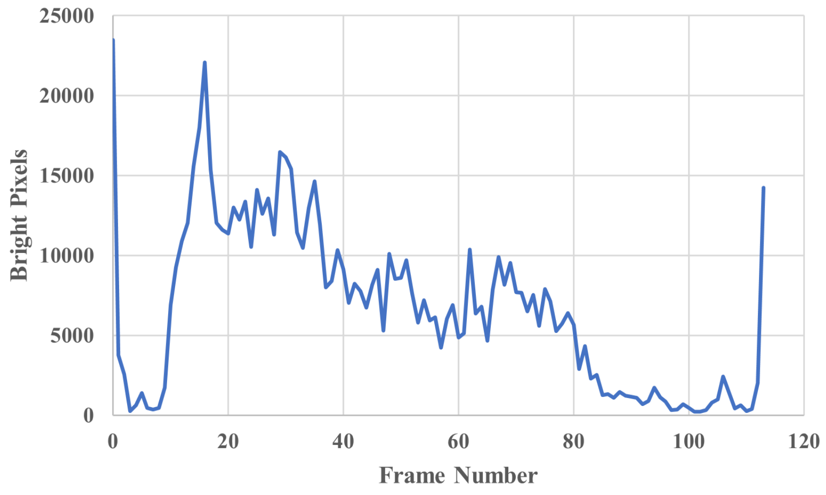


Figure 52: Plot showing the variation of pixel brightness throughout the pouring process

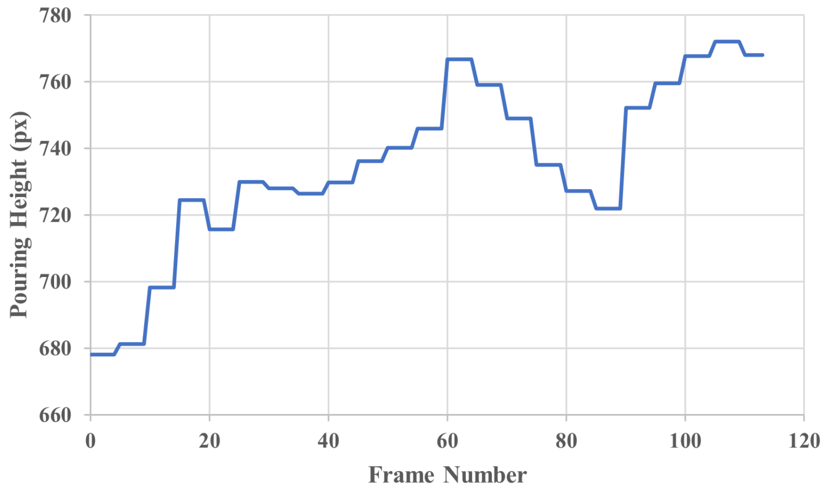


Figure 53: Plot showing the variation of ladle pouring height throughout the pouring process

This leads on to Figure 54, which shows how ladle rotation varied with time. The figure suggests that rotation angle increased gradually with time up from about 15°to 90°, which aligns with what was seen in the video. Whilst the graph suggests reasonable accuracy, the rotation angle results provide more insight later on when rate of rotation is discussed.

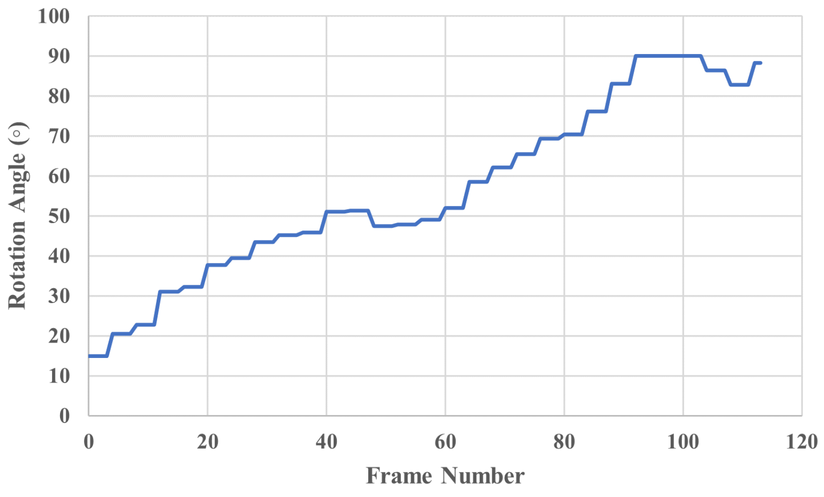


Figure 54: Plot showing the variation of ladle rotation angle, in degrees, throughout the pouring process

Collectively, these graphs suggest that the model was reasonably accurate in its measurements, since it aligns with what is observable in the footage, however the model would benefit from being validated by better established measurement methods. Furthermore, the industrial aspect of this case study was focused around building a ladle monitoring model that, if developed to production standard, could be used in two ways.

The first, more immediate application would be to apply this model to a variety of pouring footage samples with varying pouring techniques, automatically collect data on the ladle pouring behaviour (process inputs) and resulting flame emission severity (process quality indicator), and analyse the relationships between the inputs and outputs to give deeper insights into how the process inputs could be optimised to maximise process quality in terms of flame emissions.

The second, more advanced application would be to fully integrate the model into a system that also takes control of the ladle behaviour, and by doing so, automate the pouring process for minimal emission severity. Since the model, in its current primitive state, can detect when flame severity is high, it could combine this awareness with learned pouring patterns to adjust the pouring technique in real-time. In both cases, emission severity can be reduced, resulting in improved health and safety, reduced equipment degradation and overall improved operational efficiency. This benefits the environment, the economy and the operators. Further applications could include estimation of cumulative flame damage, daily emissions, process duration and possibly even energy expenditure.

In order to demonstrate the potential application of the model developed in this case study, some further analysis was conducted which aimed to gain some real-world insights on the ladle pouring process using the model outputs. The Spearman's Rank correlation coefficient was used to determine the correlation between brightness (emission severity) and pouring height and rotation angle, and the results are in Table 19. The results show that pouring height had a weak to moderate negative correlation with brightness in the flame region, suggesting a higher pouring height may reduce flame emissions. This could be due to a slightly larger cooling effect during air contact before the falling liquid metal hits the surface of the liquid metal in the furnace. The air could also improve mixing with oxygen and therefore reduce incomplete combustion which typically causes smoke. Additionally, a higher pouring height could allow the metal to reduce turbulence more before contacting the surface of the metal inside the furnace, resulting in less splashing. The results in Table 19 also show that rotation had a more negative correlation with brightness in the flame region than pouring height did, suggesting more rotation resulted in less flame emissions. This is likely to be because increasing the ladle rotation angle gradually throughout the pour, ensured a steady flow of molten metal into the furnace, therefore avoiding agitation in the furnace and splashing events. Not only this, but it likely also aided in ensuring a better temperature distribution (as an effect of steady flow rate), which further avoided agitation in the furnace. Note that in terms of minimising emissions, less agitation is better, however this is not the case for gas mixing and therefore an optimal level of agitation should be found.

Table 19: Correlation between process parameters and flame severity

Parameter	Spearman's Rank Correlation Coefficient
Pouring Height	-0.366
Rotation	-0.605

Additionally, since rotation angle and pouring speed are known to affect flow rate, rate of rotation was plotted and shown in Figure 55, as it is closely linked to both of these variables (it is the rate of change of rotation angle which directly impacts pouring speed) [244]. However, the Spearman's correlation coefficient was only 0.136, indicating flame severity increased slightly with increasing rate of rotation. This was a weak correlation and should be investigated further. It is likely that the rate of rotation had a much more significant impact than this, and one or more of pouring height, rotation angle or flame severity measurements were lacking in accuracy. It is also possible that one or more unconsidered factors contributed to flame severity, such as the fullness of the ladle, since ladle behaviour appeared to require adjustment as it became emptier. Furthermore, this evaluation was only performed on one pouring video due to the resource constraints, meaning more insights could be drawn from evaluating results on a wide variety of pouring approaches. This way, common effects such as those experienced when the ladle becomes emptier (which occurs during every pour), could be recognised and separated from other pouring parameters such as rate of rotation. Overall, with regards to industrial application, these measurements suggest that a higher pouring height reduces emission severity, a constantly increasing rotation rate reduces emission severity, and the effects of rate of rotation require further investigation.

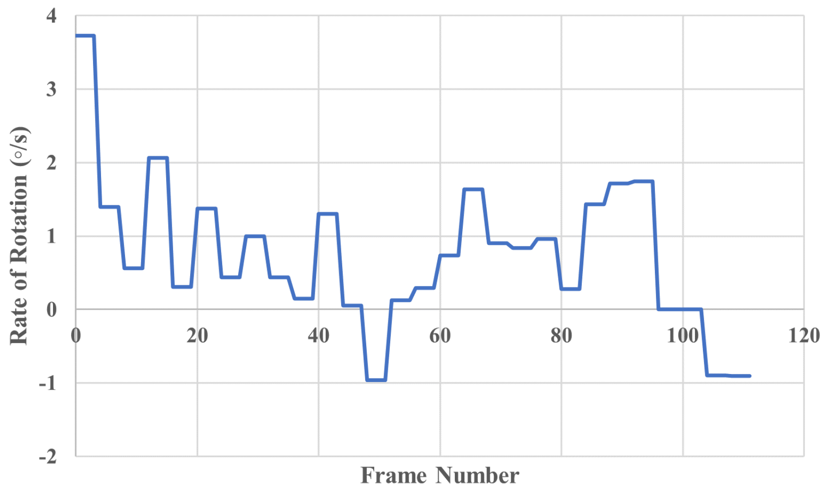


Figure 55: Plot showing the variation of ladle rotation rate, in degrees per second, throughout the pouring process

4.4.3 Model Performance Evaluation for Production-Readiness

After analysing the industrial application of the model presented, it is also important to consider the current model state in comparison to what it would require to be developed to real-world production standard. The model performance was limited mainly by the amount of data used to develop it, which was discussed throughout Section 4.3. With a minimum of a few thousand frames based on different pouring techniques in slightly different environmental conditions (e.g. lighting, dust, human presence), the reliability of the model could be improved significantly. Also, with better segmentation performance, say an mAP above 0.7 or 0.8, a fixed mask component could further improve reliability. Furthermore, ensuring that all pouring techniques are filmed from a fixed camera position, the same position intended for the production-ready system, would also increase reliability. This is different to the splatter severity measurement model discussed in Chapter 5, where for example, the model is built to be robust to changes in camera position. For a task as complex as precisely segmenting a complex ladle shape in dark and dusty environments whilst sparks, flames and smoke are emitted from it, for maximum reliability, a fixed camera would be necessary with the CV technology available today.

Another factor that affected both development and performance, was the presence of structural obstacles that partially covered the ladle as it approached the furnace. It is assumed that these are necessary, and so the model may needed to be adapted further to ensure they do not compromise performance. Other aspects such as dealing with more drastic changes in lighting, dealing with the presence of one or more groups of operators, as well as dealing with potential process failures or anomalous events, would be considered for model robustness when delivered in a production environment. It is worth noting that at the time this case study was initiated, Mask R-CNN was arguably the best available instance segmentation model available for real-world applications. However, now there are better alternatives which would provide more precise, real-time performance with much more ease such as YOLOv8-seg (You Only Look Once segmentation - see Section 2.5.6 of Chapter 2).

In the current state, it is difficult to precisely describe the accuracy of model measurements for pouring height, rotation angle and flame severity, however the results and discussion surrounding them suggest they are at minimum, an excellent foundation to production-ready systems. If they are lacking accuracy in some ways, which to an extent is quite likely, by reiterating model development with the above suggestions surrounding the likes of dataset size and camera positioning, model accuracy can certainly be improved to become precise enough for real-world application. Additionally, enhancing sensor technology by using stereo cameras or even sensor fusion approaches with camera and LiDAR for example, are likely to greatly benefit this kind of research and development.

Overall, this model lays the foundation for research and development in ladle pouring process monitoring using modern CV techniques. By using a collection of innovative methods, it shows what works and what does not work when developing a system of this kind. Also, this model is a complete evolution in comparison to traditional pouring monitoring methods such as human observation. Not only is this work a pioneering study for gaining previously uncovered insights into how the pouring process inputs affect emission severity and equipment degradation, but it also provides many insights which are applicable to other industrial applications. For example, any other processes involving large, heavy machinery with complex geometry and harsh environments such as those involving rolling, forging and casting machinery, as well as cranes and many others. If reiterated with the knowledge and experience gained throughout this case study, this model could be used for at minimum, an alarm system that notifies operators when emissions are high, as well as process optimisation using analysis insights, In its most developed state, it could be used to achieve full closed-loop control process automation whilst avoiding high emissions in real-time.

4.5 Conclusions

This chapter has presented a case study where noisy ladle pouring video data was successfully denoised and used to train a segmentation model for monitoring of process parameters and resulting process quality during hot metal ladle pouring. This work demonstrates a significant advancement in process monitoring technology through the application of traditional image processing, signal processing and modern machine learning techniques.

In this work, six different HE methods were tested and CLAHE was found to be the best across varying contrast qualities and successfully denoised good, moderately good and poor contrast quality images. Considering the harsh lighting conditions included in the data for this case study, CLAHE proved to be highly effective in mitigating this problem and demonstrates how contrast enhancement can be used to improve model robustness in scenarios where lighting is poor.

For segmentation, Mask R-CNN data pre-processing settings and hyperparameter settings were optimised via extensive experimentation using an efficient grid search approach. The optimal configuration re-trained ResNet101 from its third stage onwards, with a weight decay of 0.0001, a learning rate of 0.001, a weight loss configuration of (3, 2, 1, 3, 3) for RPN class, RPN bounding box, Mask R-CNN class, Mask R-CNN bounding box and Mask

R-CNN mask losses respectively, an RPN NMS threshold of 0.9 and a mask shape of (112, 112). This configuration achieved an mAP of 0.516.

There were a few main learning points from the optimisation such as the importance of using a sufficiently sized dataset that is also sufficiently diverse. Also, the importance of the structure and order of the optimisation particularly in relation to a complex network such as Mask R-CNN was amplified during this process. In future work, a much larger dataset consisting of several different pouring videos should be used. To obtain a larger dataset, time designated to the project would need to be increased significantly, but could also be reduced somewhat by applying an auto-labelling method once model mask predictions are quite good (similarly to Case Study 3 auto-labelling seen in Chapter 6). Note that access to a larger dataset would also be crucial in developing a production-ready ladle monitoring system. Additionally, the optimisation should be planned according to the findings presented in the optimisation discussion of this chapter. This is expected to minimise time consumption whilst maximising further findings and improving model performance further. In other words, whilst a lot was learned from this optimisation, another iteration would be highly beneficial due to the number of complexities that have been addressed during this iteration.

The optimal configuration performance was boosted to 0.606 mAP using Kalman filtering, however this had a destabilising effect on the ladle mask shape which made it erratically change across sequential frames which was negative for estimating process parameters. Kalman filtering still showed potential for future developments if harnessed correctly. One example of how this implementation may be improved is by optimising the state transition matrix for each sampled point of the original segmentation mask edge. Another example is the use of non-linear Kalman filter variants to account for the ladle rotational movements.

Applying BGS algorithms resulted in a few interesting findings. Firstly, the MOG algorithm did initially show some clear segmentation of the front edge of the ladle which could be useful for monitoring velocity, height and rotation. However, this was quickly disrupted when the scene became more complex. Nonetheless, with the correct lighting and camera setup, it may be possible to apply this in a useful way. Secondly, all algorithms showed potential in capturing the flame severity, however they are all prone to producing false positives and again, would need to be used with the correct lighting and camera setup. Also, they may need to be combined with other techniques to ensure the best quality emission capture.

Overall, the research presented in this chapter is significant to the field of steelmaking, as well as the field of CV. Within steelmaking, it is a pioneering study into ladle pouring process monitoring using cutting-edge techniques such as video segmentation. Within CV, it advances the current capabilities of existing techniques by adapting them to benefit industrial processes. Furthermore, this case study has addressed three of the research gaps identified in Chapter 2. Firstly, it has involved the generation of a novel dataset based on the ladle pouring process, which addresses the lack of datasets available for developing CV applications for steel production processes. Secondly, it has contributed to addressing the lack of real-world measurements performed by CV models. Lastly, it has addressed the lack of hybrid CV models that make use of both traditional techniques and deep learning-based techniques.

4.6 Future Work

There are a wide variety of insights drawn here such as the crucial requirement for larger datasets, to the potential benefits of changing to the current state-of-the-art networks, as well as the impact using the efficient grid search for hyperparameter optimisation has on resource requirements.

Additionally, this work opens up various future research avenues involving the automation of data collection, which includes measurement of process parameters such as cumulative flame damage, daily emissions, process duration and possibly even energy expenditure. The research here can be developed to play a major role in root-cause analysis, process optimisation, predictive maintenance and closed-loop monitoring systems. Therefore, this work can be used to guide and inspire future developments that overlap between the fields of steelmaking and computer vision.

Chapter 5: Real-Time Analysis of Zinc Splatter in Steel Galvanising

In this chapter, a novel method is presented for quantifying the severity of splatter occurring during galvanisation in real-time and in-situ using a camera. Through use of Counting (CNT) background subtraction (BGS) for splatter segmentation, combined with the YOLOv5 (You Only Look Once) object detection network for air knife detection which provides robustness to variation in camera and air knife positioning, this work is an innovative approach for splatter severity monitoring. Beyond model development, there is presentation of model deployment which proves the production-readiness of the designed system. To date, there are no other systems developed for this purpose reported in the literature, making it a substantial novel contribution to the field.

5.1 Introduction

During the galvanisation of steel, one crucial step involves submerging preheated steel strip into a bath of molten zinc. This step causes a zinc-iron alloy to form on the surface of the steel which significantly enhances corrosion resistance of the material. To ensure uniform thickness coating, a pair of air knives are used which wipe off excess zinc after submersion (see Figure 56). Ideally, the excess zinc flows smoothly down the strip and into the bath, but at high strip speeds where productivity is maximised, zinc detaches from the strip surface and sprays onto the surrounding equipment. This effect is described as zinc splatter. This means that zinc accumulates on the air knives, the electromagnetic stabilisation system and the strip itself, causing equipment degradation which leads to downtime, as well as surface defects which worsen material and energy wastage. Various process parameters such as air knife distance and air pressure affect the splatter severity and currently, severity is visually inspected by operators. This has many inherent limitations such as subjectivity, difficulties recognising splatter occurrence, and the requirement for manual effort for a single judgement. These limitations could be overcome if the splatter severity were to be measured objectively and automatically, which is the aim of this study. By achieving this, the process can be optimised for minimal splatter at high strip speeds which will lead to reduced equipment downtime and improved product quality.

The overall objective this case study was to develop a real-time analytical tool capable of quantifying the severity of molten zinc splatter occurring during the steel galvanising process. The first aim was to develop a model that overcomes the challenges of precisely monitoring dynamic fluid morphology at high speed in complex environments with moving machinery and changes in camera perspective. If deployed and implemented, this would enable operators to monitor splatter severity under varying conditions such as changes in strip speed and air knife distance, to identify relationships between process parameters and resulting process quality. These relationships could be used for optimising the galvanising process to reduce splatter whilst maximising strip speed. The second aim was to demonstrate deployment of this model.

Implementation of this work onto the line will contribute to process improvement, root-cause analysis of splatter will be more obvious, and maintenance strategies will be based on a deeper understanding of the process. Additionally, a closed-loop control system could be developed to adjust variables, such as air knife distance, in real-time depending on the splatter severity level at the current time and their effect on it. Other than previously mentioned benefits such as reduced downtime and increased product quality, this technology could be used to find previously undiscovered trends along the galvanising line, whilst also laying the foundation for similar applications on different processes. Source code is available on GitHub [245].

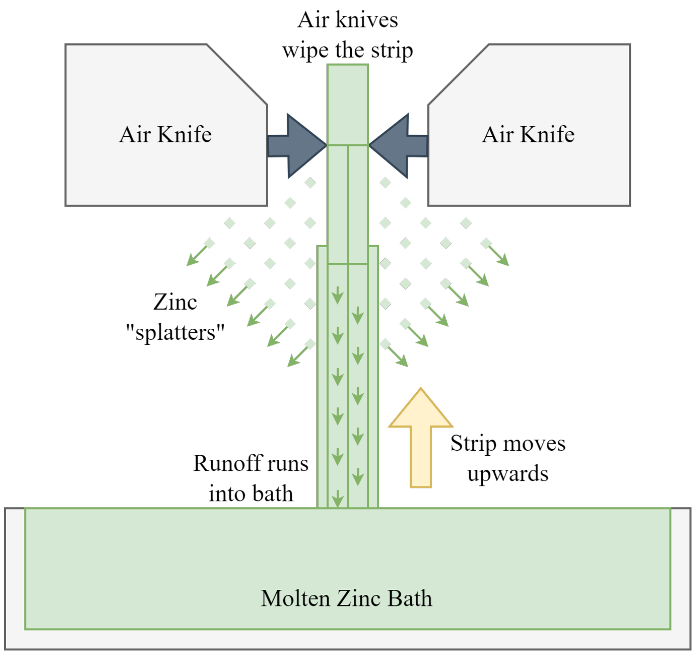


Figure 56: Schematic representing the part of the galvanising process where zinc splatter occurs

5.2 Methodology

The data for this case study was generated by recording a section of the galvanising line which is already monitored on a screen by operators 24/7. An overview of the methodology is given in Figure 57, which shows there were six main steps. The first step was data preparation, which involved labelling frames and splitting them into training, validation, testing and deployment testing sets. The data prepared in this case study addresses the gap in Chapter 2 that relates to the lack of diverse datasets available for developing computer vision (CV) applications for steel production processes. The second step was the implementation of background subtraction and denoising of the foreground mask produced. The third step involved using the foreground mask for quantifying the severity of splatter. The fourth step was the implementation of object detection, the fifth step was final validation using a range of scenarios the model will be expected to experience during application, combined with expert operator judgement against model predictions, and the final step involved the deployment of the model.

5.2.1 Data Preparation

Figure 58 gives an overview of the data strategy used for this case study, and descriptions of each source video are provided in Table 20. In VGG (Visual Geometry Group) Image Annotator (VIA), each frame was annotated with bounding boxes so that the front faces of the air knives were marked as "Knife Face", and their undersides were similarly labelled as "Knife Underside". This annotation process was applied to 30 seconds of footage from seven distinct videos (Video 1 to Video 7), each offering different camera perspectives and process conditions. Each of these videos are indicated by the blue colour coding in Figure 58. Utilising a diverse dataset ensures that model development results in robust model performance which is crucial for applications in complex environments such as those found in steel production.

With the footage running at a frame rate of 25fps, this equated to annotating 750 frames per video, resulting in a dataset of 5,250 samples for model development. This is indicated by the green colour coding in Figure 58. The dataset was strategically split, allocating roughly 80% (4,200 samples) for training, 10% (525 samples) for validation and 10% for testing, also indicated by the green colour in Figure 58. This ratio is typically successful in the field of machine learning (ML) since it ensures model training is conducted on a large enough dataset whilst reserving enough data for thorough validation and testing.

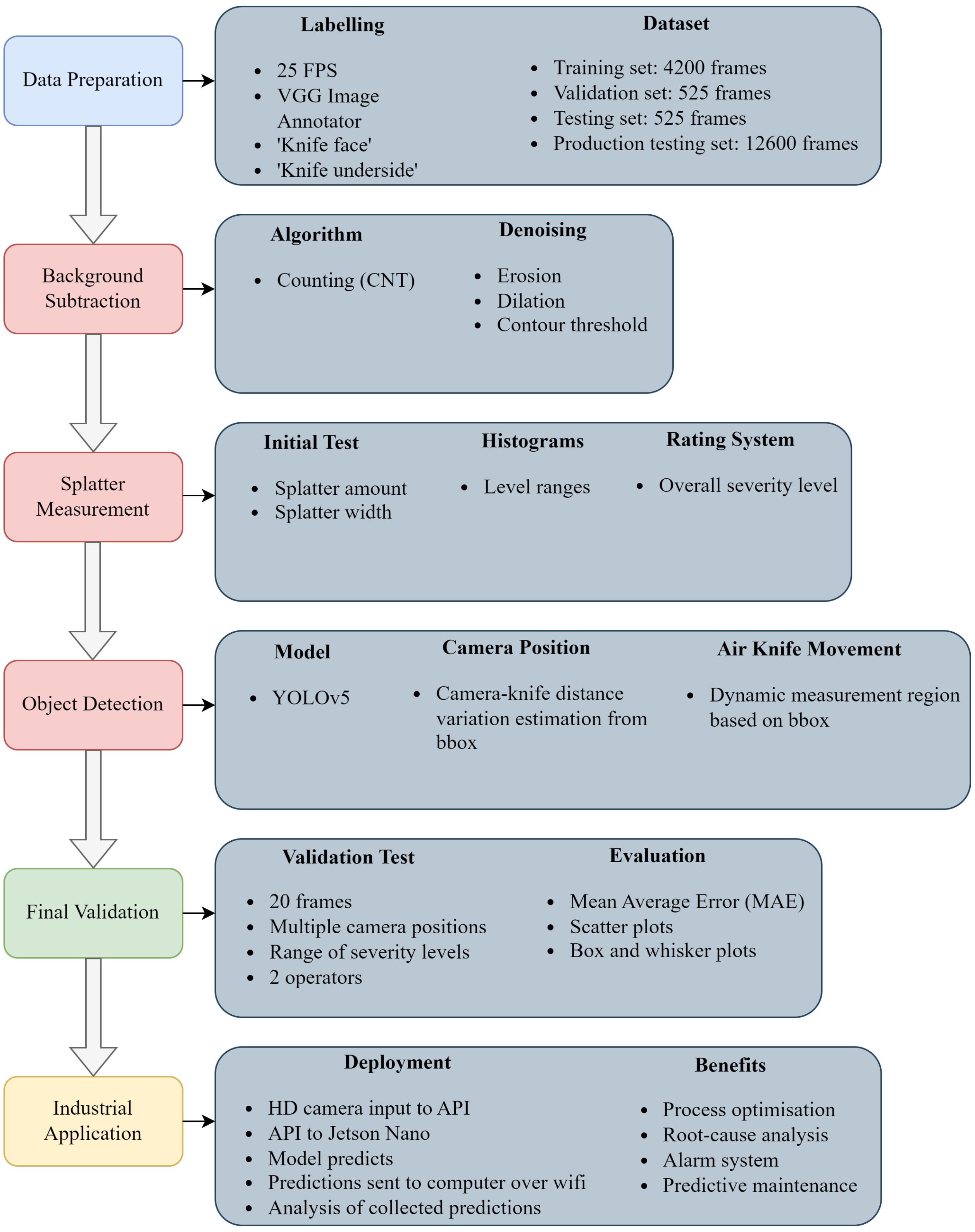


Figure 57: Overview of the methodology used for Case Study 2

Additionally, seven one-minute videos, highlighted in red, were utilised for production testing (PT) of the model. This is an extension of the conventional "testing" phase of model development seen in the field and was performed to assess the model in conditions that were as close to a production environment as possible, in order to ensure production-readiness. These one-minute videos were comprised of the same footage as the seven 30-second video sets used in the original dataset, which is indicated by the red colour coding in Figure 58.

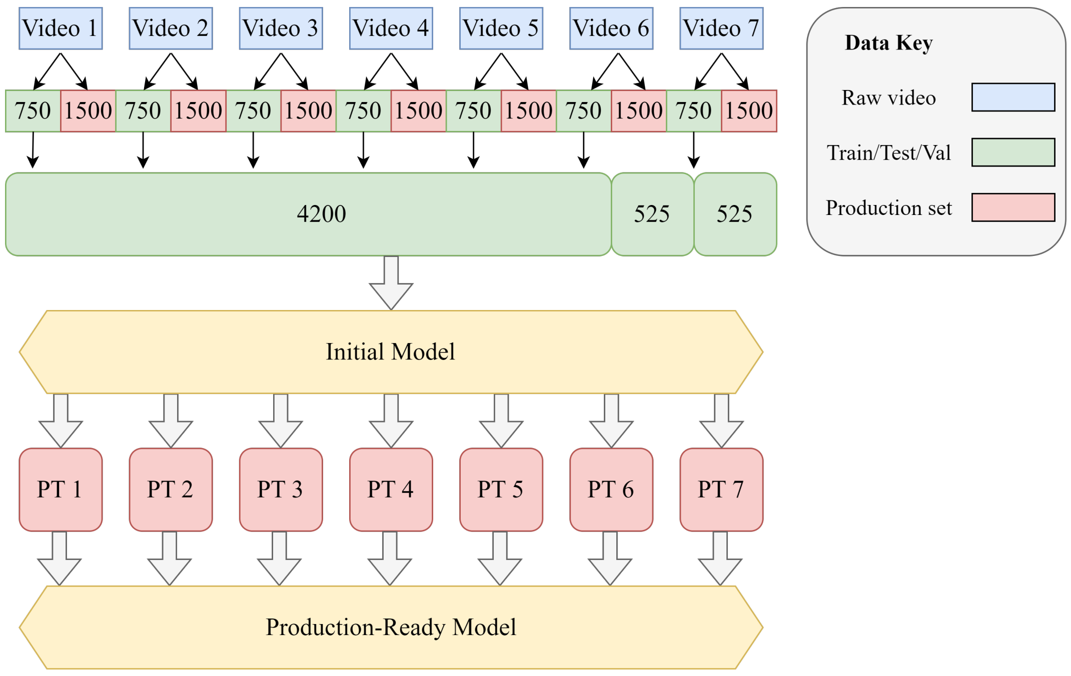


Figure 58: Diagram showing an overview of the data strategy used in this case study

Table 20: Air knife footage with varying conditions used in this case study

Name	Splatter Severity	Air Knife Movement	Camera Position
Video 1	Entire range	No	Normal distance
Video 2	Low - Mid	No	Close to knives
Video 3	Low	No	Far from knives
Video 4	Low	No	Zinc pool shown
Video 5	Low	Horizontal	Angled towards right
Video 6	Entire range	No	Angled towards right
Video 7	Low - Mid	Vertical	Close to knives

5.2.2 Background Subtraction

Following the preparation of all necessary data, the next phase involved selection of an appropriate background subtraction algorithm. Background subtraction involves a group of algorithms specifically designed to distinguish between background and foreground pixels in sequences of frames. Based on the literature presented in Section 2.6 of Chapter 2, as well as additional algorithms available in OpenCV [119], these included Mixture of Gaussians variants (MOG and MOG2), LSBP (Local Singular Value Decomposition Binary Pattern), Google Summer of Code (GSOC), Gaussian Mixture-based Background Foreground Segmentation (GMG), K-Nearest Neighbours (KNN), and CNT. The MOG, MOG2 and GMG algorithms are examples of Gaussian mixture models (GMM) which means they model each pixel over a sequence of frames as a mixture of Gaussian distributions, which exaggerates key features whilst excluding small changes such as lighting and shadows [113]. Gaussian distributions are bell-curves defined by their mean and variance, and are often utilised for modelling distributions in datasets such as those representing key features in images [246, 247].

5.2.2.1 Gaussian Mixture Models

Equation (18) calculates the probability of a given pixel having a value of x_N at time N, where each pixel is modelled by a mixture of K Gaussians [248].

$$p(\mathbf{x}_N) = \sum_{j=1}^{K} w_j \eta(\mathbf{x}_N; \theta_j) \tag{18}$$

In Equation (18), w_j is the weight parameter of the j^{th} Gaussian component, η is the probability density function and θ_j denotes the j^{th} Gaussian component parameters including mean and covariance [248]. GMMs use varying numbers of Gaussians for each pixel and this is a key aspect of their design [113]. Each component captures a different part of the background and their combined weights indicate the likelihood of different pixel values occurring [113]. For each frame, pixels not aligning with the expected background are considered foreground pixels [113].

5.2.2.2 Local Singular Value Decomposition Binary Pattern

There is limited information available on the GSOC algorithm, however it is a variant of LSBP [249, 250, 251]. LSBP is a combination of local binary patterns (LBP) and singular value decomposition (SVD) [249, 251]. LBP extracts texture-based features through neighbouring pixel comparison and encoding relationships into a binary pattern [252]. Equation (19) calculates LBP based on P neighbouring pixels at radius R, where s is the sign function used to compare intensities of the central pixel and the neighbouring pixel, g_p is the intensity of the p^{th} neighbouring pixel, g_c is the intensity of the central pixel being evaluated and 2^p is the binomial factor which corresponds to the p^{th} position of the neighbouring pixel [252].

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{19}$$

When neighbouring pixels are similar, LBP struggles with noise [249]. SVD is utilised for dimensionality reduction of rectangular matrices, and is therefore is combined with LBP to improve robustness [249]. It achieves this by exaggerating significant patterns, reducing the effect of noise, which provides improved stability of the background [249]. Equation (20) calculates SVD on a matrix B surrounding location (x, y), where U and V are orthogonal matrices, and Σ is a diagonal matrix containing singular values of $B(x, y)$ [249].

$$B(x, y) = U\Sigma V^T \tag{20}$$

LSBP begins by obtaining the LBP descriptor for each pixel using local neighbourhoods, after this it creates descriptor matrices, then it applies SVD to the matrices for principal components which reduces noise, and finally, principal components are utilised to identify the foreground and background robustly [249]. Equation (21) calculates the LSBP binary string at (x_c, y_c), where i_p is the neighbourhood value and i_c is the central value [249].

$$LSBP(x_c, y_c) = \sum_{p=0}^{P-1} s(i_p, i_c)2^p \tag{21}$$

5.2.2.3 K-Nearest Neighbours

KNN background subtraction is based on K-Nearest Neighbours, which is essentially similarity-based classification [253]. The KNN background subtraction algorithm is non-parametric and uses a kernel for classifying pixels as foreground or background [242]. The kernel is described in literature as a “balloon estimator” which has a diameter that is dynamically adjusted to cover a number of data points that is pre-defined and then adapts depending on local data density [242]. Data density is the degree of similarity between pixels in terms of different features such as colour [242] Therefore, the algorithm can effectively adapt to varying sample density, ensuring robustness to noise and gradual background changes [242]. Equation (22) calculates the non-parametric density estimate which differentiates between background (BG) components and foreground (FG) components [242]. In Equation (22), T is the number of historical frames used to adapt, t is the current time, m is the earliest frame that the algorithm iterates through until it reaches t, $\overrightarrow{x}^{(m)}$ is the RGB value of the pixel at time m, $\overrightarrow{x}$ is the RGB value of the pixel at the current time, k is the number of dataset samples X_T that lie within the hypersphere (balloon) volume V of the kernel of diameter D, and $\mathcal{K}(u)$ is the kernel function [242].

$$\hat{p}_{\text{non-parametric}}(\overrightarrow{x}|X_T, BG+FG) = \frac{1}{TV}\sum_{m=t-T}^{t}\mathcal{K}\left(\frac{\|\overrightarrow{x}^{(m)} - \overrightarrow{x}\|}{D}\right) = \frac{k}{TV} \tag{22}$$

5.2.2.4 Counting

The CNT background subtraction algorithm played a major role in quantifying zinc splatter, as well as monitoring gaseous plumes in Chapter 6. Therefore, details of the algorithm have been presented in Section 3.3.2.

All algorithms detailed in this section were implemented using OpenCV in Python. Most algorithms come with a set of adjustable parameters, although some do not offer any customisation. In order to determine the most suitable algorithm for the zinc splatter model whilst ensuring efficiency of the model development process, Video 1 was used to compare the performances of different algorithms. Performance was judged based on the precision of splatter segmentation, as well as the frame rate achieved. For parameter optimisation, there was already some idea as to which settings were likely to be ideal based on previous results with default parameter settings. Therefore, values were adjusted incrementally to balance sensitivity to splatter with resistance to noise from dust particles, heat distortion and air knife movement.

5.2.3 Splatter Severity Measurement

The optimal background subtraction algorithm produced a mask that was usable for measuring splatter. This mask underwent further processing through morphological erosion and contour thresholding to reduce noise. Morphological erosion involves thinning the external boundaries of contours. In this case study, erosion was applied to mitigate noise caused by camera movement and subtle heat wave distortions in the footage. Contour thresholding refers to the removal of contours smaller than a defined number of pixels, further eliminating residual noise. The refined foreground mask was confined to a designated splatter measurement region which was then analysed in terms of splatter amount and splatter width, both quantified in pixels. Splatter amount was the number of foreground mask pixels within the splatter region and splatter width was how widespread they were. These measurements were tracked for each frame and used to produce two histograms. These histograms were then used for the categorisation of splatter into five severity levels for both splatter amount and splatter width. An overall splatter severity score was assigned based on these levels, as shown in the splatter severity rating system in Figure 59. This scoring system operates on the premise that both splatter amount and splatter width equally influence the total severity rating, meaning the final score corresponds to the higher of the two individual severity scores.

		Splatter Width				
		0-200	**200-400**	**400-600**	**600-800**	**800+**
Splatter Amount	**30000+**	4	4	4	4	4
	22500-30000	3	3	3	3	4
	15000-22500	2	2	2	3	4
	7500-15000	1	1	2	3	4
	0-7500	0	1	2	3	4

Figure 59: The rating system used to obtain an overall splatter severity rating

5.2.4 Object Detection

After establishing a method to quantify splatter severity, it was crucial to enhance the model's robustness to changes in camera positioning. This goal was accomplished by implementing object detection to classify and localise air knives. YOLOv5 was selected for its state-of-the-art performance at the time this case study began, as shown in Section 2.3.8 of Chapter 2. Accurate detection of these knives is essential for several reasons. In the context of this research, they were used to define the splatter region, which is the area within the footage that is targeted for measurement. This was to prevent measurement of irrelevant areas that reduce the accuracy and efficiency of background subtraction. Secondly, air knife bounding box sizes were used to scale model configurations up or down based on their distance from the camera using a reference size for calibration. The relative distance between the air knives and camera was estimated using a scale factor (SF), which is shown in Equation (23) as SF and was based on the size of bounding boxes in pixels B, relative to a reference size R.

$$SF = \frac{B}{R} \tag{23}$$

The scaling factor was calculated per frame which was limited by small variations in bounding box size between every frame (since YOLOv5 predicts slightly different boxes each time). Based on this, and the fact operators only change the camera position between shifts (every few hours at least), it was more appropriate to use a moving average (MA) of the scaling factor as the final value for the model. The equation is shown in Equation (24) where X represents the SF calculated for one frame and n represents the number of averaged points.

$$MA(SF) = \frac{X_1 + X_2 + ... + X_n}{n} \tag{24}$$

The object detection process utilised YOLOv5, which required training, validation, and testing phases. Training was conducted over 30 epochs on the YOLOv5s model, utilising 4,200 samples for training and 525 samples for end-of-epoch validation (see Figure 58 in Section 5.2.1). A set of 525 test samples were then used to more accurately assess the model's generalisation performance on unseen data.

5.2.5 Final Evaluation

As model development came close to completion, validating its functionality and performance became crucial to confirm its suitability for deployment. The initial step in this process was to process seven one-minute videos through the model (see Figure 58 and Table 20 in Section 5.2.1). The resulting output videos were then visually inspected to verify that the model performed accurately across a range of potential real-world scenarios it might encounter. Following this, a practical validation test was conducted by two operators who work at the galvanising site and deal with the air knife system on a daily basis. This test required operators to assess the splatter severity across 20 carefully selected frames, characterised by diverse camera angles and process conditions. Their assessments were then compared with the ratings provided by the model to gauge its accuracy and reliability.

5.3 Results and Discussion

This section presents and discusses the results obtained from experimentation described in Section 5.2, which begins with the results of background subtraction before moving onto YOLOv5 for air knife detection. Following object detection will be definition of the splatter region and scaling factor. Finally, the results of evaluating the developed model will be presented, which consist of evaluation of performance on a set of videos with diverse environmental conditions, as well as validation of model estimations against galvanising site operator judgements.

5.3.1 Background Subtraction

Initial background subtraction results were based on the performance of each algorithm using Video 1. Only one video was required since performance differences were sufficiently distinct, eliminating the need for evaluation using additional videos. Preliminary testing indicated that variations in camera positioning did not significantly influence performance. This is likely because the algorithms are designed to handle variations by focusing on pixel-level changes as opposed to static background features. However, in Video 7, where the air knives exhibit upward movement, this resulted in a change in system appearance, therefore affecting the algorithm performance. This issue is addressed later in this section. Metrics evaluated in this initial phase were the foreground segmentation (background subtraction) precision, as well as inference time (the time taken to perform background subtraction on one frame) and the resulting processing speed, the details of which are presented in Table 21.

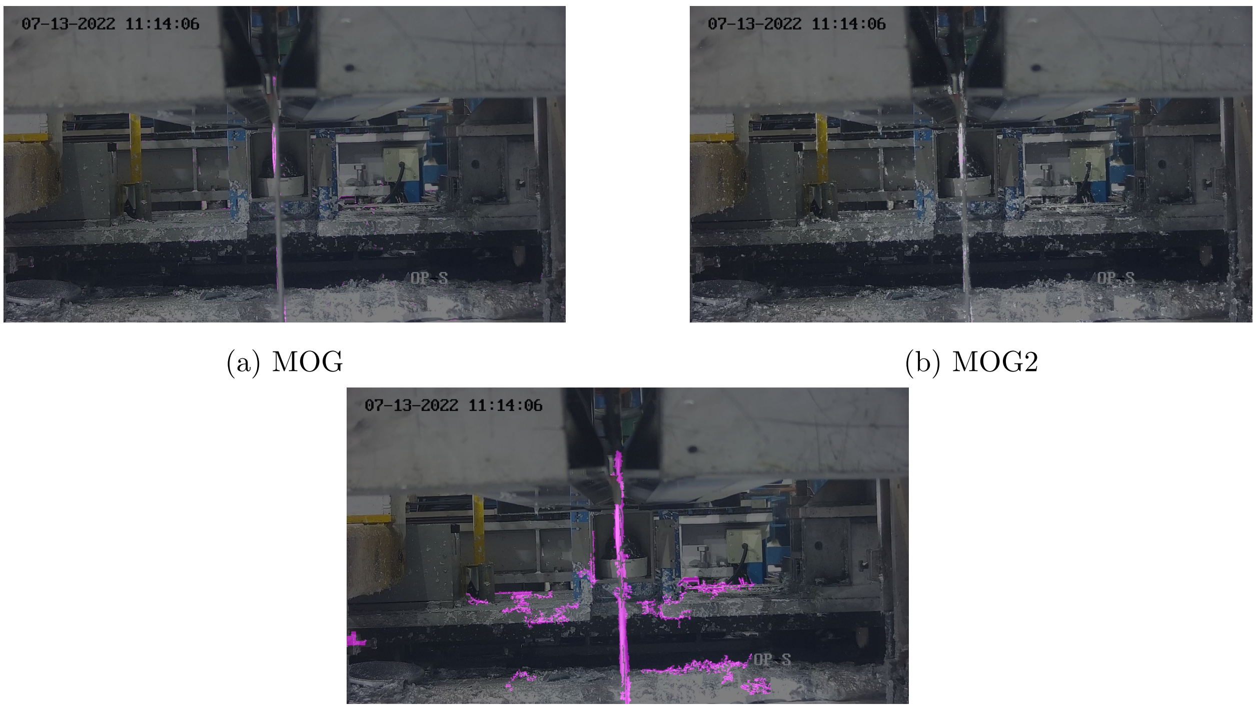
Table 21: Inference times and frame rate for various BGS algorithms

Algorithm	Inference Time (s)	Inference speed (fps)
MOG	0.095	10.515
MOG2	0.112	8.961
LSBP	1.019	0.982
GSOC	0.407	2.459
GMG	0.273	3.659
KNN	0.129	7.776
CNT	**0.051**	**19.802**

The originally captured footage operated at a frame rate of 25fps. The goal was to achieve real-time processing which meant the frames from the source could be live-streamed to the model and enough of them could be inferred upon to gain useful outputs without stopping or slowing the stream down (see Section 3.4.6). Given that each frame would undergo YOLOv5 inference, splatter measurement, and other processing steps, the speed of the selected background subtraction algorithm was a critical factor and needed to be maximised. Faster algorithms minimise processing bottlenecks and ensure that real-time analysis can be maintained. Note that the frame rates detailed in Table 21 could be significantly improved through various methods, such as utilising a GPU (graphical processing unit), resizing frames, and fine-tuning algorithm parameters. Therefore, speeds should be compared relative to one another rather than solely based on say, their ability to get close to a 25fps rate.

From the data in Table 21, MOG, MOG2, and CNT were the most promising algorithms for real-time monitoring, whilst LSBP was the least suitable by a significant gap in speed. Based on these findings, LSBP was excluded from further consideration in the final model. The next step was to look at the segmentation precision.

5.3.1.1 Analysis of Early, Low Splatter Severity Frame



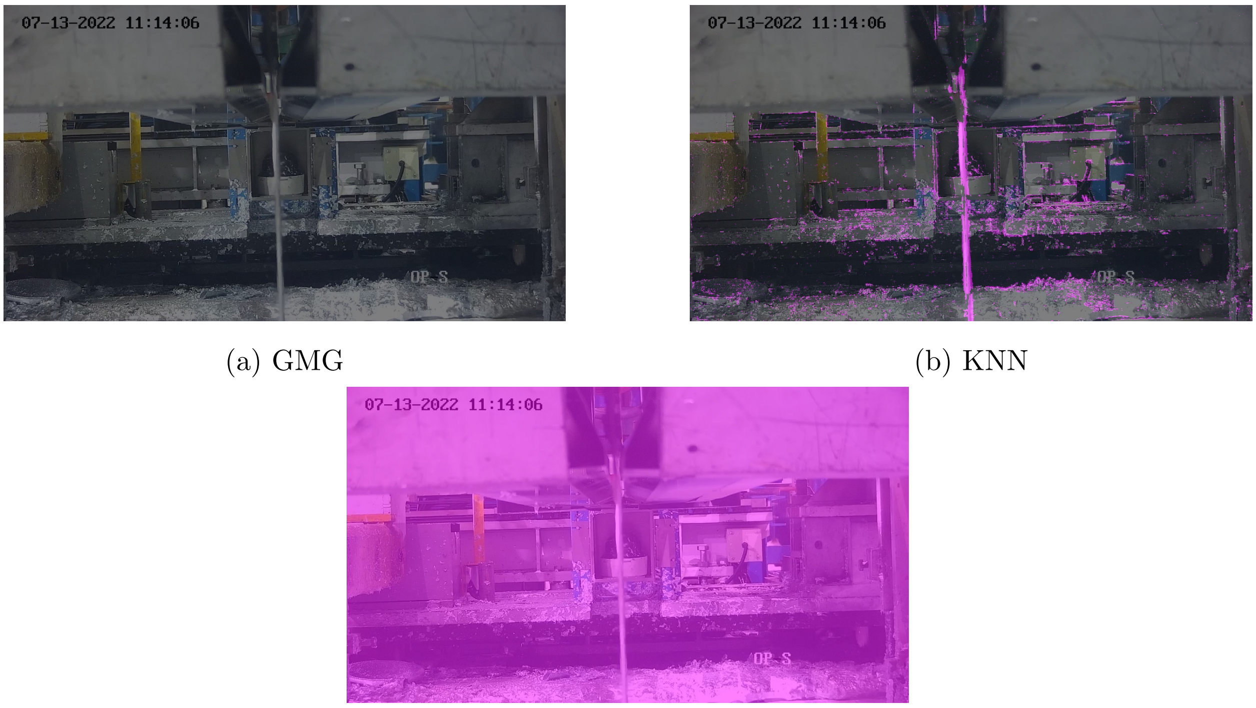
(a) MOG (b) MOG2

(c) GSOC

Figure 60: Figure showing the effects of different background subtraction algorithms (MOG, MOG2 and GSOC) on an early frame with low splatter

Figure 60 and Figure 61 show a frame during the first second of Video 1 after being processed by each of the algorithms at default settings (excluding LSBP), where splatter severity is low.

Figure 60(a) shows the MOG algorithm results where a small amount of zinc was segmented as well as a small amount of noise. Excluding the noise which is made up of many small contours and therefore easily removable using denoising techniques, the zinc segmented looks fairly precise. However, since zinc adheres to the steel strip and there is therefore not any "splatter" as such. The MOG algorithm works by modelling each background pixel as a mixture of several Gaussian distributions, which determines whether a pixel belongs to the background or foreground by evaluating how well its value fits these distributions [238]. The weights of the mixture represent relative time periods that different colours stay in the scene, and the probable background colours are the ones which stay longer and more static [238]. This approach allows the algorithm to adapt to gradual changes in the background because the Gaussian mixture model can update to reflect the most stable colours over time.



(a) GMG (b) KNN

(c) CNT

Figure 61: Figure showing the effects of different background subtraction algorithms (GMG, KNN and CNT) on an early frame with low splatter

Therefore, MOG effectively finds a balance between detecting moving objects and ignoring minor changes, which leads to fairly precise segmentation of the zinc in this scenario.

Figure 60(b) shows MOG2 algorithm results where there was no segmentation mask (fuschia) but some light grey mask was present, which was where the algorithm began to track pixel movement as shadows. Unlike MOG, MOG2 incorporates shadow detection that differentiates foreground pixels from shadow pixels [239]. This added functionality allows MOG2 to handle dynamic lighting conditions better than MOG. Distinguishing between shadows and actual foreground objects reduces false positives from shadows, leading to more accurate segmentation in environments with varying light conditions, which is shown here by detecting zinc movement without confusing it with splatter. In this case, it has detected noise as shadow but not splatter, whilst the MOG algorithm detected the noise as splatter. This indicates that MOG2 is slightly more resistant to noise than MOG. However, the MOG2 algorithm has also detected zinc that is starting to move away to the strip as shadow rather than splatter. On one hand, zinc beginning to move away from the strip is not actually "splatter" and could therefore be ignored, making the MOG2 result in Figure 60(b) an accurate representation of the severity of splatter. On the other hand, in some scenarios, it could be preferable to segment any slight disturbances along the strip, such as how MOG has in Figure 60(a).

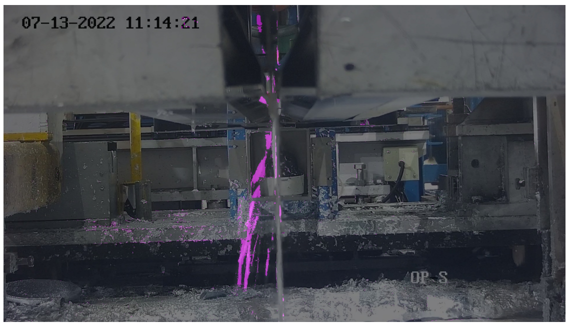
Figure 60(c) shows the GSOC algorithm results where the majority of the steel strip was segmented as well as some noise. The strip contours have much more coverage than the MOG algorithm, which could be better or worse. On one hand, the strip is moving at all times and always has some zinc moving along the surface, so more coverage would indicate better motion detection. On the other hand, it is not necessarily desirable to detect all motion along the strip, rather just the zinc that is beginning to move away from the strip. Either way, more coverage indicates higher sensitivity to movement. This aligns with the fact that the GSOC algorithm performs better on the CDNet 2014 dataset than other algorithms, which is a dataset used to evaluate the robustness of motion detection algorithms using a variety of real-world challenges such as dynamic backgrounds, camera jitter, unstable lighting conditions, and various types of motion which require an algorithm sensitive to subtle changes [240, 254]. Therefore, whilst the GSOC algorithm is designed to detect subtle changes in dynamic environments, this also makes it prone to over-segmentation. Also, whilst the algorithm has been proven to work well in complex scenarios, the particular scenario in Figure 60(c) is even more complex than CDNet samples due to the subtle differences in texture and depth of the zinc and the surrounding environment. This highlights the uniqueness and highly challenging aspects of developing CV applications for industrial environments, as opposed to more typical scenes found in the CDNet dataset [254].

Figure 61(a) shows zero background subtraction because the GMG algorithm needs a few frames to initialise. It uses Bayesian updating, essentially meaning it gives each pixel weights which are based on initial estimations from the first few frames, and then adapts them over time (similarly to Kalman filtering used in Chapter 4 which also uses Bayes' theorem [241, 255]).

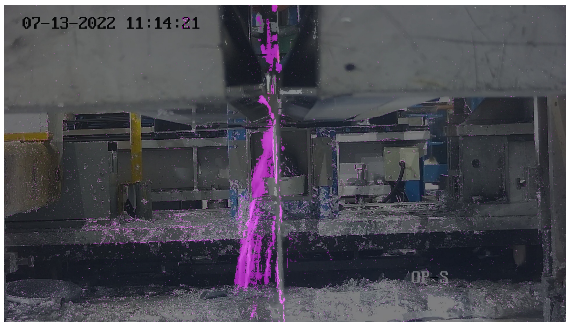
Figure 61(b) shows that similarly to the GSOC algorithm, the KNN algorithm was also over-sensitive. In this case, whilst the steel strip segmentation was similar to that seen in Figure 60(c), the surrounding noise was more widespread and more evenly distributed using many smaller contours as opposed to thicker ones seen in Figure 60(c). The nature of the noise in Figure 60(b) means it would be easier to remove using morphological operations, however there is still a lot of noise covering most of the splatter measurement zone. The KNN algorithm uses a dynamic method to analyse each pixel's surroundings [242]. It adjusts the area it examines around each pixel, depending on how many moving pixels are nearby [242]. This adaptability means it can mistake minor changes or slight camera movements as significant, especially in scenes with many edges such as this one (observable below the air knives) [242]. This approach helps capture most movement in the video, but can also make it overly sensitive to small, unimportant changes.

Finally, Figure 61(c) shows that the CNT algorithm considered the whole frame as foreground. This is opposite to the GMG algorithm that considered the whole frame as background until the algorithm had finished initialising. The CNT algorithm requires initialising because it is based on pixel stabilities, which are essentially values given to each pixel which dictate how likely it is to be part of the foreground or background. Since pixel stabilities are adapted over time, it takes at least a few initial frames for them to be established [243].

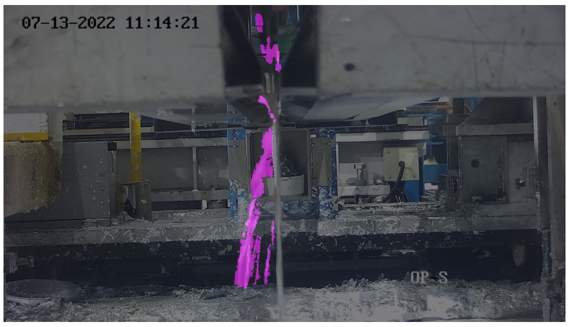
5.3.1.2 Analysis of Moderate Splatter Severity Frame



(a) MOG



(b) MOG2

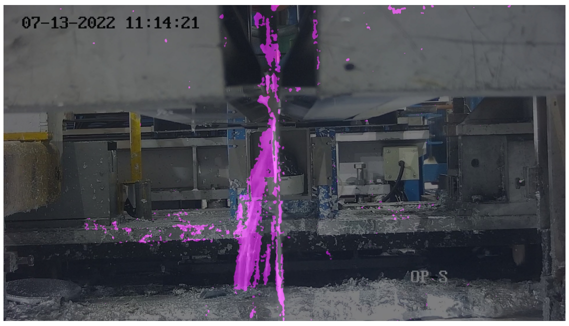


(c) GSOC

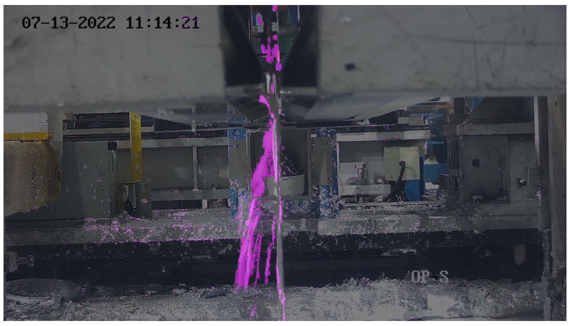
Figure 62: Figure showing the effects of different background subtraction algorithms (MOG, MOG2 and GSOC) on a frame with moderate splatter

Figure 62 and Figure 63 show a frame during the 18th second of footage processed by each of the algorithms at default settings, where splatter severity is moderate.

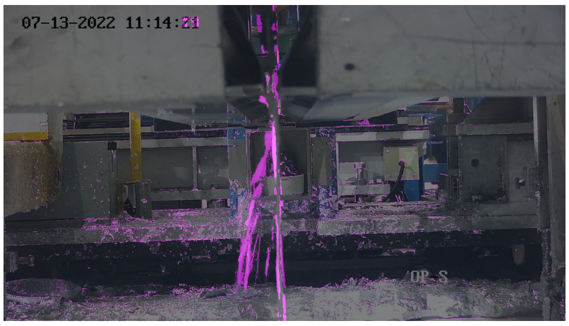
Figure 62(a) shows the MOG algorithm has successfully segmented zinc that is splattering off the steel strip is captured with good precision. As discussed, this is because



(a) GMG



(b) KNN



(c) CNT

Figure 63: Figure showing the effects of different background subtraction algorithms (GMG, KNN and CNT) on a frame with moderate splatter

the Gaussian mixture model updates distributions continuously according to changes in the scene, maintaining a balance between detecting splatter and ignoring static elements. As with Figure 60(a), there is a small amount of noise made up of many small contours that would be easily removable using denoising techniques.

Figure 62(b) shows the MOG2 result is significantly different to that shown in Figure 60(b). Whilst in the early, low severity frame in Figure 60, there were only shadows detected, in Figure 62(b) there is a significant quantity of foreground segmentation. As the algorithm processed more frames, it became better at differentiating between foreground pixels and shadows, leading to more accurate segmentation in later frames. In comparison to the MOG result, more splatter is segmented due to the increased sensitivity, however this also caused some over-segmentation, since the algorithm predicted some of the background and shadows as foreground.

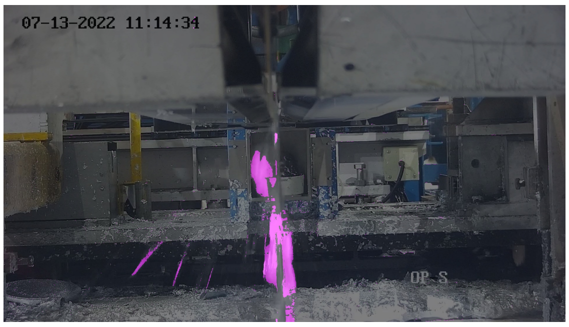
Figure 62(c) shows the GSOC result which has improved significantly from Figure 60(c). Whilst in Figure 60, most of the steel strip was segmented, as well as fairly large noise contours, in Figure 62(c) the steel strip is no longer segmented, but the splatter has been segmented quite effectively. Also, in comparison to Figure 62(a) and Figure 62(b), there are no small noise contours surrounding the splatter. The significant improvement over time indicates the strong learning capability of the GSOC algorithm. Since the algorithm is designed for complex environments such as those in CDNet 2014 [254], it is able to adapt to the scene and refine the background model, which leads to more accurate splatter segmentation as it becomes more familiar with the dynamics present in the scene. However, there is some over-segmentation between the individual streams of zinc which is, as previously explained with Figure 62(b), due to the finite details of the liquid which is a similar colour to the background colours, highlighting the difficulty of this task.

Figure 63(a) shows that the GMG algorithm had initialised by this point of the footage. The algorithm over-segmented the splatter and also produced some noise made up of contours of varying size. On the left-hand side of the image the contours are thicker and more connected, whereas on the right-hand side there are smaller contours. The over-segmentation being more extreme than in other algorithms suggests that the GMG algorithm is not as good at dealing with rapidly changing, complex scenes with subtle changes as the others. This is because Bayesian updating makes the GMG algorithm highly sensitive to changes, which can lead to over-segmentation. Varying contour sizes show that the algorithm is trying to dynamically adjust to different areas of the frame, but its high sensitivity results in more noise. The difference in left-side and right-side noise could also be due to a difference in lighting conditions. When looking more closely at the noise in Figure 62(a) and Figure 62(b), as well as Figure 63(b) and Figure 63(c), there is an observable imbalance between the left-side and right-side noises which follows the same pattern of the left-side noise being slightly more extreme than the right-side. Since Figure 62(a) shows the most over-segmentation, as well as the most extreme noise, this result suggests the GMG algorithm is the most sensitive to changes in scene and probably too sensitive for this application.

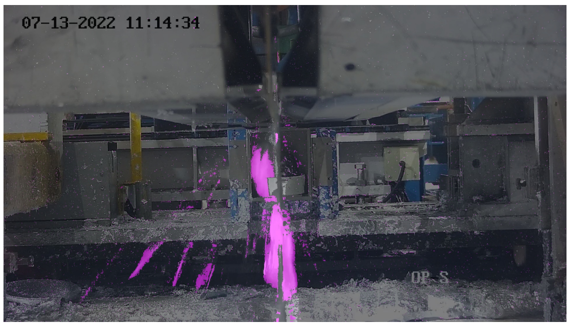
Figure 62(b) shows the KNN algorithm performed quite well with some slight over-segmentation. This is a significant improvement upon what is seen in Figure 61(b) since there is far less noise, making it much easier to remove using denoising techniques. This result suggests the KNN algorithm adapts well over time. This is likely due to the fact that the KNN algorithm changes the area of interest around each pixel depending on moving pixels nearby, and in this case of moderate splatter, has done so effectively. Despite this, it is evident that its adaptability can sometimes lead to slight over-segmentation.

Finally, Figure 63(c) shows the CNT algorithm had initialised by this point of the footage. The algorithm segmented the splatter competently with minimal over-segmentation. This is due to the pixel stability-based approach, which involves differentiating between stable background pixels and dynamic foreground objects based on how consistent their colours are. Since pixel stability thresholds were at default in this experiment, these results imply that the default values were reasonably close to optimal for this scenario. The captured foreground mask follows the natural shape formed by the splattering zinc more than other algorithms, however there is a moderate amount of noise in the form of small contours on the left and right-hand sides of the image. Since the noise is in this form, it is removable using denoising techniques.

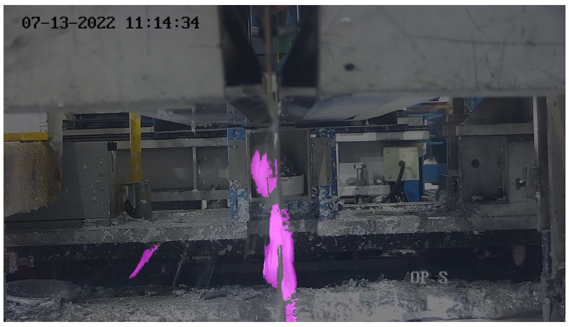
5.3.1.3 Analysis of High Splatter Severity Frame



(a) MOG



(b) MOG2

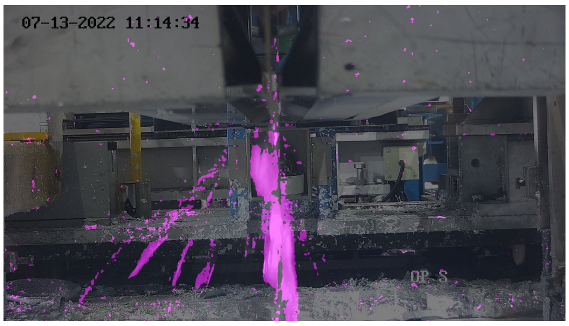


(c) GSOC

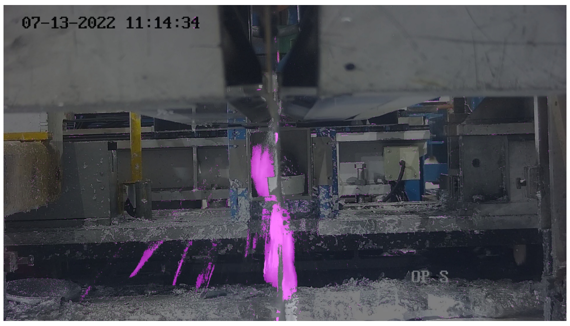
Figure 64: Figure showing the effects of different background subtraction algorithms (MOG, MOG2 and GSOC) on a frame with high splatter

Figure 64 and Figure 65 show a frame during the 33rd second of footage processed by each of the algorithms at default settings, where splatter severity is high.

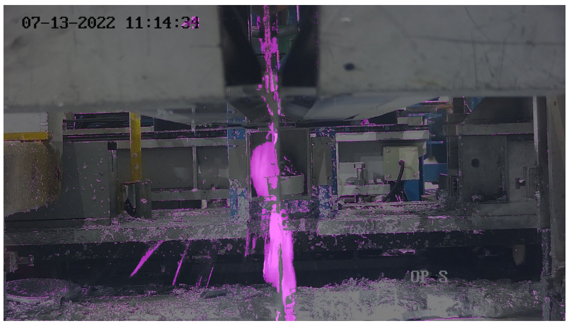
Figure 64(a) shows the MOG algorithm captured the splatter quite effectively, with some slight under-segmentation. Note that in this frame the zinc is splattering off from the strip but also moving away from it without fully splattering on the left and right sides of the strip also. The MOG algorithm has captured both the splatter and the partial separation of



(a) GMG



(b) KNN



(c) CNT

Figure 65: Figure showing the effects of different background subtraction algorithms (GMG, KNN and CNT) on a frame with high splatter

the zinc from the strip, whilst avoiding segmentation of the strip itself. This distinction is quite impressive, however as mentioned, there is some under-segmentation which can be seen towards the top-left end of the strip. High splatter scenes are typically more complex than moderate or low splatter scenes, and whilst MOG continues to effectively segment splatter by adapting to changes, the increased complexity of the scene makes it more difficult for the algorithm to fully capture all dynamic elements, which evidently can lead to slight under-segmentation.

Figure 64(b) shows the MOG2 algorithm was again more sensitive than the MOG algorithm. This resulted in slight over-segmentation of zinc as well as some small amount of noise. As discussed, the increased sensitivity of MOG2 compared to MOG can be beneficial in capturing dynamic details but can also lead to over-segmentation, which is particularly true in high splatter scenarios where there is more complexity. Based on all frames discussed in this section, it seems that MOG and MOG2 are fairly effective at background subtraction in this environment. The heightened sensitivity of MOG2 could be beneficial or detrimental depending on the use-case, however in this study the difference is not significant. Additionally, both algorithms have shown to improve their robustness to noise over time.

Figure 64(c) shows the GSOC algorithm performed very similarly to the MOG algorithm, except with slightly more severe under-segmentation. This can mainly be seen on the splattering zinc on the left-hand side, where the MOG algorithm captured three streams of splatter whereas the GSOC algorithm only captured one (which is over-segmented). Similarly to Figure 62(c), there is no scattered noise. Based on all three GSOC frames discussed in this section, it seems that whilst it is effective at adapting to dynamic environments over time and robust to small amounts of noise, it is also prone to under-segmentation. It could also be said that it is prone to over-segmentation, since in some cases it segmented between and around individual streams of splatter, which is certainly erroneous, however most algorithms were prone to this at some point due the subtle details of the splatter footage. Whilst GSOC has been proven to work effectively on complex scenes, the level of intricate detail and rapid changes in high splatter scenarios make them particularly challenging, which evidently can lead to under-segmentation as it struggles to differentiate between closely packed moving objects.

Figure 65(a) shows the GMG algorithm followed the same pattern as in Figure 63(a). There is over-segmentation of zinc that has splattered off the strip, as well as the zinc that is still not fully detached from it. There is also noise scattered around the entire frame in a more sparse distribution than other algorithms in previous frames. To an extent, this was noticeable in Figure 63(a), however here the uniqueness of it is more obvious firstly because other algorithms have eliminated most of their "scattered" noise, and secondly the noise is quite consistently spread throughout the whole image whereas in Figure 63(a) it was not so spread out. As discussed, the high sensitivity of GMG makes it prone to over-segmentation, which is amplified in high splatter scenarios since they are the most complex. From all three GMG frame results in this section, it seems the GMG algorithm is over-sensitive and is incompatible with the use-case in this case study.

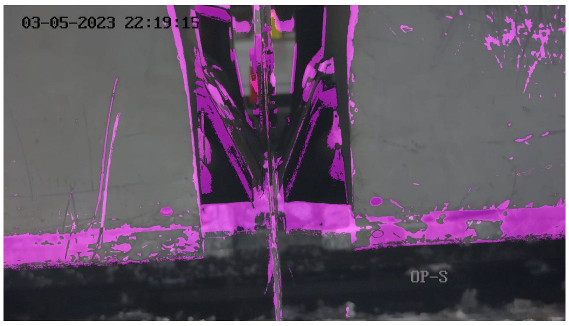
Figure 65(b) shows the KNN algorithm performed similarly to the MOG2 algorithm. There is arguably a very slight amount of under-segmentation however further research, preferably involving a contrasting coloured background, would be required to reach conclusions. Whilst the MOG2 algorithm is arguably slightly over-sensitive and definitely produces some scattered noise around the image (particularly visible on the right-hand side), the KNN algorithm is arguably slightly under-sensitive and definitely produces minimal scattered noise around the image.

This is because it adapts well to local changes which is suitable for visually complex, high splatter scenarios. However, highly complex scenarios cause the KNN algorithm to take a more conservative approach to avoid noise, which unlike in less complex scenarios (such as moderate splatter scenarios), can lead to slight under-segmentation, since it excludes some of the fine details to maintain overall precision. The results in this section suggest the KNN algorithm performs well in this scenario, becomes more robust to noise over time and may be prone to slight under-segmentation in complex scenarios (which aids noise reduction).

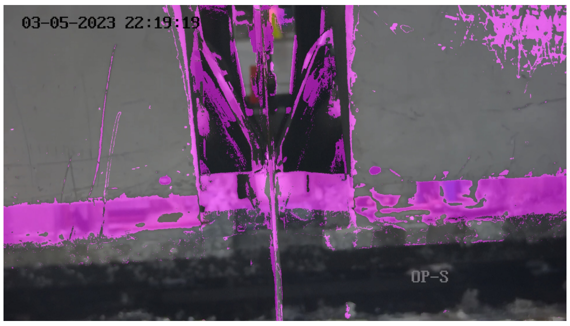
Finally, Figure 65(c) shows the CNT algorithm performed similarly to the previous frame shown in Figure 63(c). There is improved segmentation at the top-left of the strip compared to other algorithms which were prone to under-segmentation in this region. However, there was also a lot of small, scattered noise spread throughout the image in comparison to most other algorithms which had almost none of this kind of noise by this point. Still, the noise produced by the CNT algorithm was made up by many small contours and therefore would be relatively straightforward to remove. There is also some segmentation along the strip itself which is hard to distinguish as strip or zinc, but is mostly out of the splatter measurement zone anyway. The pixel stability-based approach of the CNT algorithm clearly enables it to handle high splatter scenarios effectively, since it maintains precision even in challenging regions and in comparison to other algorithm predictions, false positives are more easily mitigated using denoising techniques due to its scattered nature. The results in this section suggest the CNT algorithm performs well in this scenario, more so than all other algorithms.

5.3.1.4 Analysis of Air Knife Movement

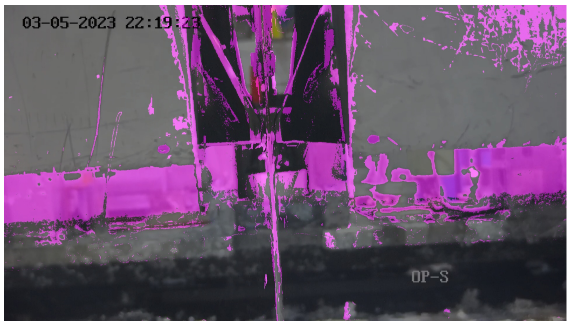
Since the CNT algorithm was the fastest by far (Table 21) and also showed the best segmentation results provided it was intended to be combined with denoising techniques, it was the background subtraction algorithm of choice for this case study. To assess the algorithm's handling of air knife movement in Video 7, it was tested with default settings. Figure 66 displays three frames at 12, 16, and 21 seconds, highlighting a noticeable issue. The CNT algorithm had established the background over multiple frames before the movement of the knives. Therefore when the knives moved, the algorithm detected this as a change, leading to inaccuracies in the segmentation mask. This inaccuracy progressively worsens across the frames shown in Figure 66.



(a) Frame 300 of Video 7



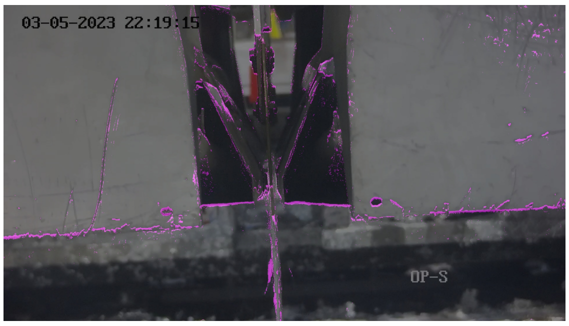
(b) Frame 420 of Video 7



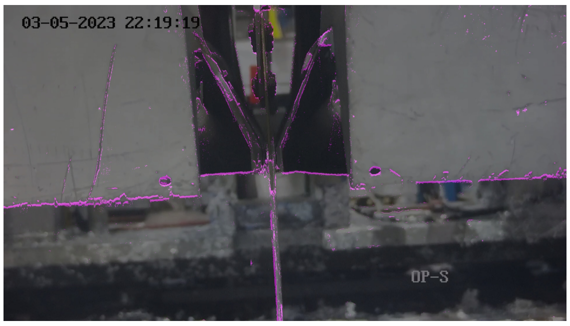
(c) Frame 540 of Video 7

Figure 66: Figure showing CNT at default settings failing to distinguish knife movement from splatter

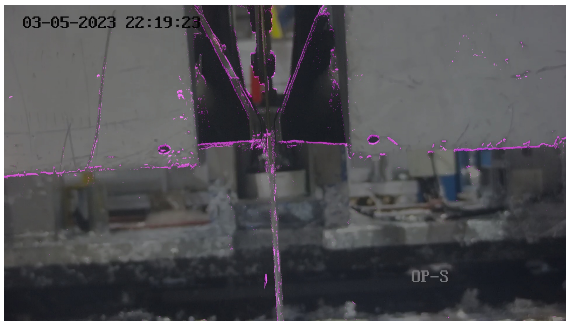
To address the air knife movement issue in Video 7, experimental adjustments were made to the CNT algorithm parameters, particularly focusing on minimum and maximum pixel stability. Minimum pixel stability is the number of frames a pixel requires constant colour for to be considered stable for segmentation, and this was manually set to fixed values within a low range (0-3) to capture the rapidly changing splatter between frames [202]. Maximum pixel stability, which is the longest possible memory of a pixel's stability, was adjusted to a value that balanced sensitivity to small changes and exclusion of occasional disturbances like air knife movement [202]. The optimal settings chosen were one for minimum and ten for maximum pixel stability respectively. As mentioned in Section 5.2.2, optimal settings were found by incrementally adjusting pixel stability thresholds from their default settings until results were satisfactory. Figure 66 and Figure 67 compare results before and after optimisation, showing improved segmentation and a slight decrease in inference time from 50.5ms to 50ms. This optimisation effectively enhanced the CNT algorithm's ability to deal with air knife movement.



(a) Frame 300 of Video 7



(b) Frame 420 of Video 7

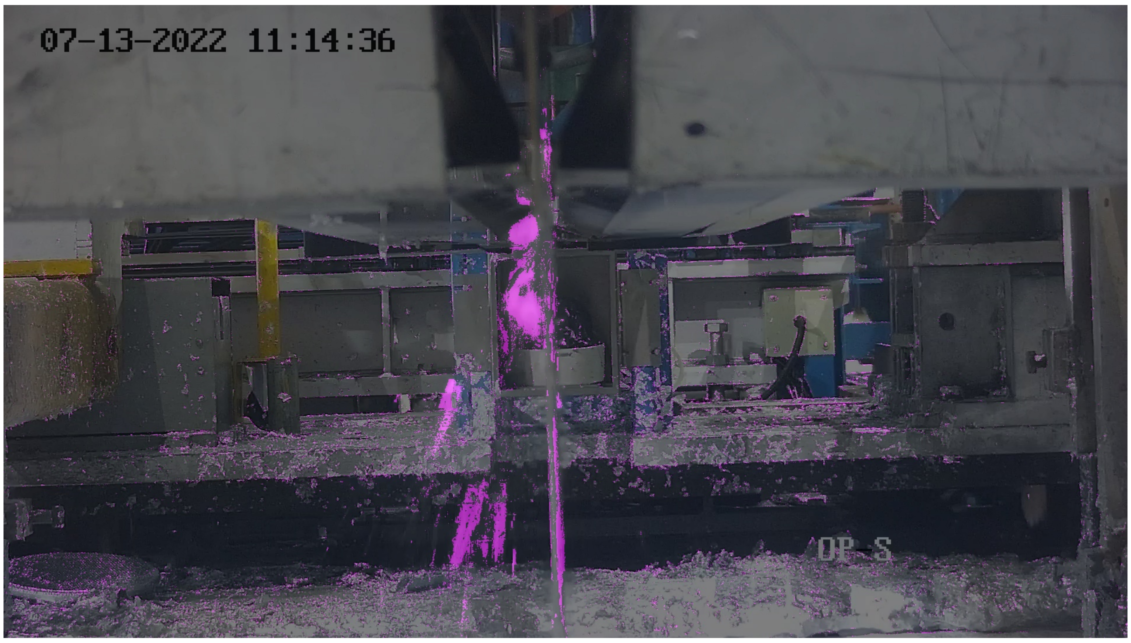


(c) Frame 540 of Video 7

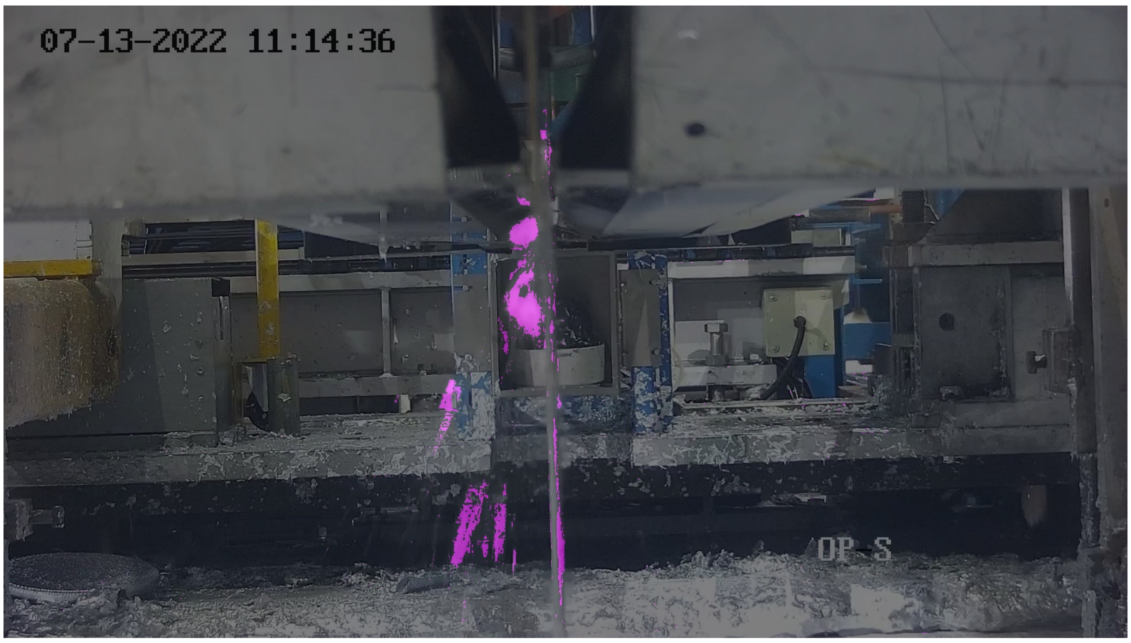
Figure 67: Figure showing the effects of knife movement on CNT background segmentation after modifying the CNT algorithm

5.3.1.5 Analysis of Noise Filtering

To tackle the excess noise caused by the CNT algorithm, erosion with a (2,2) filter was applied to the mask, removing minor background noise caused by heatwaves and camera movement. Figure 68(a) shows the result before erosion and Figure 68(b) shows the improved clarity after erosion was applied. This step significantly refined the mask. Morphological opening was considered (erosion followed by dilation), however due to there being some larger contours involved in this scenario, it was considered important to maintain these larger structures in the mask. Whilst erosion still affected them, using a small filter meant they were barely affected whilst most of the small noise contours were removed. If opening was used, the dilation step would have caused slightly more distortion of the larger contours. Dilating contours after eroding them would preserve most of the overall size of the remaining contours, but would distort the shape. Therefore, there is a trade-off depending on which approach is used and in this case, erosion was chosen for computational efficiency, since the actual difference made by the extra dilation step would be insignificant. Additionally, a contour threshold of 75 pixels was set to exclude very minor splatter contours that were widespread and could disproportionately influence the severity rating by dramatically affecting the splatter width. This ensured anomalies in the splatter distribution were excluded and the overall severity measurement accurately represented occurring splatter.



(a)



(b)

Figure 68: Figure showing the effects of contour erosion on CNT background segmentation

5.3.2 Splatter Severity Measurement

The next step after splatter segmentation was splatter measurement via splatter amount (the number of segmented pixels in the splatter region) and splatter width (the distance between the left-most and right-most segmented pixels in the splatter region). To minimise noise, the area expected to contain splatter was defined as the splatter region, which changed based on bounding boxes (discussed later). This contributes towards addressing the gap identified in Chapter 2 regarding using CV models to measure real-world characteristics.

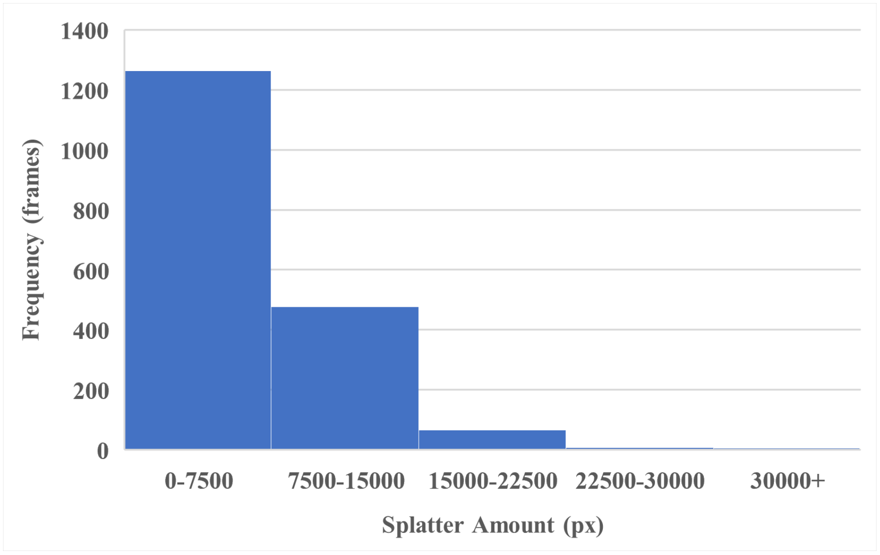


Figure 69: Histogram showing how frequent different splatter amount values occurred

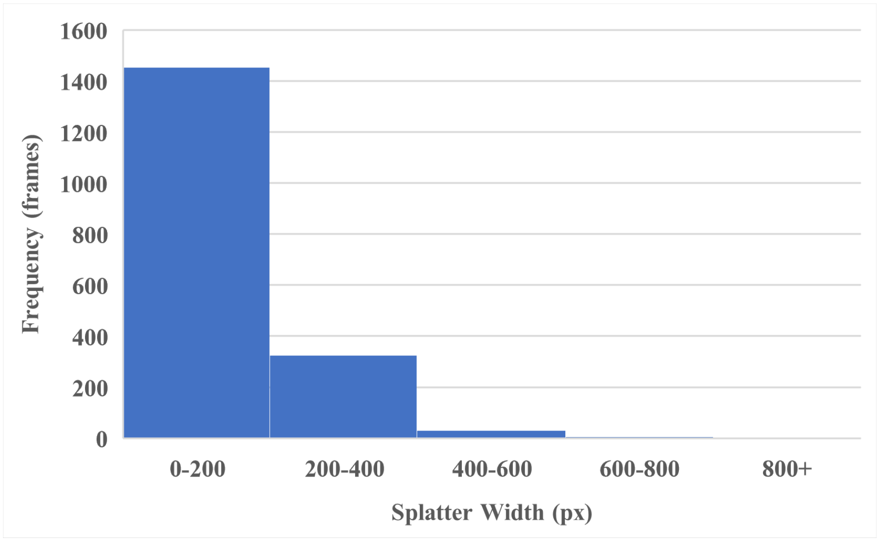


Figure 70: Histogram showing how frequent different splatter width values occurred

For every frame of Video 1, splatter amount and width were measured and plotted as histograms (Figure 69 and Figure 70). Both histograms were divided up into five ranges which each represented a different severity level for splatter amount and splatter width. Splatter amount ranged from 0 to 42655 and splatter width from 0 to 847. The majority of frames fell into the first severity level, which was the baseline state. The second range, indicating a severity of one, had fewer frames than the first but more than the subsequent levels. The third range represented a severity of two and was the outcome of significantly less frames again. The fourth and fifth ranges, representing severities three and four, contained very few frames, with the fifth having no upper limit to include extreme cases. This distribution showed higher severity levels as less common, aligning with real-world scenarios. The final splatter severity rating for each frame was determined using the system from Figure 59 in Section 5.2.3, with examples shown in the following section.

5.3.3 Object Detection

Once splatter severity measurement was achieved, integration of an object detection model was carried out. As a reminder, object detection consists of object classification (identifying the type of object), and object localisation (identifying the location of the object and enclosing it with a bounding box). In the context of this research, detection of air knives is important for defining the splatter region and scaling factor, as detailed in Section 5.2.4. The combination of background subtraction, with a modern deep learning-based technique such as YOLOv5 addresses the gap in Chapter 2 regarding a lack of applications built from combining traditional and modern techniques. The YOLOv5 object detection model was trained, validated and tested on frames where the air knives were labelled, in order to achieve automated air knife detection. The results of this model development are presented in this section.

5.3.3.1 Model Training

Figure 71 shows a loss versus epoch graph for bounding box, objectness and classification losses over 30 epochs of training of YOLOv5. Training took 14 minutes and 38 seconds per epoch, which overall took 439 minutes. All losses followed similar trends and were normal for successful training as it shows the model initially improved its ability to predict bounding box coordinates, object presence and classification labels quickly and this rate of improvement slowed down as epoch number increased. This was likely because the model initially learned more obvious features in the training data quickly, and as the remaining unlearned features were more subtle overall, training losses reduced at a slower rate. The training data consisted of all different scenarios existing in the provided footage, varying by camera position and environmental conditions (see Table 20 in Section 5.2.1). The air knives barely changed in appearance and if they did, it was due to movement. Therefore, considering a substantial amount of data was provided, including many examples of each scenario with little variation in knife appearance, it makes sense why training was straightforward.

Validation losses, which were lower than training losses, are shown in Figure 72. Typically, validation losses are slightly higher than training losses since they are based on unseen examples. This type of result could be caused by samples being distributed between training, validation and testing sets so that the validation set is easier to predict on than the training set. In this case, there is also a decreasing trend over epochs showing successful training at each epoch, and since there was not much variation of the knives in the whole dataset, these results suggest that the model learned from the training samples very quickly.

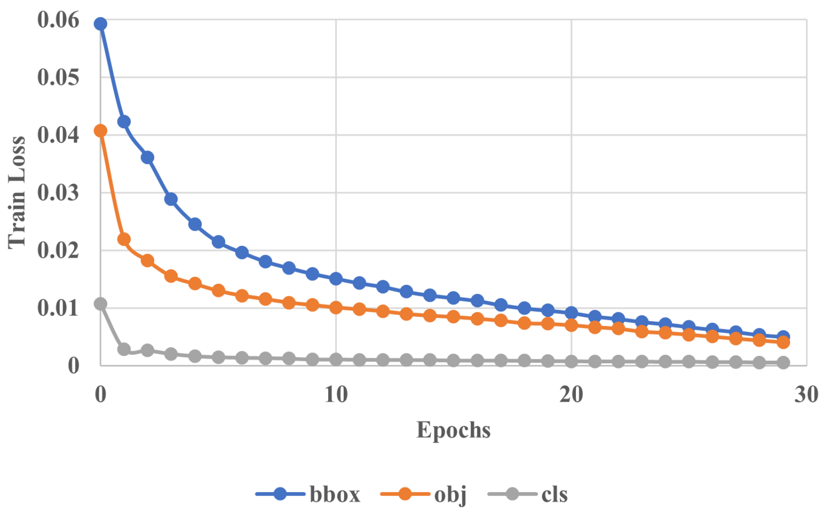


Figure 71: Graph showing how training loss changed over 30 epochs of training

Table 22 shows the precision, recall and COCO (Common Objects in Context) mAP (mean average precision) results for the intra-epoch validation which align with discussion on Figure 71 and the low validation loss shown in Figure 72. Since the dataset was large and did not vary much, YOLOv5 was able to successfully learn features with ease and therefore achieve near perfect performance.

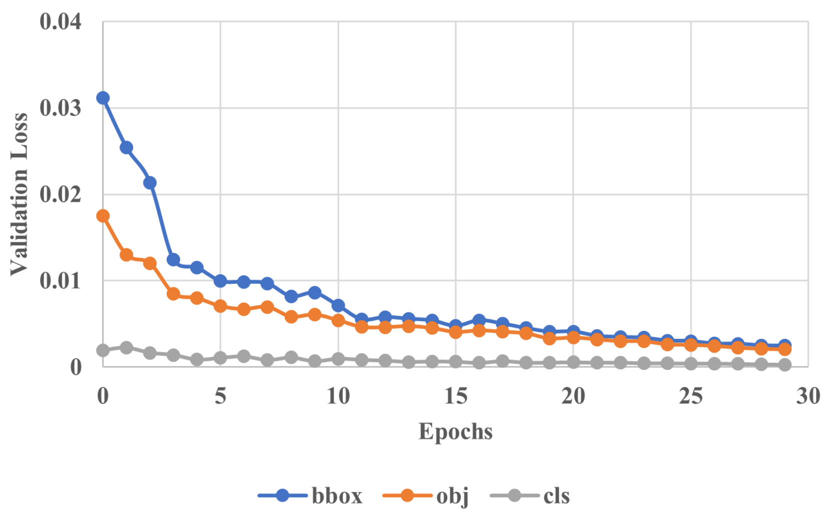


Figure 72: Graph showing how validation loss changed over 30 epochs of training

Table 22: Precision, recall and mAP results for intra-epoch validation of air knife detection

Precision	Recall	mAP_{COCO}
0.99988	1.00000	0.99414

5.3.3.2 Model Testing

Figure 73 shows model testing results in the form of a confusion matrix. For each class, the matrix shows true positive and false positive predictions, and here all predictions were correct. Figure 74 shows visualised labels and Figure 75 shows corresponding predictions, which align with the confusion matrix. As shown, the model performance is near to identical compared to the labels. This suggests the model will make good predictions during deployment. It will experience slightly different camera positions, however it has been trained on a range of different positions already and should therefore still perform well. For optimal deployment performance, the model may need to be improved iteratively based on any new camera positions.

The precision, recall and mAP results for the testing stage in Table 23 show that similarly to validation, testing resulted in near perfect model performance. Whilst the recall was perfect for both validation and testing (indicates all ground-truth labels were detected), the precision was slightly less than perfect. This indicates the model did make at least one bounding box prediction that was not in the ground-truth data. This was likely just one or a few incorrect bounding box predictions in random frames that were due to the

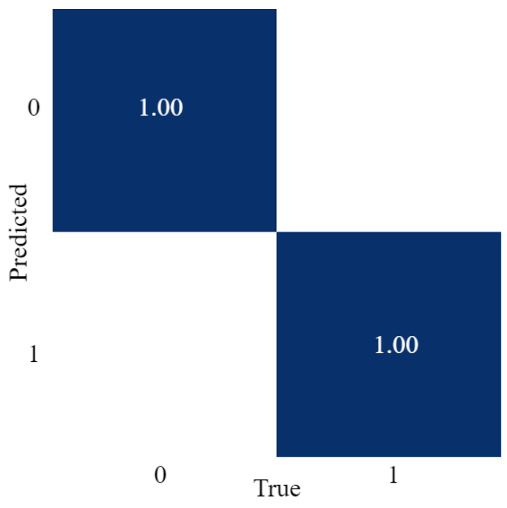


Figure 73: Confusion matrix showing 100% true positives for “knife face” class (labelled “0”) and 100% true positives for “knife underside” class (labelled “1”)

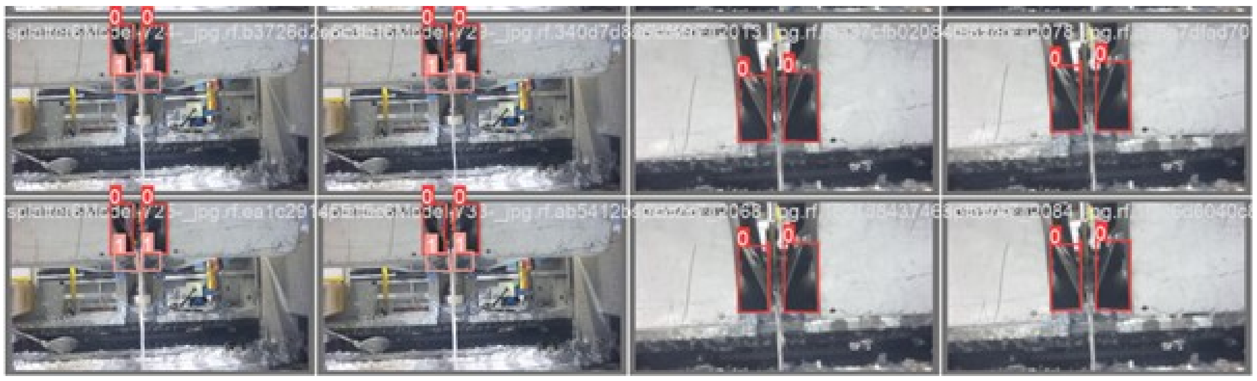


Figure 74: Visualisation of air knife labels where both classes are present on the left, and only the “knife face” class is present on the right

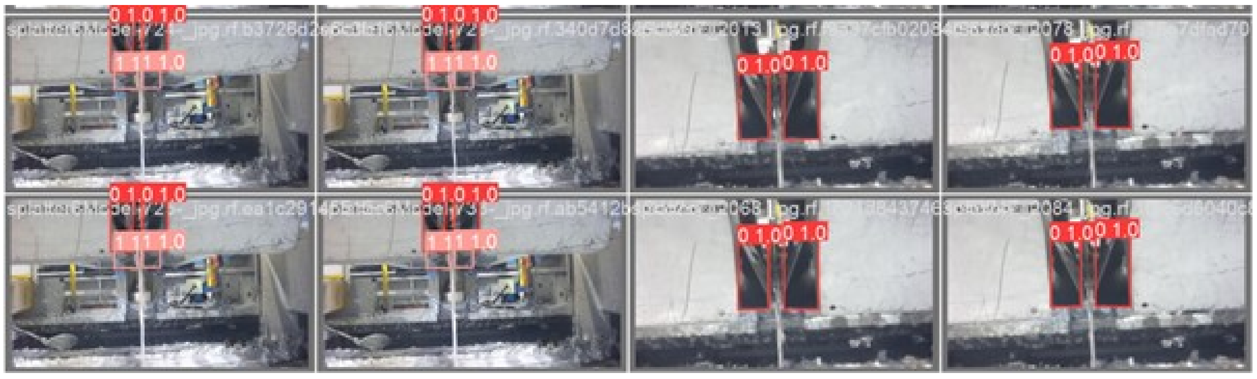


Figure 75: Visualisation of air knife YOLOv5 predictions where both classes are present on the left, and only the “knife face” class is present on the right

model recognising features incorrectly, which whilst imperfect, would not significantly impact performance during deployment (they would last less than one second). It is worth noting that taking the confusion matrix and Table 23 at face value, they do not agree. Whilst the confusion matrix indicates perfect results, Table 23 indicates slightly imperfect results. This is likely because confusion matrix values were rounded.

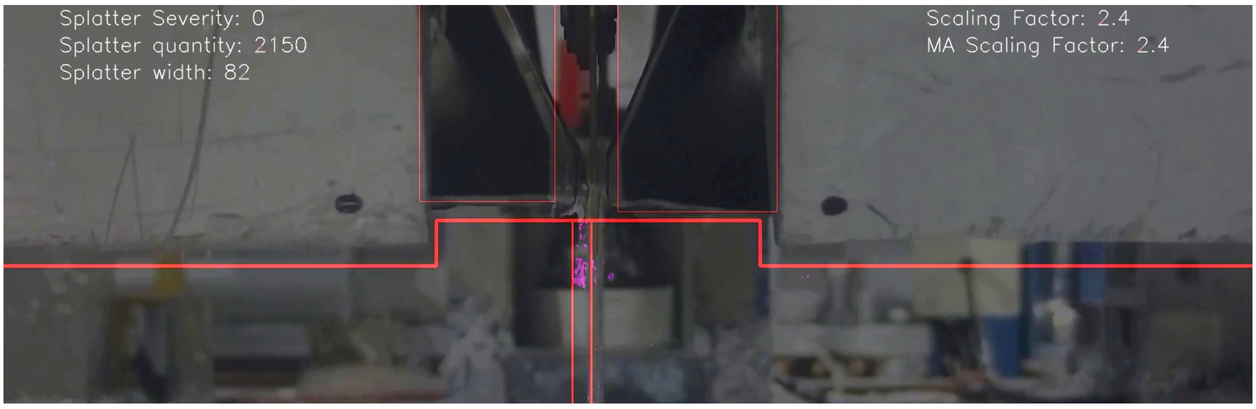
Table 23: Precision, recall and mAP results for model testing on air knife detection

Precision	Recall	mAP_{COCO}
0.99989	1.00000	0.99449

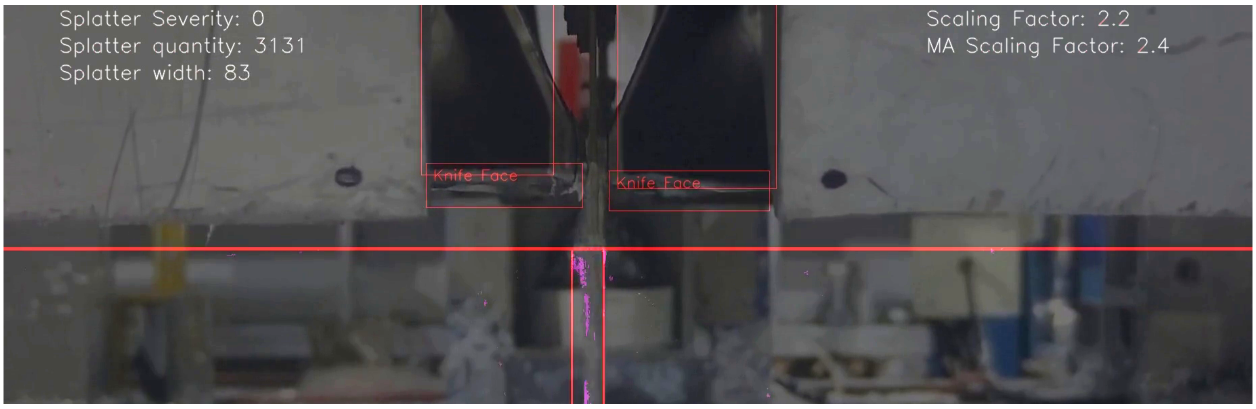
5.3.4 Splatter Region Definition

Following the success of air knife detection, it was possible to define the splatter region more precisely. Originally, the intention was to define a splatter region by a static horizontal line entered manually for each video. However, as it became known that there would be variations in camera position between operator shifts, as well as knife movement, the splatter region definition required improvement so that it could adapt to changing environments. Figure 76(a) shows the optimised splatter region line which adheres to the change in level when under the front side of the air knives. This ensured as much of the steel strip was within the splatter measurement region as possible. When underside faces were visible, there was little to no difference between the knife level and the outer level (the part the knives are attached to). Also, at some angles the splatter underside was visible at a lower point than the outer level. Therefore, it made sense to make the splatter region boundary line straight when underside faces were detected as in Figure 76(b). An example of the underside face bounding boxes sitting lower than the outer level can be seen in Figure 76(c).

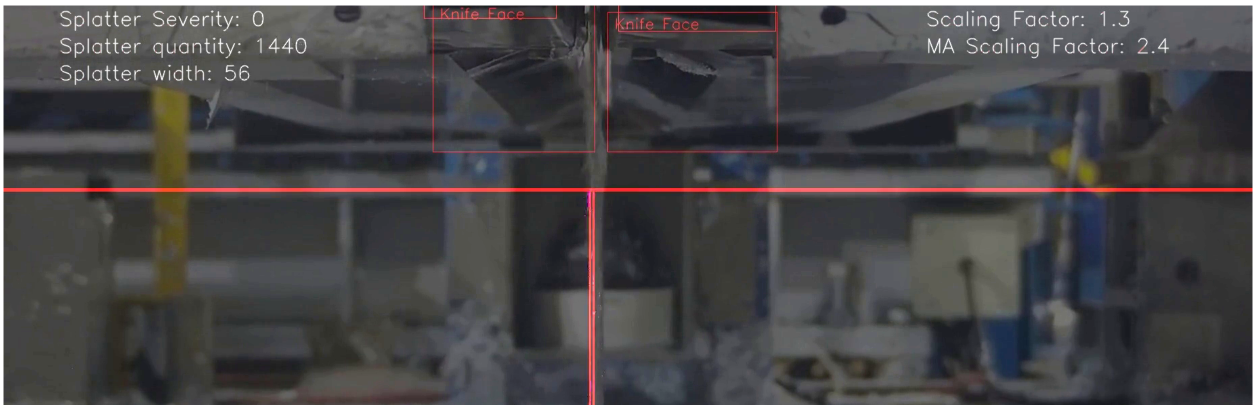
Another notable aspect of Figure 76(b) and Figure 76(c) is that the underside faces are classified as front faces ("Knife Face" as opposed to "Knife Underside"). It was found that despite the near perfect results found during model testing, the model often struggled to tell the difference between the front faces and the underside faces. This shows a limitation in the model, however it did not affect the performance of the splatter region definition, since underside faces only ever existed underneath front faces, and so the presence of underside faces could be detected based on whether there were two bounding boxes directly under the front face bounding boxes, or not. Further analysis into the YOLOv5 classification performance is a potential avenue for future work.



(a)



(b)



(c)

Figure 76: Demonstration of splatter region line (long, red, horizontal) changing depending on how the knives are positioned. In (a) it is stepped to follow the change in level between the air knives (black) and their supporting structure (greyish-brown). In (b) there is less of a level change and so the line has adapted to be straight all the way along. In (c) there is no level change and the line is appropriately straight

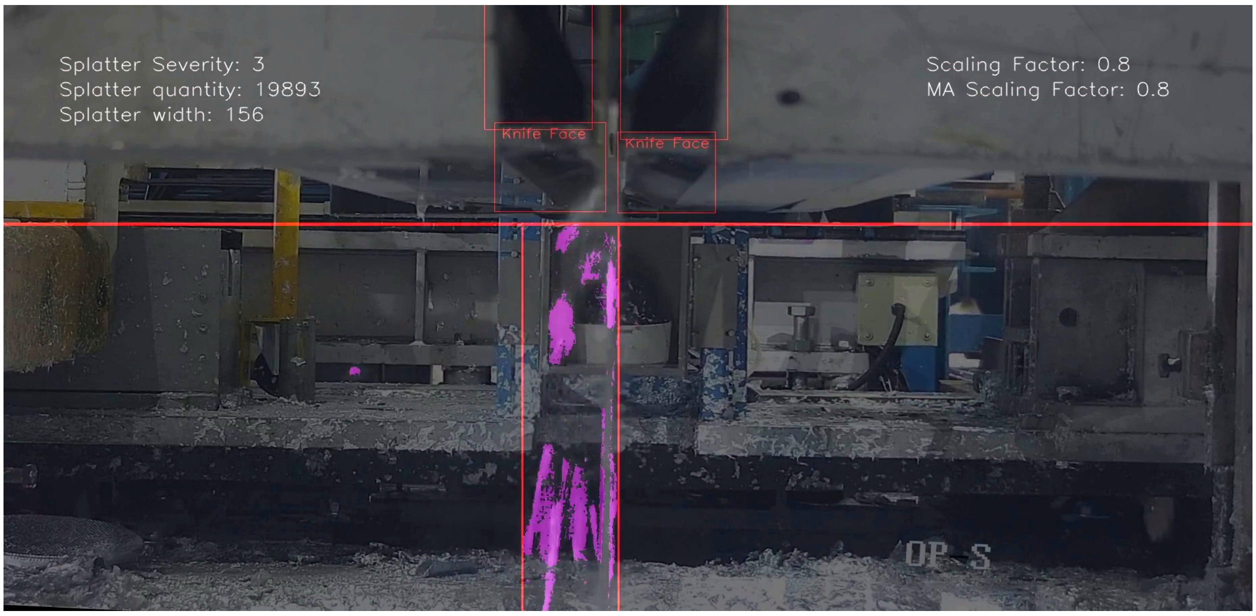
5.3.5 Scaling Factor Definition

Using air knife bounding boxes to define the splatter region was an effective aid in ensuring the model was robust to variations in camera positioning. However, the model is most reliable when the camera position is kept as similar as possible to the reference position, which is the one used for Video 1. Therefore, if the primary task of operators at any given time period is to collect splatter severity data for process optimisation, then the reference camera position should be imitated as best as possible. In future work, a camera position check could be added to automatically identify if the camera is in a similar position or not.

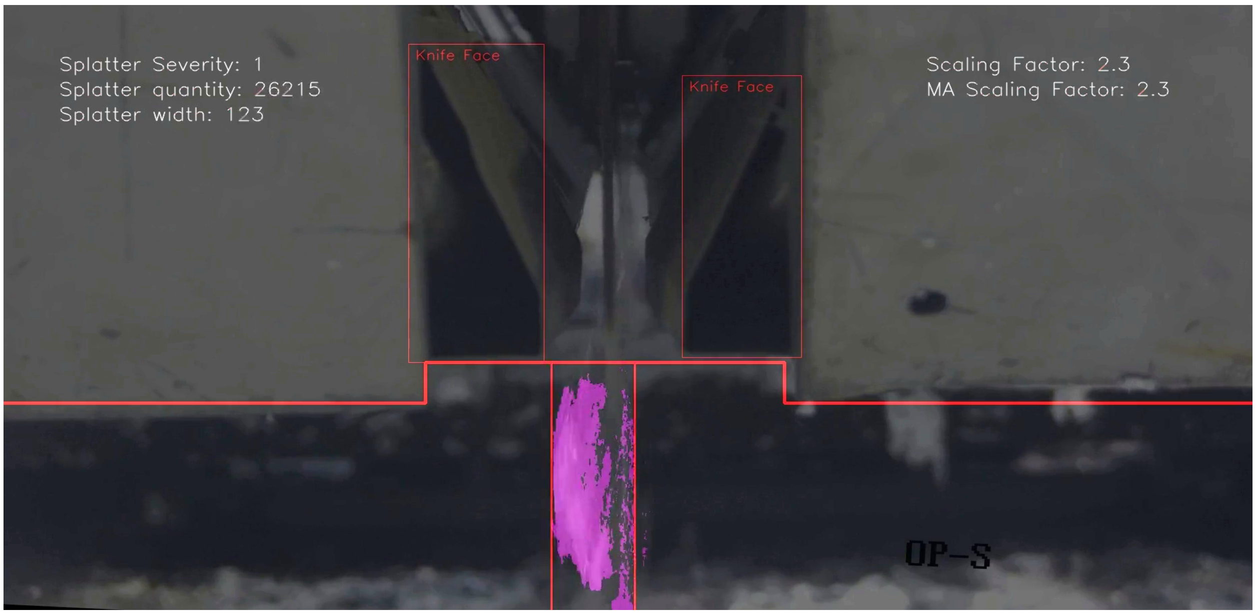
Further robustness was ensured by using the bounding boxes to calculate a scaling factor which was applied to splatter amount and splatter width severity level boundaries. The scaling factor was calculated from the pixel area of the air knives' bounding boxes compared to a reference area. Since variations in the camera-to-knives distance altered the bounding box sizes, the scaling factor was also adjusted as shown in Equation (23) of Section 5.2.4. This scaling factor was applied to the splatter amount and width thresholds to maintain accuracy in the splatter severity assessment, regardless of changes in camera distance.

Results from two different camera positions are shown in Figure 77(a) and Figure 77(b), where similar splatter amounts and splatter widths are shown, but in Figure 77(b) the severity rating is lower since the distance between the camera and knives is smaller which only gives the appearance of severe splatter.

The implementation of the scaling factor was followed by the addition of a 25,000-point moving average. This was based on the 25fps frame rate and meant the scaling factor would be refreshed approximately every 17 minutes. Under the assumption that camera adjustments by operators would occur a few hours apart, this setup provided a substantial safety factor for typical operation whilst also allowing for flexibility if the camera was to be moved more frequently.



(a)



(b)

Figure 77: Demonstration of splatter measurement boundaries adapting to changes in camera position

5.3.6 Final Evaluation

Verifying the suitability of the developed model for real-world application was crucial for evaluating its effectiveness. This comprised of two parts and the first was to evaluate the model on a range of videos with different conditions expected in production (listed in Table 20 of Section 5.2.1). The second was to validate splatter severity measurements made by the model against the judgement of operators. Both of these parts address the gap identified in Chapter 2 regarding a lack of real-world testing of models.

5.3.6.1 Video Evaluation

The model was firstly evaluated for production-readiness using the seven 60-second videos mentioned in Section 5.2.1.

In Video 1, the camera's distance from the knives was similar to the scaling factor's reference distance and exhibited all severity levels. The model's performance was largely accurate, with only a few frames showing noise-related errors. These anomalies were easily identifiable since they caused random spikes in splatter severity and could therefore be excluded or disregarded when using the model output for process optimisation. Note that even though the nature of the splatter severity is to fluctuate rapidly, this is different to the aforementioned random spikes. The difference is that when there are no anomalies from noise, the overall trend of fluctuations increases or decreases gradually, whereas anomalies cause the severity level to spike dramatically for just one or two frames from the baseline.

In Video 2, the camera was positioned more than twice the reference distance from the knives, mostly showing low severity levels but occasionally higher ones too. The model demonstrated effective performance with minimal noise-related inaccuracies.

In Video 3, the camera was positioned at about half the reference point for the scaling factor and mostly low splatter severity levels were exhibited. Inaccuracies were minimal and there were no outstanding anomalies from noise.

In Video 4, the camera was positioned at a normal distance similar to the reference distance, allowing clear visibility of the zinc pool in which the steel strip is bathed in molten zinc to form a protective coating prior to passing through the air knives. As shown in Figure 78, the zinc pool visibility led the CNT algorithm to inaccurately detect the constantly moving zinc in the pool as splatter. Whilst a method to enable the algorithm to accurately segment splatter and ignore the zinc pool is yet to be developed, the current solution for this deployment involves positioning the camera to avoid capturing the pool which is a feasible approach since the pool is typically out of view.

In Video 5, the camera was positioned about half the reference distance from the knives and the primarily low severity levels were seen. In this video, there was light reflecting near the air knives which was not observable in other videos and posed a challenge since it was incorrectly segmented by the CNT algorithm. This is shown in Figure 79 and has been considered an acceptable limitation of the model. In future work, this issue could maybe be mitigated using denoising techniques. However, this issue can currently be avoided by positioning the camera to avoid the reflective area which is a simpler and more resource-efficient solution than modifying the model. The video also displayed minor horizontal knife movement, which the YOLOv5 component handled effectively by adjusting the splatter region.

In Video 6, the camera position was close to the reference position and a range of severity levels were exhibited. The camera was angled to the right slightly which brought the reflecting light discussed in regards to Video 5 into view. Other than this known issue, the model performed well.

In Video 7, the camera was positioned at about double the distance from the reference point and low severity levels were exhibited. There was also a significant amount of vertical air knife movement ranging from the bottom to the top of the field of view. The model successfully modified the splatter region dynamically whilst air knives moved and coped well with the appearance of the underside faces.

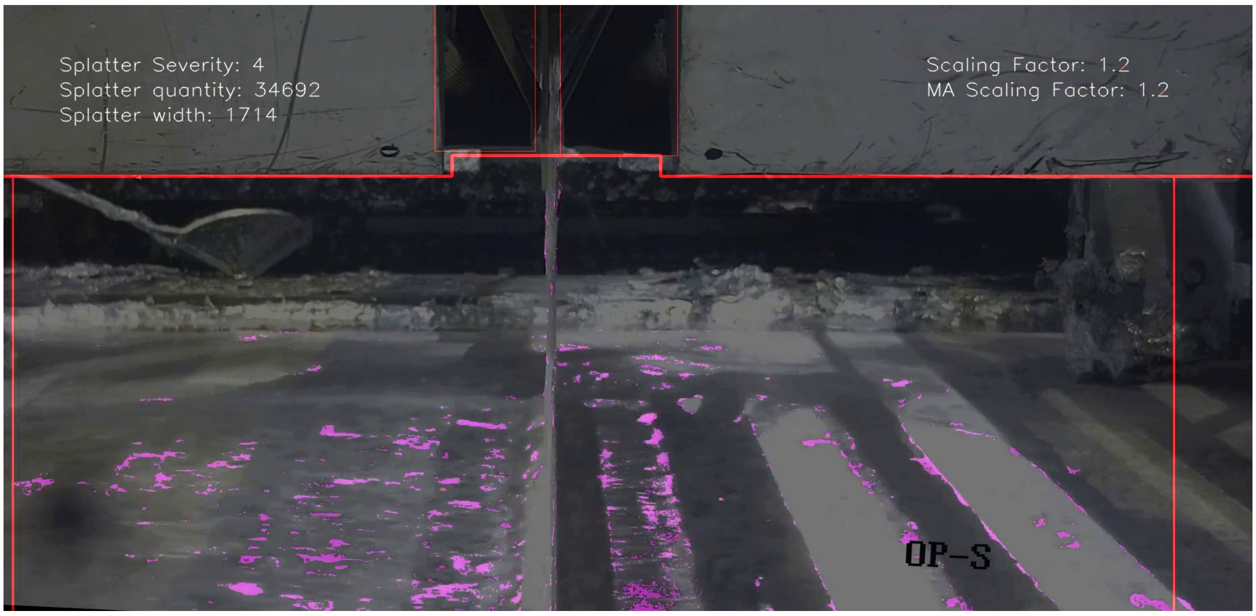


Figure 78: Frame of Video 4 showing the zinc pool effect on the CNT algorithm

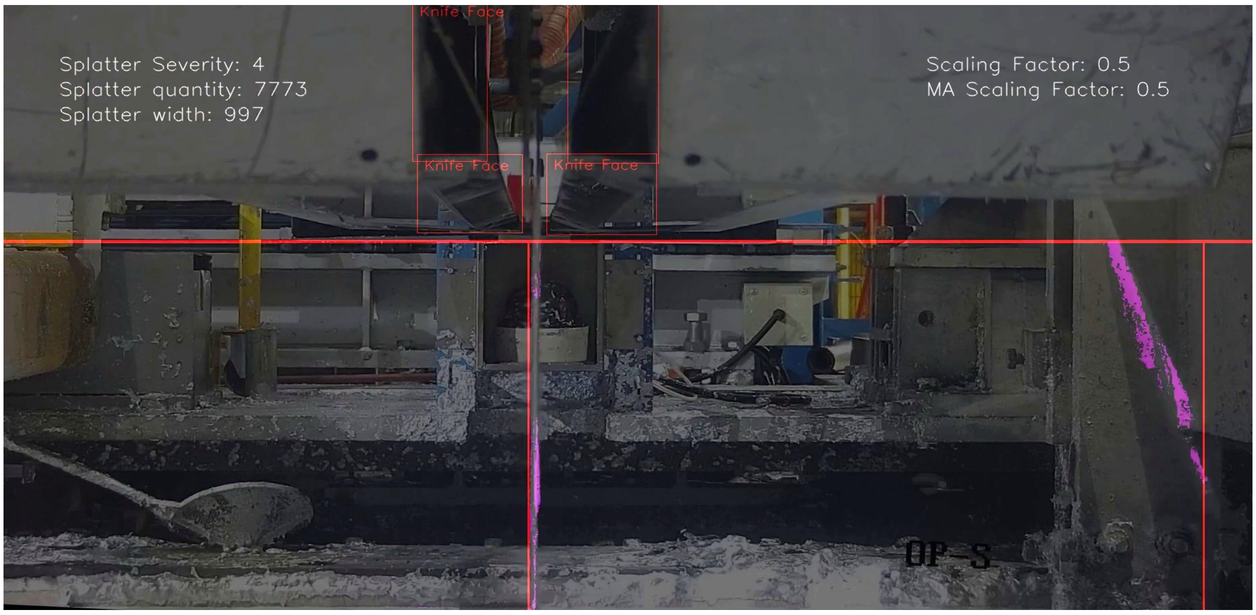


Figure 79: Frame of Video 5 showing the effect of reflecting light on the CNT algorithm

5.3.6.2 Expert Validation

Following video evaluation, splatter severity measurements made by the model were validated against the judgement of two different operators working at the galvanising site where the data was collected. The fully developed and optimised model was used to quantify the splatter severity level present in 20 hand-picked frames that represented a diverse range of camera angles and process conditions, whilst operators also estimated the splatter severity level present within these frames by eye. Operator judgement was then compared to model estimates to assess the accuracy and reliability of the model. The purpose of this validation process was to gauge the practical applicability of the model and ensure estimates aligned closely with human expert judgements.

Table 24 presents the MAE (mean average error) values comparing discrepancies between model predictions and the judgements of two zinc pot operators. The maximum MAE of the model across operators is 0.95, indicating that model predictions rarely deviate more than one level from expert assessment, which is encouraging in terms of accuracy. Interestingly, the MAE between the Pot Operator 1 (PO1) and Pot Operator 2 (PO2) is higher than between the model and PO2, suggesting either significant subjectivity in human evaluations or reliability of the model. Further testing is required to determine the significance of these discrepancies between operators, particularly to understand the influence of different datasets, environmental conditions and operator experience levels.

Table 24: Mean average error values comparing the difference between the model estimates, Operator 1 judgement and Operator 2 judgement

Model-O1 MAE	Model-O2 MAE	O1-O2 MAE
0.95	0.60	0.95

Figure 80 and Figure 81 show scatter plots which compare the model's frame-by-frame predictions with those of PO1 and PO2 respectively. The results indicate a closer alignment with PO2 than PO1, suggesting the model's assessment criteria may be more aligned with PO2's approach. However, it mostly remained between the two, suggesting model predictions may be a good reference point if there is a disagreement between operators. Model predictions fell outside both operators' judgements by one severity level in six specific frames, primarily when the camera was close to the knives. This suggests that better results were achieved when the camera was positioned closer to the reference position, which suggests the scaling factor design was sub-optimal and would benefit from more design iterations. Furthermore, of the seven close-distance frames that were used in the validation set, five of them were part of the more discrepant judgements.

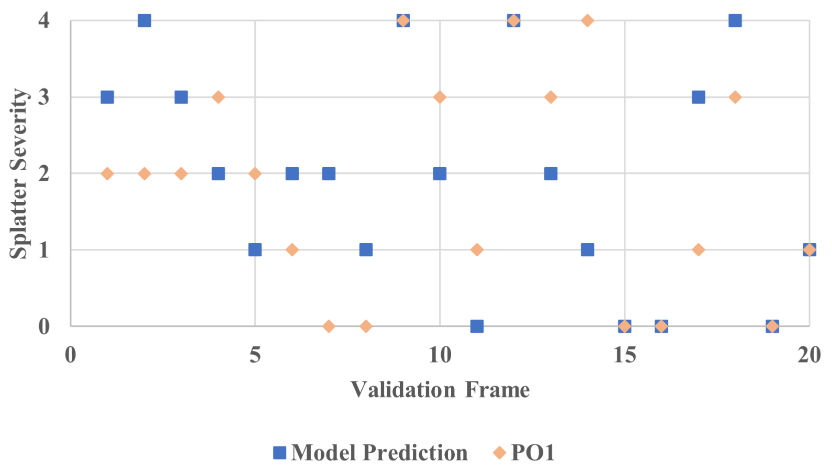


Figure 80: Scatter plot showing model predictions compared to PO1 over 20 hand-selected validation frames

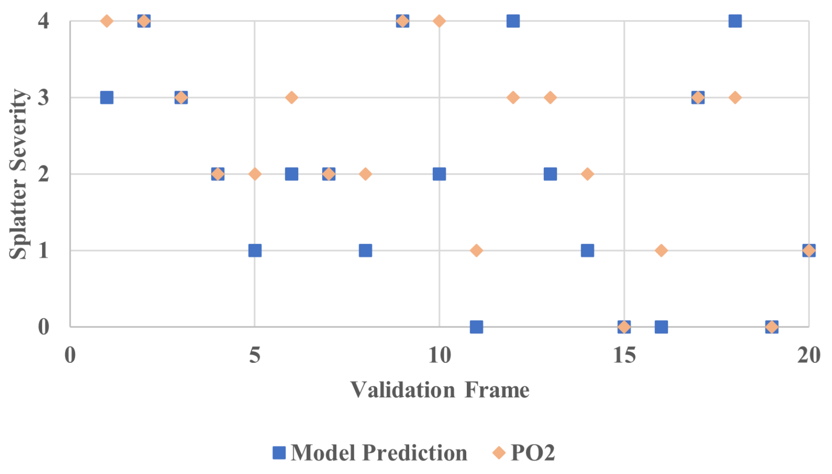


Figure 81: Scatter plot showing model predictions compared to PO2 over 20 hand-selected validation frames

Frame 11 of the expert validation set is shown in Figure 82. Whilst the model predicted a splatter severity level of zero, PO1 and PO2 judged it as one. Figure 82 shows there was not much splatter, however not much of the zinc was detected by the CNT algorithm anyway. It is debatable whether the model or the pot operators were more accurate and there was bias built into the model during the development stages when severity boundaries were decided. In this case, a small amount of splatter occurred and it is reasonable to consider this as level one severity such as how the pot operators did, however this was not how the model was designed and therefore resulted in this discrepancy. In general, in some cases the model may struggle to differentiate between minimal splatter and no splatter, since this requires a certain degree of precision. However, in most cases the model is expected to cope well.

Frame 14 of the expert validation set is shown in Figure 83. Whilst the model predicted a splatter severity of one, PO1 and PO2 estimated it as four and two respectively. This was the most significant discrepancy since the gap between PO2 and the model was three severity levels. Results show PO1 and the model differed by one severity level, which was probably due to the scaling factor multiplier. Also, PO1 and PO2 estimations differed significantly. Considering what is shown in Figure 83, at the same time as the fact that the model did not disagree with any operator judgement in any other frame by more than two severity levels, it is reasonable to assume that the value given by PO1 was misjudged.

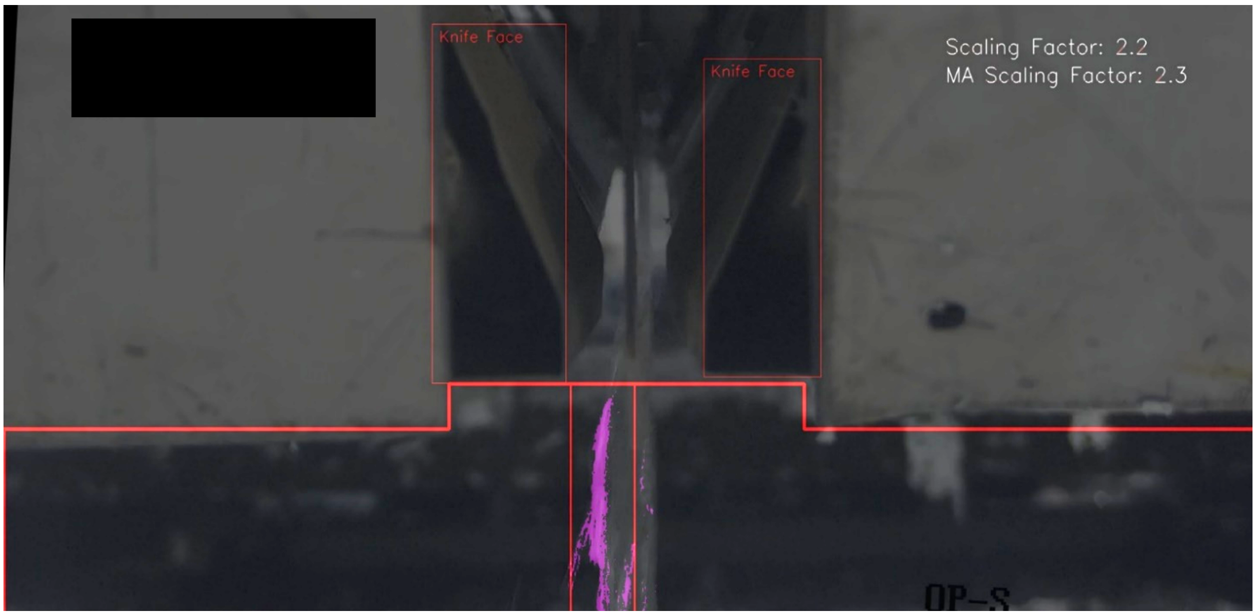

[redacted]

Figure 82: Frame 11 in the expert validation sample set where a discrepancy was found between operators and the model

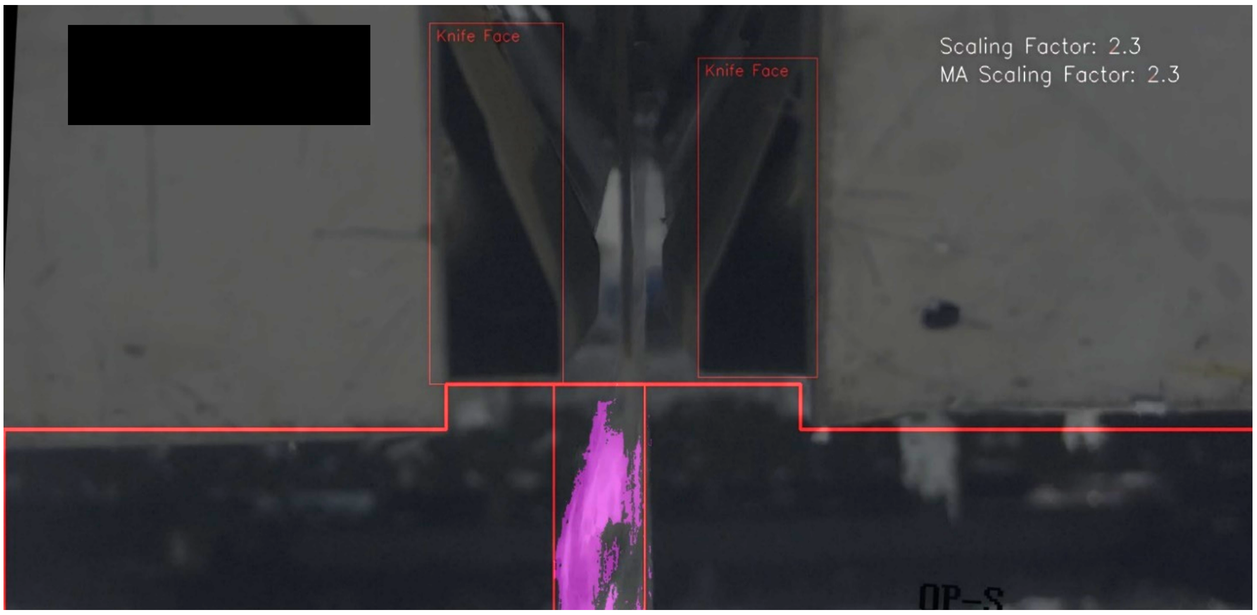

[redacted]

Figure 83: Frame 14 in the expert validation sample set where a discrepancy was found between operators and the model

Frame 18 within the expert validation set is shown in Figure 84. This frame's results were the only ones that showed clear discrepancy with operators whilst not having a scaling factor of 2.3. Whilst the model predicted a splatter severity of four, both operators judged it as three. This appears to be due to built-in bias when deciding on severity boundaries. The model was developed to consider this splatter amount as exhibiting the highest severity level, whereas the operators considered it second highest. By observing Figure 84, it is fair to state that the operators were more accurate as this is not as extreme as many splatter events occurring throughout the data. Multiple cases of this analysis suggest the scaling factor may need to be readjusted throughout design iterations. However, to be certain that it needs adjusting, extensive investigation into the expert validation set samples of choice, environmental conditions whilst operators completed the validation tests, and the pot operator judgement would be required.

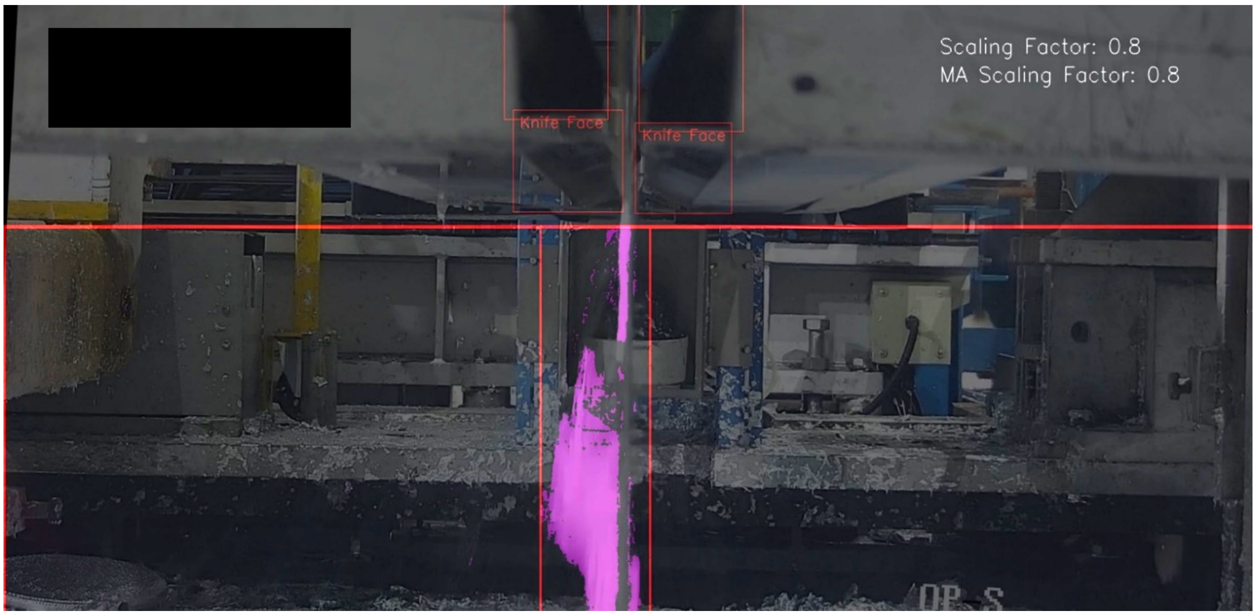

[redacted]

Figure 84: Frame 18 in the expert validation sample set where a discrepancy was found between operators and the model

Figure 85 shows a box and whisker plot which explores the relationship between model and operator judgement more deeply. Boxes represent the interquartile range of each set of results and whiskers represent the range of possible values which remained constant. The model prediction distribution was between operator distributions, as shown by the boxes. Model and PO2 were most similar for interquartile range and the model and PO1 were most similar for median. This suggests the model generated similar predictions to operators, demonstrating its potential for robustly standardising splatter severity measurements.

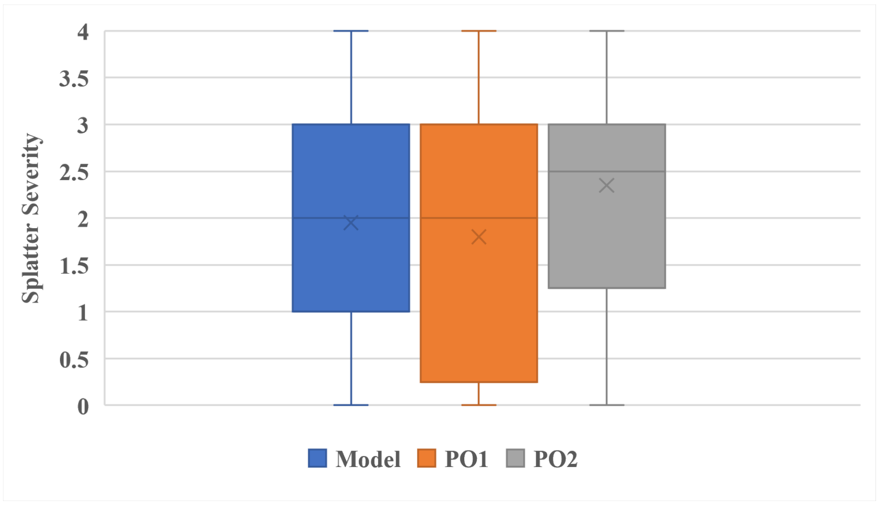


Figure 85: Box and whisker plot showing model prediction distribution compared to two different operators over 20 hand-selected validation frames

Overall, the model shows promising performance in accurately predicting zinc splatter severity, though variability among operator evaluations suggests a more complex situation than first expected with regards to manually assessing splatter severity. This opens the door to more research avenues whilst emphasising the benefit of developing a reliable computational system such as this one to avoid discrepancies between operators. The model appears to be well-suited to assist experts and could help standardise splatter severity assessments. For the model to be refined to exceed expert accuracy with confidence, a deeper study into expert judgement is essential.

5.4 Industrial Application and Model Deployment

Upon completion of the final evaluation of the model, a deployment stage was undertaken to develop the model so that it was as close to production-ready as possible. This section outlines the system architecture and setup, optimisation results using TensorRT (TRT), final deployment results and discussion of the real-world implications of the developed technology.

5.4.1 System Architecture and Setup

Figure 86 presents the data workflow of the developed system. As shown, the system is comprised of several components. Firstly, all inferences are performed on the NVIDIA Jetson Orin Nano device. This was chosen for the balance it provides between energy efficiency and computational power, which makes it suitable for deploying computer vision (CV) models on the edge [256]. This is a specialised single-board computer designed for edge AI (artificial intelligence) and deep learning tasks. It is equipped with a 6-core Arm Cortex-A78AE central processing unit (CPU), a NVIDIA GPU featuring 1024 CUDA (Compute Unified Device Architecture) cores and 32 tensor cores, and 8GB of 128-bit LPDDR5 (Low Power Double Data Rate 5) memory [256]. Additionally, it has a variety of ports and built-in Wi-Fi connectivity.

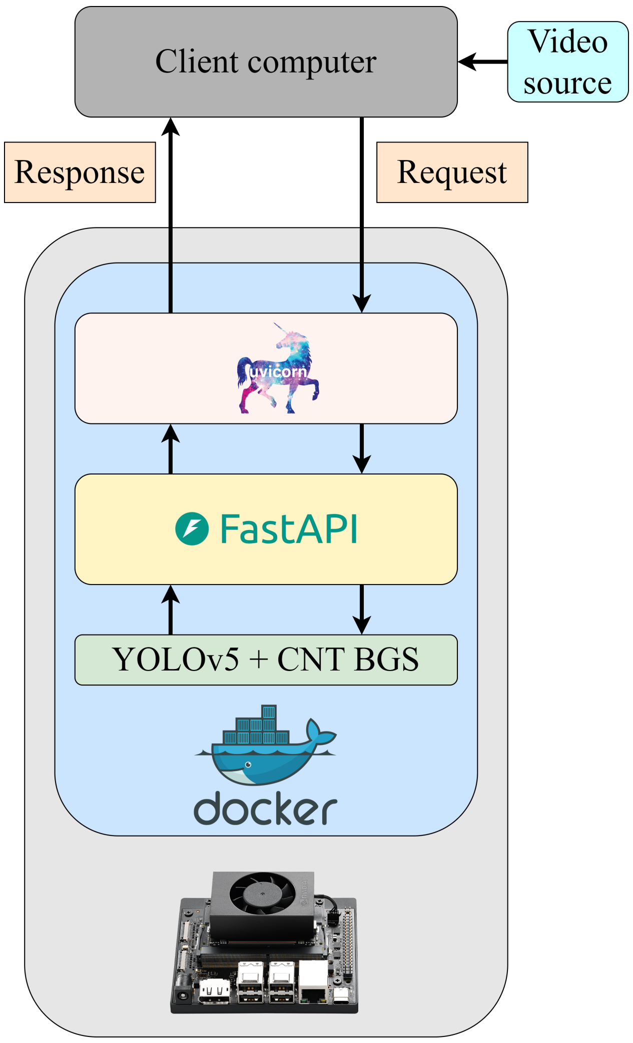


Figure 86: Schematic representing the data workflow of the deployed system

On the device, everything that is required to run the model in real-world application is stored within a Docker container. Containers are lightweight, portable, packages that contain all resources required to run a given application [257]. These include libraries, code, tools and more. Docker is the most popular containerisation platform and ensures applications run properly on different operating systems [257]. Since Docker provides a consistent, isolated environment, containerising components means the model can be rapidly deployed on other devices in future with minimal to no extra configuration. This significantly enhances the scalability of the system which is beneficial for industrial use.

To ensure that the model can interface with users, the client computer and other systems, FastAPI is used which is a high-performance web framework for building APIs (application programming interfaces) [258]. FastAPI was chosen over alternatives due to speed, robustness, automatic interactive documentation and ease of implementation [258]. In this study, speed is critical to ensure the system has real-time capabilities. One way FastAPI performs well is through asynchronous operations which enable efficient handling of I/O bound processes, making it well-aligned with real-time processing [258]. Furthermore, the interactive documentation allows for easier maintenance of the API which is ideal for development and ongoing use in industry.

Uvicorn is an ASGI (asynchronous server gateway interface) server that is typically used in conjunction with FastAPI and is a medium between the API and the web [259]. Uvicorn receives client requests and sends them to FastAPI for processing, whilst also returning responses to the client [259]. Uvicorn is capable of handling multiple client requests simultaneously using asynchronous operations, making it complimentary to FastAPI since it also operates asynchronously [259]. This means it can handle a high volume of requests with high performance, ensuring scalability of real-time applications.

In terms of the actual data workflow, it begins with a video source being streamed to the client computer. Note that the video source could also be stored on the computer itself, however in real-world application this will be a live video feed from an IP (internet protocol) camera and so the system was designed for this purpose. Assuming the FastAPI application server (hosted on Uvicorn) is online, a client-side Python script can be executed on the client computer to connect to the server. Once a connection is made, the first frame of the video source can be sent to the server with the request of performing inference on it. When the server receives the frame, the FastAPI application processes it by interacting with the computer vision model. The API provides the model with input (the video frame) and executes the inference script, which returns a splatter severity level. The severity level is then sent back to the client through the FastAPI application. At this point, the request made by the client has been successful and this loop will continue until there is no video stream available.

By using the results of the system developed in this case study in combination with known process parameter values such as strip speed and air knife distance, relationships can be identified between their settings and the resulting splatter severity. These relationships can then be used to optimise the galvanising process to minimise splatter severity at high strip speeds.

The optimisation process would first begin by collecting data during typical production cycles using the system proposed in this chapter, identifying variations in process parameters and then correlating them with changes in splatter severity measured by the system. Using statistical methods such as regression analysis, the relationships between parameters and outcomes can be identified, which can then help identify optimal settings for parameters such as air knife distance and air pressure that minimise splatter whilst maintaining high strip speeds and therefore high productivity. This could be continuously refined as more data is collected.

Additionally, once well-established, these findings can be integrated into process control systems to create a feedback loop where the galvanising line automatically adjusts parameters to control splatter. Using predictive analytics, the system could even be developed to anticipate splatter before it occurs and adjust process controls to prevent it. Implementation of these data-driven optimisations will reduce splatter occurrence whilst improving the overall efficiency of the galvanising process. Reduced splatter leads to reduced equipment downtime, reduced maintenance costs, reduced waste and enhanced product quality.

5.4.2 TensorRT Optimisation Results

The deployed system, operating on the NVIDIA Jetson Orin Nano and presented in Figure 86, is a robust solution for real-time splatter severity measurement. It was first developed on a personal computer using the PyTorch implementation of YOLOv5. For deployment, it was optimised using TensorRT (TRT) library. The results in Table 25 show the difference in performance of YOLOv5 when using PyTorch and TensorRT. All results were obtained by averaging measurements across an entire video. They show that YOLOv5 inferences using TensorRT took approximately 75% of the time that the PyTorch inferences did. This is because TensorRT optimisation applies techniques such as precision calibration, layer fusion and quantisation to enhance computational efficiency, and is highly beneficial for real-world applications [201].

Table 26 shows the difference in PyTorch and TensorRT performance of the overall model that includes YOLOv5, background subtraction and post-processing steps such as splatter region definition, scaling factor definition and splatter severity measurement. As shown, the TensorRT implementation took approximately 50% of the time that the PyTorch implementation did. This reinforces what was shown in Table 25 since not only was YOLOv5 faster with TensorRT, but other components of the full model were too. This highlights the effectiveness of using TensorRT for real-time industrial challenges.

Table 25: PyTorch vs. TensorRT YOLOv5 air knife detection inference speed

PyTorch YOLOv5 (ms)	TRT YOLOv5 (ms)
28.20	21.82

Table 26: PyTorch vs. TensorRT full splatter severity model inference speed

PyTorch Full Model (ms)	TRT Full Model (ms)
131.90	64.75

5.4.3 Final Deployment Results

Table 27: Times taken for the deployed model to perform each step from receiving a frame from the client device to returning a severity value

Step	Time (ms)
Receive Frame	65.33
Decode Frame	42.76
Pre-process Frame	0.92
Inference	21.82
Postprocess Frame	42.93
Send Severity	0.64
Total	174.41

After TensorRT optimisation was completed, the full system was tested for speed. Table 27 shows the times taken for the system to perform each step required to fully process one frame. Each step is described as follows:

Receive frame is the duration taken for the server to receive a frame from the client.

Decode frame is the time taken to convert the frame from .jpeg to a NumPy array.

Pre-process frame is the time taken for the frame to be pre-processed.

Inference is the time taken to perform YOLOv5 inference.

Postprocess frame is the background subtraction and severity measurement time.

Send severity is the time taken for splatter severity value to be sent back to the client.

Total is the time taken for all steps to be performed.

The receive, decode, inference and postprocessing steps were the most time-consuming parts of the system. This is partly because the frames were large in size which made the receiving and decoding of them relatively slow, but also because frame reception was dependent on network strength. Furthermore, the inference and postprocess steps involve major operations of the system where it makes predictions and measures splatter severity, and therefore they are expected to come at the cost of a certain degree of time expenditure. The necessity of the receive frame, decode frame and send severity steps is due to the frames being sent to the Jetson Orin Nano via a client device which sources the video. These network-based steps make up 108.73ms of the total time, which shows just how much of the overall system speed is based on network speed. Without this, the system takes 65.68ms to process each frame which is equivalent to 15.23fps. With the network cost included, this system currently operates at 5.73fps using a typical home Wi-Fi connection.

Since the original video feed ran at 25fps, the deployed model is capable of making inferences on just over 60% of the frames locally and just over 20% over a network, which is sufficient for real-world application. This is highly successful, however there are various opportunities to improve this further. For example, image size was large at 1920x1080px and this could be reduced to as low as 25% of the original size (i.e., downsampling the frames to approximately 480x270px). This would not significantly affect the quality of the inferences since the high resolution is not essential for detecting the objects of interest, but it would considerably improving the speed of YOLOv5 and CNT by reducing the amount of data processed per frame, leading to faster inference times without compromising detection accuracy.

Also, whilst the currently implementation is highly efficient, direct integration of C++ where there are bottlenecks could significantly improve speed since it is much faster than Python. This addresses the gap in Chapter 2 regarding a lack of real-time edge-based solutions using computer vision in steel production scenarios.

5.4.4 Discussion on Real-World Applicability

Discussing the potential real-world impact of the developed system is important to assess its value. Technologically there are a wide variety of real-world benefits that this system brings. Firstly, it enables the numerical measurement of splatter severity which is unprecedented. The existing method is visual inspection via line operators which is subjective and is susceptible to many biases including operator location, operator experience and operator mood. In contrast, the deployed model measures splatter severity consistently and eliminates these biases. This is not only beneficial for reliability of results, but also could be used as a standardisation tool for splatter measurements when optimising the process over a long period of time. The automatic aspect also reduces resource requirements after setup.

The availability of severity measurements opens various doors such as root-cause analysis of splatter events, preventive and predictive maintenance, and process optimisation. The capability to measure changes in splatter when adjustments are made to the process means pinpointing the cause of it will be much easier. Preventive maintenance in the form of cleaning is simplified by monitoring the accumulated splatter on surrounding equipment over a given period of time (either through manual data analysis or this could be automated as an additional feature to the model). Predictive maintenance is also simplified by knowing how much splatter has accumulated. Also, collected data be analysed to discover new trends which can guide process improvements which reduce splatter at high strip speeds. This will reduce equipment downtime and increase productivity.

This model and its application is novel and combines advanced CV algorithms to monitor a previously unmeasured process variable in real-time, which means it lays the foundation for future developments of CV in manufacturing, particularly those involving liquid with fast-paced morphology changes. For example, an air knife distance measuring feature could be developed to more closely examine any relationships between air knife distance and splatter severity. If relationships were found, air knife distance could be automatically adjusted in real-time to minimise splatter severity.

Examples of applications this kind of model would be suitable for firstly include automating the evaluation of spray coverage when coating aircraft components. Overspray detection prevents wasting materials and unnecessary expense, whilst underspray detection avoids compromising performance. This would aid quality control and sustainability. Another example is leak detection for food and drinks manufacturing to improve adaptability, production efficiency and waste reduction of the process. A final example is real-time assessment of lubrication spray for coverage, volume and consistency to prevent excessive wear, decrease downtime and ensure high standards.

The deployed model can be used to enhance the galvanising process in terms of reducing defects. Therefore, producing the same quantity of galvanised steel requires less energy and materials. Therefore, this is both environmentally and economically beneficial to the process. In terms of social benefits, implementation of this model would reduce the amount of time workers spend cleaning zinc from the floor and surrounding equipment, which reduces their workload. This also improves health and safety by minimising the amount of time workers spend in close proximity to the hazardous equipment on the galvanising line. Additionally, the fact this model automates the previous visual inspection performed by workers means workload is further reduced, whilst worker confidence and awareness is boosted when making decisions related to zinc splatter.

It is also crucial to acknowledge the limitations of the proposed approach. A significant challenge of this approach is the initial setup of the model which includes tasks such as data preparation, model training and model deployment. These tasks consume a considerable amount of time and resources which could potentially be problematic for small and medium-sized enterprises (SMEs) which may lack computational resources and technical expertise. In contrast, the current approach only requires on-site operators. Additionally, adapting the model for varying environments would require the setup process to be reiterated with new datasets which prevents rapid deployment. However, techniques such as transfer learning and auto-labelling could be used to minimise these additional resource requirements. Overall, whilst human readiness and adaptability is advantageous, the benefits provided by an automated, objective system surpass low resource demands.

5.5 Conclusions

This chapter has presented a case study where real-time quantification of zinc splatter severity during the steel galvanising process was achieved through use of CNT background subtraction for splatter segmentation and YOLOv5 for robustness to changing environmental conditions. This research and development addresses the limitations of subjective visual inspections by operators by offering an objective and automatic alternative. Therefore, this work demonstrates a significant advancement in industrial technology.

In this work, seven different background subtraction algorithms were tested and the CNT algorithm was found to be the best in terms of performance and real-time suitability. The CNT algorithm parameters were optimised so that the minimum and maximum pixel stability thresholds were one and ten respectively. Furthermore, denoising techniques such as morphological erosion and contour thresholding were used to refine CNT masks. YOLOv5 was trained on 4200 images which resulted in excellent precision, recall and mAP results, which were all 1, indicating perfect performance. Also, promising results were drawn from comparison of model predictions and operator judgements where the model was more similar to individual operators than they were to each other.

Successful model deployment exemplifies the production-readiness of the developed system and is exhibited through the result of 5.73fps on a NVIDIA Jetson Orin Nano which includes client requests and responding over a Wi-Fi network. Whilst TensorRT optimisation was highly successful and an incredibly valuable aspect of this study, allowing inferences to run at over 15fps with no image resizing on 1920x1080px images, resizing images before initiating YOLOv5 model training would largely improve inference speed. Furthermore, the use of Docker and FastAPI in this study have ensured the model is scalable.

This chapter contributes to the advancement of steel galvanisation monitoring technology whilst demonstrating the suitability of integrating modern computer vision techniques in industrial environments. Therefore, it is pioneering research that offers great insight that could guide the development of future technologies in CV and manufacturing. Furthermore, this case study has addressed four of the research gaps identified in Chapter 2. Firstly, it has addressed the lack of real-time edge-based CV applications developed for steel production processes. Secondly, it has involved the generation of a novel dataset based on the galvanising process which addresses the lack of datasets available for developing CV applications for steel production processes. Thirdly, it has contributed to addressing the lack of real-world measurements performed by CV models. Lastly, it has addressed the lack of hybrid CV models that make use of both traditional and deep learning-based techniques.

5.6 Future work

In future work, implementing the splatter severity system into the galvanising line would contribute to enhancing process efficiency and quality control through data-driven insights during tasks such as root-cause analysis, process optimisation and maintenance planning.

Additionally, there is opportunity to develop this work into a closed-loop control system that automatically adjusts process variables depending on the resulting splatter in real-time, as well as integrate it into an alarm system that automatically alerts operators of high splatter severity events. Therefore, this work paves the way for discovering new trends in the galvanising process and promotes future developments based on similar methodologies that accelerate the progress of manufacturing technology.

Chapter 6: Analysis of Gaseous Plume Dynamics, Refractory Wear and Stirring Efficiency in Gas Stirring

In this chapter, a novel approach using computer vision (CV) is presented to monitor plume characteristics from experiments simulating gas stirring in a basic-oxygen furnace (BOF). The CV model uses the YOLOv5 (You Only Look Once) detection network for plume detection, DeepSORT (Deep Simple Online and Real-Time Tracking) for plume tracking, and Counting (CNT) background subtraction (BGS) for plume segmentation. This innovative method provides insights into plume dynamics, refractory wear, and stirring efficiency of different gas stirring configurations, whilst demonstrating a significant improvement over traditional measurement techniques.

6.1 Introduction

During steel production, the refinement process within the basic-oxygen furnace is crucial, and optimising this process could significantly enhance the efficiency and quality of steel production [260]. Gas stirring is a technique that facilitates improved mixing between steel and slag which increases yield and scrap charge (i.e., more recycled steel can be used), improves phosphorus removal, reduces skulling and reduces consumption of aluminium and flux [260, 261]. However, gas injection into the BOF is achieved via a mixing element and this technique has been found to increase wear on the refractory lining surrounding the mixing element, which is due to thermo-mechanical stresses from gas injection and compressive failure of the refractory material [262, 263]. Previous approaches for assessing this wear include time-domain reflectometry, laser scanning and temperature monitoring, which offer valuable insights but are limited by their static nature and point-specific measurements [264].

This objective of this case study was to develop an analytical tool for monitoring helium gas plumes in a water column setup that replicated BOF gas stirring using a specific mixing element design called the annular tuyère, which is proposed in [264]. A schematic of the simulation setup is shown in Figure 87, where water acts as liquid steel and helium acts as injected gas (argon in real application). Whilst gas stirring is known to improve the efficiency of metal purification, the incorporation of a mixing element into the furnace setup causes refractory wear in the area adjacent to the element. By detecting and tracking plumes in different column setups, plume characteristics can be measured which provide insights into how different furnace setups affect wearing of the refractory lining, as well as the efficiency of the stirring process. These insights can be used for optimising the tuyère design to minimise refractory wear and maximise stirring efficiency. Therefore, another goal of this case study was to demonstrate how the analytical tool could be used for real-world application by using it to gain insight on different column setups. This work overcomes the challenges of monitoring several characteristics of gaseous plumes that are complex due to their dynamic behaviour, indefinite shapes and inconsistent patterns of merging and splitting.

The primary contribution of this study is a novel application of CV that indicates optimal gas stirring configurations that reduce refractory wear and increase stirring efficiency, leading to extended equipment life and enhanced product quality. Additionally, work presented here paves the way for integrating real-time monitoring systems, advancing predictive maintenance technology and further enhancing operational efficiency. The model is likely to be applicable to other mixing element designs without further training, which would enable better design decisions beyond gas stirring configuration. Furthermore, this approach could be adapted to other processes, which would widen current monitoring capabilities and deliver more benefits to industry. Source code is available on GitHub [265].

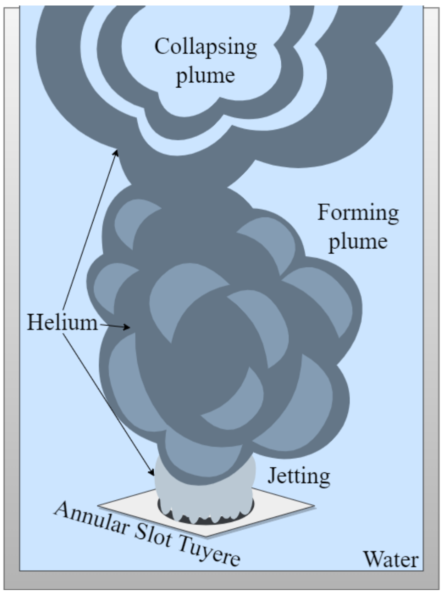


Figure 87: Schematic representing the gas stirring simulation setup. During experimentation, helium is injected into the system via the annular tuyère which induces jetting. As it rises, plumes form, expand and then begin collapsing towards the top area

6.2 Methodology

The data for this case study was generated by recording a series of experiments designed to simulate gas stirring in the BOF using water instead of liquid steel and helium to simulate argon. These experiments were conducted by another researcher who captured footage using six different configurations. An overview of the methodology is given in Figure 88, which shows there were seven main steps. The first step of this case study was to generate an initial dataset by labelling one of the six sets of footage that were available, and then splitting it up into training, validation and testing sets. The data prepared in this case study addresses the gap in Chapter 2 regarding the lack of diverse datasets available for developing CV applications for steel production processes. The second step was to experiment with three different object detection models and establish which one was most suitable for this application. The third step involved comprehensive preparation of all unlabelled datasets using an efficient auto-labelling approach, to build a dataset sufficient for maximising model performance in later stages. The fourth step entailed model development for taking real-world measurements, which included experimentation of various background subtraction algorithms to establish which one was most suitable for this application. The fifth stage involved optimisation of the object detection network to maximise performance. Once these stages were complete, the model was validated on real-world data to assess performance, before it was utilised to calculate wear rate factor (WF), which indicates the expected wear on the furnace refractory lining and stirring efficiency (SE), which indicates the effectiveness of the stirring process. These metrics were calculated for different gas stirring configurations to draw industrial insights on the process.

6.2.1 Initial Data Preparation

All datasets prepared for this case study were prepared using footage of gas stirring using the annular slot tuyère design proposed in [264]. There were six different gas injection rates used and therefore the data used for this case study is shown in Table 28.

Each of the six datasets contained approximately 750 frames and were originally provided in frame format rather than video. Whilst model predictions were visualised in video format, developing a real-time analysis tool was outside the scope of this case study since the aim was to develop a tool used for postprocessing, and therefore whilst a higher inference speed was preferable for minimising time consumption, it was not a significant concern in comparison to model accuracy.

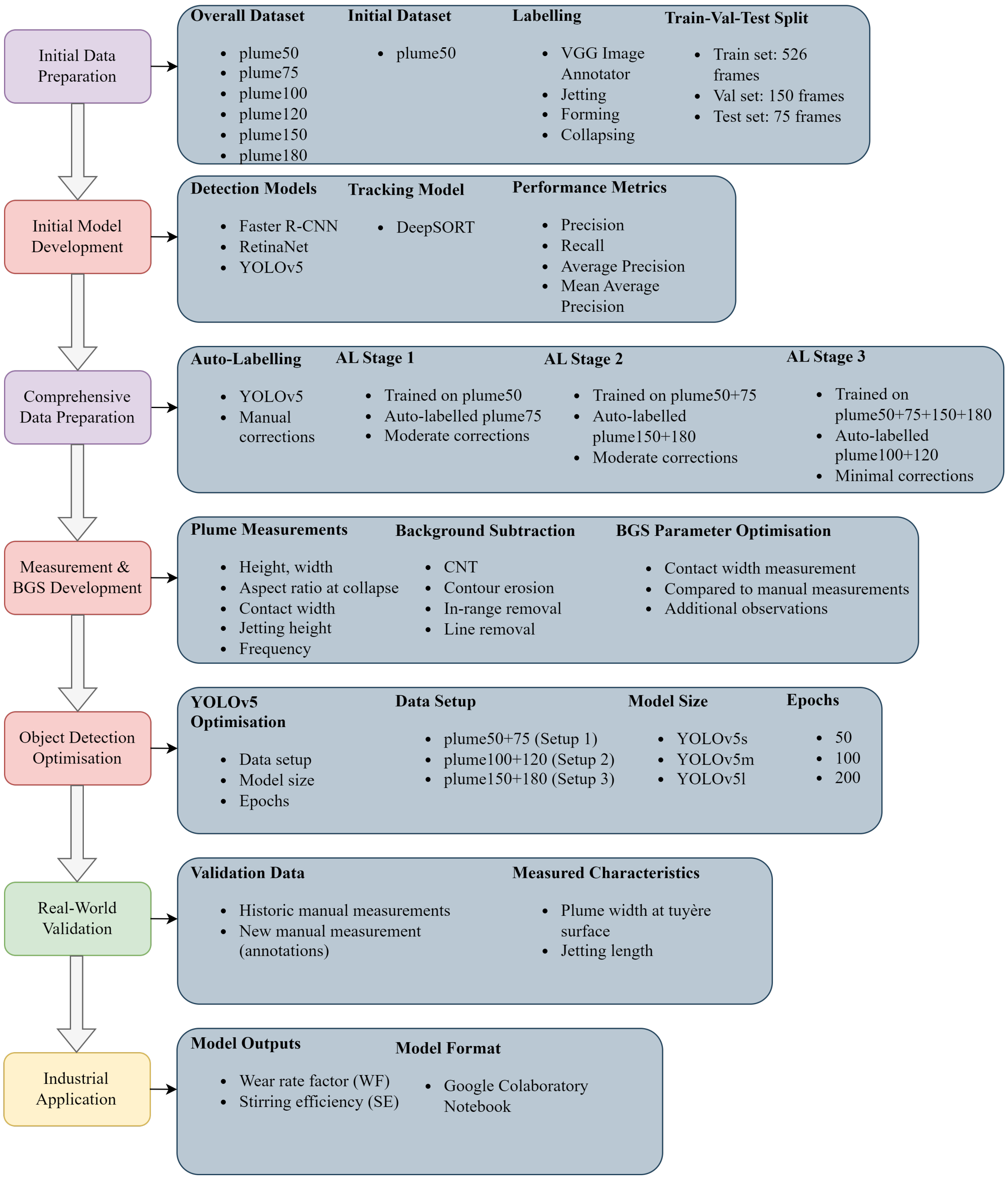


Figure 88: Overview of the methodology used for Case Study 3

Table 28: Original datasets used in Case Study 3

Injection Rate (Nm^{3}h^{-1})	Dataset Name
50	Plume50
75	Plume75
100	Plume100
120	Plume120
150	Plume150
180	Plume180

Since the labelling process was time-consuming and complex, Plume50 was the only dataset used for initial model development as this was sufficient for comparing several object detection networks and assessing their suitability for this use-case. Each plume instance throughout Plume50 was labelled in terms of location and and classification using the VGG (Visual Geometry Group) Image Annotator (VIA) [215]. The overall classifications for this case study were as follows: plumes exhibiting jetting phenomena were marked as “jetting”, plumes in a state of continuous expansion were marked as “forming” and plumes that had achieved maximum diameter and had began shrinking and dispersing were identified as “collapsing”. However, for the initial data preparation stage only “forming” and “collapsing” were used to simplify the process. This annotation process was particularly challenging due to the dynamic and complex behaviour of the gaseous plumes which were often structurally chaotic and were constantly evolving, splitting and merging. These challenges are addressed in-depth in Section 6.3.1.

The number of instances of each class in Plume50 are shown in Table 29. Plume50 was split using a 70:20:10 ratio for training, validation and testing sets, which resulted in the subset sizes shown in Table 30. Note that jetting instances were not used during initial model development to maintain simplicity of the process.

Table 29: Number of instances of each class in Plume50

Class	Number of Instances
Forming	561
Collapsing	1238
Jetting	128

Table 30: Number of frames of each subset of Plume50

Subset	Frames
Training	526
Validation	150
Testing	75
Total	751

6.2.2 Initial Model Development

To effectively monitor plume behaviour, it was essential to implement a high performance method for object detection and tracking. Three distinct detection networks underwent evaluation to determine their practicality for this case study, before proceeding to develop a comprehensive measurement tool. The networks selected for evaluation were Faster R-CNN (Faster Region-based Convolutional Neural Network) [48], RetinaNet [55], and YOLOv5 [63]. Each network was integrated with the DeepSORT tracking algorithm [92] and was subjected to training, validation, and testing using the Plume50 dataset. These models were selected for their promising performance, as evidenced in Section 2.3 of Chapter 2. Performance was measured using metrics such as precision, recall, average precision (AP), mean average precision (mAP) and inference time. These were also chosen based on what was shown as common practice throughout Chapter 2. The final choice of the detection model was primarily influenced by its COCO (Common Objects in Context) mAP score, although other metrics were also considered.

For this case study, models were developed in Google Colaboratory (differently to other case studies which were developed on local resources). It was challenging to align all three models to perform the same task and evaluate them using the same metrics. This is because all three models varied in terms of their development history and underlying architectures. Each model (Faster R-CNN, RetinaNet and YOLOv5), was originally developed by different teams. This meant that a deep understanding of their unique implementations was required.

Google Colaboratory was partly used for this case study due to the availability of pre-built Colab notebooks that either already integrated DeepSORT, or made it easier to integrate it manually. In the case of YOLOv5, it was possible to find a notebook with DeepSORT integrated [266], whereas with Faster R-CNN and RetinaNet it was necessary to manually integrate it [267, 268, 269]. It was possible to implement these notebooks locally, however

Google Colaboratory was used so that the final model was easily accessible for the collaborating researcher, without needing Python experience or software to be downloaded locally as it was all available on the cloud. This is an example of good scalability since the model can easily be distributed to anyone wishing to use it to collect results.

Also, each model needed to be trained, validated and tested using annotated data of different formats which can be seen in Table 31. They were exported in the VIA JSON (JavaScript object notation) format and converted to their respective formats using online conversion tools [216, 270]. Note that PASCAL VOC XML refers to "Pattern Analysis, Statistical Modelling and Computational Learning Visual Object Classes extensible markup language". PASCAL is a collaborative network of excellence contributing to the advancement of pattern analysis and machine learning (ML), VOC is a dataset developed by the PASCAL network (see Section 2.1 of Chapter 2), and .xml is a data format used for annotating datasets.

Furthermore, the evaluation functions built-in to each model by their respective developers were not equal and in some cases, non-existent. This was particularly true for the mAP which as mentioned in Section 3.4.4 of Chapter 3, can be calculated in different ways. Therefore, each model needed to be modified to output evaluation metrics that were calculated in the exact same way every time and included the most important metric, the COCO mAP.

Table 31: Annotation format required for each initial model

Model	Annotation Format
Faster R-CNN	COCO JSON
RetinaNet	PASCAL VOC XML
YOLOv5	.txt

6.2.3 Comprehensive Data Preparation

Following initial model development, it was necessary to label the remaining available data to ensure the final model was suitable for application. Since labelling was time-consuming and complex, an auto-labelling approach was taken to adapt to resource-constraints and maximise efficiency. There were three auto-labelling (AL) stages (shown in Figure 88).

Initially, the model trained on Plume50 was used to auto-label Plume75, as it most closely resembled Plume50. Note that the "jetting" instances had to be added manually to Plume50 and used to re-train the model before initiating auto-labelling. Following the first auto-labelling task, manual adjustments were still necessary for correcting imprecise bounding boxes predicted by the model. However, this process was notably quicker than prior manual labelling from scratch.

In the second stage, the model was trained on a dataset comprised of both Plume50 and Plume75. This newly trained model was then applied to label Plume150 and Plume180, creating a need for a moderate degree of manual correction. Although Plume150 and Plume180 differed more from Plume50 and Plume75 than Plume100 and Plume120, the labelling of Plume100 and Plume120 would still require moderate adjustments after auto-labelling but would not expose the model to the full extreme of possibilities in the following stage.

The final stage required minimal corrections. This approach was proposed as efficient due to the model's training on the two extremes (Plume50 and Plume75, and Plume150 and Plume180), providing it with reference points for interpolation when encountering new, unseen data. In contrast, training the model with only the first four datasets lacks exposure to the unique features present in the extremes of Plume150 and Plume180, therefore limiting its potential for interpolation capabilities.

6.2.4 Measurement and Background Subtraction Development

This stage focused on enabling the object detection model established in the initial development phase to accurately measure various plume characteristics across different injection rates. Collectively, these measurements could provide insights into the wear and stirring characteristics of the tuyère design and therefore aid in optimisation for minimising wear and maximising stirring efficiency. Key characteristics measured include plume state, plume height, plume width, contact width, jetting length and plume frequency.

Plume state (jetting, forming or collapsing) was described by the classification results, whereas plume height and width were derived using bounding box dimensions. Then, using the scale in Figure 89, these pixel-based measurements were converted into millimeters. The plume-tuyère contact width was used to calculate washing severity (WS) which impacted wear rate factor (WF) since washing phenomena causes refractory wear, the plume height and width were used to calculate plume volume using the ellipsoid volume equation [271], which impacts stirring efficiency (SE), plume frequency contributed to both WF and SE, and the jetting length was measured for additional insight but not fully integrated into the overall configuration assessment. The practical application of these variables is detailed in Section 6.4.1 and the results of application are presented in Section 6.4.2.

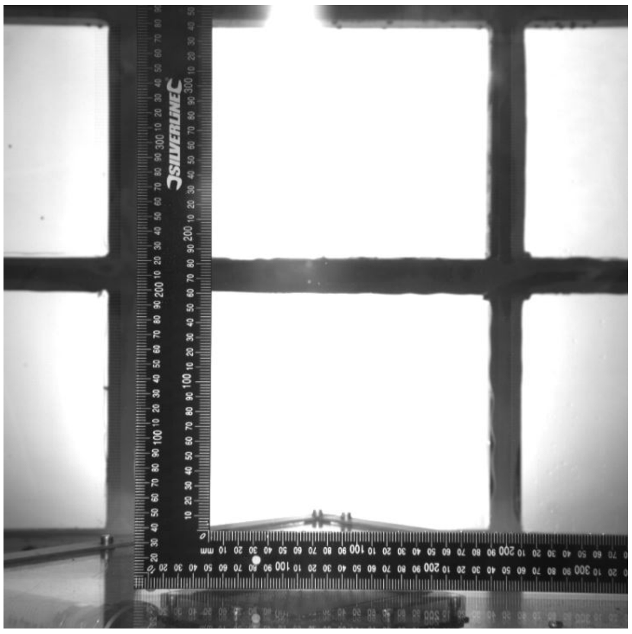


Figure 89: Image of a ruler within the experimental setup that was used to scale pixels to millimetres

Whilst bounding boxes were sufficient for overall plume height and width measurements, measuring the plume-tuyère contact width was more complex since typically, the contact width was not the same as the overall plume width. This created the need for plume width measurement at a certain level within the bounding box (the level of the tuyère surface). This capability was achieved by assessing the performance of multiple background subtraction algorithms and selecting the best one. These algorithms have been detailed in Section 5.2.2 of Chapter 5 (all algorithms except CNT) and Section 3.3.2 of Chapter 3 (for CNT). Additionally, morphological operations [28], in-range colour removal [272], and Hough transformations [273] were used to denoise the segmentation mask.

Case Study 2 only used erosion when applying morphological techniques, however this case study used morphological opening (erosion and dilation) [28]. As discussed in Chapter 5, after erosion removes small noise contours, subsequent dilation can distort contour shapes but preserve size. Unlike in Chapter 5, this case study did not hold real-time performance as a goal, meaning the extra computation required for the dilation step was not detrimental. Also, whilst contour shapes were important, sizes were more important to ensure major geometrical features of plumes such as height and width, were preserved.

To further refine BGS masks, a technique called in-range colour removal was implemented [272]. This method involves eliminating colours within a pre-defined range of RGB values.

It was used to remove a noticeable aspect of the mask that surrounded the plumes but were not part of them, and appeared to be a different colour to the accurate part of the mask. The area prone to this error was probed 20 times to obtain RGB values, and this was to remove as much of the unwanted mask as possible.

Hough line transforms are a method for line detection [273] and were applied to identify lines within the segmentation mask. These lines were specifically targeted for their location and angle, which matched the straight edges of the tuyère. The column setup occasionally experienced instability, leading to movements of the tuyère which resulted in the edges of the tuyère being mistakenly identified as gas in the segmentation process. By applying Hough line detection, it was possible to consistently eliminate these inaccurately detected edges from the segmentation mask. This was an unconventional but innovative application of Hough line transforms.

Optimisation of the parameter settings for each technique was conducted using grid searches. The effectiveness of these settings was evaluated by comparing the plume-tuyère contact width measurements from the model against manual measurements from previous research [264]. Since some of the validation data seemed imprecise, a careful comparison of model predictions, earlier manual measurements from [264], and new manual observations was undertaken. This cross-examination was essential for maximising the reliability of the model.

6.2.5 Object Detection Optimisation

Following the completion of BGS development, the object detection model underwent further optimisation to enhance performance. A comprehensive grid search [274] was conducted which applied every possible combination of selected parameter settings. Optimisation was carried out on three parameters, each at three distinct levels. The strategy for data setup drew inspiration from k-fold cross-validation and involved pairing the six datasets to create three new sets. For each variant of data setup, two pairs were designated for training and a different pair was allocated for testing. This ensures model performance was not biased by the selection of particularly favorable or unfavorable sets for training and testing [275]. Model sizes compared were the small (YOLOv5s), medium (YOLOv5m), and large (YOLOv5l) versions of YOLOv5. Architecturally, these networks differ in complexity and size, with YOLOv5s having the fewest convolutional layers and the lightest structure, whilst YOLOv5l has the most layers, making it better suited for handling more complex detection tasks. An overview of the differences between these networks are presented in Section 3.3.1

[196]. In terms of training epochs, 50, 100, and 200 were used.

Throughout the grid search, training loss, validation loss, performance metrics such as mAP, as well as model predictions of real-world characteristics were used to evaluate the performance of each model design. Overall, these results allowed for thorough assessment of which design produced the most accurate results. Table 32 shows the experimental setup which includes 27 experiments in total.

Table 32: Grid search experimental setup involving three different data setups, YOLOv5 model sizes (S - small, M - medium, L - large), and numbers of training epochs

Exp. No.	Data Setup	Model Size	Number of Epochs
1	1	S	50
2	2	S	50
3	3	S	50
4	1	M	50
5	2	M	50
6	3	M	50
7	1	L	50
8	2	L	50
9	3	L	50
10	1	S	100
11	2	S	100
12	3	S	100
13	1	M	100
14	2	M	100
15	3	M	100
16	1	L	100
17	2	L	100
18	3	L	100
19	1	S	200
20	2	S	200
21	3	S	200
22	1	M	200
23	2	M	200
24	3	M	200
25	1	L	200
26	2	L	200
27	3	L	200

6.2.6 Real-World Validation

Upon completion of model optimisation, it was necessary to validate its ability to measure real-world characteristics. This process aimed to address the gap identified in Chapter 2 regarding the lack of real-world validation of computer vision model measurements.

The validation process entailed comparing predictions with manual measurements obtained from previous research, as well as new annotations produced manually for training the object detecton model. The validation focused on key plume characteristics including plume width at the tuyère surface and jetting length.

6.3 Results and Discussion

This section presents and discusses the results obtained from experimentation described in Section 6.2, which begins with the results of initial data preparation and model development, followed by comprehensive data preparation and model development. Finally, the model estimations of real-world characteristics are validated against manual measurements.

6.3.1 Initial Data Preparation

The initial phase of data preparation presented various challenges, mainly posed by the constantly changing nature of the plumes as they formed and collapsed. An example of this complexity is illustrated in Figure 90 which shows an unlabelled frame from Plume50. The unclear boundaries of each plume in the image make it challenging to accurately measure the plume geometry, leading to potential ambiguity in identifying the number of plumes, which could be anywhere from two to five depending on how the frame is observed. When patterns across series' of frames were examined, as opposed to individual frames, the labelling process became even more complex. This meant labelling required careful consideration, practice, and multiple revisions for each frame to ensure the chosen approach would yield the most effective measurement tool during deployment.

One of the main labelling challenges was based on occlusions. The gaseous nature meant plumes were always partially merged and therefore defining the precise edges of plume bounding boxes was difficult. Additionally, the overlapping nature of plumes further complicated the process. Note that overlapping is not the same as merging. Whilst overlapping refers to one plume occluding the other from the camera perspective, merging refers to the actual combination of two or more plumes. The fact that the approach in this case study used 2D data, which lacks depth information, was one of the reasons for the aforementioned labelling challenges. In future work, it would be beneficial to experiment with 3D data however this would introduce new challenges surrounding the complexity of data, increased computational resource requirements and the necessity to use more sophisticated algorithms. In terms of partial merging of plumes, a decision was made to avoid identifying exact physical boundaries for each plume since this was virtually impossible to do perfectly using the human eye. Instead, bounding boxes were assigned as best as possible and no attempt was made to account for overlapping. Figure 91 exemplifies this labelling approach.

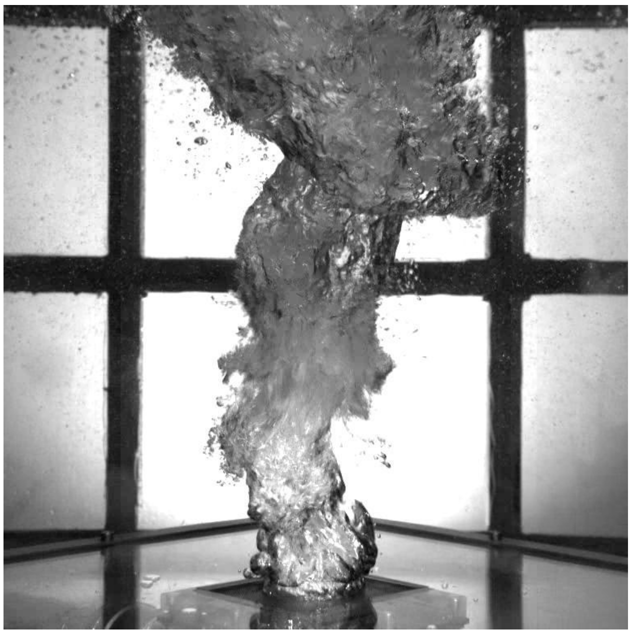


Figure 90: Frame of Plume50 demonstrating the ambiguous nature of plumes

Another challenge arose from the seemingly random forming and merging of plumes. In certain instances, rather than adhering to the typical jetting-forming-collapsing pattern, plumes unexpectedly merged with neighbouring plumes during formation. This complexity introduced challenges in the labelling process, as judgement was required to determine when two constituent plumes should be treated as a single entity. The dynamic nature of the gas which has previously been described, meant it was not always evident which plumes were undergoing merging. In Figure 90 this uncertainty is shown in the bottom third of the image, where it is unclear whether one or two plumes exist. Similarly, ambiguity persists in the top two-thirds of the image, and deciding whether the partially collapsed gas in the middle belongs to plumes above or below adds further complexity to the labelling process.

Labelling challenges also included the ambiguous definition of the outer edges of plumes, primarily caused by random gas streams partially detaching from the plumes and creating

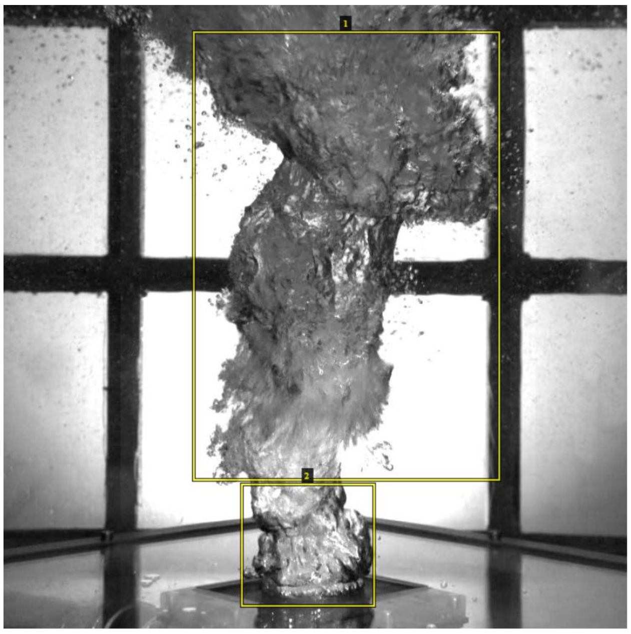


Figure 91: Labelled frame of Plume50 showing plume ambiguity

confusion of where exact boundaries were. This phenomenon is exhibited in Figure 92. To address this issue, labels were applied with a focus on capturing the main body of plumes, allowing for a slight allowance for diverging gas until detachment began. This approach is shown in Figure 92, where the detached gas stream in the bottom left of the image is excluded at a specific extent of separation.

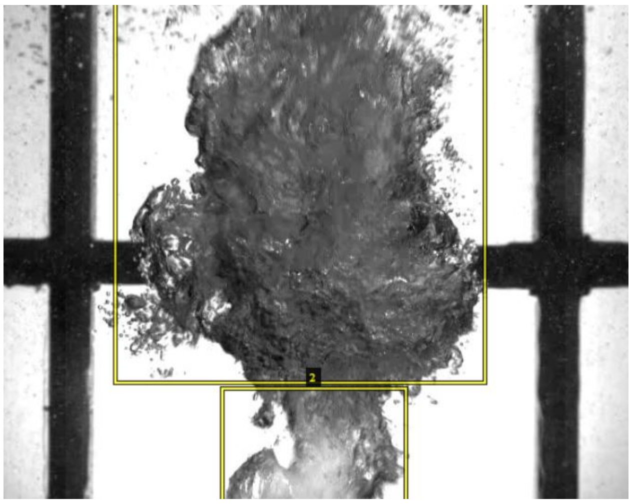


Figure 92: Labelled frame of Plume50 showing diverging gas

6.3.2 Initial Model Development

The initial model development stage entailed the evaluation of several existing object detection architectures in order to choose the most appropriate one for this case study. The three models chosen were Faster R-CNN, RetinaNet and YOLOv5. It is important to note that whilst comprehensive testing results are available for all three models, the training and validation data, which would have provided additional insights into the learning dynamics and generalisation capabilities of models, are not present. This absence is due to an oversight during initial model development, where most of the focus was placed on ensuring DeepSORT was correctly integrated with each model, aligning all three models to perform the same task and ensuring uniform evaluation criteria despite their structural differences. This led to training results not being stored carefully for later use. Further complicating this issue, the Python version of Google Colaboratory updated in this time, which posed challenges in re-training the models, as they were originally developed for an earlier version. Whilst re-training and data collection remain feasible, the Google Colaboratory update has made this process significantly more time-consuming. To mitigate issues in future projects, it is essential to implement more documentation protocols, backup procedures and version control using GitHub from the outset.

Fortunately, the results in Tables 33, 34, and 35 show the aim of the initial model development stage, which was to find the most appropriate model for this application, was still achieved. The models were evaluated on the task of localising gaseous plumes with bounding boxes and classifying them as either “forming” or “collapsing”, which overall is an assessment of their ability to perform object detection on every plume that was visible in the footage. The number of plumes detected was not a specific consideration in the evaluation of model performance, as the focus was on the overall accuracy of detection and classification rather than the count of individual detections. Whilst the ability to analyse model behaviour during the training phase is limited, the testing phase results still offer a robust basis for evaluating and comparing the performance of these models. Also, experience was gained in that using a cloud service such as Google Colaboratory means certain precautions should be taken in the preliminary stages to ensure models are adapted to handle any uncontrollable updates. Therefore, not only will documentation, backup and version control activities be practiced thoroughly in future projects, but extra precaution will be taken to prepare for the utilisation of cloud computing resources. All models were trained for 30 epochs on Plume50 and the epoch with the lowest validation loss was used to produce the results presented.

The testing was conducted across ten IoU (intersection-over-union) thresholds, adhering to COCO mAP standards, in order to ensure a thorough comparison of model performances for this task whilst avoiding any biases to particular IoU thresholds. Despite the missing training and validation data, these results provide a strong indication of the superiority of YOLOv5 in terms of accuracy and efficiency in the context of this case study. For initial model development (assessment of three architectures), tabular data that follows was much more insightful than visual results, which are therefore not presented for conciseness. However, visual results will be presented in later elements of this section where observations were more significant.

Table 33: Testing results for object detection of plumes using Faster R-CNN across varying IoU thresholds (0.50 to 0.95 from most relaxed to most strict)

IoU Thresh.	#Ground Truths	#Predictions	TP	FP	Precision	Recall	Avg. Inference Time (s)	Forming AP	Collapsing AP	mAP_{COCO}
0.50	359	410	340	70	0.829	0.947	0.092	0.805	0.879	0.842
0.55	359	410	96	314	0.234	0.267	0.092	0.000	0.292	0.146
0.60	359	410	10	400	0.024	0.028	0.092	0.000	0.019	0.010
0.65	359	410	3	407	0.007	0.008	0.092	0.000	0.005	0.003
0.70	359	410	0	410	0.000	0.000	0.092	0.000	0.000	0.000
0.75	359	410	0	410	0.000	0.000	0.092	0.000	0.000	0.000
0.80	359	410	0	410	0.000	0.000	0.092	0.000	0.000	0.000
0.85	359	410	0	410	0.000	0.000	0.092	0.000	0.000	0.000
0.90	359	410	0	410	0.000	0.000	0.092	0.000	0.000	0.000
0.95	359	410	0	410	0.000	0.000	0.092	0.000	0.000	0.000
Overall	359	410	45	365	0.110	0.125	0.092	0.080	0.120	0.100

Table 33 shows the testing results for Faster R-CNN. At the lowest IoU threshold of 0.50, Faster R-CNN performance was quite good with a precision of 0.829 and a recall of 0.947. The primary metric of COCO mAP was also good at 0.842, indicating the effectiveness of Faster R-CNN for this task when the threshold for an acceptable prediction is relaxed. On the other hand, there were still 19 ground-truth instances missed as shown by comparing ground-truths to true positives (TP), as well as 70 instances predicted that did not match any ground-truth instance as shown by false positives (FP).

A significant decline in performance was observed as the IoU threshold increased. Beyond a threshold of 0.50, both precision and recall experienced a marked decrease due to the model predicting less than a third of the true positives previously predicted whilst over quadrupling the number of false positives. From thresholds 0.70 and above, a point of zero performance was reached. To clarify, the cause of this is that the IoU threshold (a measure of the overlap between predictions and ground-truth data) that must be met for a prediction to be considered a true positive is stricter than all of the predictions that were made. In other words, all predictions made had an IoU score below 0.70 and did not overlap enough with the labelled data to meet the stricter criteria. This trend emphasises the limitations of Faster R-CNN in achieving accurate predictions under moderately strict criteria. The collapsing and forming class AP values followed a similar trend, starting strong at the 0.50 threshold but reducing rapidly when the threshold was increased. An interesting aspect of this is that the ability of Faster R-CNN to predict forming declined much quicker than collapsing. This was because, as mentioned in Section 6.2.1, Plume50 contained 561 forming instances and 1238 collapsing instances. Therefore, Faster R-CNN had learned the features of collapsing instances significantly better than those for forming instances.

Overall, the COCO mAP achieved by Faster R-CNN was 0.100 which reflects the deteriorating performance at higher thresholds, and again suggests limitations in its ability to make precise predictions. It is also worth noting the inference time of 0.092 seconds which suggests reasonable efficiency in processing. However, given the overall performance trends it is clear that whilst Faster R-CNN can effectively detect objects at lower precision requirements, it is limited in ability to perform high-precision detection. Therefore, it is not suitable for an application such as the one in this case study.

Table 34: Testing results for object detection of plumes using RetinaNet across varying IoU thresholds (0.50 to 0.95 from most relaxed to most strict)

IoU Thresh.	#Ground Truths	#Predictions	TP	FP	Precision	Recall	Avg. Inference Time (s)	Forming AP	Collapsing AP	mAP_{COCO}
0.50	359	2063	359	1704	0.174	1.000	2.770	0.985	0.985	0.985
0.55	359	2063	359	1704	0.174	1.000	2.770	0.985	0.985	0.985
0.60	359	2063	359	1704	0.174	1.000	2.770	0.985	0.985	0.985
0.65	359	2063	357	1706	0.173	0.994	2.770	0.980	0.974	0.977
0.70	359	2063	349	1714	0.169	0.972	2.770	0.961	0.945	0.953
0.75	359	2063	334	1729	0.162	0.930	2.770	0.900	0.921	0.910
0.80	359	2063	311	1752	0.151	0.866	2.770	0.802	0.865	0.833
0.85	359	2063	260	1803	0.126	0.724	2.770	0.593	0.724	0.658
0.90	359	2063	168	1895	0.081	0.468	2.770	0.294	0.355	0.325
0.95	359	2063	45	2018	0.022	0.125	2.770	0.056	0.025	0.040
Overall	359	2063	290	1773	0.141	0.808	2.770	0.754	0.776	0.765

Table 34 shows the testing results for RetinaNet. At the lowest threshold of 0.50, RetinaNet achieved a perfect recall of 1.000 and a COCO mAP of 0.985, indicating its ability to detect all ground-truth instances. However, this was accompanied by a high number of False Positives (1704), resulting in a low precision of 0.174. This trend of high recall but low precision existed across all IoU thresholds, suggesting that whilst RetinaNet was effective in detecting the presence of objects (high recall), it struggled with the precision of these detections (high FP). The high number of predictions (2063) compared to the number of ground-truths (359) further indicates that the model had a tendency to over-predict. Despite detecting all or nearly all ground-truth instances, the substantial number of false positives significantly impacted precision.

In real-world application of object detection, models are typically designed by considering a trade-off between precision and recall. This is managed through confidence thresholding which is a post-processing step that follows detection, where only predicted bounding boxes with confidence scores (how likely the model thinks a prediction is correct) above the set threshold are kept, effectively filtering out low-confidence predictions. This leads to fewer, but more reliable detections. RetinaNet showed high recall but low precision due to a large number of predictions. Adjusting the confidence threshold upwards can significantly improve precision by only accepting higher-confidence detections, consequently reducing false positives. This adjustment, however, typically decreases recall. The optimal threshold varies based on the application the model is used for and is generally higher where high precision is required, and lower where missing detections is highly undesirable. Balancing this trade-off is essential, especially in industrial applications like this case study, to ensure effective detection with minimal false positives. For this part of model development, the confidence threshold was left at default. For Faster R-CNN and RetinaNet this was 0.05, whereas for YOLOv5 this was 0.001. This was to avoid interfering with thresholds set by the developers until it was necessary to optimise the best model for real-world application.

As the IoU threshold increased, there was a gradual decrease in both precision and recall. At an IoU of 0.95, the precision to dropped to 0.022, and the recall to 0.125, with a corresponding decrease in COCO mAP to 0.040. For any model that does not have perfect performance, a decline in performance is expected as the IoU threshold increases. However, comparing Faster R-CNN results to RetinaNet results, it can be seen that the performance of RetinaNet does not appear to degrade as rapidly as Faster R-CNN. In the case of Faster R-CNN, there was failure to predict over 90% of the ground-truth instances from a threshold of 0.60 and above, whilst in the case of RetinaNet, there was failure to predict 87% of ground-truth instances reached at a threshold of 0.95 which is significantly better. However, Faster R-CNN only made 410 predictions during a single testing experiment, whereas RetinaNet made 2063 which is much less efficient, especially when considering there were only 359 ground-truths. This is why Faster R-CNN achieved a higher precision than RetinaNet until a threshold of 0.60.

Collapsing and forming AP values remained relatively high across most thresholds and compared to Faster R-CNN, there was significantly less degradation in performance with increasing threshold, as well as less difference between classes. At most thresholds, collapsing AP was higher than forming AP, which aligns with previous discussion surrounding the number of training samples. However, there were a few cases where forming samples were predicted slightly better than collapsing samples, which suggests the model learned both classes well.

Overall, the COCO mAP achieved by RetinaNet was 0.765, which reflects reasonably good performance across the range of thresholds. However, this was at the computational cost of many false predictions and performance still degraded significantly from a threshold of 0.80 to 0.95. This suggests that whilst RetinaNet had the capability to capture ground-truth instances quite well, confidence threshold optimisation would be required to minimise false positives if it were to be used in a real-world applications. Also, other hyperparameters such as the learning rate and focal loss parameters would need to be optimised to improve overlap between prediction and ground-truth boxes to improve results at higher IoU thresholds. Another important aspect to consider is the average inference time of 2.770 seconds. This is considerably higher than Faster R-CNN, further supporting the suggestion that whilst RetinaNet may have more ability to detect objects, it does so at the cost of efficiency. With optimisation, RetinaNet may be suitable for an application such as the one in this case study, however it is quite slow and inefficient. If a real-time system was developed in future projects, RetinaNet may be unsuitable for this reason.

Table 35: Testing results for object detection of plumes using YOLOv5 across varying IoU thresholds (0.50 to 0.95 from most relaxed to most strict)

IoU Thresh.	#Ground Truths	#Predictions	TP	FP	Precision	Recall	Avg. Inference Time (s)	Forming AP	Collapsing AP	mAP_{COCO}
0.50	359	1913	359	1554	0.188	1.000	0.006	0.976	0.983	0.979
0.55	359	1913	359	1554	0.188	1.000	0.006	0.976	0.983	0.979
0.60	359	1913	358	1555	0.187	0.997	0.006	0.976	0.983	0.979
0.65	359	1913	357	1556	0.187	0.994	0.006	0.970	0.983	0.976
0.70	359	1913	354	1559	0.185	0.986	0.006	0.936	0.980	0.958
0.75	359	1913	348	1565	0.182	0.969	0.006	0.915	0.980	0.947
0.80	359	1913	330	1583	0.173	0.919	0.006	0.840	0.962	0.901
0.85	359	1913	282	1631	0.147	0.786	0.006	0.648	0.869	0.759
0.90	359	1913	187	1726	0.098	0.521	0.006	0.318	0.500	0.409
0.95	359	1913	60	1853	0.031	0.167	0.006	0.041	0.168	0.105
Overall	359	1913	299	1614	0.157	0.834	0.006	0.760	0.839	0.799

Table 35 shows the testing results for YOLOv5. At the two lowest thresholds of 0.50 and 0.55, YOLOv5 was able to detect all ground-truth instances, achieving a perfect recall of 1.000 and an impressive COCO mAP of 0.979. However, this high detection rate is accompanied by a considerable number of False Positives (1554), leading to a low precision of 0.1877. Similarly to RetinaNet, the trend of high recall and low precision suggests that whilst YOLOv5 is highly effective in identifying objects, it also over-predicts. Previous discussion on confidence thresholds with respect to RetinaNet addresses the over-predictive behaviour of YOLOv5.

As the IoU threshold increases, both precision and recall gradually decrease. At the highest threshold of 0.95, the recall drops significantly to 0.167, and precision to 0.031, with the COCO mAP correspondingly falling to 0.105. As previously discussed, performance is expected to deteriorate with increasing threshold, especially at a threshold of 0.95. Considering this, in comparison to Faster R-CNN the rate of deterioration was much better. In comparison to RetinaNet, the rate of deterioration was similar. YOLOv5 made 150 less predictions than RetinaNet and therefore had slightly better precision however exactly the same recall. The performance results of YOLOv5 for collapsing and forming followed the same trend as Faster R-CNN and RetinaNet. This was again, likely due to the class imbalance in the training set. Table 36 shows the key results for each model tested during this development stage.

Table 36: Overall testing results for plume detection using Faster R-CNN, RetinaNet and YOLOv5

Model	Precision	Recall	Inference Time (s)	mAP_{COCO}
Faster R-CNN	0.110	0.125	0.092	0.100
RetinaNet	0.141	0.808	2.770	0.765
YOLOv5	**0.157**	**0.834**	**0.006**	**0.799**

Overall, the COCO mAP achieved by YOLOv5 was 0.799, which suggests generally good performance across most thresholds and the best out of the three models evaluated. Similarly to RetinaNet, there were many false predictions made which is a concern for real-world application due to prediction reliability and model efficiency. However, confidence threshold discussion has suggested it may be possible to improve the precision of YOLOv5 significantly by sacrificing a relatively small amount of recall performance. Additionally, YOLOv5 achieved an average inference time of 0.006 seconds which significantly surpasses Faster R-CNN and RetinaNet.

YOLOv5 did not outperform RetinaNet to a great extent in terms of mAP, however it was significantly better in terms of inference speed. Whilst real-time performance was not a primary aim of this case study, a faster model was still preferable. Therefore, YOLOv5 was used for the remainder of the case study. Visually, it was observed that the bounding boxes predicted by RetinaNet and YOLOv5 were generally tighter compared to Faster R-CNN, suggesting better localisation of plumes.

The results and discussion surrounding confidence thresholds proved crucial to the success of real-world application and therefore some further experimentation was conducted on YOLOv5 to observe the effects of changing this hyperparameter. The default confidence threshold value of YOLOv5 was 0.001 as previously mentioned. Note that there are multiple "default values" for this hyperparameter in the Ultralytics distribution depending on which script is used and for validation, 0.001 is the default. This is to ensure that even lower-confidence predictions are included in the results, allowing for a comprehensive evaluation of model performance and the identification of patterns in detection that may not be apparent when only higher-confidence predictions are considered. For extended experimentation, the value was changed to 0.25 and the results are shown in Table 37.

The results in Table 37 show that when the confidence threshold was raised to 0.25, there was a significant increase in precision across all IoU thresholds due to the drastic decrease in false positives. At an IoU threshold of 0.50, the precision is 0.928 with only 27 false positives, as opposed to the previous 0.188 precision with 1554 false positives. This improvement indicates that the model was generating fewer but more accurate predictions overall. Meanwhile, recall was only reduced marginally at each threshold, indicating that YOLOv5 can effectively detect most ground-truth instances with minimal false detections. There was also a slight decrease in true positives but this was a small cost relative to the large reduction in false positives. These results are promising for real-world application.

The mAP values were slightly lower for lower IoU thresholds but as the criteria for an acceptable detection became stricter, the higher confidence threshold model began to surpass the original model. The COCO mAP was higher for the new model (0.818 compared to 0.799) indicating the superiority of the new confidence threshold. There was still a decline in performance as IoU threshold increased and this became more drastic in the highest thresholds, however the confidence modification made a significant improvement to overall performance and suitability for application.

Table 37: YOLOv5 confidence threshold testing results at a threshold value of 0.25

IoU Thresh.	#Ground Truths	#Predictions	TP	FP	Precision	Recall	Avg. Inference Time (s)	Forming AP	Collapsing AP	mAP_{COCO}
0.50	359	376	349	27	0.928	0.972	5.897	0.983	0.970	0.976
0.55	359	376	349	27	0.928	0.972	5.897	0.983	0.970	0.976
0.60	359	376	348	28	0.926	0.969	5.897	0.983	0.968	0.975
0.65	359	376	346	30	0.920	0.964	5.897	0.983	0.961	0.972
0.70	359	376	335	41	0.891	0.933	5.897	0.978	0.929	0.953
0.75	359	376	330	46	0.878	0.919	5.897	0.978	0.916	0.947
0.80	359	376	315	61	0.838	0.877	5.897	0.961	0.964	0.912
0.85	359	376	274	102	0.729	0.763	5.897	0.900	0.714	0.807
0.90	359	376	183	193	0.487	0.510	5.897	0.586	0.407	0.496
0.95	359	376	59	317	0.157	0.164	5.897	0.243	0.080	0.162
Overall	359	376	289	87	0.768	0.805	5.897	0.858	0.778	0.818

6.3.3 Comprehensive Data Preparation

Initial model development resulted in YOLOv5 emerging as superior for this application (see Section 6.3.2). Therefore, YOLOv5 was used to auto-label remaining data during the comprehensive data preparation stage, which was a prerequisite to later development stages.

Results of the auto-labelling process are presented in Table 38. As expected, the initial labelling phase consumed the most time. Manually correcting Plume75 required 80% of the time it took to label Plume50 from scratch, proving less efficient than anticipated, but still advantageous. Plume150 and Plume180 required slightly less time than Plume75, indicating that the model had successfully learned features from Plume50 and Plume75. However, an additional factor influencing these findings was the improvement of manual labelling ability over time. As labelling experience increased, it was possible to label plumes more accurately in less time. This factor might account for the one-hour discrepancy between Plume150 and Plume180, for instance. Plume100 and Plume120 proved to be the quickest and significantly less effort-intensive to label than other datasets. Plume120, in particular, was labelled in just 20% of the time it took to label Plume50, proving the effectiveness of the method.

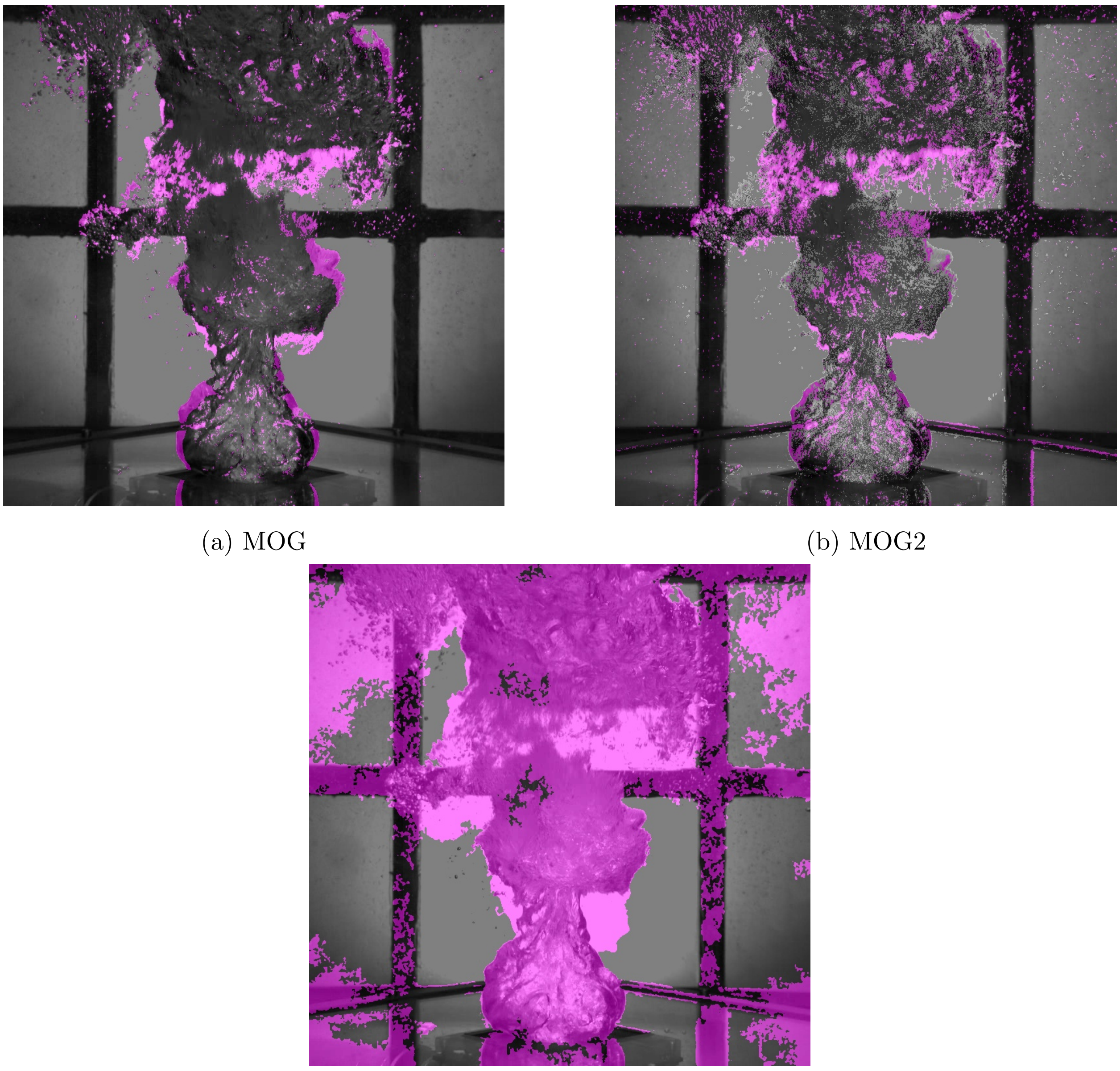
Table 38: Times taken to label each subset

Dataset	AL Stage	Time (Hours)
Plume50	Initial	20
Plume75	1	14
Plume100	3	5
Plume120	3	4
Plume150	2	11
Plume180	2	12

6.3.4 Measurement and Background Subtraction Development

After YOLOv5 was shown to be the best model, the model was developed for real-world measurement. Combining BGS with detection and tracking enabled the model to measure the contact width between plumes and the tuyère surface. This addresses the gap identified in Chapter 2 regarding combining traditional and modern techniques. Additionally, this addresses the gap identified in Chapter 2 regarding using CV techniques to measure real-world characteristics. BGS algorithms included MOG, MOG2 (Mixture of Gaussians variants), GMG (Gaussian Mixture-based Background Foreground Segmentation), GSOC (Google Summer of Code), KNN (K-Nearest Neighbours) and CNT.

6.3.4.1 Analysis of Early Video Frame



(a) MOG

(b) MOG2

(c) GSOC

Figure 93: Figure showing the effects of different BGS algorithms (MOG, MOG2 and GSOC) on a early frame in Plume50

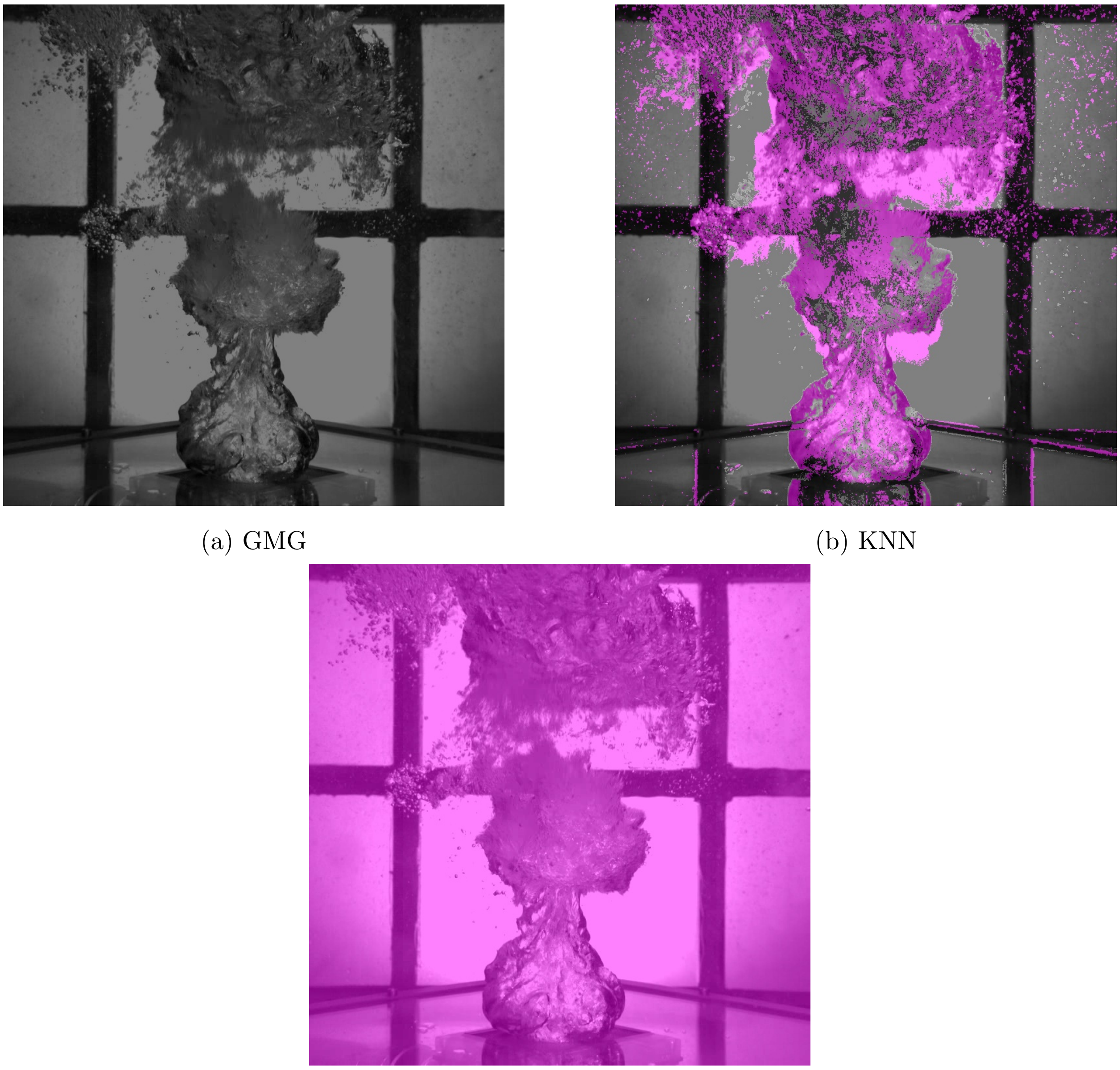

(a) GMG
(b) KNN
(c) CNT

Figure 94: Figure showing the effects of different BGS algorithms (GMG, KNN and CNT) on a early frame in Plume50

Figure 93 and Figure 94 show an early frame of Plume50 footage after being processed by each of the algorithms. In Figure 93(a), MOG algorithm results show its capability to extract the gas edges to some extent. However, its effectiveness in capturing the actual gaseous body is limited. The MOG algorithm captures background pixels through multiple Gaussian distributions and primarily focuses on changes in the scene, which explains its partial success in edge detection but its failure to recognise the fuller body of the gas. This observation aligns with the algorithm design which is to detect changes in pixel values rather than capturing larger, more cohesive moving objects.

In Figure 93(b), the MOG2 algorithm is shown to be slightly more sensitive compared to MOG. This is evident by its marginally better performance in capturing the gas edges as well as parts of plume bodies. It is possible that the improved sensitivity of MOG2 is due to its shadow detection feature that distinguishes between foreground and shadows. However, similar to MOG, MOG2 also struggled with accurately representing the entirety of plumes which suggests their limitations in dealing with larger dynamic fluid structures.

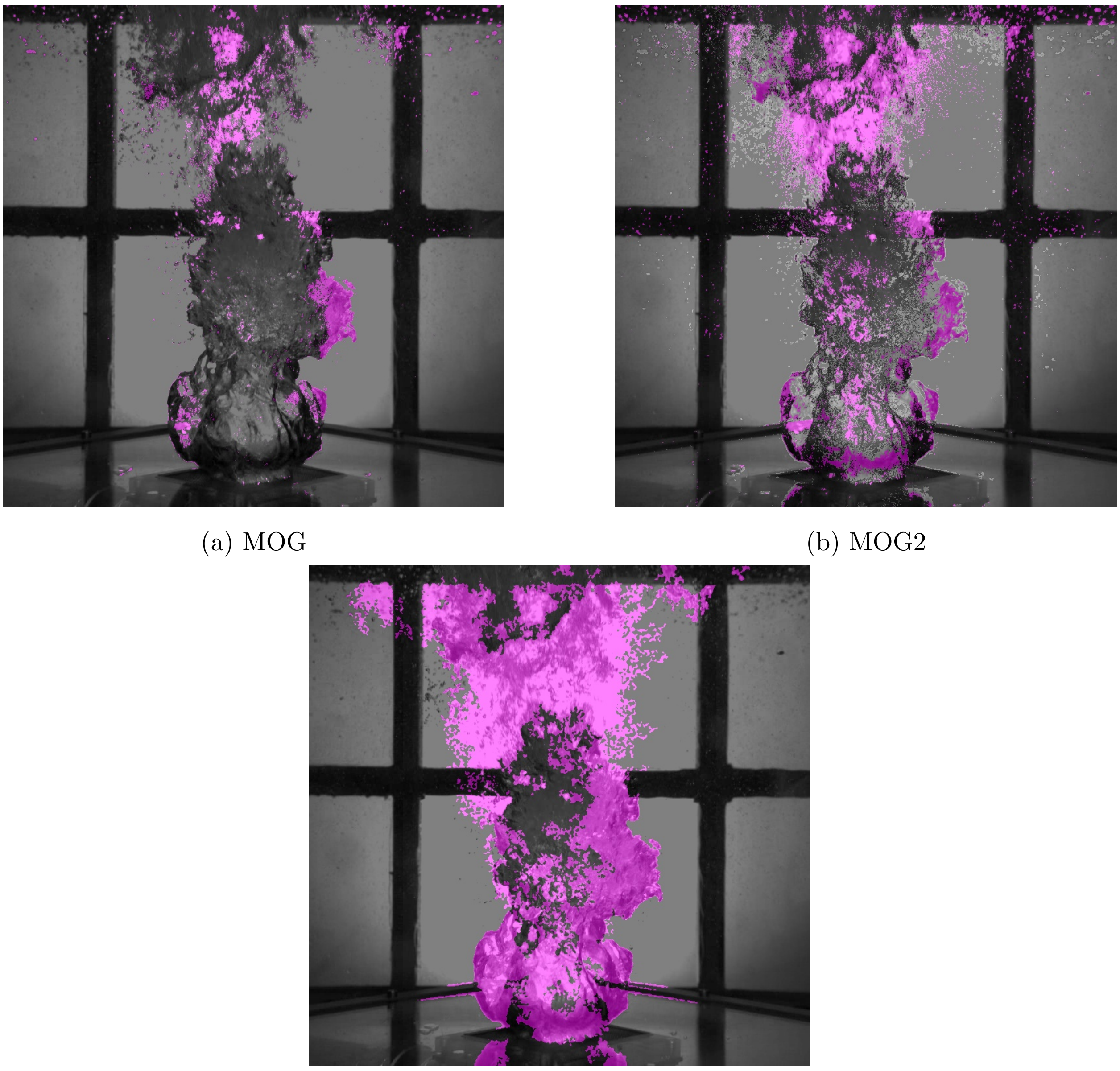
Figure 93(c) presents GSOC results. Whilst it managed to segment most of the gas, it appears overly sensitive, as it also segments parts of the background. The excess segmentation is quite severe since it not only includes areas adjacent to the gas but also the image corners, and even some of the dark column structure. This over-sensitivity is likely due to lighting conditions, since it segmented shadows in the corners and the dark structures which are similar in colour. This behavior suggests that sensitivity to subtle changes, whilst useful in some contexts, leads to significant over-segmentation in this scenario.

In Figure 94(a), it can be seen that the GMG algorithm interpreted the entire scene as background due to GMG Bayesian updating mechanism, which requires a few frames to initialise and adapt to the data. Therefore, early frames are not accurately processed since it is necessary for the algorithm to establish a reliable background model.

Figure 94(b) shows KNN results, which appears to be the most effective among the tested methods. It successfully captured both the gaseous body and its edges with reasonable precision. KNN performs dynamic analysis of the surroundings of each pixel which allows it to adapt effectively to the scene, although it does exhibit some over-sensitivity around the gas edges. This performance suggests a good balance in sensitivity, enabling it to detect and segment the gas more adequately than the other algorithms already discussed.

Finally, Figure 94(c) shows CNT results, which in contrast to GMG, interpreted the entire scene as foreground. This is due to relying on establishing pixel stabilities over time, which requires a few frames to differentiate between foreground and background effectively.

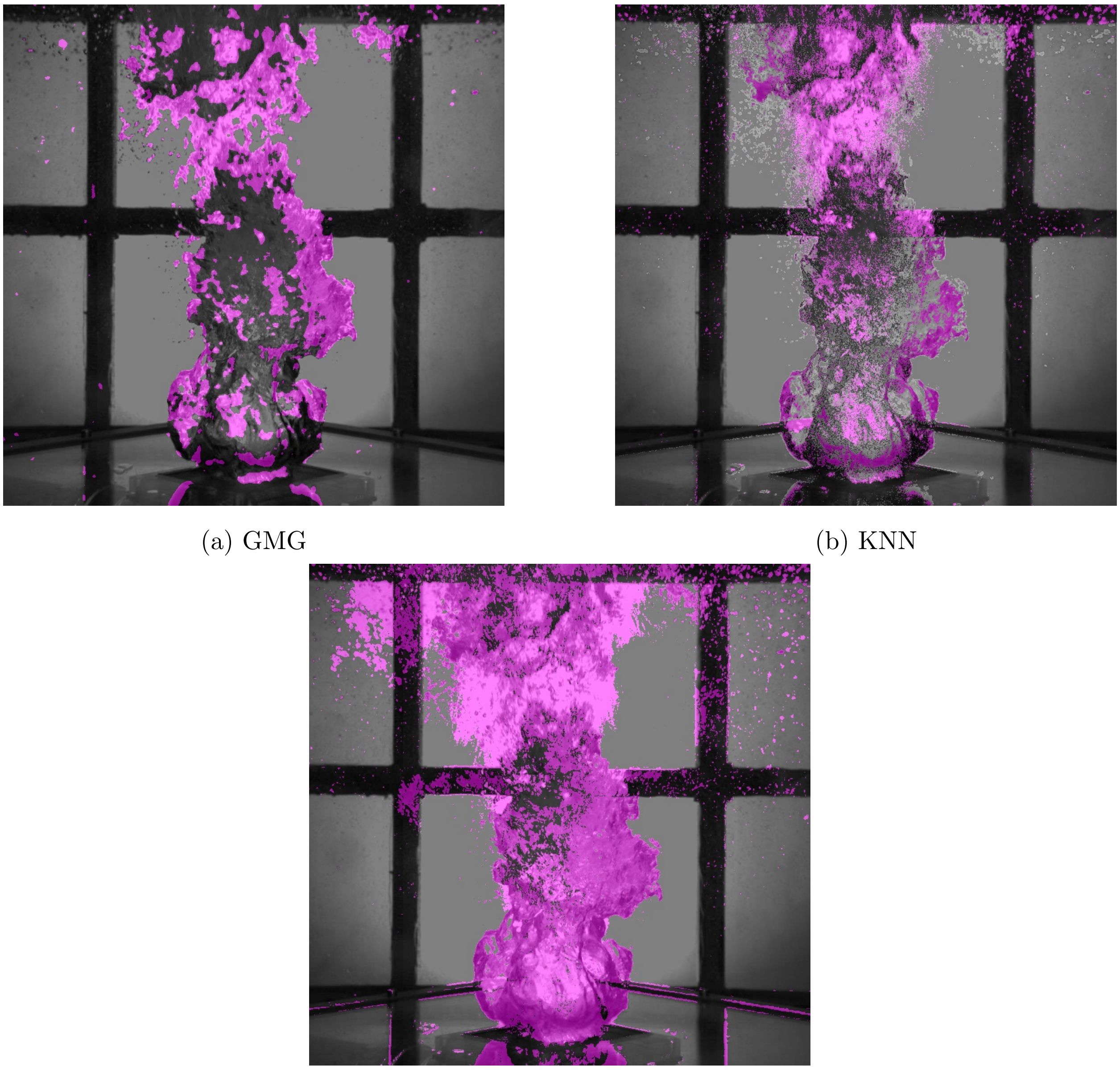
6.3.4.2 Analysis of Mid-Video Frame


(a) MOG

(b) MOG2

(c) GSOC

Figure 95: Figure showing the effects of different BGS algorithms (MOG, MOG2 and GSOC) on a mid-video frame in Plume50


(a) GMG (b) KNN

(c) CNT

Figure 96: Figure showing the effects of different BGS algorithms (GMG, KNN and CNT) on a mid-video frame in Plume50

Figure 95 and Figure 96 show a frame within the 8th second of footage after being processed by each of the algorithms. In Figure 95(a), the MOG algorithm results show a reduced ability to segment the gas. The edges of the gas are defined worse than in the earlier frame in Figure 93(a) and there is no improvement in segmentation of the plume body. This suggests that the MOG algorithm is not suitable for this application. This is because the Gaussian distributions are failing to accurately capture changes in the scene, which is likely due to the fluidity of plumes causing Gaussians to struggle modelling the continuous nature of the gas.

Figure 95(b) shows the more sensitive MOG2 algorithm performed better than the MOG2 algorithm as it managed to capture some edges of the plumes and small regions of the body. Whilst the MOG2 algorithm performance did not suffer as much as the MOG algorithm when moving from Figure 93 to Figure 95, the edge capturing ability does appear to have reduced slightly. This suggests that either this particular frame is difficult for MOG-based algorithms to perform on, or this application is generally not well-suited for MOG or MOG2 use. The performance difference between MOG and MOG2 is due to the increased sensitivity and shadow detection capability of MOG2, indicating the potential benefit of increased sensitivity when modelling fluid structures with Gaussians, as well as the benefit of using MOG2 to reduce the impact of lighting changes.

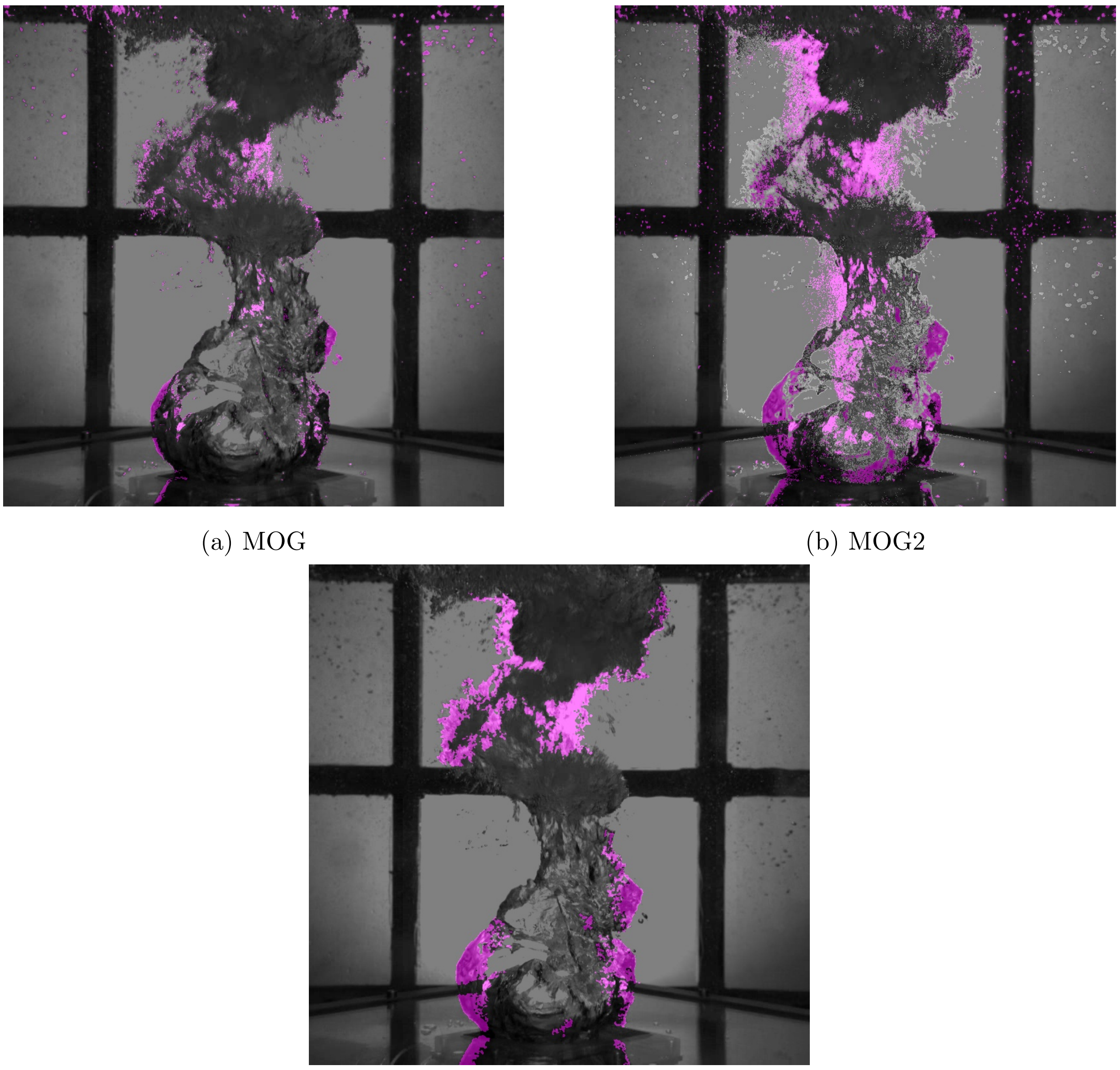
In Figure 95(c), the GSOC algorithm shows significant improvement from its initial over-sensitivity. Whilst it continued to segment some unnecessary background elements, its ability to adapt to the scene over time is evident. This is because it is designed to adapt to dynamic backgrounds, camera jitter, unstable lighting conditions, various types of motion and other subtle changes [240, 254]. The algorithm captured the majority of the gas whilst reducing unnecessary background segmentation. However, it still shows some over-sensitivity towards the top half of the frame and some under-segmentation in the middle section of the frame. The improvement of the GSOC algorithm over time suggests it may be useful for this application, but may need a long time to properly adapt to footage.

Figure 96(a) presents the GMG algorithm's initial segmentation results. After allowing it time to initialise, GMG began to detect the gas edges and parts of the body. The performance can be considered quite good, since most of the edges are captured well and these are more important than the body, which is due to the high sensitivity of the Bayesian updating mechanism through constant adjustments based on new data. However, it struggled to segment plume bodies, indicating that Bayesian updating is suitable for complex edge detection but not the inner section of objects. Therefore, it has potential for this application.

In Figure 96(b), the KNN algorithm shows a significant difference in approach compared to in Figure 94(b). Whilst it still captured most edges and some parts of the body, it adapted towards using a much more conservative approach. This indicates the complexity of the scene, since KNN adapts conservatively to avoid over-segmenting fine details. This is also evidenced by the fact that the plume edges are less complex than the bodies. This means that whilst there is far less noise, there is also far less segmentation in general, meaning important features are potentially excluded. Further experimentation would be required to reveal whether the performance shown in Figure 96(b) is inferior or superior to that shown in Figure 94(b), and therefore whether performance is improving or deteriorating over time.

Finally, Figure 96(c) shows the CNT algorithm now effectively differentiating the gas from the background. It accurately segmented most of the gas, with a slight decline in performance towards the top as the gas disperses and therefore changes morphology. This is because the pixel stability-based approach enables the algorithm to learn which pixels are static and which are changing over time. Since it is based on pixel stability thresholds, the default threshold values used here must be near optimal for this application. This pattern of segmentation aligns well with the objective of measuring plume width at the tuyère surface since that is the area of the scene in which precise segmentation is most crucial, making the CNT algorithm particularly suitable for this aspect of the study. Despite some over-sensitivity, the detailed capture of the gas structure suggests its potential as the preferred choice for background subtraction if used in combination with additional processing techniques to address over-segmentation.

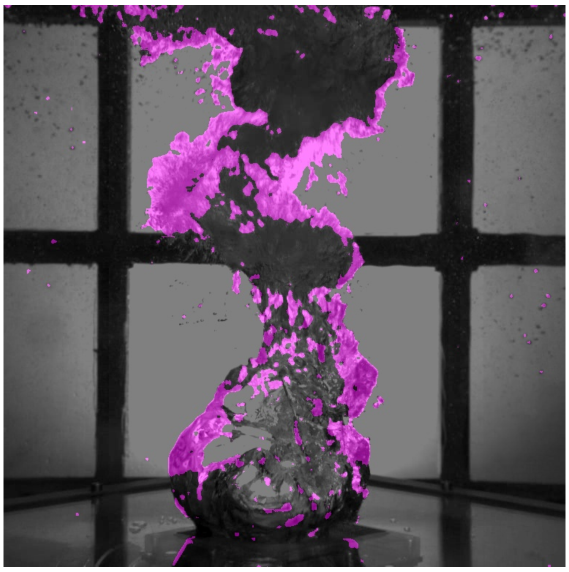
6.3.4.3 Analysis of Late Video Frame



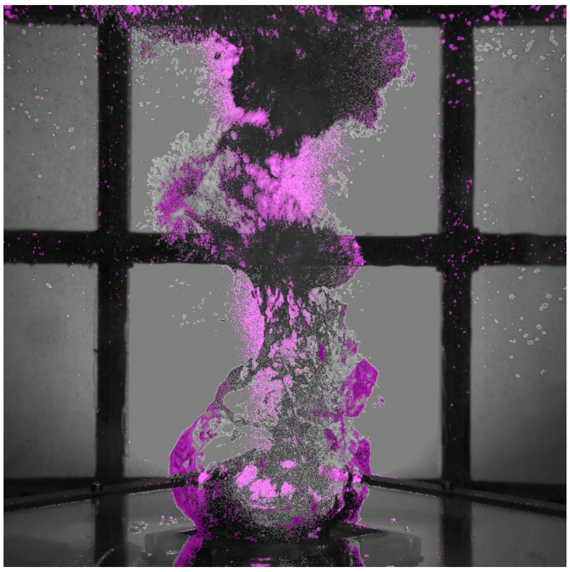
(a) MOG (b) MOG2

(c) GSOC

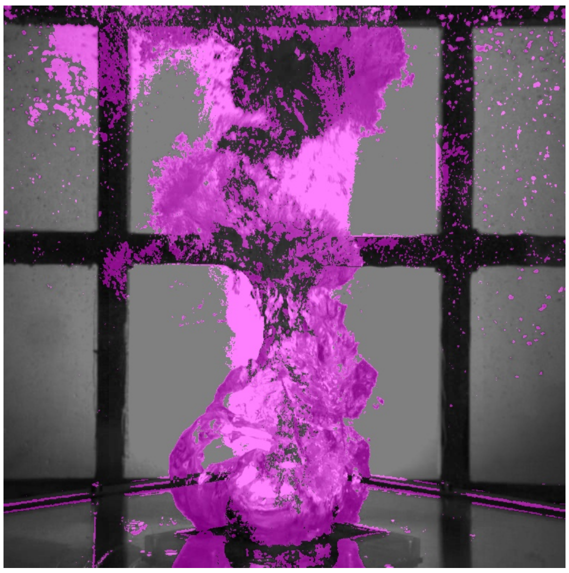
Figure 97: Figure showing the effects of different BGS algorithms (MOG, MOG2 and GSOC) on a late video frame in Plume50



(a) GMG



(b) KNN



(c) CNT

Figure 98: Figure showing the effects of different BGS algorithms (GMG, KNN and CNT) on a late video frame in Plume50

Figure 97 and Figure 98 show a frame within the 28th second of footage after being processed by each algorithm. In Figure 97(a), MOG results continue to show a lack of improvement in segmenting the gas. Its inability to accurately define either the edges or the body of the gas is consistent with observations in Figure 95(a), reinforcing the conclusion that MOG is not optimal for this task due to Gaussian modelling struggling with the fluidity of the gas structure.

In Figure 97(b), the MOG2 algorithm, despite being more sensitive than MOG, also failed to deliver effective results and continued to struggle with accurately capturing the gas. Whilst MOG2 has shown to be slightly better than MOG, it is still not an ideal choice for this specific application based on the results. As discussed, the increased sensitivity and shadow detection improve suitability for this application, however they still result in unacceptable performance. Developing an MOG variant more sensitive than MOG2 may be beneficial.

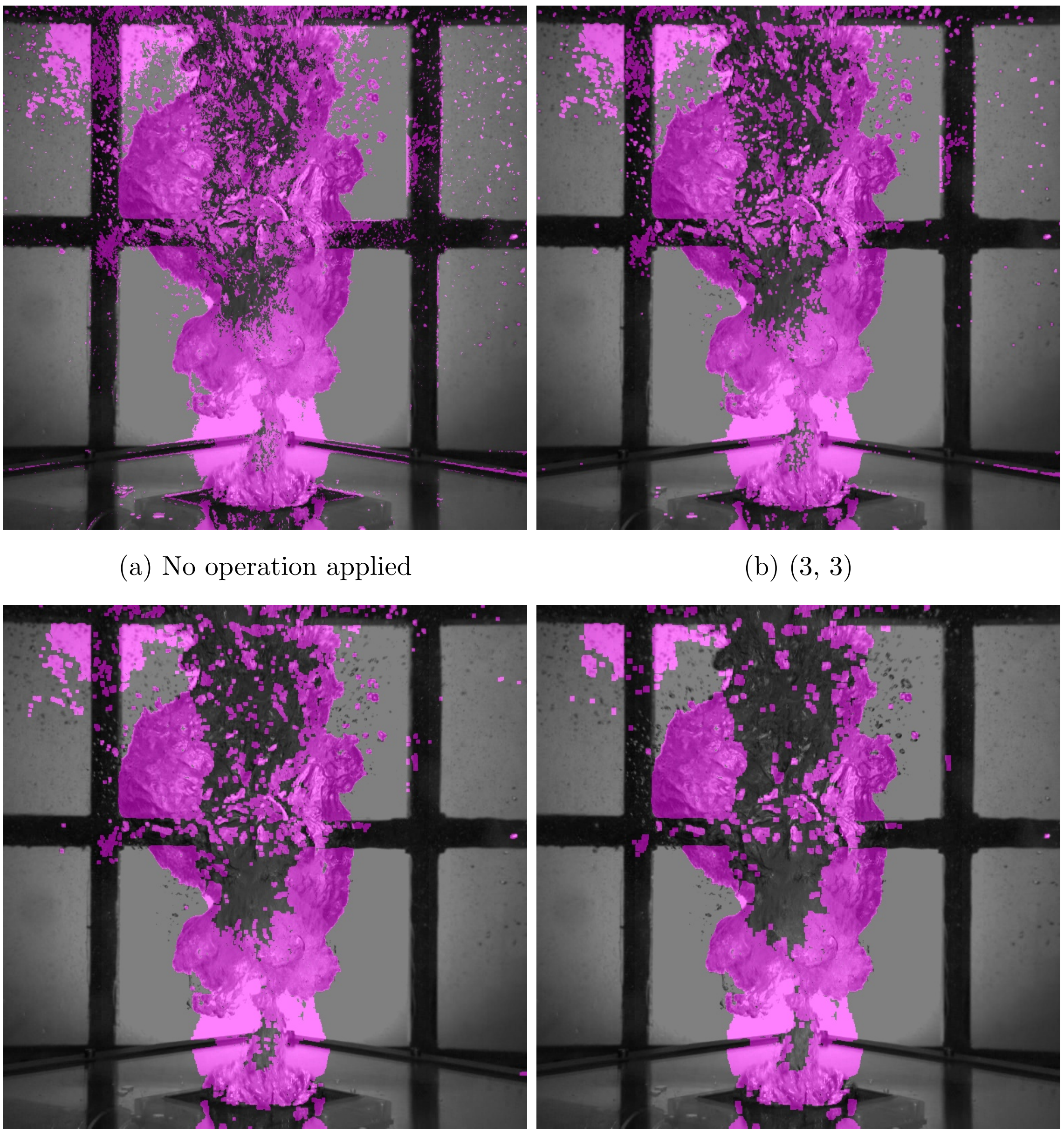
In Figure 97(c), GSOC again shows significant change from the previous frame. Whilst Figure 93(c) showed large over-segmentation, Figure 95(c) showed a more conservative estimation. Meanwhile, Figure 97(c) shows a significantly more conservative estimation than both previous frames, suggesting high adaptiveness, which is due to GSOC being designed to adapt to subtle changes. In Figure 97(c), some edges are captured however it is not substantial enough for application, whilst minimal body regions are captured.

Figure 98(a) shows the GMG algorithm maintained similar performance to that shown in Figure 96(a). There is good performance shown in terms of edge segmentation and body segmentation is nearly non-existent. These results suggest that once the Bayesian updating mechanism has initialised, the GMG algorithm does not change much over time. This is because it becomes less sensitive to new changes as it is continuously refined, and eventually stabilises. Despite this, it could be useful for edge detection of complex dynamic structures.

Figure 98(b) shows KNN algorthm performance declined in effectiveness over time. It still captures some edges and some parts of the body but has become too conservative, leading to lack of segmentation of key features. This suggests that whilst KNN initially showed promise, its adaptability over time was detrimental. As discussed, this has been caused by the complexity of the scene causing the algorithm to adapt by avoiding over-segmentation of fine details, resulting in it under-segmenting plume bodies. Whilst the algorithm may be effective at adapting to dynamic scenes, this particular scenario is too complex.

Finally, Figure 98(c) presents CNT performance, which remains consistent with previous results. It successfully segments the majority of the gas but tends to over-segment more dynamic regions of the video. Overall, the CNT algorithm appears able to maintain high accuracy, particularly at the bottom of the frame which aligns well with the objective of measuring plume-tuyère contact width, indicating its suitability for application. As discussed, this is because the pixel stability-based approach enables the algorithm to distinguish background and foreground pixels over time. Analyses and discussion indicate CNT was the most appropriate choice for this study. After selecting CNT for background subtraction, focus shifted to mitigating the tendency of the algorithm to over-segment.

6.3.4.4 Analysis of Noise Removal



(a) No operation applied

(b) (3, 3)

(c) (5, 5)

(d) (7, 7)

Figure 99: Figure showing the effects of different kernels on a frame in Plume50

Following algorithm selection, morphological operations were used to reduce noise. Morphological opening (as mentioned in Section 6.2.4) consists of erosion (size reduction) to remove minor contours that added noise, followed by dilation (expansion) to restore the size of the remaining segmentation mask. Morphological operation tests used a standard grid kernel (a square matrix where all values are equal to one) and sizes varied between 3x3, 5x5, and 7x7 pixels, to determine the most effective size for noise reduction. Figure 99(a) shows the frame prior to any processing, where there is noise in the form of minor contours and specks. These small artifacts are the result of the initial segmentation by the CNT algorithm, which is sensitive to minor variations in pixel intensity.

Figure 99(b) shows the effect of a (3,3) kernel, which marginally reduces the noise around the gas, noticeable by the disappearance of some specks particularly in the top-right quarter of the image where the vertical black column is located. This is because the erosion process in morphological opening removes small, isolated pixel groups. However, the small kernel size means that not all noise is effectively removed, highlighting the trade-off between preserving fine details and reducing noise.

The result of using a (5,5) kernel is presented in Figure 99(c), where a significant portion of the surrounding noise is eliminated, though a few specks remain. The larger kernel is more effective at removing isolated noise without overly distorting the shape of the segmented gas areas. This balance is due to the increased erosion strength followed by dilation, which helps in maintaining the original shape of larger features whilst eliminating smaller artifacts.

With a (7,7) kernel, as seen in Figure 99(d), nearly all the noise is removed. However, the square shape of the kernel introduces a grid-like pattern which causes the original mask to lose natural shapes and details. This occurs because the kernel's uniform structure averages out the pixel values in a grid formation, resulting in straight edges and corners that do not accurately represent the true contours of the objects being processed. This makes it unsuitable for making precise measurements and demonstrates the importance of selecting an appropriately sized kernel to balance noise reduction and shape preservation. Therefore, the (5,5) kernel was selected as the most suitable option.

In hindsight, a (7,7) kernel may not have been as detrimental to plume-tuyère contact width measurement as first expected and to uncover whether this is true or false, further experimentation would be required. However, observation of the tuyère region in Figure 99(a) to (d) shows that (5,5) was the smallest kernel size that successfully removed noise in this area, and therefore had the least chance of compromising measurement accuracy, which made it the most logical choice at this point of the model development process.

After morphological opening was implemented, it became apparent that the segmented plume areas were surrounded by a residual mask from previous frame tracking, as shown in Figure 100(a). To address this, the technique of in-range colour removal was applied.

The unwanted mask was easily distinguishable by its lighter shade of fuchsia compared to the desired mask, and was therefore sampled multiple times to determine its RGB value

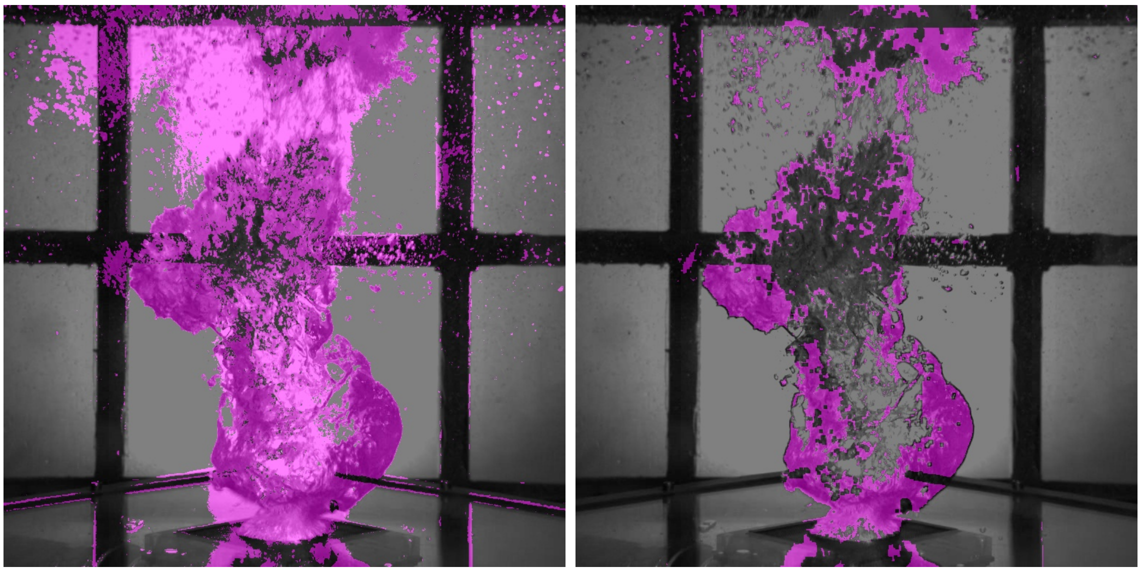
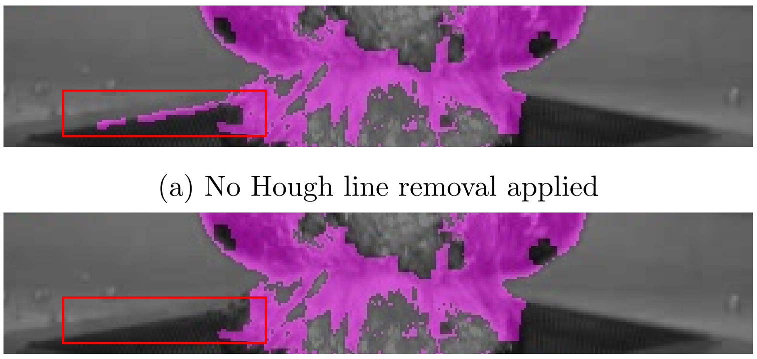

(a) No in-range removal applied (b) In-range removal applied

Figure 100: Figure showing the effects of in-range colour removal on a frame in Plume50

range. The minimum and maximum RGB values sampled were then used as parameters for the in-range removal process to eliminate mask pixels where image pixels fell within that range. RGB values of the image created by superimposing the mask onto the original image were probed which resulted in the minimum RGB values being (200, 85, 190) and the maximum RGB values being (255, 149, 255). This approach was particularly effective due to the bright background of the column setup emphasising the unwanted mask, which is valuable knowledge for future projects. This method successfully preserved key features whilst removing the residual mask, and the result is shown in Figure 100(b).


(a) No Hough line removal applied

(b) Hough line removal applied

Figure 101: Figure showing the effects of Hough line removal on a frame in Plume50

The last denoising step involved correcting the occasional segmentation of tuyère edges in some frames. This issue arose as the tuyère itself seemed to vibrate during the footage, leading the CNT algorithm to interpret it as movement. To counter this, the Hough line transform technique was used to identify lines in the segmentation mask in the tuyère area and remove them. This required some trial-and-error of the Hough transform function which required custom input on the expected angle and length of lines detected. The transform was applied once on the left side of the tuyère and once on the right side since these lines were at different angles. Figure 101(a) and Figure 101(b) depict the tuyère before and after the application of Hough line removal, respectively. The bottom image clearly shows the absence of the lines in the segmentation mask that are present in the top image, demonstrating the success of the approach. This adjustment was crucial, particularly since these erroneous lines affected the accuracy of the plume-tuyère contact width measurements.

6.3.5 Object Detection Optimisation

The grid search results for optimisation of YOLOv5 are shown in Table 39. The setup for each experiment has already been presented in Table 32 of Section 6.2.5. To address inference times, the lowest inference time of 4.6ms was achieved by Experiment 3 which used Data Setup 3, small YOLOv5 and 50 epochs of training. It is assumed that data setup does not significantly impact speed since there was no correlation found between the two. The results also showed no correlation between the inference speed and the number of training epochs. Changing epochs typically affect model performance in terms of mAP but not inference speed, so this was as expected. Interestingly, inference times decreased as model size increased, which is expressed in Figure 102.

Figure 102 shows inference speeds for experiments using YOLOv5s ranged from 4.64ms to 4.80ms, whilst experiments using YOLOv5m inferred at speeds between 10.27ms to 10.79ms, and experiments using YOLOv5l ranged from 16.81ms to 17.05ms. This makes sense because the increasing size of the YOLOv5 architecture means the data must be processed by more layers, which results in more time consumption. Whilst speed is an important factor of any CV model, in this case study real-time performance was not a primary goal. Therefore, whilst lower inference speeds were both preferred, and more promising for future projects evolving from this one, the primary aim was to maximise precision, recall and mAP. It is worth noting that even the slowest inference time here was 17.05ms which is 58.65fps, making all experiments fast enough for real-time application (keep in mind this is just for YOLOv5 inferences and not any additional processing such as background subtraction).

Table 39: Grid search results for optimisation of YOLOv5 involving different data setups, network sizes and numbers of training epochs

Exp. No.	Precision	Recall	Forming AP	Collapsing AP	Jetting AP	mAP_{COCO}	Inference Time (ms)
1	0.822	0.862	0.712	0.793	0.591	0.699	4.802
2	0.854	0.878	0.834	0.780	0.668	0.761	4.746
3	0.807	0.791	0.643	0.518	0.632	0.598	**4.635**
4	0.828	0.858	0.753	0.811	0.597	0.720	10.789
5	0.869	0.873	0.843	0.788	0.670	0.767	10.496
6	0.811	0.780	0.656	0.547	0.619	0.607	10.388
7	0.802	0.880	0.763	0.837	0.607	0.736	16.814
8	0.854	0.869	0.849	0.798	0.675	**0.774**	16.853
9	0.831	0.757	0.672	0.558	0.609	0.613	16.871
10	0.809	0.860	0.716	0.800	0.609	0.708	4.687
11	0.858	0.879	0.845	0.770	0.703	0.773	4.654
12	0.812	0.797	0.667	0.520	0.622	0.603	4.657
13	0.814	0.851	0.735	0.823	0.594	0.718	10.311
14	0.847	0.904	0.852	0.786	0.683	0.773	10.500
15	0.800	0.795	0.663	0.542	0.612	0.606	10.407
16	0.809	0.846	0.732	0.823	0.562	0.706	17.020
17	0.839	0.877	0.848	0.778	0.680	0.769	17.034
18	0.798	0.788	0.667	0.554	0.589	0.603	16.961
19	0.817	0.839	0.706	0.797	0.586	0.696	4.742
20	0.854	0.901	0.846	0.775	0.701	**0.774**	4.747
21	0.809	0.767	0.658	0.531	0.606	0.598	4.776
22	0.806	0.852	0.722	0.835	0.563	0.706	10.274
23	0.872	0.858	0.831	0.761	0.682	0.758	10.289
24	0.800	0.769	0.679	0.548	0.566	0.598	10.278
25	0.801	0.833	0.726	0.827	0.545	0.700	16.981
26	0.863	0.848	0.833	0.755	0.641	0.743	17.053
27	0.817	0.775	0.660	0.554	0.591	0.602	17.033

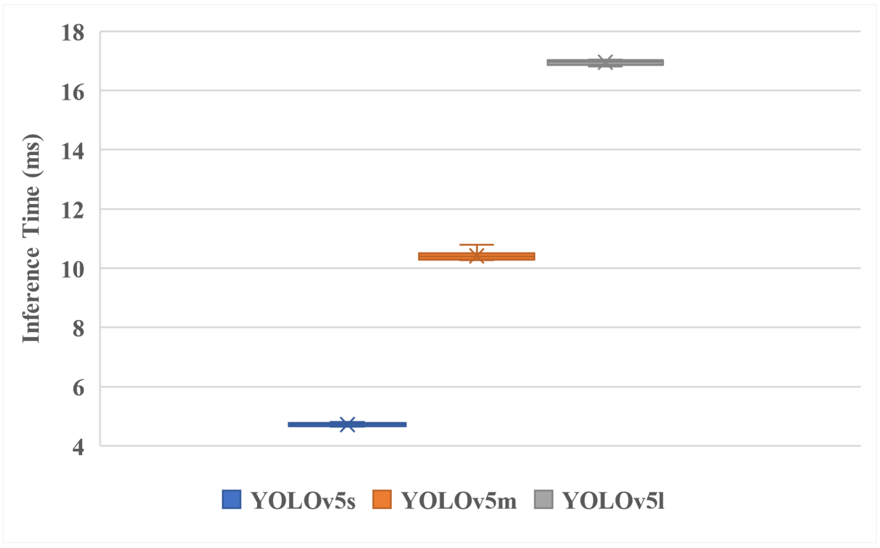


Figure 102: Box plot showing how inference speed changed with model size. YOLOv5s, YOLOv5m and YOLOv5l are the small, medium and large variants, respectively

Table 39 does not suggest any correlation between model size and mAP since the two best experiments used the small and large YOLOv5 variants. Furthermore, comparing mAP and model size of every experiment showed no correlation. It may be expected that a larger, more complex model is capable of capturing more intricate features and would therefore surpass the precision of smaller networks, however the results show that this was not the case here, and is likely due to YOLOv5s being large enough for effective feature extraction which would mean layers in larger variants were excessive. Similarly, it would be logical to assume that more training epochs would increase precision, however this is often not the case in practice and there is typically a cut-off point where increasing epochs causes overfitting. Experiment 8 and Experiment 20 achieved the highest mAPs. Data Setup 2 was used in both of these which suggests it was the best data setup. This is probably due to the same principle used in the auto-labelling approach for the comprehensive data preparation stage. To elaborate, Data setup 2 used Plume50, Plume75, Plume150 and Plume180 for training and was tested on Plume100 and Plume120. Therefore, the model was trained on two extremes of possible scenarios and tested on data in the middle of these extremes, allowing it to capture and interpolate between features from both sides. This aligns with the motive behind the auto-labelling approach and verifies the successful results of that stage. Also, experiments using Data Setup 2 consistently achieved the highest mAP performances which is expressed in Figure 103. Note that in Experiment 20, training took one minute and 34 seconds to train per epoch which overall took approximately 313 minutes.

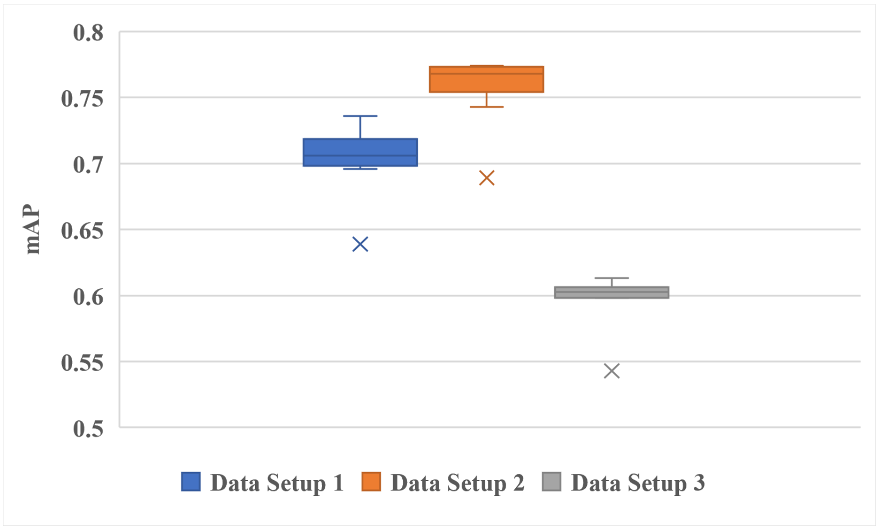


Figure 103: Box plot showing how mAP changed with different data setups

Figure 103 shows all Data Setup 2 experiments achieved a COCO mAP between 0.743 and 0.774, all Data Setup 1 experiments achieved a COCO mAP between 0.696 and 0.736, and all Data Setup 3 experiments achieved a COCO mAP between 0.598 and 0.613. These results show Data Setup 2 was the best setup which can be explained by discussion for Table 39 (in this section) related to exposing the model to a wide range of scenarios, which enhanced its ability to generalise on unseen scenarios. Additionally, the difference in mAP values between Data Setup 1 and Data Setup 3 are due to less new features being introduced after the injection rate was increased from 100Nm^{3}h^{-1}to 180Nm^{3}h^{-1}. Therefore, since Plume50 and Plume75 had more unique features than the higher injection rate videos, they were harder to predict on without explicit training on those unique features. Also, since the results show that all of the highest performing models used Data Setup 2, followed by Data Setup 1, followed by Data Setup 3, this is a clear indication that the data setup used in any given experiment dictated the overall performance achieved. This emphasises the importance of a well-curated dataset over a complex neural network or training strategy.

Looking at Data Setup 2 experiments, model performance still exhibited no relationship with model size or the number of epochs. In real-world application, it therefore is most efficient to use the smallest possible network and the lowest number of epochs to minimise inference time and computational requirements. Closer analysis reveals Experiment 8 or Experiment 20 should be chosen to align with the goal of maximising the mAP. Experiment 20 required a one-off training requirement of 200 epochs, whilst Experiment 8 used YOLOv5l which significantly increases computational requirements for both training and inference, and also takes almost four times the amount of time to make inferences. Additionally, the recall of Experiment 20 marginally surpassed that of Experiment 8, further supporting the case for using the design of Experiment 20 in real-world application. Note that Experiment 2 achieved an mAP of 0.761 and an inference time of 4.75ms whilst using YOLOv5s and only 50 training epochs which made it competitive. Whilst this may be useful for future projects, in this case, the aim was to maximise mAP and therefore it was not selected.

After analysing the results across experiments in Table 39, discussion on differences between Table 39 results and Table 37 results (confidence-adjusted YOLOv5 results in Section 6.3.2) brings up some interesting points of discussion regarding the overall progress of model development. Note that whilst Table 37 explicitly shows the results at each IoU threshold and includes true positives and false positives, Table 39 shows the overall results after averaging these values for each experiment. Also, whilst Table 37 results were produced using Plume50 with YOLOv5 at a confidence threshold of 0.25, Table 39 results used all six injection rate datasets. This means differences in metrics such as precision and recall are based on the prediction of many more samples. For clarification, Data Setup 1 tested on 3019 instances (Plume150 and Plume180), Data Setup 2 tested on 3187 instances (Plume100 and Plume120) and Data Setup 3 tested on 3650 instances (Plume50 and Plume75). This is in comparison to the 1799 instances in Plume50 (excluding jetting instances). Also note the difference in jetting instances adding a whole new class to the challenge during Table 39 experiments. For brevity and clarity, discussion will primarily focus on the best experiment for this application, which was Experiment 20, in comparison to the results in Table 37.

Comparison shows that firstly, the precision in every experiment in Table 39 surpasses the overall precision in Table 37 whereas the recall in Table 39 roughly lies in the middle of the recall values for experiments in Table 37. This suggests that the increase in training data (roughly four times as much), enhanced precision regardless of different data setups, network sizes, training epochs, an extended testing set and an additional testing class (jetting). This emphasises the importance of a sufficiently large training set. Also, the results indicate that recall was more variable and this may have been due to the change in the number of instances across different data setups. Experiment 20 surpassed the results achieved in Table 37 in terms of both recall and precision, demonstrating the effectiveness of this design.

Secondly, differently to Table 37 and other initial model development results, class APs in Table 39 varied in terms of which classes were predicted the best. Overall forming instances were generally predicted the best, followed by collapsing instances, followed by jetting instances. This follows a different pattern to that found and discussed in Section 6.3.2 which was based on the number of training samples for each class.

Generally the collapsing instances were more difficult to accurately localise than other class instances because they were the most complex and most changeable, especially considering the plumes left the field of view in a collapsed form which required some subjectivity during labelling as to when they were no longer detectable. The effect of this issue appeared to be non-existent in earlier stages of model development. The model actually performed better on collapsing instances than forming instances. However, Table 39 shows the model struggled slightly more with collapsing than forming when larger datasets with more variety were introduced. At this point, the labelling issue may have come into effect.

Jetting AP results were generally the lowest of the three classes, indicating the theory on training sample size previously discussed is still valid, but with the full dataset, other factors (such as the complexity of predicting specific classes) came into play. However, this order did vary. For example, almost all experiments using Data Setup 1 favoured collapsing instances, followed by forming instances, followed by jetting instances. This could suggest that the first four injection rate datasets provided superior collapsing samples, or that the last two injection rate datasets included collapsing samples that were easier to predict.

Generally, experiments using Data Setup 2 and Data Setup 3 favoured forming instances over collapsing and jetting instances. Similarly to Data Setup 1 discussion, this could suggest that the first four datasets included forming samples that were easier to predict. Data Setup 3 typically favoured jetting instances over collapsing instances. This suggests that the complexity of predicting collapsing instances was particularly high in the first two datasets, and emphasises the dominant effect of a complex class over the number of training data samples. Whilst there was some variation in the number of samples for each class across different data setups, collapsing instances were still significantly dominant over the other two classes, and forming instances were significantly dominant over jetting instances.

The superior and inferior classes in Table 37 and Experiment 20 in Table 39 swapped, and Table 37 achieved higher class APs and a higher overall COCO mAP. This is due to the fact that there was an extra class predicted in Table 39, as well as roughly double the number of instances in the test set. Considering this, the difference in COCO mAP is actually impressive and shows that the model from Experiment 20 in Table 39 is a significantly evolved version of the model reported in Table 37.

Finally, the inference time in Table 37 is marginally higher than YOLOv5s model experiments in Table 39, which is expected since they both used YOLOv5s. This could be due to random fluctuations in system performance at the time of obtaining results.

6.3.6 Real-World Validation

It was essential to validate the effectiveness of models in measuring real-world plume characteristics, beyond standard object detection metrics. This addresses the gap identified in Chapter 2 regarding a lack of real-world validation of models. A key aspect of this assessment involved judging the accuracy of plume width measurements at the tuyère surface. It was observed that whilst the model generally performed well, it did not consistently succeed in every frame. In some instances, it failed to make detections, which could be due to a need for more training data despite already using over 3000 frames with more than 11000 labels, or a limitation inherent to the YOLOv5 model itself which could be improved by using a more advanced model such as YOLOv8. Furthermore, the width of the tuyère design was known to be 68.5mm and so any measurements falling below this threshold were considered to be either the result of minor noise or missed detections, and were therefore ignored. Model measurements were validated against data from prior research [264], which involved manual measurement techniques. Initially, these measurements were intended for validation purposes, but it was realised that in some instances, the computer vision model actually provided more accurate results than the original manual methods. This is further explored later in this section.

Figure 104 exhibits a comparison of the model predictions and manual measurements from [264] for the Plume100 data. It is observable that the manual measurements exhibit a relatively smooth, cyclic pattern, whereas the model's measurements show more variation within these cycles, suggesting a noisier output. Typically, the natural dynamics of plumes expanding and detaching from the tuyère surface would create a regular oscillation in width measurements. However, other than occasional inaccuracies of the model already addressed, there were some inter-cycle fluctuations which were partly due to its reliance on a fixed row of pixels for measurements. If this row encounters sub-optimal lighting conditions or camera angles in any given frame, the model measurement accuracy was affected. In contrast, manual measurement allows for more flexibility, as a human observer might adjust the measurement row for optimal visibility. This complication was recognised during this research and could be addressed in future work by enhancing the gas stirring simulation setup for a clearer view of the tuyère surface, or by integrating multiple sensing technologies, such as combining stereo vision with depth cameras or LiDAR sensors, to achieve more robust measurements.

Figure 105 highlights frames where significant differences were observed between model predictions and the original manual measurements from [264]. Frame 66, is shown in Figure 105(a), where the model estimated the contact width to be approximately 20mm wider

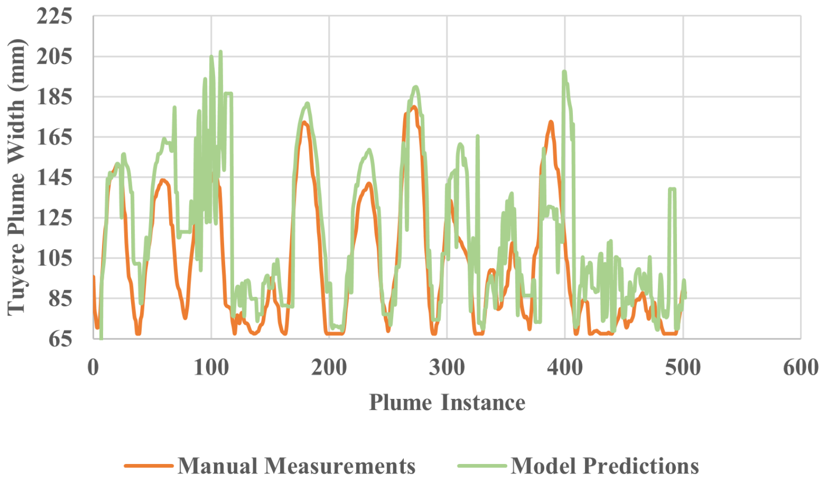
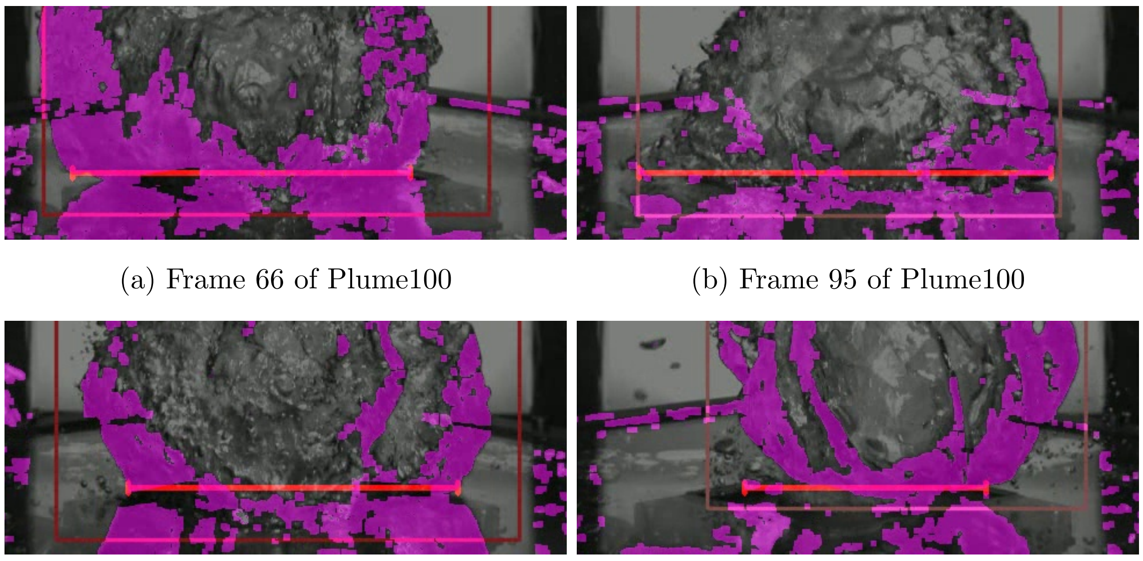


Figure 104: Graph comparing model predictions to manual measurements performed in another author's work



(a) Frame 66 of Plume100

(b) Frame 95 of Plume100

(c) Frame 315 of Plume100

(d) Frame 439 of Plume100

Figure 105: Figure showing model predictions on frames of significant discrepancy between predictions and manual measurements

than the manually measured width. The red horizontal line in Figure 105(a) illustrates the measurement taken by the model. Upon closer inspection, it is evident that whilst the segmentation was accurate, the pre-set measurement line was positioned slightly higher than the actual surface, which led to a wider section of the plume being measured. This issue was anticipated early in the development process but due to occasional shaking of the tuyère surface during injection, consistently measuring along an accurate line proved challenging. Alternative methods, such as using multiple lines for measurement were explored, but the chosen single-line approach yielded the most consistent results. This problem opens up an avenue for future work.

Next along Figure 104, the most obvious inconsistency occurs across Frame 95 and surrounding frames, presented in Figure 105(b). Here, the model failed to accurately segment the plume edges. Although the measurement line in Figure 105(b) appears accurate, this was coincidental. The overall results suggest that the segmentation algorithm had difficulties in this section of the footage.

In Frame 315, shown in Figure 105(c), there is a discrepancy of around 40mm which is observable in Figure 104. Unlike the earlier case in Figure 105(a), the measurement line here was appropriately placed. Adjusting the line lower would not result in a significant reduction in the measured width, despite less than ideal segmentation in the lower pixel rows. This, and other similar instances, indicate that the original manual measurements were not consistently reliable and that, in some cases, the model measured the width more accurately. Figure 106 shows the contact width manually measured is approximately 155cm as opposed to the original manual measurement of approximately 110cm.

Lastly, also seen as a discrepancy in Figure 104, Frame 439 in Figure 105(d) exhibited a similar fluctuation to what was seen around Frame 95. In this instance, gas bubbles detaching from the left side of the plume interfered with the segmentation algorithm. Whilst these bubbles are ideally ignored since they do not contribute to the washing effect, some residual noise was poorly eliminated. This resulted in the model overestimating the contact width.

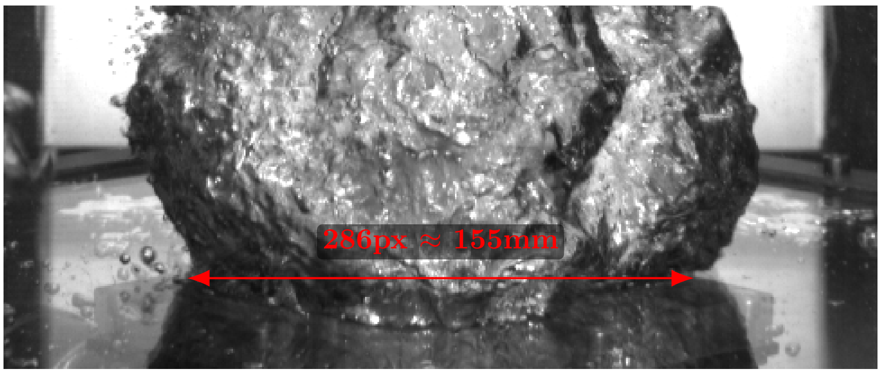


Figure 106: Frame 315 labelled manually by authors of this thesis and converted from pixels to millimetres, showing improvement of model over original human measurements

Another real-world plume characteristic that was of interest was the jetting length, which occurs during the initial gas injection into the liquid. Object detection identifies different stages of gas behaviour such as jetting, forming, and collapsing plumes, which facilitated these measurements. The model determined the jetting length using the height of bounding boxes associated with the jetting class. Previously recorded data suggested that the maximum jetting length, observed across various injection rates, was between 12mm to 15mm. It was hypothesised that this maximum length would occur at the highest injection rate of 180Nm^{3}h^{-1}, following the logic that increased injection rates lead to longer jetting lengths.

The final model in this research was trained using data from Plume50, Plume75, Plume150, and Plume180, and therefore jetting estimation analysis began with predictions on Plume100 and Plume120 which are shown in Figure 107 and Figure 108 respectively. Analysis of these figures reveals two key observations. Firstly, the jetting lengths for Plume100 and Plume120 were measured to be approximately 20mm to 26mm, and 18mm to 39mm, respectively. This aligns with the expectation that higher injection rates result in longer jetting phenomena. Secondly, both measurements significantly exceed the previous maximum jetting length for Plume180 in [264]. This difference raised questions about the accuracy of either model measurements or the original manual measurements. To look further into this, further clarification is provided with the aid of Figure 109 and Figure 110.

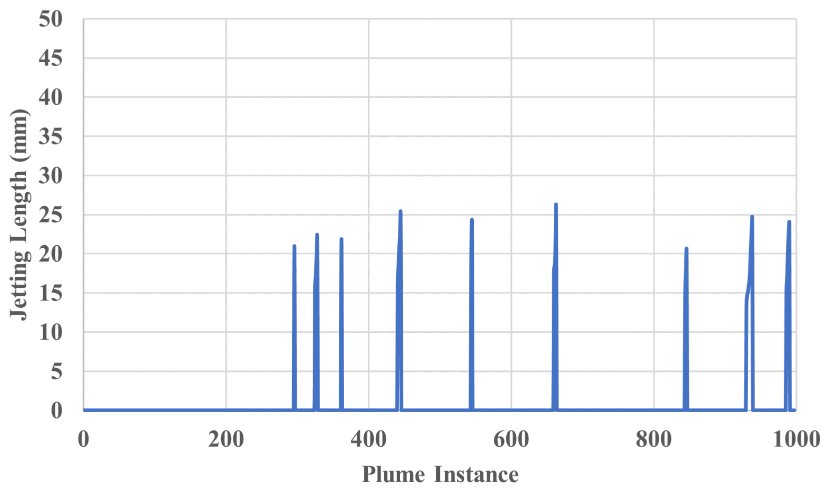


Figure 107: Graph showing model jetting length predictions across Plume100

Figure 109 presents the new manual measurements of jetting lengths for Plume180 which were taken from the model annotations. They show that the maximum jetting length reaches roughly 61mm, reinforcing the previously proposed trend where an increase in the injection

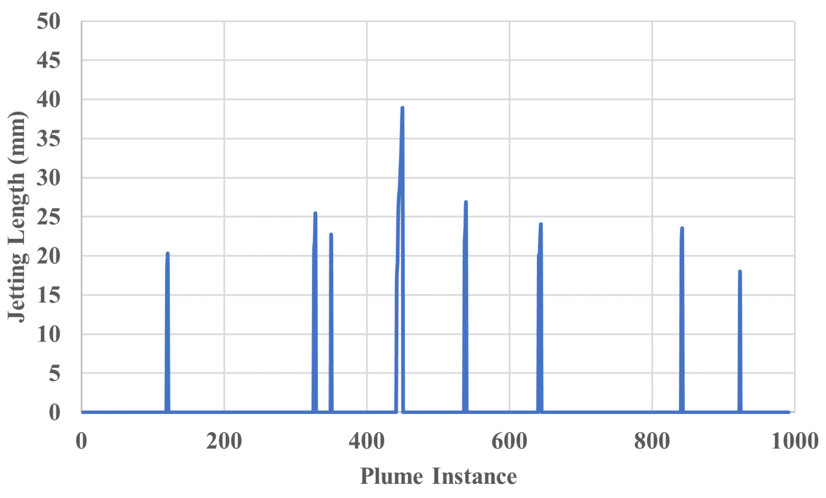


Figure 108: Graph showing model jetting length predictions across Plume120

rate corresponds to a larger jetting lengths. However, this measurement quadruples the maximum length of 12mm to 15mm reported in earlier research. Therefore, whilst the manual annotations and inference measurements of this case study appear to align, there are discrepancies with the previous manual measurements. This considerable difference emphasised the need for observing the actual images.

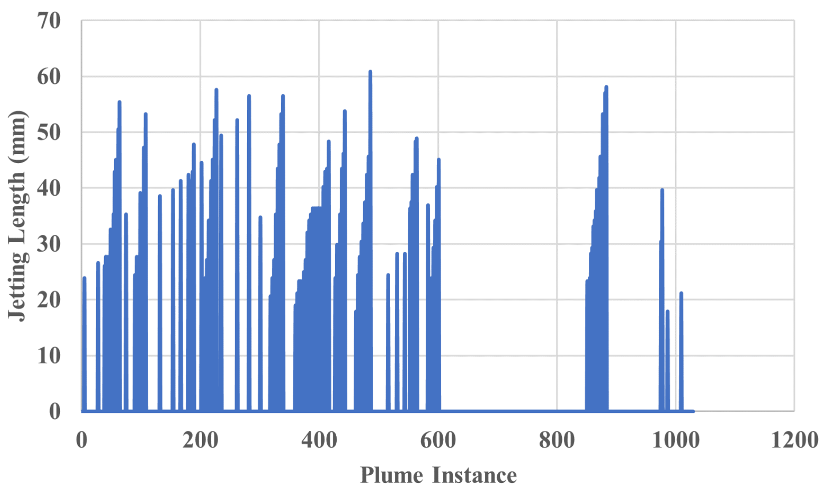


Figure 109: Graph showing Plume180 annotated jetting lengths as part of this work

Figure 110 shows Frame 299 (instance 486) from the Plume180 dataset, which Figure 109 shows as the instance with the longest jetting length across all predictions of all injection rates. The jetting length in this frame is highlighted with a red double-ended arrow and is annotated to be precisely 112 pixels in length. Applying the pixel-to-millimeter conversion rate of 1.84, this measurement translates to approximately 60.87mm. This measurement closely matches the data in Figure 109 which therefore validates the accuracy of the model predictions of jetting length. This finding also challenges the accuracy of the previously reported range of 12mm to 15mm for jetting length from earlier authors [264].

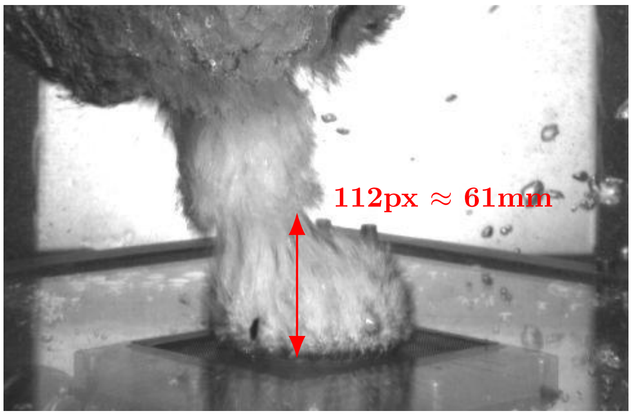


Figure 110: Image of the 486th instance in Plume180 with the red label proving accurate model predictions

6.4 Industrial Application

The model outputs the state of each plume (classified as jetting, forming, or collapsing), the width and height of each plume, the height of jetting events, plume-tuyère contact width and plume frequency. Using these outputs along with some known environmental variables, it was possible to determine the wear rate factor WF, as well as a stirring efficiency rating SE. This section will detail the calculation of WF and SE values, which were devised as part of this project, and present results.

6.4.1 Methodology for Calculating Wear Rate Factor and Stirring Efficiency

Equation (25) shows how WF has been calculated, which is constituted by three different variables. Washing severity WS indicates the severity of plume-tuyère contact events, velocity ratio VR is the ratio between the inlet velocity of injected gas, to the maximum velocity reached (which is 180Nm^{3}h^{-1}), and the frequency of plumes occurring, F_{plume}.

$$WF = WS \times VR \times F_{plume} \tag{25}$$

Equation (26) shows how WS has been calculated, which is essentially the width of the contact event, $W_{contact}$, normalised using the maximum surface width W_{max}. Since washing events cause refractory wear, the more severe they are, the more wear is occurring. Normalisation is performed to ensure WS reflects the contact width in relation to the column width, and therefore adapting the approach to work across column setups of varying width.

$$WS = \frac{W_{contact}}{W_{max}} \tag{26}$$

Equation (27) shows how VR has been calculated and is the ratio of the gas inlet velocity U_{inlet}, to the maximum velocity record U_{max}. Higher inlet velocities aid gas stirring but also induce stress on the refractory lining. Normalisation is performed to ensure the inlet velocity is accounted for in relation to the maximum gas velocity, which ensures the approach is consistent across column configurations of different scales (some simulations may be 10% of the furnace size whereas others may be 50% of it, for example).

$$VR = \frac{U_{inlet}}{U_{max}} \tag{27}$$

Equation (28) shows how the plume frequency F_{plume} was calculated, where N is the number of plumes occurring over a video and D is the video duration in seconds. Plumes promote gas stirring but also contribute to refractory wear.

$$F_{plume} = \frac{N}{D} \tag{28}$$

The approach presented here also extends to the calculation of stirring efficiency which has been performed using Equation (29). This approach assumes that larger volume plumes result in improved stirring efficiency, and therefore VP_{plume} is included, which is the ratio of instantaneous plume volume to the maximum plume volume, as well as the the velocity ratio and plume frequency as these also affect how effective stirring is.

$$SE = VP_{plume} \times VR \times F_{plume} \tag{29}$$

Equation (30) shows how the plume volume proportion has been calculated, which is the ratio of the instantaneous plume volume $V_{instantaneous}$, to the maximum plume volume at the point just before collapse, V_{max}. The instantaneous point was chosen arbitrarily as four frames before the point just before collapse, which was used consistently for all measurements.

$$VP_{plume} = \frac{V_{instantaneous}}{V_{max}} \tag{30}$$

6.4.2 Results of Wear Rate Factor and Stirring Efficiency Calculations

For assessing different configurations, the wear rate factor and stirring efficiency values for Plume100 and Plume120 were calculated and the results are shown in Table 40 and Table 41.

Table 40: Results of wear rate factor calculation for Plume100 and Plume120

Dataset	*WS*	*VR*	F_{plume}	*WF*
Plume100	5.770	0.556	0.247	0.792
Plume120	4.741	0.667	0.198	**0.626**

Table 41: Results of stirring efficiency calculation for Plume100 and Plume120

Dataset	VP_{plume}	*VR*	F_{plume}	*SE*
Plume100	0.826	0.556	0.247	0.113
Plume120	0.871	0.667	0.198	**0.115**

Table 40 shows Plume100 had more severe washing overall than Plume120, suggesting that an injection rate of 100Nm^{3}h^{-1} results in more contact between plumes and the tuyère surface, and therefore more refractory wear is experienced, which is reflected by the higher wear rate factor of Plume100. A lower velocity ratio was seen in Plume100 to Plume120, which is due to the lower injection rate, and a higher frequency was seen in Plume100, suggesting lower injection rates result in more frequent plume occurrence. Essentially, the injection rate of 120Nm^{3}h^{-1} resulted in less wear.

Table 41 shows a higher injection rate resulted in a marginally higher plume volume proportion which suggests better mixing. Also, stirring efficiency was found to be slightly higher for Plume120 than Plume100. Therefore, these results show that the increase in injection rate may have improved stirring efficiency but not significantly.

Out of the two configurations evaluated, Plume120 appears better due to less intense washing events and therefore less wear, as well as marginally better stirring efficiency. Overall, this suggests an injection rate of 120Nm^{3}h^{-1} is superior. However, whilst this demonstrates the effectiveness of the model developed, a more extensive set of experiments should be analysed to build a comprehensive understanding of different setups and how they affect wear rate and stirring efficiency.

6.5 Conclusions

This chapter has presented a case study where various plume characteristics have been measured throughout video footage of an experimental simulation of gas stirring in the BOF, using YOLOv5 for plume detection, DeepSORT for plume tracking and CNT background subtraction for plume segmentation. Using plume measurements, this work has presented an approach for calculating a wear rate factor and a stirring efficiency rating for any given simulation experiment. Assessing these variables across a range of setups aids gas stirring design optimisation which is highly beneficial for the process and industry as a whole, since it helps to minimise equipment degradation and improve mixing.

This study explored various object detection models and YOLOv5 was the best. In the final model, YOLOv5 was trained on approximately 3000 images and tested on approximately 1500. This resulted in precision, recall and COCO mAP results of 0.854, 0.901 and 0.774 respectively. The full model ran at 7.6fps, which is a good starting point for any future developments requiring real-time application, provided the model is optimised for speed. This study also explored various background subtraction algorithms and the CNT algorithm was the best. This algorithm was then combined with morphological opening, in-range removal and Hough line removal for denoising purposes, which ensures robustness.

Furthermore, contact width and jetting length measurements were compared to validation data which found that the proposed approach was accurate for most frames but still had room for improvement, and in some cases, outperformed the manual measurements taken in previous research. Wear rate factor and stirring efficiency rating calculations showed an injection rate of 120Nm^{3}h^{-1} was superior to 100Nm^{3}h^{-1}, since it resulted in less wear and higher stirring efficiency.

In conclusion, research contributes significantly to the advancement of steel production technology in terms of process monitoring, particularly in the context of fluid dynamics, whilst improving design capabilities through automation, efficiency and novel insights. Furthermore, this research contributes to CV through the innovative application of techniques such as detection, tracking and segmentation to gain an understanding of complex and dynamic industrial environments. Through use of Google Colaboratory, which does not have any requirements other than an internet connection, the model can easily be distributed to anyone wishing to analyse gas stirring simulation experiments, which ensures it is scalable for its current purpose.

This case study has addressed three of the research gaps identified in Chapter 2. Firstly, it has involved the generation of a novel dataset based on the ladle pouring process, which addresses the lack of datasets available for developing CV applications for steel production processes. Secondly, it has contributed to addressing the lack of real-world measurements performed by CV models. Lastly, it has addressed the lack of hybrid CV models that make use of both traditional and deep learning-based techniques.

6.6 Future Work

Future work should initially include detection refinement to overcome missed detections, followed by BGS refinement to reduce noise issues and therefore enhance measurement accuracy.

Additionally, there are various pathways for future work related to this case study, such as improving inference times and deploying the model onto an Internet of Things (IoT) device for real-time feedback. Another pathway could be to extend the model by integrating a time series model and applying it to the current model's output for advancing predictive maintenance capabilities. The current model could also be applied to varying mixing element designs, or even different processes. Therefore, this work lays a foundation for monitoring various dynamic fluid characteristics and drawing insights that are valuable for enhancing the operational efficiency and quality of steelmaking processes.

Chapter 7: Case Study Comparison

This chapter will contain a comparative analysis of the three case studies presented in Chapters 4, 5 and 6 by looking at them holistically in terms of their methodologies and results. In the methodological comparison, notable similarities and differences will be discussed in terms of tools utilised, data preparation, model development and evaluation metrics. In the results comparison, discussion will be based on model performance, robustness, scalability and limitations.

7.1 Introduction

The rise of computer vision (CV) technology has begun to rapidly improve the state of steel production technology by enhancing a wide variety of processes through aspects such as operational efficiency and product quality. Chapters 4, 5 and 6 explored the application of CV for three processes: hot metal ladle pouring (Case Study 1), steel galvanisation (Case Study 2), and basic oxygen furnace (BOF) gas stirring (Case Study 3). Each case study highlighted the applicability of CV to complex industrial challenges, whilst also contributing findings unique to each case study.

The objective of this chapter is to synthesise insights from individual studies whilst identifying some of the wider implications of this project that only become apparent when viewing case studies as cohesive components. By doing this, it is ensured that analysis of the work undertaken is thorough.

The overall contribution of this chapter is that it provides valuable insights into developing CV applications for steel production. It offers a comparison of different methodological aspects across different scenarios, whilst also evaluating model performances, assessing the robustness and scalability of systems in real industrial environments, and addressing commonly occurring limitations when undertaking challenges based on application of CV to steel processes.

7.2 Methodological Comparison

This section compares the methodologies of each case study in terms of the software and hardware tools used, the data preparation, how models were developed and how they were evaluated.

7.2.1 Software Tools

For all three case studies, similar software tools were used. This primarily includes Python, Python libraries and the VGG (Visual Geometry Group) Image Annotator (VIA) [276] labelling tool. Microsoft Excel was also used for all three studies when outputting real-world measurements from Python, as this was a more appropriate format when dealing with sponsors, collaborating researchers and writing research papers. However, there were a few notable differences between the case studies.

Firstly, as discussed, Case Study 2 and Case Study 3 used Roboflow dataset conversion tools [216], whereas annotations for Mask R-CNN (Mask Region-based Convolutional Neural Network) in Case Study 1 required no conversion from VIA. Secondly, Case Study 3 used Google Colaboratory, which was primarily for ease of access when collaborating with another researcher who used the model to gain insights. This was an advantage when collaborating, however as mentioned in Section 6.3.2 of Chapter 6, there was a downside to this which created an issue when revisiting models.

Both cloud and local approaches were used in this body of work and in future, it would be preferable to operate locally but upload models to the cloud when collaborating with researchers who do not specialise in machine learning (ML). This way, the advantages of both approaches are gained. For deployment of the splatter severity measurement model demonstration, Docker [257], FastAPI [258], Uvicorn [259] and TensorRT [201] were used which was unique to Case Study 2.

7.2.2 Hardware Tools

There were some differences between the hardware used for each study. For example, Case Study 1 was entirely conducted using the NVIDIA RTX 2070 Super GPU (graphical processing unit), whilst Case Study 2 used the same GPU for development but for deployment used the NVIDIA Jetson Orin Nano's GPU. Case Study 3 did not use either as it made use of the Google Colaboratory GPU which was the Tesla T4. The differences in GPU selection affected various aspects of the development process of each study. For example, the 2070 Super GPU is more powerful than the T4 overall, and is therefore better for model development, but the T4 is specifically built for machine learning training and inference, meaning it is likely more suitable for inference and especially for deployment since it is much more power efficient [217, 277]. Also, the Jetson Orin Nano GPU is less powerful than the 2070 Super GPU, but is intended to be embedded into other systems for Internet of Things (IoT) applications, and is therefore much more efficient in terms of size and power consumption

[256]. There was benefit in using the 2070 Super and T4 simultaneously when working on multiple case studies in parallel was required, and having access to multiple GPUs would certainly be advantageous for time efficiency in real-world model development scenarios.

7.2.3 Data Preparation

In all three studies, data was provided by industrial sponsors or researchers and required processing from their raw format into formats appropriate for model development. As mentioned, all data was then labelled using VIA [276]. However, the way in which data was treated from that point onwards differed for each study. As mentioned in Chapters 4, 5 and 6, every case study addressed the research gap identified in Chapter 2 regarding the lack of datasets available for developing CV applications for steel production.

7.2.3.1 Case Study 1

In Case study 1 frames were extracted to produce 1fps sequences to ensure inferences could be performed on complete pouring videos without taking too long. Contrast-Limited Adaptive Histogram Equalisation (CLAHE) was also applied to combat poor lighting. When developing the model for industrial application, the training set was extended whilst the validation set remained unchanged. As explained in Section 4.2.1 of Chapter 4, Mask R-CNN used the same set for intra-epoch validation as it did for model testing, which is a common approach used when available data is limited. This contrasted with other case studies that used four distinct datasets which included extra sets for model testing and production testing.

7.2.3.2 Case Study 2

In Case Study 2 frames were extracted at the natural rate of 25fps, split into training, validation, testing and production testing sets and labelled. After labelling, the Roboflow conversion tool [216] was used to convert the dataset from VIA JSON (JavaScript object notation) format to YOLOv5 (You Only Look Once) Pytorch .txt format. Unlike Case Study 1, four separate sets were used since there was much more data available. The training set was obviously used for training, the validation set was used for intra-epoch validation, the testing set was used to evaluate the performance of the trained model on unseen data, and the production set was used to assess and adapt the model performance to perform robustly in real-world scenarios. Therefore, this study involved more complex data management than Case Study 1 and facilitated a more rigorous assessment of model robustness.

7.2.3.3 Case Study 3

In Case Study 3 frames were provided individually, not as video, and were split into training, validation and testing sets and labelled. After labelling, Roboflow conversion tools [216] were used to convert the dataset from VIA JSON format to YOLOv5 Pytorch .txt and PASCAL VOC (Pattern Analysis, Statistical Modelling and Computational Learning Visual Object Classes) XML (extensible markup language) format for the YOLOv5 and RetinaNet models respectively [216, 270]. For Faster R-CNN, annotations were exported in the COCO (Common Objects in Context) JSON format. Labelling was conducted in two stages, the initial manual labelling of Plume50, and the auto-labelling of the remaining plume datasets using YOLOv5. Finally, the production testing involved an approach inspired by k-fold cross validation which used all datasets for training and testing in turn to mitigate the effects of any biases in the data. The multi-format, auto-labelling and validation approaches used in this case study make the most advanced study overall in terms of data strategy.

7.2.4 Model Development

In all three studies, existing neural networks were trained for manufacturing-based applications, validated and tested. In every case, deep learning was complemented by various other machine learning and image processing techniques. Each network, as well as any additional techniques incorporated, served a purpose that contributed towards to monitoring of a real or simulated process.

7.2.4.1 Case Study 1

Case Study 1 used Mask R-CNN for its state-of-the-art instance segmentation performance at the time of project initiation, as reported in Section 2.5.1 of Chapter 2, to achieve precise ladle segmentation in poor lighting conditions. Meanwhile, Case Study 2 and Case Study 3 used YOLOv5 for its state-of-the-art object detection performance at the time they were initiated, which is shown in Section 2.3.8 of Chapter 2. Also, the second and third case studies used background subtraction (BGS) rather than instance segmentation. See Section 2.6 of Chapter 2 for the distinction between segmentation networks and background subtraction algorithms. Although background subtraction results were presented in Chapter 4 (Case Study 1), this was additional experimentation that was not a core element to the model developed. In all three studies, the COCO dataset was used to pre-train models to give a foundational understanding of what objects look like. This was inspired by the fact that literature in Chapter 2 frequently reported model evaluation via the COCO dataset and COCO mean average precision (mAP) metric.

Case Study 1 model development was challenging due to limited data availability, meaning focus was primarily placed on compensating for a smaller dataset through hyperparameter optimisation. Through completion of the case studies, it has been learned that the dataset often dictates model performance more than the model itself. This was not fully understood during Case Study 1 and regardless, data available was limited and labelling ladle masks in poorly-lit images was far more time-consuming than labelling bounding boxes on air knives and plumes.

7.2.4.2 Case Study 2

Whilst Case Study 2 and Case Study 3 both used YOLOv5 and background subtraction, these stages were conducted differently in each case study. Note that this addressed the research gap identified in Chapter 2 regarding the lack of hybrid CV models that make use of both traditional and deep learning-based techniques. In Case Study 2, the overall aim of the study was to develop a tool that could provide a zinc splatter severity level in real-time during galvanising. For this study, YOLOv5 was used in two main ways. Primarily, it was used to adjust the measurement region in real-time as the air knives moved upwards and downwards. The relevant region of splatter always remained below the air knives, and since the CNT (Counting) background subtraction algorithm was the basis of the splatter measurement, and one of the limitations of BGS algorithms is dynamic backgrounds (see Section 2.6 of Chapter 2), YOLOv5 was used to detect the locations of each air knife and therefore identify where the splatter was expected to occur. Secondarily, in the provided footage, and through discussions with operators, it was clear that the cameras at the area of interest were moved regularly (in-between operator shifts). Therefore, any measurement methods needed to be scaled differently depending on the camera position to ensure model robustness. By using YOLOv5, the sizes of predicted bounding boxes for air knives (in pixels) could be used to give an indication of the distance between the camera and the air knives. Based on a reference bounding box size, it was possible to scale splatter severity measurement boundaries up or down accordingly.

With the exception of the air knives which moved up and down infrequently and were handled by YOLOv5, and of course the splattering zinc which was desirable to measure, there was almost no movement. Therefore, the CNT algorithm was used to segment moving splatter from the background in real-time. The main advantage of using the CNT algorithm is that it is significantly faster than all available BGS algorithms whilst maintaining high precision [8]. This advantage was well-suited to Case Study 2 since real-time performance was desirable. Another advantage of the CNT algorithm is that it adapts well to environmental changes, and since the splatter was erratic, fluctuating dramatically multiple times per second, the CNT algorithm was even more fitting for this application [8].

7.2.4.3 Case Study 3

In terms of Case Study 3, the overall aim of the study was to develop a tool that could monitor and record gaseous plume states and geometries for design optimisation of gas mixing elements. For this study, YOLOv5 had several purposes. Firstly, the classification element of detection was used to distinguish between three different plume states which were jetting, forming and collapsing. Secondly, the bounding boxes were used to measure the height and width of plumes in order to calculate their aspect ratio at the time just before they collapsed, which was the last frame on which they were classified as forming. Thirdly, the bounding boxes were used to identify and isolate plumes that were washing the mixing element surface, so that the BGS mask was only measured in the area of interest when measuring plume-tuyère contact width. Finally, the bounding boxes and classification were used to identify and measure the height of jetting events.

With the exception of small gas bubbles and streams detaching from the main plumes, which were dealt with using a variety of denoising techniques, the background was mostly static. Therefore, the CNT algorithm was used to segment plume motion from injection until they collapsed and left the visible region of the water column. Like with Case Study 2, the CNT algorithm's adaptability to the environment was key in the success of the model development, since plume behaviours were complex and at times, unpredictable. The speed of the CNT algorithm was not as crucial in this case study, however it contributed to the overall quality and industrial impact of the final model.

The main use of CNT BGS in this case study was to read plume width when it formed on the mixing element surface, as this affected refractory wear. Whilst YOLOv5 located a mixing element washing event using a bounding box, BGS measured plume width precisely within the box at pixel-level, which was not possible using object detection alone.

Furthermore, Case Study 3 involved experimentation with Faster R-CNN, RetinaNet, and YOLOv5 in the initial stages, as opposed to just choosing one model based on literature review. This was because experience, and therefore efficiency, had been improved over the course of the two previous case studies.

7.2.5 Model Evaluation Metrics

The main difference between computer vision metrics used in different case studies was that segmentation uses masks to calculate intersection-over-union (IoU) whereas detection uses bounding boxes. This is explained in Section 3.4.1 of Chapter 3. Other than this difference, conventional metrics remain unchanged between detection and segmentation, as evidenced in model performances reported in the literature in Section 2.3 and Section 2.5 of Chapter 2. In terms of evaluating real-world measurements such as ladle parameters, splatter severity and jetting length, there were some trends as well as differences.

In Case Study 1 there was no real-world evaluation data available for pouring angle or pouring height, which limited the ability to fully assess the model's robustness in real application. It was during this case study that the possibility of adapting a computer vision model for measuring real-world process parameters was first considered.

In Case Study 2, there was no evaluation data in terms of exact splatter severity levels since this effect had never been quantified before, however two operator judgements were used to evaluate the effectiveness of the model using metrics such as mean average error (MAE), median and interquartile range. Therefore, the real-world evaluation of Case Study 2 was significantly more rigorous than in Case Study 1.

Whilst some characteristics measured in Case Study 3 did not have real-world measurements, data was available for characteristics such as plume-tuyère contact width and jetting length. This highlighted the importance of reliable real-world measurements, as well as the need for novel approaches in generating them to accommodate the development of novel deep learning-based monitoring techniques.

7.3 Results Comparison

This section compares the results of each case study in terms of the performances achieved by each model, the robustness of each model to changes in environmental conditions, the potential scalability of each model, and finally, the limitations experienced.

7.3.1 Model Performance

Table 42 shows the precision, recall, COCO mAP and speed of the main models developed as part of this project. As shown, the precision, recall and mAP of the splatter model is the highest and virtually perfect, followed by plume model performance which is very good, followed by the ladle model performance which is good considering the scenario. Note that the Kalman filter ladle model in Case Study 1 was not used for comparison since it reduced the suitability of the model for real-world application, despite improving CV metrics. The splatter study had the easiest object detection task with the least variation, as well as the most training data. The plume study involved a fairly difficult object detection task with a lot of variation and had slightly less training data. The ladle study involved a very challenging object segmentation task with a complex shape in a harsh environment, and had the least available training data by a significant amount, near 10% of the available data for the splatter study. This discussion suggests that task complexity and available training data were the two primary factors that affected model performance.

In terms of speed, the splatter was the fastest, followed by the plume model and then the ladle model. The splatter model was optimised with TensorRT and the speed of 5.73fps includes network overhead. Without it, the model achieved over 15.23fps with large images, meaning use of image resizing could significantly improve this speed (see Section 5.4 of Chapter 5). For the use-case of the splatter model, the 5.73fps speed was sufficient for real-time monitoring on the NVIDIA Jetson Orin Nano and there were clear opportunities to speed this up further, such as resizing images from large to small. This study addressed the research gap identified in Chapter 2 regarding the lack of real-time edge-based CV applications developed for steel production processes.

For the plume model, real-time was not an aim, however for future developments 7.6fps is a good start, especially considering it utilised PyTorch and was therefore not optimised using TensorRT. Regarding the ladle model, it is far from being applicable in real-time. Even if the natural frame rate of the video stream is diluted, 2.17fps is not sufficient. However, this was developed using Mask R-CNN which was state-of-the-art when the study initiated, and has now been surpassed by the likes of YOLOv8-seg (see Section 2.5.1 and Section 2.5.6 of Chapter 2). If a closed-loop control system were to be built using the ladle model, it would certainly need to be redesigned using a network such as YOLOv8-seg.

Table 42: Comparison of model performances across different case studies

Model	Precision	Recall	mAP_{COCO}	Inference Speed (fps)
Ladle (RTX)	0.477	0.516	0.516	2.170
Splatter (Jetson)	1.000	1.000	0.994	15.230
Splatter (Jetson - Over network)	1.000	1.000	0.994	5.730
Plume (T4)	0.854	0.901	0.774	7.600

7.3.2 Robustness

Robustness of models was a key discussion point throughout this work and is included here to highlight the environmental challenges involved in each case study. All case studies involve the development of models that can measure real-world process variables and are robust to different environmental conditions. Therefore, they all address the research gap identified in Chapter 2 regarding the lack of real-world measurement, testing and validation surrounding CV applications.

Case Study 1 dealt with harsh lighting conditions and an obstacle blocking the view of the ladle. Through application of CLAHE, the negative effect of the harsh lighting conditions was significantly reduced, proving the effectiveness of this approach and suggesting potential applicability in other similar scenarios. Through the labelling approach described in Section 4.2.1 of Chapter 4, the occlusion was mitigated and this could also be beneficial in future scenarios.

Case Study 2 dealt with variations in camera position and air knife movement. Through application of YOLOv5, robustness to camera position variations was improved significantly, however the results and discussion showed that for optimal model performance, a fixed camera position should be used every time. Also through use of YOLOv5, the air knife movement aspect was managed with minimal issues, however there was a tendency for the model to struggle distinguishing between front knife faces and underside knife faces, but this did not impact deployment performance.

Due to the experimental nature of the data provided in Case Study 3, there were minimal variations in the environment that required extra attention. One example was that the tuyère tended to vibrate, which caused segmentation mask noise that was addressed by using Hough line transforms to remove the segmented edge during these occurrences.

7.3.3 Scalability

When considering industrial application of models, scalability becomes important for real-world deployment and especially for the future advancement of computer vision in steel production. This refers to the ability of a model to be replicated across various devices, potentially operating simultaneously on different operating systems and in diverse scenarios. Also whilst doing this, good scalability means model performance and efficiency will not be sacrificed when it is applied in different situations. During Case study 2 and Case Study 3 it was ensured that models developed can be utilised on different devices by different users, which addressed the research gap identified in Chapter 2 regarding the lack of real-world measurement, testing and validation surrounding CV applications.

For Case Study 1, scalability was limited. The model is capable of processing multiple videos, however can not be considered ready for real-world deployment until it is trained and tested on much larger datasets.

For Case Study 2, scalability is promising. The model has successfully been developed on a personal computer and deployed on a NVIDIA Jetson Orin Nano using FastAPI, Uvicorn and Docker. FastAPI allows operators to easily interface with the model without having coding experience, Uvicorn provides a server for the API (application programming interface) to connect to the web, and Docker ensures the model is lightweight, portable and functional on any operating system. Uvicorn and FastAPI both support asynchronous operations which enhances efficiency, enables multiple simultaneous request handling, and ensures scalability.

For Case Study 3, the scalability is also promising. Using Google Colaboratory, researchers with minimal to no coding experience and no local software or libraries can easily access and operate the model simply by uploading plume footage, typically waiting a few minutes and then downloading the predictions. This could easily be shared and duplicated to run many tests at once.

7.3.4 Limitations

7.3.4.1 Case Study 1

Case Study 1 experienced more limitations than the other two studies for various reasons. Firstly, since it was the first study of the whole project, experience at tackling CV tasks was at a minimum. In contrast, Case Study 3 benefited from improved familiarity with CV, leading to more efficient and effective completion. Furthermore, due to the emergent nature of CV, learning resources, literature, existing networks and community contributions were limited, which made progressing more difficult. This ties in with other limitations such as data availability, time consumption of labelling data and performance of state-of-the-art networks.

Since Mask R-CNN was the highest performance, applicable, instance segmentation model at the time of initiation, this was the best choice. However, even on a much more powerful GPU than the RTX 2070 Super (which not many existed that were better than this at the time), the model would not be able to reach near real-time performance. For this reason, and for the high time consumption of labelling, pouring videos were simplified down to 1fps.

Also, there were only four pouring videos available, and in total this gave less than

600 frames for the entire dataset. Considering the harsh environmental conditions in the footage, as well as the complexity of instance segmentation, as well as the complex shape of the ladle, this was not enough data to develop a high-performance model. Despite this, the performance achieved at the end of the study achieved through methods such as the hyperparameter optimisation, was impressive considering the high complexity and minimal data availability during the task. This is discussed more in Section 4.4 of Chapter 4.

7.3.4.2 Case Study 2

Case Study 2 limitations were somewhat similar to Case Study 1 but with some variation. In terms of experience, a large amount had been gained from the first case study which meant the rate of progress improved significantly and there were less technical pitfalls when dealing with things like debugging coding issues. Also, compared to Mask R-CNN, YOLOv5 was less complex with more community contributions for help with issues. However, there were still challenges such as data access and scoping the problem for real industrial benefit.

In terms of data access, it was necessary to communicate with operators at the galvanising site to request different kinds of footage that could be used for model development. For example, a range of splatter severities were required to ensure the model could handle the whole range of magnitudes, footage of air knife movement was required to ensure the object detection worked, and footage from various camera positions were required so the model could be built to be robust to this. COVID-19 restrictions making the site inaccessible were active during much of this period, which made data acquisition difficult. Therefore, unlike the other case studies where all available data was provided from the beginning, in Case Study 2 there was an element of uncertainty in terms of what data the model would need to be developed for.

In terms of scoping the problem, Case Study 1 industrial aims were provided gradually throughout the study by a collaborating researcher which complicated the process and will be avoided in future, whereas Case Study 2 aims were specified at the beginning of the study which was crucial for the success of it. When developing computer vision systems for stakeholders that have no computer vision experience, it is necessary to bridge the gap in understanding the capabilities and limitations, as well as identify exactly how specific stakeholder objectives are achievable using different techniques. For example, technologists at the galvanising site requested a tool that could quantify the severity of splatter in real-time and this was initially achieved on one video using background subtraction. However, it later became known that the camera position was changed by operators between each shift and the air knives actually moved up and down during the process. Therefore, the model had to be further developed. This highlighted the importance of gaining a deep understanding of the problem early on.

7.3.4.3 Case Study 3

Case study 3 limitations included the large time consumption of labelling thousands of complex plumes, but was overcome through the auto-labelling approach. Raw data availability was of no issue since it was all readily available and easily transferred by a collaborating researcher. This contrasts from other case studies where data was either unavailable or difficult to access. However, there were limitations surrounding the complexity of the task.

For example, it has been mentioned in Section 6.3.1 of Chapter 6 that plumes were difficult to capture individually with 2D bounding boxes due to the complex, gaseous nature of them. This was unique compared to the other case studies which involved objects with more consistent shapes. This not only made labelling difficult, but potentially affected the quality of results. This is not to say the results were poor, because they were not, but using 3D data capturing methods would greatly benefit this kind of task.

There were only two other notable limitations of Case Study 3 which were firstly, the movement of the tuyère. This was mitigated competently using the Hough line removal, however without movement this would have meant less processing for the model and in some frames, possibly an improvement in precision. The other limitation is that the water column had a dark grid-like structure behind the plumes, which if removed, would have made the setup optimal for feature extraction due to the contrast between plumes and the background. This aspect, along with the moving camera position in Case Study 2, highlighted the importance of optimising the environment and camera setup for computer vision systems to maximise results.

7.4 Conclusion

The comparative analysis conducted in this chapter has presented the application of CV technologies across three steel production processes. Through methodological analysis, valuable insights have been drawn such as the versatility of CV models when monitoring various aspects of different industrial environments, the crucial requirement for sufficient quantity and quality of data for achieving desired model performances, and the importance of selecting the right tools and evaluation metrics to facilitate efficient and effective development.

Key insights have also been drawn from results analysis, which include the importance of choosing the right device for development and deployment depending on what the use-case is for a given model, the subtle details to environmental changes that models need to account for in real-world applications in order to be robust, as well as the aspects of development that need to be considered to ensure models can be integrated at scale.

Furthermore, this chapter has uncovered some of the common challenges and limitations of CV that are important to consider when developing CV technology for industry. These contribute to current understanding of CV application to industrial processes, whilst also paving the way for better practices in future projects.

As discussed in Chapters 4, 5 and 6, this comparative analysis has highlighted how the three case studies have addressed several significant gaps identified in the literature in Chapter 2. These include enabling real-time processing capabilities in edge-based CV systems, generating diverse datasets that represent the complexity of real industrial environments, conducting real-world application, testing and validation of CV systems, as well as synthesising traditional techniques with more modern, deep learning-based techniques. By addressing these gaps, this research advances the field of computer vision with regards to application in the steel industry.

Chapter 8: Conclusion

This thesis has demonstrated the capabilities of machine learning-based computer vision (CV) for advancing the current state of manufacturing technology. By tackling three different case studies based on three entirely different processes within steelmaking, this work has demonstrated how versatile CV is as a tool, and how it can be used to monitor highly complex and variable environments in ways that were previously unattainable. Additionally, the impact of integrating CV into steel production has been critically evaluated in terms of enhancing operational efficiency and product quality, whilst reducing equipment degradation and ensuring safer working conditions. The primary research objectives of this body of work are listed below.

8.1 Research Objectives

1. *Conduct a comprehensive literature review of modern advances in computer vision techniques with emphasis on industrial and manufacturing scenarios. This includes establishing a clear understanding of the state-of-the-art, critically analysing existing literature, and identifying research gaps.*

In terms of the first objective, Chapter 2 presents a thorough literature review on modern computer vision techniques and their applications to industrial scenarios, with particular emphasis on manufacturing scenarios and primary focus on steel processing. Through this, the state-of-the-art has been clearly established, critically analysed, and used to identify research gaps which have either been addressed in this work or recommended for future endeavours. The gaps addressed include the need for more real-time edge-based CV systems on production lines, the need for more steel production datasets that are based on a range of different processes, the lack of innovation in the field with regards to combining traditional and modern CV techniques, and the lack of application of CV for making real-world measurements and validating them.

Collectively, addressing these gaps has significantly contributed to the field. Remaining gaps such as establishing standardised evaluation approaches for fairer comparison of model performances in industrial environments, analysing cost-effectiveness of CV implementation and accounting for human factors, are left open for future endeavours.

2. *Design novel computer vision systems tailored for different processes within steel production (hot metal ladle pouring, galvanisation, and gas stirring) to monitor and analyse key process variables that give insights for enhancing operational efficiency and product quality.*

In terms of the second objective, all case studies advance CV application in steel manufacturing through process monitoring and analysis of process variables for insights into improving the efficiency and effectiveness of production methods.

In Chapter 4, ladle process parameters and resulting emission severity were, for the first time, monitored using CV to give insights that improve operational efficiency. This sets a new standard for process monitoring within the steel industry. The novel dataset developed addressed the gap identified in Chapter 2 related to the lack of steel production process datasets, the combination of Mask R-CNN and Kalman filtering addressed the gap related to the lack of innovation involving traditional and DL-based CV techniques in combination, and the measurement of pouring height, rotation angle, and flame severity was pioneering and addressed the gap related to the measurement of real-world process variables using CV.

In Chapter 5, splatter severity, which has never been quantified before, was monitored in real-time on an edge device to enable operators to draw insights that improve both operational efficiency and product quality. This has demonstrated the capability of computer vision to monitor new process variables that were previously not measured. The novel dataset developed addressed the gap identified in Chapter 2 related to the lack of steel production process datasets, the combination of YOLOv5 with background subtraction addressed the gap related to the lack of innovation involving traditional and DL-based CV techniques in combination, and the measurement and validation of zinc splatter severity was revolutionary to the process whilst addressing the gap related to the measurement and validation of real-world process variables using CV.

In Chapter 6, refractory wear was monitored by analysing plumes in an innovative way to give novel insights that improve operational efficiency and product quality. This has opened up many new possibilities in terms of optimising the gas stirring process. The novel dataset developed addressed the gap identified in Chapter 2 related to the lack of steel production process datasets, the combination of YOLOv5 with background subtraction addressed the gap related to the lack of innovation involving traditional and DL-based CV techniques in combination, and the measurement of plume characteristics and validation of contact width and jetting length was groundbreaking for optimising gas stirring whilst addressing the gap of measurement and validation of real-world process variables using CV.

3. *Investigate the deployment of computer vision technologies into steel production environments as both post-processing and real-time monitoring applications. This includes the consideration of robustness and scalability of systems to ensure they can withstand the complexities and variations of deploying to real industrial settings.*

In terms of the third objective, each study explores practical deployment of computer vision technologies, whether this is for real-time monitoring or post-processing analysis.

In Chapter 4, the model developed is robust to poor lighting conditions and can be used as a post-processing tool. However, state-of-the-art instance segmentation has advanced since the initiation of the study and with a larger training set, as well as application of a more recently released segmentation network, it could be deployed in real-time and potentially used for closed-loop control of ladle motion.

In Chapter 5, the model developed was deployed onto an edge device, operates in real-time, and is ready for integration into a real-world application. It is robust to changes in camera position, changes in air knife position, dust particles and heat distortion. It is also deployed within a Docker container and accessed through an application programming interface (API), both of which ensure scalability. This addresses the gap identified in Chapter 2 related to the lack of real-time edge-based CV systems on production lines.

In Chapter 6, the model produced is available as a post-processing tool and has been used by another researcher for this purpose. It is robust to tuyère vibrations and can easily be distributed as a Google Colaboratory notebook, demonstrating scalability. With collaboration from gas stirring process experts, it is possible to further develop the model for real-time application.

In Chapter 7, a comprehensive comparative analysis of the case studies emphasised the importance of data quality and quantity, the selection of appropriate models and evaluation metrics, and tools that were crucial for both development and deployment. Additionally, it discussed the practicality, robustness and scalability of the developed models, whilst also detailing the limitations with regards to development and deployment.

Overall, deployment has been a key topic investigated in every case study. This is particularly true in Chapter 5, where the technology was taken from theoretical performance evaluation to deployment, and Chapter 6, where it was passed over to another researcher for use. These are significant contributions to both CV and manufacturing.

8.2 Key Findings & Implications

The findings of each case study are highly beneficial for future research on CV applications for process monitoring. Chapter 4 laid the foundation for future advancements in monitoring ladle pouring by offering a novel approach that can be developed to reduce emissions and equipment degradation whilst improving safety.

Chapter 5 demonstrated the practical application of a CV model to quantify zinc splatter severity occurring on the galvanising line, using a deployed edge-based system with Wi-Fi and real-time capabilities, ready for implementation. The system built in this study can be implemented to reduce equipment degradation and the occurrence of defects, whilst improving operational efficiency.

Chapter 6 provided a tool for easily and automatically comparing the wear rate and stirring efficiency of different gas stirring configurations, which contributes to the reduction of equipment degradation and improvement of operational efficiency and product quality.

The implications of this work are vast. Firstly, this research contributes to progressing the integration of CV technology into industrial environments and is evidence of the benefits that can be gained by doing so. Applying CV to three entirely different areas of steel production emphasises the adaptability and widespread potential of CV technology across many industrial processes, which greatly contributes to the field of manufacturing technology, whilst also contributing to the field of CV. Secondly, the successful deployment of one model, and the successful handover of another, demonstrate the feasibility of utilising CV models in real-world steel production environments for tangible benefits. This challenges existing perspectives on the limitations of CV technology in complex industrial environments, whilst laying the foundation for future developments across the entire manufacturing sector. Lastly, the limitations found in this work are valuable to future projects in both the academic space and the industrial space. Addressing these challenges has provided guidance for developing and implementing CV solutions in a wide variety of scenarios.

8.3 Limitations

Key limitations discovered firstly include the complexities of dealing with harsh industrial environments that make high model performance and model deployment particularly challenging to achieve. These environments introduced significant sources of noise and variability such as poor lighting, heat distortion and vibrations, which impacted model performance and required extensive data pre-processing and robust model design to overcome.

Also, there were limitations surrounding problem scoping and data acquisition which highlighted the challenges in communicating with individuals from different professional backgrounds and obtaining datasets of a sufficient volume, diversity and quality from colleagues for satisfactory model performance upon training. Problem scoping was challenging due to the general lack of familiarity with computer vision within the steel sector, whilst data acquisition for real industrial environments was particularly difficult due to their variable nature.

Furthermore, labelling thousands of frames, often containing complex shapes such as plumes and ladles, was highly time consuming. This highlighted the need for more resource-efficient approaches such as the auto-labelling approach demonstrated in Chapter 6.

Additionally, the necessity for efficient approaches to hyperparameter optimisations became evident throughout Chapter 4. Computer vision models often have many different hyperparameters and one training session (and therefore one experiment), can take a considerable amount of time to complete. Therefore, in this field efficient experimental design is crucial in order to develop high-performing models within computational power and time constraints.

Finally, deployment of a real-time edge-based system in Chapter 5 highlighted various limitations such as requiring maximum efficiency for real-time performance on limited computational power, meaning only essential code can be processed. For example, once the model has been validated as reliable, removing the visual display of measurements may be necessary depending on project demands.

These limitations highlight areas that should be carefully considered in future work to avoid some of the pitfalls and bottlenecks experienced during this project. Through awareness of these challenges, which are all common in industrial CV projects, future developments will progress more efficiently and more effectively. Therefore, these limitations are valuable to future researchers.

8.4 Future Research Directions

Due to the emergent nature of computer vision, there are a vast number of potential future research directions which not only involve further development of the case studies in this project, but also diversifying models to be used in other steel production processes (or even in other industries). This section will firstly focus specifically on ways of addressing the identified limitations, and secondly how work here can generally be developed further. In order to address the limitations, the following improvements can be made:

- In all future research projects, establishing a thorough understanding of the problem statement, relevant environmental conditions and data availability will be imperative. This will ensure that the full problem is understood by all parties involved from project initiation, and that every stage of the project is conducted with this in mind.
- If extensive data labelling is required, efficient methods such as auto-labelling (or transfer learning) should be used to minimise time consumption. This can significantly reduce project duration and significantly increase the volume of labelled data (provided there is enough data originally acquired). The performance of models in all studies could be improved significantly with larger datasets.
- Hyperparameter optimisation should be tackled with more advanced approaches that reduce computational overhead and shorten the overall time to complete a series of experiments. An example of this could be early stopping, which stops training when performance plateaus.
- Model deployment for real-time application should be approached with more focus on enhancing computational efficiency through efficient use of data types, data structures and memory allocation, as well as using a minimalistic design of model features, and maximising the use of optimisation techniques such as quantisation and knowledge distillation.
- To address remaining gaps in the literature, an evaluation methodology could be established that is suitable for all case studies, to work towards more comparable outcomes across the entire field. Furthermore, cost-effectiveness and human factors could be investigated to better understand the economic impacts and user interactions with these technologies, as well as promote wider acceptance and utilisation within the industry.

There are also several potential research avenues that could be pursued to further develop each case study:

- For Chapter 4, the ladle model could be redeveloped using a model such as YOLOv8-seg to improve both speed and precision. This should be conducted using a larger dataset that includes thousands of labelled frames rather than hundreds, whilst also validating model measurements against reliable real-world measurements. These improvements would enhance insights, and with extensive testing for full physical integration into the pouring site, could result in a closed-loop control system that minimises emissions by adjusting pouring controls in real-time.

- For Chapter 5, the splatter model could be implemented onto the ZODIAC galvanising line. This way it could be used for an alarm system to alert operators of high splatter severity, as well as for data collection and analysis to optimise process parameters for maximum strip speed and minimum splatter.

- For Chapter 6, the plume model is currently usable for comparing different gas stirring configurations, but would benefit from using YOLOv8, and maybe a different column setup more adapted for computer vision application, to further improve performance. If this system was to be integrated directly into the basic-oxygen furnace, it would be revolutionary for the stirring process. Additionally, the plume model itself could be evolved by taking a 3D approach rather than a 2D approach.

References

[1] I. H. Sarker, "AI-Based Modeling: Techniques, Applications and Research Issues Towards Automation, Intelligent and Smart Systems," *SN Comput. Sci.*, vol. 3, no. 2, p. 158, 2022. DOI: https://doi.org/10.1007/s42979-022-01043-x.

[2] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC)*, pp. 128–144, 2020. DOI: https://doi.org/10.1007/978-3-030-17795-9_10.

[3] A. L'Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine Learning With Big Data: Challenges and Approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017. DOI: https://doi.org/10.1109/ACCESS.2017.2696365.

[4] D. G. Hong, W. H. Han, and C. H. Yim, "Convolutional recurrent neural network to determine whether dropping slag dart fills the exit hole during tapping in a basic oxygen furnace," *Metall. Mater. Trans. B.*, vol. 52, pp. 3833–3845, 2021. DOI: https://doi.org/10.1007/s11663-021-02299-z.

[5] P. Damacharla, A. R. MV, J. Ringenberg, and A. Y. Javaid, "TLU-Net: A Deep Learning Approach for Automatic Steel Surface Defect Detection," in *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, pp. 1–6, 2021. DOI: https://doi.org/10.1109/ICAPAI49758.2021.9462060.

[6] A. Bansal, X. Chen, B. Russell, A. Gupta, and D. Ramanan, "Pixelnet: Representation of the pixels, by the pixels, and for the pixels," *arXiv*, 2017. DOI: https://doi.org/10.48550/arXiv.1702.06506.

[7] C. O'Donovan, I. Popov, G. Todeschini, and C. Giannetti, "Ladle pouring process parameter and quality estimation using Mask R-CNN and contrast-limited adaptive histogram equalisation," *Int. J. Adv. Manuf. Technol.*, vol. 126, pp. 1397–1416, 2023. DOI: https://doi.org/10.1007/s00170-023-11151-4.

[8] C. O'Donovan, C. Giannetti, and C. Pleydell-Pearce, "Real-Time Monitoring of Molten Zinc Splatter using Machine Learning-Based Computer Vision," *J. Intell. Manuf.*, pp. 1–27, 2024. DOI: https://doi.org/10.1007/s10845-024-02418-y.

[9] C. O'Donovan, M. Kamran, C. Giannetti, and C. Pleydell-Pearce, "Monitoring of Gaseous Plumes in Basic Oxygen Furnace Gas Stirring Simulation using Machine Learning-Based Computer Vision," 2024. Under review.

[10] C. O'Donovan, C. Giannetti, and C. Pleydell-Pearce, "Revolutionising the Sustainability of Steel Manufacturing Using Computer Vision," in *2024 International Conference on Industry 4.0 and Smart Manufacturing (ISM)*, pp. 1729–1738, 2024. DOI: https://doi.org/10.1016/j.procs.2024.01.171.

[11] C. O'Donovan, C. Giannetti, and C. Pleydell-Pearce, "Enabling Sustainable Steel Production with Computer Vision," in *2023 International Conference on Manufacturing Research (ICMR)*, pp. 37–42, 2023. DOI: https://doi.org/10.3233/ATDE230897.

[12] PASCAL, "The PASCAL Visual Object Classes Homepage." http://host.robots.ox.ac.uk/pascal/VOC/, 2012. Accessed: 2024 Mar 12.

[13] Meta AI, "MS COCO (Microsoft Common Objects in Context)." https://paperswithcode.com/dataset/coco, n.d. Accessed: 2024 Apr 24.

[14] Microsoft, "detection-2016.htm." https://github.com/cocodataset/cocodataset.github.io/blob/master/dataset/detection-2016.htm, 2018. Accessed: 2024 Apr 24.

[15] Roboflow, "Roboflow 100: A Multi-Domain Object Detection Benchmark." https://blog.roboflow.com/roboflow-100/, 2024. Accessed: 2024 Apr 24.

[16] M. Kaur, J. Kaur, and J. Kaur, "Survey of Contrast Enhancement Techniques based on Histogram Equalization," *Int J Adv Comput Sci Appl.*, vol. 2, no. 7, pp. 137–141, 2011. DOI: https://doi.org/10.14569/IJACSA.2011.020721.

[17] Towards Data Science, "Histogram equalization." https://towardsdatascience.com/histogram-equalization-5d1013626e64, 2017. Accessed: 2022 Feb 6.

[18] The Mathworks, Inc, "Adjust Image Contrast Using Histogram Equalization." https://uk.mathworks.com/help/images/histogram-equalization.html, 2024. Accessed: 2024 Nov 7.

[19] M. Tiwari and B. Gupta, "Brightness preserving contrast enhancement of medical images using adaptive gamma correction and homomorphic filtering," in *2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, pp. 1–4, 2016. DOI: https://doi.org/10.1109/SCEECS.2016.7509287.

[20] C. F. J. Kuo and H. C. Wu, “Gaussian probability bi-histogram equalization for enhancement of the pathological features in medical images,” *Int. J. Imaging Syst. Technol.*, vol. 29, pp. 132–145, 2019. DOI: https://doi.org/10.1002/ima.22307.

[21] A. Horé and D. Ziou, “Image Quality Metrics: PSNR vs. SSIM,” in *2010 20th International Conference on Pattern Recognition*, pp. 2366–2369, 2010. DOI: https://doi.org/10.1109/ICPR.2010.579.

[22] V. Panse and R. Gupta, “Medical Image Enhancement with Brightness Preserving Based on Local Contrast Stretching and Global Dynamic Histogram Equalization,” in *2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 164–170, 2021. DOI: https://doi.org/10.1109/CSNT51715.2021.9509670.

[23] Y. T. Kim, “Contrast enhancement using brightness preserving bi-histogram equalization,” *IEEE Trans. Consum. Electron.*, vol. 43, pp. 1–8, 1997. DOI: https://doi.org/10.1109/30.580378.

[24] A. Sarrafzadeh, F. Rezazadeh, and J. Shanbehzadeh, “Brightness Preserving Fuzzy Dynamic Histogram Equalization,” in *International MultiConference of Engineers and Computer Scientists (IMECS)*, pp. 467–471, 2013. https://www.iaeng.org/publication/IMECS2013/IMECS2013_pp467-471.pdf.

[25] L. A. Zadeh, “Fuzzy logic,” *Computer*, vol. 21, no. 4, pp. 83–93, 1988. DOI: https://doi.org/10.1109/2.53.

[26] B. Kaur, “BPDFHE based Hybrid Pre-Processing Methodology of Leaf Images for Efficient Disease Detection,” in *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*, pp. 1704–1708, 2021. DOI: https://doi.org/10.1109/ICOSEC51865.2021.9591766.

[27] M. F. Khan, E. Khan, M. M. Nofal, and M. Mursaleen, “Fuzzy Mapped Histogram Equalization Method for Contrast Enhancement of Remotely Sensed Images,” *IEEE Access*, vol. 8, pp. 112454–112461, 2020. DOI: https://doi.org/10.1109/ACCESS.2020.3001658.

[28] OpenCV developers, “Morphological Operations.” https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html, 2023. Accessed: 2024 Mar 12.

[29] The Mathworks, Inc, "Remove Thin Lines Using Erosion." https://uk.mathworks.com/help/images/erode-an-image.html, 2024. Accessed: 2024 Nov 7.

[30] The Mathworks, Inc, "Dilate an Image to Enlarge a Shape." https://uk.mathworks.com/help/images/dilate-an-image.html, 2024. Accessed: 2024 Nov 7.

[31] D. Lee, Y. Kang, C. Park, and S. Won, "Defect Detection Algorithm in Steel Billets Using Morphological Top-Hat filter," in *IFAC Proceedings Volumes*, pp. 209–212, 2009. DOI: https://doi.org/10.3182/20091014-3-CL-4011.00038.

[32] J. P. Yun, Y. Park, B. Seo, S. W. Kim, S. H. Choi, C. H. Park, H. M. Bae, and H. W. Hwang, "Development of Real-time Defect Detection Algorithm for High-speed Steel Bar in Coil (BIC)," in *2006 SICE-ICASE International Joint Conference*, pp. 2495–2498, 2006. DOI: https://doi.org/10.1109/SICE.2006.314680.

[33] M. Heydari, R. Amirfattahi, B. Nazari, and P. Rahimi, "An industrial image processing-based approach for estimation of iron ore green pellet size distribution," *Powder Technology*, vol. 303, pp. 260–268, 2016. DOI: https://doi.org/10.1016/j.powtec.2016.09.020.

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014. DOI: https://doi.org/10.1109/CVPR.2014.81.

[35] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, 2010. DOI: https://doi.org/10.1109/TPAMI.2009.167.

[36] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, pp. 154–171, 2013. https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013.

[37] X. Wang, M. Yang, S. Zhu, and Y. Lin, "Regionlets for Generic Object Detection," in *2013 IEEE International Conference on Computer Vision*, pp. 17–24, 2013. DOI: https://doi.org/10.1109/ICCV.2013.10.

[38] S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun, “Bottom-Up Segmentation for Top-Down Detection,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3294–3301, 2013. DOI: https://doi.org/10.1109/CVPR.2013.423.

[39] R. Girshick, “Fast R-CNN,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015. DOI: https://doi.org/10.1109/ICCV.2015.169.

[40] J. Deng, W. Dong, R. Socher, L. J. Li, L. Kai, and F. F. Li, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. DOI: https://doi.org/10.1109/CVPR.2009.5206848.

[41] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional Architecture for Fast Feature Embedding,” in *Proceedings of the 22nd ACM International Conference on Multimedia*, pp. 675–678, 2014. DOI: https://doi.org/10.1145/2647868.2654889.

[42] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, “Return of the Devil in the Details: Delving Deep into Convolutional Nets.” https://arxiv.org/abs/1405.3531, 2014. Accessed: 2024 Mar 12.

[43] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” in *International Conference on Learning Representations*, pp. 1–14, 2015. DOI: https://doi.org/10.48550/arXiv.1409.1556.

[44] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” in *Computer Vision - ECCV 2014*, pp. 346–361, 2014. DOI: https://doi.org/10.1007/978-3-319-10578-9.

[45] T. Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” in *Computer Vision - ECCV 2014*, pp. 740–755, 2014. DOI: https://doi.org/10.1007/978-3-319-10602-1_48.

[46] M. Lin, Q. Chen, and S. Yan, “Network in network,” *arXiv*, 2013. DOI: https://doi.org/10.48550/arXiv.1312.4400.

[47] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler, “segDeepM: Exploiting segmentation and context in deep neural networks for object detection,” in *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, pp. 4703–4711, 2015. DOI: https://doi.org/10.48550/arXiv.1502.04275.

[48] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, pp. 1–9, 2015. https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[49] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks," in *Computer Vision - ECCV 2014*, pp. 818–833, 2014. DOI: https://doi.org/10.1007/978-3-319-10590-1_53.

[50] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016. DOI: https://doi.org/10.1109/CVPR.2016.91.

[51] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv*, vol. abs/1612.08242, 2016. DOI: https://doi.org/10.48550/arXiv.1612.08242.

[52] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Computer Vision - ECCV 2016*, pp. 21–37, 2016. DOI: https://doi.org/10.1007/978-3-319-46448-0_2.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. DOI: https://doi.org/10.1109/CVPR.2016.90.

[54] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018. DOI: https://doi.org/10.48550/arXiv.1804.02767.

[55] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017. DOI: https://doi.org/10.1109/ICCV.2017.324.

[56] W. Wu, L. Guo, H. Gao, Z. You, Y. Liu, and Z. Chen, "YOLO-SLAM: A semantic SLAM system towards dynamic environment with geometric constraint," *Neural Comput. Appl.*, vol. 34, no. 8, pp. 6011–6026, 2022. DOI: https://doi.org/10.1007/s00521-021-06764-3.

[57] J. Hui, "Understanding Feature Pyramid Networks for object detection (FPN)." https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c, 2018. Accessed: 2024 Apr 24.

[58] J. Terven, D. M. Córdova-Esparza, and J. A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Mach. Learn. Knowl. Extr.*, vol. 5, no. 4, pp. 1680–1716, 2023. DOI: https://doi.org/10.3390/make5040083.

[59] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8759–8768, 2018. DOI: https://doi.org/10.1109/CVPR.2018.00913.

[60] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7464–7475, 2023. DOI: https://doi.org/10.1109/CVPR52729.2023.00721.

[61] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8." https://github.com/ultralytics/ultralytics, 2023. Accessed: 2024 Mar 12.

[62] Deci, "YOLO-NAS by Deci Achieves SOTA Performance on Object Detection Using Neural Architecture Search." https://deci.ai/blog/yolo-nas-object-detection-foundation-model/, 2023. Accessed: 2024 Mar 12.

[63] G. Jocher, "Ultralytics YOLOv5." https://github.com/ultralytics/yolov5, 2020. Accessed: 2024 Mar 12.

[64] Y. Lee, J. An, and I. Joe, "Deep-Learning-Based Object Filtering According to Altitude for Improvement of Obstacle Recognition during Autonomous Flight," *Remote Sens.*, vol. 14, no. 6, p. 1378, 2022. DOI: https://doi.org/10.3390/rs14061378.

[65] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781–10790, 2020. DOI: https://doi.org/10.1109/CVPR42600.2020.01079.

[66] RangeKing, "Brief summary of YOLOv8 model structure." https://github.com/ultralytics/ultralytics/issues/189, 2023. Accessed: 2024 Apr 24.

[67] Deci, "The Ultimate Guide to Deep Learning Model Quantization and Quantization-Aware Training." https://deci.ai/quantization-and-quantization-aware-training, 2023. Accessed: 2024 Mar 12.

[68] G. Welch and G. Bishop, "An Introduction to the Kalman Filter." https://perso.crans.org/club-krobot/doc/kalman.pdf, 2001. Accessed: 2024 Mar 12.

[69] S. H. P. Won, F. Golnaraghi, and W. W. Melek, "A Fastening Tool Tracking System Using an IMU and a Position Sensor With Kalman Filters and a Fuzzy Expert System," *IEEE Trans. Ind. Electron.*, vol. 56, no. 5, pp. 1782–1792, 2009. DOI: https://doi.org/10.1109/TIE.2008.2010166.

[70] X. D. Gao and S. J. Na, "Detection of weld position and seam tracking based on Kalman filtering of weld pool images," *J. Manuf. Syst.*, vol. 24, no. 1, pp. 1–12, 2005. DOI: https://doi.org/10.1016/S0278-6125(06)00002-1.

[71] A. N. F., D. Ulutan, and L. Mears, "In-process Tool Flank Wear Estimation in Machining Gamma-prime Strengthened Alloys Using Kalman Filter," *Procedia Manuf.*, vol. 1, pp. 696–707, 2015. DOI: https://doi.org/10.1016/j.promfg.2015.09.018.

[72] N. F. Akhavan, M. Michel, and L. Mears, "State of health monitoring in machining: Extended Kalman filter for tool wear assessment in turning of IN718 hard-to-machine alloy," *J. Manuf. Process.*, vol. 24, pp. 361–369, 2016. DOI: https://doi.org/10.1016/j.jmapro.2016.06.015.

[73] R. Soman, K. Majewska, M. Mieloszyk, P. Malinowski, and W. Ostachowicz, "Application of Kalman Filter based Neutral Axis tracking for damage detection in composites structures," *Compos. Struct.*, vol. 184, pp. 66–77, 2018. DOI: https://doi.org/10.1016/j.compstruct.2017.09.092.

[74] F. Cao, P. D. Docherty, S. Ni, and X. Chen, "Contact force and torque sensing for serial manipulator based on an adaptive Kalman filter with variable time period," *Robot. Comput.-Integr. Manuf.*, vol. 72, p. 102210, 2021. DOI: https://doi.org/10.1016/j.rcim.2021.102210.

[75] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Adv. Neural. Inf. Process. Syst.*, vol. 25, 2012. DOI: https://doi.org/10.1145/3065386.

[76] M. H. Rahimi, H. N. Huynh, and Y. Altintas, "On-line chatter detection in milling with hybrid machine learning and physics-based model," *CIRP J. Manuf. Sci. Technol.*, vol. 35, pp. 25–40, 2021. DOI: https://doi.org/10.1016/j.cirpj.2021.05.006.

[77] K. Xu, G. Zheng, and F. Zhang, "Tracking Steel Coils Using CV Algorithm Based on Deep Learning and Kalman Filter," in *2021 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pp. 1–6, 2021. DOI: https://doi.org/10.1109/ICCAIS52680.2021.9624513.

[78] S. Purushothaman, "Tool wear monitoring using artificial neural network based on extended Kalman filter weight updation with transformed input patterns," *J. Intell. Manuf.*, vol. 21, no. 6, pp. 717–730, 2010. DOI: https://doi.org/10.1007/s10845-009-0249-y.

[79] H. Guo, Y. Li, C. Liu, Y. Ni, and K. Tang, "A Deformation Force Monitoring Method for Aero-Engine Casing Machining Based on Deep Autoregressive Network and Kalman Filter," *Appl. Sci.*, vol. 12, no. 14, p. 7014, 2022. DOI: https://doi.org/10.3390/app12147014.

[80] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network," *Phys. D: Nonlinear Phenom.*, vol. 404, p. 132306, 2020. DOI: https://doi.org/10.1016/j.physd.2019.132306.

[81] A. A. Rahman, "LSTM Networks." https://www.linkedin.com/pulse/lstm-networks-abdullah-al-rahman/, 2023. Accessed: 2024 Apr 24.

[82] Z. Shi, A. A. Mamun, C. Kan, W. Tian, and C. Liu, "An LSTM-autoencoder based online side channel monitoring approach for cyber-physical attack detection in additive manufacturing," *J. Intell. Manuf.*, pp. 1–17, 2022. DOI: https://doi.org/10.1007/s10845-021-01879-9.

[83] J. Zhang, P. Wang, R. Yan, and R. X. Gao, "Long short-term memory for machine remaining life prediction," *J. Manuf. Syst.*, vol. 48, pp. 78–86, 2018. DOI: https://doi.org/10.1016/j.jmsy.2018.05.011.

[84] J. Li, Y. Huang, Q. Li, and Y. Li, "Closed-LSTM neural network based reference modification for trajectory tracking of piezoelectric actuator," *Neurocomputing*, vol. 467, pp. 379–391, 2022. DOI: https://doi.org/10.1016/j.neucom.2021.10.012.

[85] N. Qadeer, J. H. Shah, M. Sharif, M. A. Khan, G. Muhammad, and Y. D. Zhang, "Intelligent Tracking of Mechanically Thrown Objects by Industrial Catching Robot for Automated In-Plant Logistics 4.0," *Sensors*, vol. 22, no. 6, p. 2113, 2022. DOI: https://doi.org/10.3390/s22062113.

[86] H. Wang, S. Fu, B. Peng, N. Wang, and H. Gao, "Equipment Health Condition Recognition and Prediction Based on CNN-LSTM Deep Learning," in *International Conference on Maintenance Engineering*, pp. 830–842, 2020. DOI: https://doi.org/10.1007/978-3-030-75793-9_78.

[87] R. Yu, J. Kershaw, P. Wang, and Y. Zhang, "How to Accurately Monitor the Weld Penetration From Dynamic Weld Pool Serial Images Using CNN-LSTM Deep Learning Model?," *IEEE Robot. Autom. Lett.*, vol. 7, 2022. DOI: https://doi.org/10.1109/LRA.2022.3173659.

[88] J. H. Lee, J. Kang, W. Shim, H. S. Chung, and T. E. Sung, "Pattern Detection Model Using a Deep Learning Algorithm for Power Data Analysis in Abnormal Conditions," *Electronics*, vol. 9, no. 7, p. 1140, 2020. DOI: https://doi.org/10.3390/electronics9071140.

[89] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE international conference on image processing (ICIP)*, pp. 3464–3468, 2016. DOI: https://doi.org/10.1109/ICIP.2016.7533003.

[90] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3645–3649, 2017. DOI: https://doi.org/10.1109/ICIP.2017.8296962.

[91] R. Pereira, G. Carvalho, L. Garrote, and U. J. Nunes, "Sort and deep-sort based multi-object tracking for mobile robotics: Evaluation with new data association metrics," *Applied Sciences*, vol. 12, no. 3, pp. 1–18, 2022. DOI: https://doi.org/10.3390/app12031319.

[92] H. M. Ahmad, A. Rahimi, and K. Hayat, "Deep Learning Transforming the Manufacturing Industry: A Case Study," in *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Sys-*

tems & Application (HPCC/DSS/SmartCity/DependSys), pp. 1286–1291, 2021. DOI: https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00196.

[93] T. Xie and X. Yao, "Smart Logistics Warehouse Moving-Object Tracking Based on YOLOv5 and DeepSORT," *Appl. Sci.*, vol. 13, no. 17, p. 9895, 2023. DOI: https://doi.org/10.3390/app13179895.

[94] A. A. Tulbure, C. Covaciu, A. A. Tulbure, C. Safirescu, and E. H. Dulf, "Novel Defect Tracking Using DeepSORT, ScaledYOLOv4 and the Clip Model," in *2022 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pp. 1–6, 2022. DOI: https://doi.org/10.1109/AQTR55203.2022.9801989.

[95] K. Bernardin, A. Elbs, and R. Stiefelhagen, "Multiple Object Tracking Performance Metrics and Evaluation in a Smart Room Environment," in *Computer Vision - ECCV 2016*, 2006. https://api.semanticscholar.org/CorpusID:315852.

[96] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, "A review of semantic segmentation using deep neural networks," *Int. J. Multimed. Inf. Retr.*, vol. 7, pp. 87–93, 2018. DOI: https://doi.org/10.1007/s13735-017-0141-z.

[97] A. M. Hafiz and G. M. Bhat, "A survey on instance segmentation: state of the art," *Int. J. Multimed. Inf. Retr.*, vol. 9, no. 3, pp. 171–189, 2020. DOI: https://doi.org/10.1007/s13735-020-00195-x.

[98] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017. DOI: https://doi.org/10.1109/ICCV.2017.322.

[99] J. Dai, K. He, and J. Sun, "Instance-Aware Semantic Segmentation via Multi-task Network Cascades," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3150–3158, 2016. DOI: https://doi.org/10.1109/CVPR.2016.343.

[100] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2359–2367, 2017. DOI: https://doi.org/10.1109/CVPR.2017.472.

[101] C. M. Brown and D. Terzopoulos, *Real-time computer vision.* Cambridge: Cambridge University Press, 1995.

[102] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Real-time computer vision with OpenCV," *ACM*, vol. 55, no. 6, p. 61–69, 2012. DOI: https://doi.org/10.1145/2184319.2184337.

[103] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-Time Instance Segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9157–9166, 2019. DOI: https://doi.org/10.1109/ICCV.2019.00925.

[104] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT++ Better Real-Time Instance Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 2, p. 1108–1121, 2022. DOI: https://doi.org/10.1109/TPAMI.2020.3014297.

[105] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "SOLO: Segmenting objects by locations," in *Computer Vision - ECCV 2020*, pp. 649–665, 2020. DOI: https://doi.org/10.1007/978-3-030-58523-5_38.

[106] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "Solov2: Dynamic and fast instance segmentation," in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pp. 17721–17732, 2020. https://proceedings.neurips.cc/paper_files/paper/2020/file/cd3afef9b8b89558cd56638c3631868a-Paper.pdf.

[107] G. Jocher, "v7.0 - YOLOv5 SOTA Realtime Instance Segmentation." https://github.com/ultralytics/yolov5/releases/v7.0, 2023. Accessed: 2024 Mar 12.

[108] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8 Instance Segmentation." https://docs.ultralytics.com/tasks/segment/#models, 2023. Accessed: 2024 Mar 12.

[109] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, "Fast Segment Anything." https://arxiv.org/abs/2306.12156, 2023. Accessed: 2024 Mar 12.

[110] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W. Y. Lo, P. Dollár, and G. R, "Segment anything," *arXiv*, 2023. DOI: https://doi.org/10.48550/arXiv.2304.02643.

[111] K. Goyal and J. Singhai, "Review of background subtraction methods using Gaussian mixture model for video surveillance systems," *Artif. Intell. Rev.*, vol. 50, pp. 241–259, 2018. DOI: https://doi.org/10.1007/s10462-017-9542-x.

[112] OpenCV Developers, "How to Use Background Subtraction Methods." https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html, 2022. Accessed: 2024 Apr 26.

[113] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, pp. 28–31, 2004. DOI: https://doi.org/10.1109/ICPR.2004.1333992.

[114] OpenCV developers, "Background Subtraction MOG2." https://docs.opencv.org/3.4/de/df4/tutorial_js_bg_subtraction.html, 2024. Accessed: 2024 Mar 12.

[115] Z. Qu, M. Yu, and J. Liu, "Real-time traffic vehicle tracking based on improved MoG background extraction and motion segmentation," in *3rd International Symposium on Systems and Control in Aeronautics and Astronautics (ISSCAA)*, pp. 676–680, 2010. DOI: https://doi.org/10.1109/ISSCAA.2010.5633717.

[116] J. Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994. DOI: https://doi.org/10.1109/CVPR.1994.323794.

[117] J. Shin, S. Kim, S. Kang, S. W. Lee, J. Paik, B. Abidi, and M. Abidi, "Optical flow-based real-time object tracking using non-prior training active feature model," *Real-Time Imag.*, vol. 11, no. 3, pp. 204–218, 2005. DOI: https://doi.org/10.1016/j.rti.2005.03.006.

[118] P. M. Dhulavvagol, A. Desai, and R. Ganiger, "Vehicle Tracking and Speed Estimation of Moving Vehicles for Traffic Surveillance Applications," in *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, pp. 373–377, 2017. DOI: https://doi.org/10.1109/CTCEEC.2017.8455043.

[119] OpenCV Developers, "cv::bgsegm Namespace Reference." https://docs.opencv.org/4.x/df/d5d/namespacecv_1_1bgsegm.html, 2022. Accessed: 2024 May 15.

[120] T. Hyla and N. Wawrzyniak, "Automatic Ship Detection on Inland Waters: Problems and a Preliminary Solution," in *14th International Conference on Systems (ICONS)*, pp. 56–60, 2019. http://personales.upv.es/thinkmind/dl/conferences/icons/icons_2019/icons_2019_3_30_40033.pdf.

[121] T. Trnovszký, P. Sýkora, and R. Hudec, "Comparison of Background Subtraction Methods on Near Infra-Red Spectrum Video Sequences," *Procedia Eng.*, vol. 192, pp. 887–892, 2017. DOI: https://doi.org/10.1016/j.proeng.2017.06.153.

[122] G. S. Priya, M. Latha, K. Manoj, and S. Prakash, "Unusual Activity And Anomaly Detection In Surveillance Using GMM-KNN Model," in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pp. 1450–1457, 2021. DOI: https://doi.org/10.1109/ICICV50876.2021.9388587.

[123] A. Poonsri and W. Chiracharit, "Fall detection using Gaussian mixture model and principle component analysis," in *2017 9th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pp. 1–4, 2017. DOI: https://doi.org/10.1109/ICITEED.2017.8250441.

[124] G. Conti, M. Quintana, P. Malagón, and D. Jiménez, "An FPGA Based Tracking Implementation for Parkinson's Patients," *Sensors*, vol. 20, no. 11, p. 3189, 2020. DOI: https://doi.org/10.3390/s20113189.

[125] P. Machado, A. Oikonomou, J. F. Ferreira, and T. M. Mcginnity, "HSMD: An Object Motion Detection Algorithm Using a Hybrid Spiking Neural Network Architecture," *IEEE Access*, vol. 9, pp. 125258–125268, 2021. DOI: https://doi.org/10.1109/ACCESS.2021.3111005.

[126] P. Christiansen, L. Nielsen, K. Steen, R. Jørgensen, and H. Karstoft, "DeepAnomaly: Combining Background Subtraction and Deep Learning for Detecting Obstacles and Anomalies in an Agricultural Field," *Sensors*, vol. 16, no. 11, p. 1904, 2016. DOI: http://dx.doi.org/10.3390/s16111904.

[127] T. Yu, J. Yang, and W. Lu, "Combining Background Subtraction and Convolutional Neural Network for Anomaly Detection in Pumping-Unit Surveillance," *Algorithms*, vol. 12, no. 6, p. 115, 2019. DOI: http://dx.doi.org/10.3390/a12060115.

[128] A. Sklorz, S. Janßen, and W. Lang, "Application of a miniaturised packed gas chromatography column and a SnO2 gas detector for analysis of low molecular weight hydrocarbons with focus on ethylene detection," *Sens. Actuators. B Chem.*, vol. 180, pp. 43–49, 2013. DOI: https://doi.org/10.1016/j.snb.2011.12.110.

[129] B. Gruber, T. Groeger, D. Harrison, and R. Zimmermann, “Vacuum ultraviolet absorption spectroscopy in combination with comprehensive two-dimensional gas chromatography for the monitoring of volatile organic compounds in breath gas: A feasibility study,” *J. Chromatogr. A*, vol. 1464, pp. 141–146, 2016. DOI: https://doi.org/10.1016/j.chroma.2016.08.024.

[130] F. W. Karasek and R. E. Clement, *Basic Gas Chromatography-Mass Spectrometry: Principles and Techniques.* Amsterdam: Elsevier Science, 1988.

[131] P. Trefz, M. Schmidt, P. Oertel, J. Obermeier, B. Brock, S. Kamysek, J. Dunkl, R. Zimmermann, J. K. Schubert, and W. Miekisch, “Continuous Real Time Breath Gas Monitoring in the Clinical Environment by Proton-Transfer-Reaction-Time-of-Flight-Mass Spectrometry,” *Anal. Chem.*, vol. 85, no. 21, pp. 10321–10329, 2013. DOI: https://doi.org/10.1021/ac402298v.

[132] H. Edner, J. Sandsten, Y. Saito, J. Smith, S. Svanberg, and P. Weibring, “Novel remote gas monitoring and imaging in the IR spectral region,” in *Technical Digest. CLEO/Pacific Rim ’99. Pacific Rim Conference on Lasers and Electro-Optics*, pp. 577–578, 1999. DOI: https://doi.org/10.1109/CLEOPR.1999.811578.

[133] M. Hinnrichs, “Remote sensing for gas plume monitoring using state-of-the-art infrared hyperspectral imaging,” in *Environmental Monitoring and Remediation Technologies*, pp. 370–381, 1999. DOI: https://doi.org/10.1117/12.339016.

[134] G. Somesfalean, Z. G. Zhang, M. Sjöholm, and S. Svanberg, “All-diode-laser ultraviolet absorption spectroscopy for sulfur dioxide detection,” *Appl. Phys. B*, vol. 80, no. 8, pp. 1021–1025, 2005. DOI: https://doi.org/10.1007/s00340-005-1835-0.

[135] G. J. S. Bluth, J. M. Shannon, I. M. Watson, A. J. Prata, and V. J. Realmuto, “Development of an ultra-violet digital camera for volcanic SO2 imaging,” *J. Volcanol. Geotherm. Res.*, vol. 161, no. 1, pp. 47–56, 2007. DOI: https://doi.org/10.1016/j.jvolgeores.2006.11.004.

[136] S. Svanberg, “Geophysical gas monitoring using optical techniques: volcanoes, geothermal fields and mines,” *Opt. Lasers Eng.*, vol. 37, no. 2, pp. 245–266, 2002. DOI: https://doi.org/10.1016/S0143-8166(01)00098-7.

[137] H. Edner, P. Ragnarson, S. Svanberg, E. Wallinder, R. Ferrara, R. Cioni, B. Raco, and G. Taddeucci, "Total fluxes of sulfur dioxide from the Italian volcanoes Etna, Stromboli, and Vulcano measured by differential absorption lidar and passive differential optical absorption spectroscopy," *J. Geophys. Res. Atmos.*, vol. 99, no. D9, pp. 18827–18838, 1994. DOI: https://doi.org/10.1029/94JD01515.

[138] T. Lappalainen and J. Lehmonen, "PAPER PHYSICS: Determinations of bubble size distribution of foam-fibre mixture using circular hough transform," *Nord. Pulp Pap. Res. J.*, vol. 27, no. 5, pp. 930–939, 2012. DOI: https://doi.org/10.3183/npprj-2012-27-05-p930-939.

[139] Y. Y. Zuo, M. Ding, A. Bateni, M. Hoorfar, and A. W. Neumann, "Improvement of interfacial tension measurement using a captive bubble in conjunction with axisymmetric drop shape analysis (ADSA)," *Colloids Surf. A: Physicochem. Eng. Asp.*, vol. 250, no. 1, pp. 233–246, 2004. DOI: https://doi.org/10.1016/j.colsurfa.2004.04.081.

[140] T. Xue, L. Qu, Z. Cao, and T. Zhang, "Three-dimensional feature parameters measurement of bubbles in gas–liquid two-phase flow based on virtual stereo vision," *Flow Meas. Instrum.*, vol. 27, pp. 29–36, 2012. Special Issue: Multiphase Flow, DOI: https://doi.org/10.1016/j.flowmeasinst.2012.07.007.

[141] T. Haas, C. Schubert, M. Eickhoff, and H. Pfeifer, "BubCNN: Bubble detection using Faster RCNN and shape regression network," *Chem. Eng. Sci.*, vol. 216, p. 115467, 2020. DOI: https://doi.org/10.1016/j.ces.2019.115467.

[142] I. Malakhov, A. Seredkin, A. Chernyavskiy, V. Serdyukov, R. Mullyadzanov, and A. Surtaev, "Deep learning segmentation to analyze bubble dynamics and heat transfer during boiling at various pressures," *Int. J. Multiph. Flow*, vol. 162, p. 104402, 2023. DOI: https://doi.org/10.1016/j.ijmultiphaseflow.2023.104402.

[143] A. Chandran and S. Kavitha, "A Smart Gas Stove with Gas Leakage Detection and Multistage Prevention System Using IoT," *Int. J. Mod. Trends Sci. Technol.*, vol. 1, no. 9, p. 5–9, 2022. https://journal.ijmdes.com/ijmdes/article/view/79.

[144] J. Ryu and D. Kwak, "A Study on a Complex Flame and Smoke Detection Method Using Computer Vision Detection and Convolutional Neural Network," *Fire*, vol. 5, no. 4, p. 108, 2022. DOI: https://doi.org/10.3390/fire5040108.

[145] J. G. Peacey and W. G. Davenport, *The Iron Blast Furnace: Theory and Practice.* Oxford: Pergamon Press, 1979.

[146] J. Lee, G. T. Ahn, and S. Y. Park, "Slag Removal Path Estimation by Slag Distribution Image and Multi-Task Deep Learning Network," *IEEE Access*, vol. 9, pp. 118541–118552, 2021. DOI: https://doi.org/10.1109/ACCESS.2021.3107677.

[147] D. Pan, Z. Jiang, Z. Chen, W. Gui, Y. Xie, and C. Yang, "Temperature Measurement and Compensation Method of Blast Furnace Molten Iron Based on Infrared Computer Vision," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 10, pp. 3576–3588, 2019. DOI: https://doi.org/10.1109/TIM.2018.2880061.

[148] A. Lay-Ekuakille, M. A. Ugwiri, J. D. Okitadiowo, C. Chiffi, and A. Pietrosanto, "Computer Vision for Sensed Images Approach in Extremely Harsh Environments: Blast Furnace Chute Wear Characterization," *IEEE Sens. J.*, vol. 21, no. 10, pp. 11969–11976, 2021. DOI: https://doi.org/10.1109/JSEN.2021.3063264.

[149] W. Deng, Q. Zheng, and L. Chen, "Regularized extreme learning machine," in *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pp. 389–395, 2009. DOI: https://doi.org/10.1109/CIDM.2009.4938676.

[150] A. M. Yang, J. M. Zhi, K. Yang, J. H. Wang, and T. Xue, "Computer Vision Technology Based on Sensor Data and Hybrid Deep Learning for Security Detection of Blast Furnace Bearing," *IEEE Sens. J.*, vol. 21, no. 22, pp. 24982–24992, 2021. DOI: https://doi.org/10.1109/JSEN.2021.3077468.

[151] S. Li, X. Y. Tang, X. Wang, and X. Fu, "Blast Furnace Charging State Recognition based on YOLOv5," in *2022 China Automation Congress (CAC)*, pp. 6809–6814, 2022. DOI: https://doi.org/10.1109/CAC57257.2022.10055833.

[152] K. Jiang, Z. Jiang, Y. Xie, D. Pan, and W. Gui, "Prediction of Multiple Molten Iron Quality Indices in the Blast Furnace Ironmaking Process Based on Attention-Wise Deep Transfer Network," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–14, 2022. DOI: https://doi.org/10.1109/TIM.2022.3185325.

[153] J. S. Kim, G. T. Ahn, and S. Y. Park, "Estimation of slag removal path using CNN-based path probability of ladle image blocks," *Int. J. Control Autom. Syst.*, vol. 18, pp. 791–800, 2019. DOI: https://doi.org/10.1007/s12555-019-0019-3.

[154] D. G. Hong, W. H. Han, and C. H. Yim, “Tapping stream tracking model using computer vision and deep learning to minimize slag carry-over in basic oxygen furnace,” *Electron. res. arch.*, vol. 30, no. 11, pp. 4015–4037, 2022. DOI: https://doi.org/10.3934/era.2022204.

[155] L. Zhang and B. G. Thomas, “Inclusions in continuous casting of steel,” in *XXIV National Steelmaking Symposium*, pp. 26–28, 2003. http://ccc.illinois.edu/PDF%20Files/Publications/03_Mexico_Nov_Inclusion_review_v5a_updated.pdf.

[156] J. Zhang, X. Kang, H. Ni, and F. Ren, “Surface defect detection of steel strips based on classification priority YOLOv3-dense network,” *Ironmak. Steelmak.*, vol. 48, no. 5, pp. 547–558, 2020. DOI: https://doi.org/10.1080/03019233.2020.1816806.

[157] A. Grishin, BorisV, iBardintsev, Inversion, and Oleg, “Severstal: Steel Defect Detection.” https://kaggle.com/competitions/severstal-steel-defect-detection, 2019. Accessed: 2024 Mar 12.

[158] M. Wieler, T. Hahn, and F. A. Hamprecht, “Weakly Supervised Learning for Industrial Optical Inspection.” https://zenodo.org/records/8086136, 2023. Accessed: 2024 Mar 12.

[159] D. Tabernik, S. Šela, J. Skvarč, and D. Skočaj, “Segmentation-Based Deep-Learning Approach for Surface-Defect Detection,” *J. Intell. Manuf.*, 2019. DOI: https://doi.org/10.1007/s10845-019-01476-x.

[160] Meta AI, “KolektorSDD (Kolektor Surface-Defect Dataset).” https://paperswithcode.com/dataset/kolektorsdd, n.d. Accessed: 2024 Apr 24.

[161] Y. He, K. Song, Q. Meng, and Y. Yan, “An End-to-End Steel Surface Defect Detection Approach via Fusing Multiple Hierarchical Features,” *IEEE Trans. Instrum. Meas.*, vol. 69, no. 4, pp. 1493–1504, 2020. DOI: https://doi.org/10.1109/TIM.2019.2915404.

[162] Y. Huang, C. Qiu, Y. Guo, X. Wang, and K. Yuan, “Surface Defect Saliency of Magnetic Tile,” in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 612–617, 2018. DOI: https://doi.org/10.1109/COASE.2018.8560423.

[163] A. Kim, “Magnetic Tile Surface Defects.” https://www.kaggle.com/datasets/alex000kim/magnetic-tile-surface-defects/data, 2020. Accessed: 2024 Apr 24.

[164] W. Yan, C. Chen, and D. Zhang, "U-Net-based medical image segmentation algorithm," in *2021 13th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–5, 2021. DOI: https://doi.org/10.1109/WCSP52459.2021.9613447.

[165] L. Gao, J. Zhang, C. Yang, and Y. Zhou, "Cas-VSwin transformer: A variant swin transformer for surface-defect detection," *Comput. Ind.*, vol. 140, p. 103689, 2022. DOI: https://doi.org/10.1016/j.compind.2022.103689.

[166] M. Li, H. Wang, and Z. Wan, "Surface defect detection of steel strips based on improved YOLOv4," *Comput. Electr. Eng.*, vol. 102, p. 108208, 2022. DOI: https://doi.org/10.1016/j.compeleceng.2022.108208.

[167] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018. DOI: https://openaccess.thecvf.com/content_ECCV_2018/papers/Sanghyun_Woo_Convolutional_Block_Attention_ECCV_2018_paper.pdf.

[168] A. Vozmilov, V. Urmanov, and A. Lisov, "Using Computer Vision to Recognize Defects on the Surface of Hot-rolled Steel," in *2022 International Ural Conference on Electrical Power Engineering (UralCon)*, pp. 21–25, 2022. DOI: https://doi.org/10.1109/UralCon54942.2022.9906737.

[169] V. Nath, C. Chattopadhyay, and K. A. Desai, "NSLNet: An improved deep learning model for steel surface defect classification utilizing small training datasets," *Manuf. Lett.*, vol. 35, pp. 39–42, 2023. DOI: https://doi.org/10.1016/j.mfglet.2022.10.001.

[170] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6569–6578, 2019. DOI: https://openaccess.thecvf.com/content_ICCV_2019/papers/Duan_CenterNet_Keypoint_Triplets_for_Object_Detection_ICCV_2019_paper.pdf.

[171] R. Tian and M. Jia, "DCC-CenterNet: A rapid detection method for steel surface defects," *Measurement*, vol. 187, p. 110211, 2022. DOI: https://doi.org/10.1016/j.measurement.2021.110211.

[172] Z. Guo, C. Wang, G. Yang, Z. Huang, and G. Li, “Msft-yolo: Improved yolov5 based on transformer for detecting defects of steel surface,” *Sensors*, vol. 22, no. 9, p. 3467, 2022. DOI: https://doi.org/10.3390/s22093467.

[173] N. Imamoglu, W. Lin, and Y. Fang, “A Saliency Detection Model Using Low-Level Features Based on Wavelet Transform,” *IEEE Trans. Multimedia.*, vol. 15, no. 1, pp. 96–105, 2013. DOI: https://doi.org/10.1109/TMM.2012.2225034.

[174] G. Lin, A. Milan, C. Shen, and I. Reid, “RefineNet: Multi-path Refinement Networks for High-Resolution Semantic Segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1925–1934, 2017. DOI: https://doi.org/10.1109/CVPR.2017.549.

[175] T. Liu and Z. He, “TAS 2-Net: Triple-Attention Semantic Segmentation Network for Small Surface Defect Detection,” *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–12, 2022. DOI: https://doi.org/10.1109/TIM.2022.3142023.

[176] S. Azimi, D. Britz, M. Engstler, M. Fritz, and F. Mücklich, “Advanced Steel Microstructural Classification by Deep Learning Methods,” *Sci. Rep.*, vol. 8, 2018. DOI: https://doi.org/10.1038/s41598-018-20037-5.

[177] B. L. DeCost, B. Lei, T. Francis, and E. A. Holm, “High Throughput Quantitative Metallography for Complex Microstructures Using Deep Learning: A Case Study in Ultrahigh Carbon Steel,” *Microsc. Microanal*, vol. 25, no. 1, pp. 21–29, 2019. DOI: https://doi.org/10.1017/S1431927618015635.

[178] B. Mulewicz, G. Korpala, J. Kusiak, and U. Prahl, “Autonomous Interpretation of the Microstructure of Steels and Special Alloys,” *Mater. Sci. Forum.*, vol. 949, pp. 24–31, 2019. DOI: https://doi.org/10.4028/www.scientific.net/MSF.949.24.

[179] W. Zhang, C. F. Yang, F. Jiang, and X. Z. Gao, “Safety Helmet Wearing Detection Based on Image Processing and Deep Learning,” in *2020 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pp. 343–347, 2020. DOI: https://doi.org/10.1109/CISCE50729.2020.00076.

[180] S. Kang and H. Wang, “Crane Hook Detection Based on Mask R-CNN in Steel-making Plant,” *J. Phys. Conf. Ser.*, vol. 1575, 2020. DOI: https://doi.org/10.1088/1742-6596/1575/1/012151.

[181] Q. J. Zhao, P. Cao, and D. W. Tu, “Toward intelligent manufacturing: label characters marking and recognition method for steel products with machine vision,” *Adv. Manuf.*, vol. 2, pp. 3–12, 2014. DOI: https://doi.org/10.1007/s40436-014-0057-2.

[182] S. J. Lee, J. P. Yun, G. Koo, and S. W. Kim, “End-to-end recognition of slab identification numbers using a deep convolutional neural network,” *Knowl.-Based Syst.*, vol. 132, pp. 1–10, 2017. DOI: https://doi.org/10.1016/j.knosys.2017.06.017.

[183] S. J. Lee, S. W. Kim, W. Kwon, G. Koo, and J. P. Yun, “Selective Distillation of Weakly Annotated GTD for Vision-Based Slab Identification System,” *IEEE Access*, vol. 7, pp. 23177–23186, 2019. DOI: https://doi.org/10.1109/ACCESS.2019.2899109.

[184] T. Caldeira, P. M. Ciarelli, and G. A. Neto, “Industrial optical character recognition system in printing quality control of hot-rolled coils identification,” *J. Control Autom. Electr. Syst.*, vol. 31, pp. 108–118, 2020. DOI: https://doi.org/10.1007/s40313-019-00551-1.

[185] F. Chang, M. Dong, M. Liu, L. Wang, and Y. Duan, “A Lightweight Appearance Quality Assessment System Based on Parallel Deep Learning for Painted Car Body,” *IEEE Trans. Instrum. Meas.*, vol. 69, no. 8, pp. 5298–5307, 2020. DOI: https://doi.org/10.1109/TIM.2019.2962565.

[186] M. J. Shafiee, M. Famouri, G. Bathla, F. Li, and A. Wong, “Tinydefectnet: Highly compact deep neural network architecture for high-throughput manufacturing visual quality inspection,” *arXiv*, 2021. DOI: https://doi.org/10.48550/arXiv.2111.14319.

[187] Z. Mo, L. Chen, and W. You, “Identification and Detection of Automotive Door Panel Solder Joints Based on YOLO,” in *2019 Chinese Control And Decision Conference (CCDC)*, pp. 5956–5960, 2019. DOI: https://doi.org/10.1109/CCDC.2019.8833257.

[188] X. Zheng, J. Chen, H. Wang, S. Zheng, and Y. Kong, “A deep learning-based approach for the automated surface inspection of copper clad laminate images,” *Appl. Intell.*, vol. 51, pp. 1262–1279, 2021. DOI: https://doi.org/10.1007/s10489-020-01877-z.

[189] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018. DOI: https://doi.org/10.1109/CVPR.2018.00474.

[190] J. Li, J. Gu, Z. Huang, and J. Wen, "Application Research of Improved YOLO V3 Algorithm in PCB Electronic Component Detection," *Appl. Sci.*, vol. 9, no. 18, p. 3750, 2019. DOI: https://doi.org/10.3390/app9183750.

[191] Y. Zhang, H. G. Soon, D. Ye, J. Y. H. Fuh, and K. Zhu, "Powder-Bed Fusion Process Monitoring by Machine Vision With Hybrid Convolutional Neural Networks," *IEEE Trans. Ind. Informatics.*, vol. 16, no. 9, pp. 5769–5779, 2020. DOI: https://doi.org/10.1109/TII.2019.2956078.

[192] Z. Tan, Q. Fang, H. Li, S. Liu, W. Zhu, and D. Yang, "Neural network based image segmentation for spatter extraction during laser-based powder bed fusion processing," *Opt. Laser Technol.*, vol. 130, p. 106347, 2020. DOI: https://doi.org/10.1016/j.optlastec.2020.106347.

[193] J. Chai, H. Zeng, A. Li, and E. W. T. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100134, 2021. DOI: https://doi.org/10.1016/j.mlwa.2021.100134.

[194] D. Tian, Y. Han, B. Wang, T. Guan, H. Gu, and W. Wei, "Review of object instance segmentation based on deep learning," *Journal of Electronic Imaging*, vol. 31, no. 4, pp. 041205–041205, 2022. DOI: https://doi.org/10.1117/1.JEI.31.4.041205.

[195] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards Real-Time Multi-Object Tracking," in *Computer Vision - ECCV 2020*, pp. 107–122, 2020. DOI: https://doi.org/10.1007/978-3-030-58621-8_7.

[196] Ultralytics, "Train Custom Data." https://docs.ultralytics.com/yolov5/tutorials/train_custom_data/#1-create-dataset, 2023. Accessed: 2024 Mar 12.

[197] G. Jocher, "hyp.scratch-low.yaml." https://github.com/ultralytics/yolov5/blob/master/data/hyps/hyp.scratch-low.yaml, 2020. Accessed: 2024 Mar 12.

[198] E. Phaisangittisagul, "An Analysis of the Regularization Between L2 and Dropout in Single Hidden Layer Neural Network," in *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pp. 174–179, 2016. DOI: https://doi.org/10.1109/ISMS.2016.14.

[199] G. Jocher, “TFLite, ONNX, CoreML, TensorRT Export.” https://docs.ultralytics.com/yolov5/tutorials/model_export/, 2020. Accessed: 2024 Mar 12.

[200] ONNX, “Open Neural Network Exchange.” https://onnx.ai/, 2019. Accessed: 2024 Mar 12.

[201] NVIDIA, “TensorRT SDK.” https://developer.nvidia.com/tensorrt, 2024. Accessed: 2024 Mar 12.

[202] S. Zeevi, “BackgroundSubtractorCNT Documentation.” https://sagi-z.github.io/BackgroundSubtractorCNT/doxygen/html/index.html, 2023. Accessed: 2024 Mar 12.

[203] Sagi-z, “BackgroundSubtractorCNT.” https://github.com/sagi-z/BackgroundSubtractorCNT/tree/master, 2020. Accessed: 2024 Mar 12.

[204] A. Rosebrock, “Mean Average Precision (mAP) using the COCO Evaluator.” https://pyimagesearch.com/2022/05/02/mean-average-precision-map-using-the-coco-evaluator/, 2022. Accessed: 2024 Mar 12.

[205] Scikit Learn, “Precision-Recall.” https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html, 2023. Accessed: 2024 Mar 12.

[206] J. Hui, “mAP (mean Average Precision) for Object Detection.” https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173, 2018. Accessed: 2024 Mar 12.

[207] Microsoft, “COCO Detection Evaluation.” https://cocodataset.org/#detection-eval, 2019. Accessed: 2024 Mar 12.

[208] Microsoft, “cocoeval.py.” https://github.com/cocodataset/cocoapi/blob/master/PythonAPI/pycocotools/cocoeval.py, 2019. Accessed: 2024 Mar 12.

[209] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. da Silva, “A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit,” *Electronics*, vol. 10, no. 3, pp. 1–28, 2021. DOI: https://doi.org/10.3390/electronics10030279.

[210] H. Zhang, A. Rogozan, and A. Bensrhair, “An enhanced N-point interpolation method to eliminate average precision distortion,” *Pattern Recognit. Lett.*, vol. 158, pp. 111–116, 2022. DOI: https://doi.org/10.1016/j.patrec.2022.04.028.

[211] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 237–242, 2020. DOI: https://doi.org/10.1109/IWSSIP48289.2020.9145130.

[212] P. Dendorfer, A. Ošep, A. Milan, K. Schindler, D. Cremers, I. Reid, S. Roth, and L. Leal-Taixé, "MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking," *Int. J. Comput. Vis.*, vol. 129, no. 4, pp. 845–881, 2021. DOI: https://doi.org/10.1007/s11263-020-01393-0.

[213] Microsoft, "COCO - Common Objects in Context Homepage." https://cocodataset.org/#home, 2020. Accessed: 2024 Mar 12.

[214] Microsoft, "COCO - Common Objects in Context." https://cocodataset.org/#download, 2020. Accessed: 2024 Mar 12.

[215] A. Dutta, A. Gupta, and A. Zisserman, "VGG Image Annotator." https://www.robots.ox.ac.uk/~vgg/software/via/via_demo.html, 2019. Accessed: 2024 Mar 12.

[216] J. Solawetz, "How To Convert VGG Image Annotator JSON to YOLOv5 PyTorch TXT." https://roboflow.com/convert/via-json-to-yolov5-pytorch-txt, 2023. Accessed: 2024 Mar 12.

[217] NVIDIA, "GeForce RTX 2070 Super User Guide." https://www.nvidia.com/content/geforce-gtx/GEFORCE_RTX_2070_SUPER_User_Guide.pdf, 2018. Accessed: 2024 Mar 12.

[218] NVIDIA, "CUDA Toolkit." https://developer.nvidia.com/cuda-toolkit, 2023. Accessed: 2024 Mar 12.

[219] NVIDIA, "CUDA Deep Neural Network." https://www.nvidia.com/content/geforce-gtx/GEFORCE_RTX_2070_SUPER_User_Guide.pdf, 2023. Accessed: 2024 Mar 12.

[220] C. O'Donovan, "Ladle Tracking." https://github.com/Callum232310/Hot-Metal-Ladle-Tracking, 2024. Accessed: 2024 Mar 12.

[221] K. Kuter, "Probability Mass Functions (PMFs) and Cumulative Distribution Functions (CDFs) for Discrete Random Variables." https://stats.libretexts.org/

Courses/Saint_Mary's_College_Notre_Dame/MATH_345__-_Probability_(Kuter)/3%3A_Discrete_Random_Variables/3.2%3A_Probability_Mass_Functions_(PMFs)_and_Cumulative_Distribution_Functions_(CDFs)_for_Discrete_Random_Variables, 2020. Accessed: 2024 Mar 12.

[222] OpenCV Developers, "Histogram Equalization." https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html, 2022. Accessed: 2024 Mar 12.

[223] S. Arora, M. Agarwal, V. Kumar, and D. Gupta, "Comparative study of image enhancement techniques using histogram equalization on degraded images," *Int. J. Eng. Sci. Technol.*, vol. 7, pp. 468–471, 2018. DOI: https://doi.org/10.14419/ijet.v7i2.8.10487.

[224] S. K. Shome and S. R. K. Vadali, "Enhancement of Diabetic Retinopathy Imagery Using Contrast Limited Adaptive Histogram Equalization," *Int. J. Comput. Sci. Inf. Technol.*, vol. 2, pp. 2694–2699, 2011. https://api.semanticscholar.org/CorpusID:15587978.

[225] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *J. Mach. Learn. Res.*, vol. 13, no. 10, pp. 281–305, 2012. http://jmlr.org/papers/v13/bergstra12a.html.

[226] R. K. Roy, *Design of Experiments Using The Taguchi Approach 16 Steps to Product and Process Improvement.* New Jersey: Wiley, 2001.

[227] Matterport, "Mask R-CNN for Object Detection and Segmentation: Configuration File." https://github.com/matterport/Mask_RCNN/blob/master/mrcnn/config.py, 2017. Accessed: 2024 Mar 12.

[228] S. Song, J. Zhu, X. Li, and Q. Huang, "Integrate MSRCR and Mask R-CNN to Recognize Underwater Creatures on Small Sample Datasets," *IEEE Access*, vol. 8, pp. 172848–172858, 2020. DOI: https://doi.org/10.1109/ACCESS.2020.3025617.

[229] Matterport, "Mask R-CNN for Object Detection and Segmentation: Model File." https://github.com/matterport/Mask_RCNN/blob/master/mrcnn/model.py, 2017. Accessed: 2024 Mar 12.

[230] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "7.4 Momentum - Dive into Deep Learning." https://d2l.ai/chapter_optimization/momentum.html, 2020. Accessed: 2024 Mar 12.

[231] Q. Fang, C. Ibarra-Castanedo, I. Garrido, Y. Duan, and X. Maldague, "Automatic Detection and Identification of Defects by Deep Learning Algorithms from Pulsed Thermography Data," *Sensors*, vol. 23, no. 9, p. 4444, 2023. DOI: https://doi.org/10.3390/s23094444.

[232] "SGD with Momentum." https://paperswithcode.com/method/sgd-with-momentum, 2020. Accessed: 2024 Mar 12.

[233] X. Chen, Z. S. Wu, and M. Hong, "Understanding gradient clipping in private SGD: a geometric perspective," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020. https://proceedings.neurips.cc/paper/2020/file/9ecff5455677b38d19f49ce658ef0608-Paper.pdf.

[234] J. Hosang, R. Benenson, and B. Schiele, "Learning Non-Maximum Suppression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6469–6477, 2017. DOI: https://doi.org/10.1109/CVPR.2017.685.

[235] WOLFRAM Demonstrations Project, "Convex hull." https://demonstrations.wolfram.com/ConvexHull/, 2007. Accessed: 2024 Jan 20.

[236] D. T. Lee, "On Finding the Convex Hull of a Simple Polygon," *Int. J. Comput. Inf.*, vol. 12, pp. 87–98, 1983. DOI: https://doi.org/10.1007/BF00993195.

[237] R. Sadli, "Object tracking: 2-D object tracking using kalman filter in python." https://machinelearningspace.com/2d-object-tracking-using-kalman-filter/, 2020. Accessed: 2024 Jan 20.

[238] OpenCV developers, "Background Subtraction MOG." https://docs.opencv.org/3.4/d8/d38/tutorial_bgsegm_bg_subtraction.html, 2024. Accessed: 2024 Mar 12.

[239] OpenCV developers, "BackgroundSubtractorMOG2 Class Reference." https://docs.opencv.org/4.x/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html, 2024. Accessed: 2024 Mar 12.

[240] OpenCV developers, "BackgroundSubtractorGSOC Class Reference." https://docs.opencv.org/4.x/d4/dd5/classcv_1_1bgsegm_1_1BackgroundSubtractorGSOC.html#details, 2024. Accessed: 2024 Mar 12.

[241] A. B. Godbehere, A. Matsukawa, and K. Goldberg, "Visual Tracking of Human Visitors Under Variable-Lighting Conditions for a Responsive Audio Art Installation," in *2012*

American Control Conference (ACC), pp. 4305–4312, 2012. DOI: https://doi.org/10.1109/ACC.2012.6315174.

[242] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 773–780, 2006. DOI: https://doi.org/10.1016/j.patrec.2005.11.005.

[243] OpenCV developers, "BackgroundSubtractorCNT Class Reference." https://docs.opencv.org/3.4/de/dca/classcv_1_1bgsegm_1_1BackgroundSubtractorCNT.html, 2024. Accessed: 2024 Mar 12.

[244] United States Navy, "Foundry Manual," 1958. Accessed: 2024 Mar 12.

[245] C. O'Donovan, "Splatter Severity Measurement." https://github.com/Callum232310/Splatter-Severity-Measurement, 2024. Accessed: 2024 Mar 12.

[246] Newcastle University, "Normal Distribution." https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/distributions/normal-distribution.html, 2023. Accessed: 2024 Mar 12.

[247] OpenCV, "Image Filtering." https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html, 2023. Accessed: 2024 Mar 12.

[248] P. KaewTraKulPong and R. Bowden, *An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection*, pp. 135–144. Springer US, 2002. DOI: https://doi.org/10.1007/978-1-4615-0913-4_11.

[249] L. Guo, D. Xu, and Z. Qiang, "Background Subtraction Using Local SVD Binary Pattern," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1159–1167, 2016. DOI: https://doi.org/10.1109/CVPRW.2016.148.

[250] OpenCV, "BackgroundSubtractorGSOC Class Documentation." https://docs.opencv.org/4.x/d4/dd5/classcv_1_1bgsegm_1_1BackgroundSubtractorGSOC.html, 2023. Accessed: 2024 Mar 12.

[251] J. Bobulski and K. L., "Background Segmentation Method for Autonomous Car," in *26th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI)*, pp. 19–22, 2022. DOI: https://doi.org/10.54808/WMSCI2022.01.19.

[252] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, 2002. DOI: https://doi.org/10.1109/TPAMI.2002.1017623.

[253] Scikit-learn Developers, "sklearn.neighbors.KNeighborsClassifier - Scikit-learn Documentation." https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html, 2023. Accessed: 2024 Mar 12.

[254] changedetection.net, "CDW-2014." http://changedetection.net/, 2024. Accessed: 2024 Mar 12.

[255] J. Joyce, "Bayes' Theorem." https://plato.stanford.edu/archives/fall2021/entries/bayes-theorem/, 2021. Accessed: 2024 Mar 12.

[256] NVIDIA, "NVIDIA Jetson Orin Nano Developer Kit Datasheet." https://docs.rs-online.com/4051/A700000009607470.pdf, 2023. Accessed: 2024 Mar 12.

[257] Docker, "Docker overview." https://docs.docker.com/get-started/overview/, 2024. Accessed: 2024 Mar 12.

[258] Tiangolo, "FastAPI." https://fastapi.tiangolo.com/, 2024. Accessed: 2024 Mar 12.

[259] Uvicorn, "Uvicorn." https://www.uvicorn.org/, 2024. Accessed: 2024 Mar 12.

[260] R. J. Fruehan, *The Making, Shaping and Treating of Steel.* Pittsburgh: The AISE Steel Foundation, 1998.

[261] N. L. Kotraba, *PNEUMATIC STEELMAKING.* Warrendale: The Iron and Steel Society, 1988.

[262] T. Aoki, "The mechanism of the back-attack phenomenon on a bottom blowing tuyere investigated in model experiments," *ISIJ*, vol. 76, no. 11, pp. 1996–2003, 1990. DOI: https://www.jstage.jst.go.jp/article/tetsutohagane1955/76/11/76_11_1996/_pdf.

[263] K. Andreev, W. M. Heijboer, R. Siebring, Y. Bi, M. Frith, J. P. Everstein, S J. andBrockhoff, M. C. Davies, and C. Vaughan, "BOF bottom wear around tuyeres and blowing elements - thermo-mechanical investigation to optimise the lining performance," in *Proceedings of the Unified International Technical Conference on Refractories (UNITECR 2007)*, pp. 318–321, 2007. DOI:

https://www.researchgate.net/publication/262437797_BOF_bottom_wear_around_tuyeres_and_blowing_elements_-_investigation_to_optimise_the_lining_performance.

[264] S. Kubal, *A New Approach to BOF Mixing Element Design: Development of Annular Slot Tuyère*. PhD thesis, Swansea University, Swansea, UK, 2017. Engineering Doctorate Thesis.

[265] C. O'Donovan, "Plume Monitoring." https://github.com/Callum232310/Plume-Monitoring, 2024. Accessed: 2024 Mar 12.

[266] M. Broström, "Real-time multi-object tracker using YOLOv5 and deep sort." https://github.com/mikel-brostrom/Yolov5_DeepSort_Pytorch, 2020. Accessed: 2024 Mar 12.

[267] Y. Wu, A. Kirillov, F. Massa, W. Y. Lo, and R. Girshick, "Detectron2." https://github.com/facebookresearch/detectron2, 2019. Accessed: 2024 Mar 12.

[268] P. Dwivedi, "Aerial Pedestrian Detection." https://github.com/priya-dwivedi/aerial_pedestrian_detection, 2019. Accessed: 2024 Mar 12.

[269] S. R. Maiya, "Nanonets Object Tracking." https://github.com/abhyantrika/nanonets_object_tracking, 2021. Accessed: 2024 Mar 12.

[270] J. Solawetz, "How To Convert VGG Image Annotator JSON to Pascal VOC XML." https://roboflow.com/convert/via-json-to-pascal-voc-xml, 2023. Accessed: 2024 Mar 12.

[271] D. Kleitman, "Volume of an Ellipsoid: the Disk Method." https://math.mit.edu/~djk/18_01/chapter17/section02.html, n.d. Accessed: 2024 Mar 12.

[272] OpenCV developers, "Thresholding Operations Using InRange." https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html, 2023. Accessed: 2024 Mar 12.

[273] OpenCV developers, "Hough Line Transform." https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html, 2023. Accessed: 2024 Mar 12.

[274] scikit-learn developers, "Tuning the hyper-parameters of an estimator." https://scikit-learn.org/stable/modules/grid_search.html, 2023. Accessed: 2024 Mar 12.

[275] Scikit-learn developers, "sklearn.model_selection.KFold." https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html, 2023. Accessed: 2024 Mar 12.

[276] Visual Geometry Group, "Visual Geometry Group at the University of Oxford." http://www.robots.ox.ac.uk/~vgg/. Accessed: 2024 Mar 12.

[277] NVIDIA, "NVIDIA T4." https://www.nvidia.com/en-gb/data-center/tesla-t4/, 2024. Accessed: 2024 Mar 12.