Design and testing of a position adaptation system for KUKA robots using photoelectric sensors

A dissertation submitted to the University of Swansea, for the degree of

MSc by Research in Mechanical Engineering

by

Adam James Morgan

College of Engineering

Swansea University

Bay Campus

Swansea

SA1 8EN

2020

ABSTRACT

This thesis presents the development and analysis of a position monitoring and adaptation system to be used in conjunction with a KUKA KR16-2 articulated robot using components readily available in most manufacturing settings. This system could be beneficial in the manufacturing sector in areas such as polymer welding and spray painting. In the former it could be used to maintain an effective distance between a welding end effector laying molten plastic and the surface area of the parts being welded, or in the case of the latter the system would be useful in painting objects of unknown shape or objects with unknown variations in the surface level. In the case of spray painting if you spray to close to an object you will get an inconsistent amount of paint applied to an area. This system would maintain the programmed distance between the robot system and target object. Typically, systems that achieve this level of control rely on expensive sensors such as force torque sensors. This research proposes to take the first step in trying to address the technical problems by introducing a novel way of adapting to a target surface deformation using comparably low cost photoelectric diffuse sensors.

The key outcomes of this thesis can be found in the form of a software package to interface the photo-electric sensors to the KUKA robot system. This system is operated by a custom-built algorithm which is capable of dynamically calculating robot movements based off the sensor input. Additionally, an optimum system setup is developed with different configurations of sensor mounting and speeds of robot operation discussed and tested. The viability of the photo-electric diffuses sensors used in this application is also considered with further works suggested. Finally, a secondary application is developed for recording and analysing KUKA robot movements for use in other research activities.

ACKNOWLEDGMENTS

I cannot express enough thanks to my primary supervisor Dr Christian Griffiths for his patience, commitment and support throughout writing this thesis. My completion of this project could not have been accomplished without his support and encouragement. I would also like to thank my secondary supervisor Dr Ashraf Fahmy for the inspiration and support that inspired me to start this journey of enlightenment.

I would like to express my sincere gratitude to Professor Johann Sienz and ASTUTE2020 for providing me with this extraordinary opportunity to study and to improve myself at Swansea University. I want to thank all the ASTUTE staff for their support and encouragement along the way.

Finally, I would like to thank my family and my partner Michelle Osborne in particular, if it wasn't for her I would not have started or finished this MSc. Her constant encouragement and support have been a light through the darker and more difficult times over the last two years and for that I will always be thankful.

DECLARATION

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree

Signed…  ndidate)
Date …02/03/2021……

Statement 1

This thesis is the result of my own investigation, exception where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed…  Candidate)
Date …02/03/2021……

Statement 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisatio

Signed…  .... (Candidate)
Date ……02/03/2021……..

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

NOTATION

TCP	Tool Centre Point
RSI	Robot Sensor Interface
UDP	User Datagram Protocol
RUR	Rossum's Universal Robots
ASEA	Allmänna Svenska Elektriska Aktiebolaget
ABB	ASEA Brown Boveri
TIG	Tungsten Inert Gas
MIG	Metal Inert Gas
UAV	Unmanned Aerial Vehicle
APC	Amazon Picking Challenge
Cobots	Collaborative Robots
HRI	Human Robot Interface
AUV	Autonomous Underwater Vehicle
SPURV	Self-propelled Underwater Research Vehicle
IoT	Internet of Things
AGV	Automatic Guided Vehicle
VR	Virtual Reality
AR	Augmented Reality
FMS	Flexible Manufacturing Systems
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue, Depth
PC	Personal Computer
PTP	Point to Point movement
LIN	Linear movement
CIRC	Circular movement
SPTP	Spline PTP Movement
SLIN	Spline LIN Movement
SCIRC	Spline CIRC Movement
PROFIBUS	Process Field Bus
BMBF	German Department of Education and Research
ZVEI	German Central Association for the Electrical Industry
PROFIBUS DP	PROFIBUS Decentralised Periphery

KRL	KUKA Robot Language
XML	Extensible Mark-up Language
MRAC	Model Reference Adaptive Control
PID	Proportional Integral Derivative
PLC	Programmable Logic Controller
I/O	Input/output
IP	Internet Protocol
USB	Universal Serial Bus
UI	User Interface
DAQ	Data Acquisition Device
CSV	Comma-Separated Values
IPOC	Movement Timestamp
RIst	Setpoint position of robot TCP
RSol	Actual position of robot TCP
AIPos	Setpoint position of robot axis
ASPos	Actual position of robot axis
PCB	Printed Circuit Board

1. INTRODUCTION

This thesis is a culmination of study and development towards the goal of creating a robot end effector position varying system that is capable of maintaining a specified distance between the robot tool centre point (TCP) and a target work surface. The system needs to be able to adapt to an unknown geometry or surface that has inconsistent surface levels. This project was inspired by research on a plastic welding system capable of adapting to deformations of target polymers in a live environment using a force torque sensor. This thesis takes the step of generalising such a system so it may be applied to various other manufacturing processes such as spray painting.

With the ever-increasing applications of robot systems in manufacturing, businesses deploying such systems are always looking to increase their return on investment. With this in mind, one of the first objective for this research was to develop an application that could be easily introduced into manufacturing environments utilising existing technology. Using a KUKA KR16-2 a six-axis articulated robot as an example system, a return on investment could be achieved by utilising Python programming to develop an application that would interface two photoelectric diffuse sensors and the robot's KUKA C4 controller via a computer utilising the KUKA Robot Sensor Interface (RSI) software. To test the viability of this system, a series of experiments were conducted using a KUKA KR16-2 robot paired with a KUKA C4 controller.

First, a study of existing methods of interfacing with a KUKA robot was undertaken to establish what software is available on the market or through existing research. Additional focus was spent investigating existing polymer welding methods, and current applications in robotics. Finally, a study in to the manual methods of polymer welding was undertaken to establish an understanding of the required robotic control needed for an automated application. Initially an interface was designed to allow communications between a PC and the KUKA C4 using the KUKA Robot Sensor Interface. A User Datagram Protocol (UDP) point-to-point network was utilised with Python programming to build the underlying communication method. Once satisfactory communication was achieved, a study and trial on how to control the KUKA KR16-2 robots' motion entirely from the PC rather than the KUKA pendant device was conducted. Different combinations of RSI configuration were tested to suit the requirements of the project and to establish better understanding of the software's

capabilities. Through this testing an additional benefit of this research was achieved where a method of recording data from a KUKA robot was created for robots utilising the RSI package. Information such as Cartesian coordinates, angle position, torque and voltage values are recorded from the robot while a KUKA program is in motion, allowing for analysis and testing of robot programs/motions. Details of this research are also contained in this thesis.

When the movements of the robot were satisfactorily established, an understanding of how to perform concurrent communication and calculations simultaneously was needed to allow parallel communication with the robot and calculation or robot movements based off user input. This resulted in using parallel processing functions to create processes that were assigned to separate computing threads to avoid conflict and to allow this functionality to run without conflict. Additionally, an understanding of how to connect and control sensory devices via a PC is required. Research into software and hardware requirements was completed with an overview of different types of sensors applicable to this research. Finally, all these steps were then tied together in the final application presented in this thesis and the testing surrounding its suitability is presented.

1.1 RESEARCH OBJECTIVES

The aim of this research is to develop a way for an industrial robot to adapt to deformations in a surface material to maintain a specific distance between a robot end effector and said surface using photoelectric diffuse sensors. Analysis of robot movements using calculated geometry as obstacles to influence robot movement is conducted. The effectiveness of the motion control in the desired task is measured by extracting data from the robot and comparing the position to predicted movements. To assist analysing experimental data, a system needs to be investigated to record the robot operations and to be able to visual this data in a useful manner.

To summarise, the objects of this research are:

- Develop a communication interface with a KUKA KR16 robot.
- Develop a robot and sensor interface for positional accuracy.
- Develop a system to adapt to sensory input.
- Design and implement a novel system to extract robot data for post experiment analysis.
- Create a process for analysis of the data extracted.

1.2 THESIS ORGANISATION

This thesis is structured in a rather linear fashion where the reader is guided through the process of designing an application for robot position adaptation from its inception to a working prototype. The first chapter lays out the objectives of the thesis (detailed in the previous section). Chapter 2 is an overview of all the literature reviewed before undertaking this project. This involves applications in industry where the position adaptation system could potentially be beneficial such as plastic welding and painting. This chapter 3 contains a detailed examination of the KUKA KR16 industrial robot and associated user interfaces. The focus being on the equipment and configuration of the work area of the robot and the software configurations for interfacing with the robot controller. Next, a review of the sensors being utilised in the project along with the equipment needed to operate them is presented, finalising in a set of investigatory experiments used to confirm a sufficient understanding of the robot and systems has been established. Chapter 4 presents the development of the application with a focus on the program development. Chapter 5 introduces the KUKA logging software developed to aid with robot analysis during this project. In chapter 6 a series of experiments is conducted and where the program developed is optimised further. Finally, in chapter 7 a review the project as a whole is completed where the merits are discussed as well as its drawbacks and future prospects of the proposed automation system.

2. LITERATURE REVIEW

2.1 Introduction

This chapter sets out the groundwork required for this project. The intention is to lay out for the reader a comprehensive history of robotics to build a better understanding of where robotics came from and where it is going. The following sections will take the reader from the earliest origins of the word robot up until modern day manufacturing examples, addressing all the major steps along the way. This literature review then becomes more focussed and address relevant literature in areas such as robotic welding which links directly to this project and highlights some key knowledge gaps that present an opportunity for producing some novel research.

2.2 History of robotics

Human obsession with robotics can be traced back to the eighteenth century. One of the earliest inventions that had fascinated Europeans was the Canard Digérateur or by its more common name the digestion duck [1], later known as the Vaucanson's duck, a mechanical device that had the appearance of being able to eat and digest corn and grain and to later excrete the processed food that operated on a clockwork system of gears and pipes that could mimic the movement of a duck. While it was later found to not actually process the food but rather excrete stored waste, it was described by Vaucanson as a machine or automaton. It was not a very complex device but this was one of many important first steps by man towards todays robotics and led to Vaucanson becoming an Associated Mechanician in the Paris Academy of Sciences. [2, p. 601]. Vaucanson moved into investigating the mechanisation of silk weaving after the success of his automated duck. Work he completed in this field was later built upon by Joseph Marie Jacquard which led to one of the first examples of a programmable robot that came in 1804 in the form of the Jacquard loom [3, p. 30]. A machine that could be instructed which patterns were to be woven by using a chain of holed punch cards. The Jacquard loom is an interesting invention because not only was it a hallmark of early robotics by being able to perform a physical task and be programmable [4, p. 1] but it also provided one of the very first examples of a programmed computer system or computer assisted robotics.

Whilst these were early interpretations of robots, the term robot did not actually appear until 1920. From a Czech science fiction play write known as Karel Čapek in

his play Rossumovi Univerzální Roboti or by its English subtitle “Rossum’s Universal Robots” (RUR), the term robot was coined by his brother Josef, in the Czech language means serf labour but colloquially meant hard work or drudgery, an appropriate term considering how robots would be used in times to come. In the play RUR the robots were artificial people or androids who later rebelled against the human race. This play later turned out to be well before its time, where androids is now the official terminology used for describing human looking robots, at the time this was seen as pure fiction.

One of the most impactful science fiction authors to influence the field of robotics is Isaac Asimov. Asimov was born in the early 1920’s in Russia and later emigrated to America at a young age. He became a professor of biochemistry at the University of Boston whilst becoming an accomplished author. Asimov’s contribution to robotics largely came from his literary work, he is attributed to coining the term “robotics” in his short story called “Liar!” in 1941. Through the multitudes of books and short stories authored by Asimov, the field of robotics was expanded greatly as he offered through literary means a view point of the ethics and social issues that would arise through the developments of robot systems. Asimov is probably most famous for creating the three laws or robotics. In his short story “Runaround” written in 1942 the 3 laws were created;

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

These were seen as an important step for man as it would protect us from any harm by way of the robot. Even though these laws are still not needed today due to the level of independence robots have, they remain an important reminder or the caution that is needed as robots develop. These laws would later be adapted to include a fourth law. Although taking multiple revisions, the final version was adapted in Asimov’s “Foundation and Earth”. Considered the zeroth law it precedes all other laws and states;

0. A robot may not injure humanity, or, by inaction, allow humanity to come to harm.

These laws were widely popularised through the motion picture adaptation of Asimov's' story "I Robot" and would go on to influence the field for years to come. Asimov was a widely regarded as a visionary in the field. His work has led on to influence later science fiction. One such influence came where he coined the term positronic brains, this would become a famous example where the character Data, a cybertronic android created in Star Trek: The Next Generation, a show that would go on to inspire many thousands of children to work in the sciences. Asimov's work has arguably shaped perceptions of what robots are for generations of people.

Whilst Vaucanson, Jacquard, Čhapek and Asimov had important impacts in shaping the perceptions of robotics, the first patent filed to be considered a robot by modern standards would be for a mechanical arm with a gripper that was attached to a track. Created by George C. Devol, the patent was issues in 1961 [5]. This device's motions were controlled by patterns encoded on a magnetic drum and represented the first real robot that could be used for multiple tasks by way of different programmed sequences. Devol went and founded Unimation with Joseph Engelberger in 1956, this was the first company to offer robot systems as their product. Their first industrial robot was installed in 1961 at a General Motors factory in America. In 1968 Unimation Inc licensed the first robot to Kawasaki Heavy Industries which in turn produced the first Japanese industrial robot. Despite Japan already having a long running fascination with robots, all previous incarnations took the form of automata, similar to the Vaucanson's duck [6]. Devol and Engelberger's vision was to have millions of robots working in industrial settings. By looking at modern manufacturing you can clearly say that this vision has been realised, and today robots are used for such tasks as welding, painting, loading and unloading, electronic assembly [4, p. 2]. Unimation went through many changes in following years mainly by being bought out by multiple companies as they expanded. The technology and people from this company went on to be extremely influential in the field of robotics.

In 1969 Victor Scheinman at Stanford University invented the Stanford arm. This would be a defining point for robotics as it was the first all-electric 6 degree of freedom articulated robot arm that was controlled entirely by computer control [7]. Whilst not

truly anthropomorphic - as it contained five revolute joints and one prismatic, this arm is what popularised arm-type robots by proving its mobility and flexibility made it superior to other types of robot. This design was later sold to Unimation where it was further developed and deployed into manufacturing settings. At about the same time, in 1975 a Swiss company named Allmänna Svenska Elektriska Aktiebolaget (ASEA) developed the ASEA IRB, another fully electrically driven robot. It was a first in robotics because it had a microprocessor-controller built in that relied on Intel's first chipset. In 1988 ASEA went on to merge with another Swiss company, Brown, Boveri & Cie to form the ABB Group (ASEA Brown Boveri). ABB went on to establish themselves as a key player in the robotics arena and has been a Fortune 500 company for 24 years. In 1973 KUKA, another company involved in industrial automation unveiled the FAMULUS industrial robot [8]. A 6-axis revolute robot arm that would go on to revolutionise the automotive industry and to be the first example of one of the most popular robot types for years to come.

2.3 Automation

Robots from the 60s onwards have continued to play a major part in shaping manufacturing and pushing what is possible in terms of automation. Automation being the technology which takes a process or procedure and creates a method of reproducing a task with minimal human intervention. The international Society of Automation defines automation as "the creation and application of technology to monitor and control the production and delivery of products and services" [9]. There are many areas where automation can be useful such as banking, laboratories, space, medical and at home which mean a vast array of technologies converge under this one topic. In manufacturing terms, the main purpose of automation is to increase productivity. With the start of the industrial revolution many jobs were replaced or assisted by new tools designed with the sole task of replacing a repetitive task. Jacques de Vaucanson's automated loom was a prime example of this. In modern times robotics and automation almost, fit hand in hand, particularly when it comes to industrial automation. Industry 4.0, representing the fourth industrial revolution has spurred a number of innovations since its conception. The Robotic Industries Association expects most advances in automation to be related to connectivity and communication, software architectures and security as increased internet connectivity permeates manufacturing [10]. The

International Federation of Robotics released a press release in 2020 discussing the explosion of robotics in industry, expecting two million new robots to enter the manufacturing arena between 2020-2022, the advances in Industry 4.0 mean robots are getting smarter and more connected [11]. With the increase of connectivity, the use of sensors is greatly increasing. By reducing computation time of the robot by linking it to more powerful systems, more advances sensors can be used in conjunction with the robot. Finally, versatility is a huge challenge for the industry, current limitations from a cost standpoint come with paying for an expensive robot that is suited to a singular task. Through the use of multi end-effector setups and creative programming, robots are being adapted to fit into different manufacturing scenarios.

Within industrial automation lays three distinct fields of automation, fixed, programmable and flexible. Fixed automation, also known as hard automation is when a production line is fixed in the sense that the automation is completed by machines that use cams, gears and hard-wired machinery that cannot be easily changed from making one product to another. They tend to have high initial investment costs and high production volume. Programmable automation operates in the sense of being able to manufacture multiple batches of different products but only manufacturing one type at a time. This can range from a dozen to several thousand products. At the end of the production run the equipment is reprogrammed for a new product. This means the production run is followed by a period of non-productive time, followed by a new batch. This sort of automation comes with a generally lower production rate when compared to fixed automation. Finally, flexible automation comes as an extension of programmable automation, however with flexible automation the factory setup is limited to a smaller set of products so the change over time is smaller than programmable due to the defined nature of each product [12], [13].

2.4 Types of automation robots

2.4.1 Welding Robots

Robot welding has become almost ubiquitous in manufacturing. Offering greater precision than human counterparts, having superior repeatability enabling increased production throughput as well as being unable to tire, robot welding was partly responsible for the increased uptake in automation during the 1980s. Some of the most common types of welding consist of Arc, Resistance, Spot, TIG, MIG, Laser and

Plasma welding [14]. Whilst welding is a long-established manufacturing method of assembly, the development of robot welding has been focused mainly on optimisation of the existing process. An example of this optimisation is by pushing an automated welding system more into the category of autonomous welding. Where most welding systems are created through a teach and playback method, where an operator runs through a weld manually whilst recording all the steps so the robot can re-play the process, it would be a more efficient system if the robot was able to detect the seam that needed welding and to weld it automatically without any human intervention. An example of modern development in this field, Yuan Li et all presented a defect detection system for weld beads [15] where they designed a new sensor capable of detecting defects that is attached to the robot. By increasing the robot's capability to see they have taken it another step towards being truly autonomous. Another example of increasing the vision of a welding robot can be seen in work by Kitti Suwanratchatamanee et al [16], tactile sensors were developed to give robots the ability to visualise 3D objects, specifically in the case of welding, giving the robot the ability to detect the edges of objects requiring welding. Utilising these new types of sensors and combining with Artificial Intelligence, Yunhe Feng discussed a method of processing multiple optical sensors using various neural networks [17].

In the following sub-section, hot gas welding is discussed specifically. As this topic is a relatively new application of robot automation it means there is a lack of existing literature so what is available should be noted.

2.4.1.1 Hot Gas Welding

Hot gas welding is a manual plastic welding process that is being used in current manufacturing processes in the manufacturing of industrial sized plastic piping. Developed in the mid-20th century, the process involves using a hot-gas torch to direct high temperature welding gas (usually air) towards a plastic weld rod that is inserted between another thermoplastic object and the target surface. Heating the materials to a temperature until they reach their softening temperature. The pressure of the weld rod creates a bonding effect between the two materials [18].

There are two types of hot-gas welding, both hand welding and speed welding are common in the industry. Whereas hand welding is a technique where the weld rod is applied directly to the weld joint and the pressure that is applied is controlled by hand.

Speed welding on the other hand has a nozzle specific design that combines the hot gas torch and welding rod apparatus together into a single unit. The benefits to this design are it allows the user a more controlled application of pressure. An example of this tool is shown in Figure 2-1. There are different designs of welder which include different shape weld rods, these come as round or triangular. Whereas hand welding is better suited to constrained areas or designs which contain complex geometry due to the design of the weld gun, speed welding is the opposite, it is better suited for simple joint design due to orientating the welder of that size. There are four parameters that are important to consider when hot gas welding, gas temperature and flow rate, pressure, travel speed and orientation of the tool.

Literature appears to be sparse on the subject of robotic hot gas welding however a few interesting points of note have been found. Valk Welding, a company based out of the Netherlands has developed their own hot gas welding end effector for the purpose of welding plastic tanks shown in Figure 2-2 [19]. Similarly, Eugen Riexinger GmbH & Co. KG have demonstrated a dual end effector setup utilising KUKA robots where the milling process and welding process are combined (Figure 2-3) using tools switching stations.

Figure 2-1 An example of a hot gas speed welding apparatus

Figure 2-2 Hot gas welding end effector

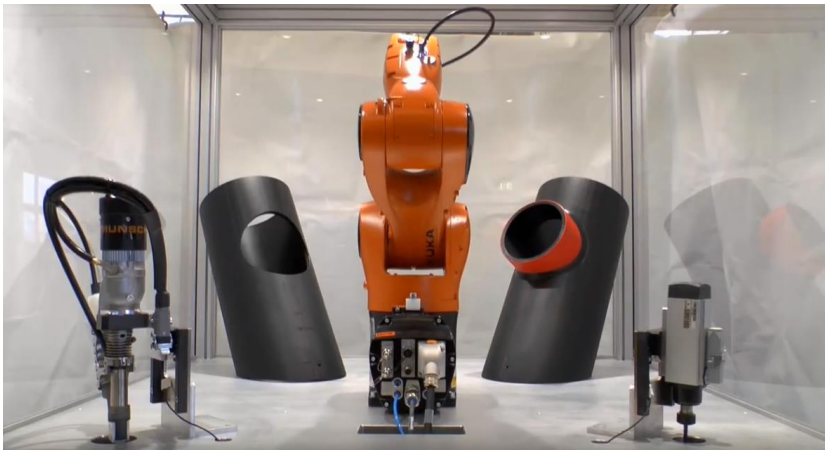


Figure 2-3 CNC & Welding end effector dual configuration [20]

2.4.2 Painting Robots

Industrial painting robots largely came to existence due to the automotive industry. Originally being very large an expensive, recent developments in low cost robotics has resulted in rather unique applications being developed that are not constrained the manufacturing domain as it largely was during the 80s and 90s. Painting applicators have now moved away from being a role suited best by articulated robot arms and become more available for other systems such as UAVs [21]. Within manufacturing robotic painting has its advantages due to increased quality and efficiency of the sprayer as well as a wider range of operation all whilst removing a human operator from working with hazardous chemicals. Despite being used within industry for many years

there is still the challenges to overcome when painting very large objects. Meng Z et all proposes a positioning system to overcome working with extremely large surfaces where a robot is incapable of maintaining position and location within the ship painting industry [22] while Chen et all discuss the problems of compensating for distortions when working with large scale structures [23]. Another area of research is on how to achieve different types of paint effects. The paper presented by Helou et al discusses using deep learning to generate appropriate spray commands to create textured painted surfaces [24]. Within manufacturing, development has mainly focused around optimisation, From et al who discusses optimal paint gun orientations to maintain constant velocities during the paint process [25] and Yu proposes an algorithm to efficiently control the overlaps on a painting task [26].

2.4.3 Pick and Place Robots

Pick and place robots are a staple of the manufacturing sector. They utilise a variety of grippers to pick and place material. Some examples are bag grippers used to work with soft materials such as sand, soils and absorbent materials, magnetic grippers used to work with heavier steel products, suction grippers which utilise vacuums and cups typically used to manipulate hard and flat products such as boxes and claw grippers which use two of more fingers to grasp objects and move them, typically used for moving boxes and packing materials. Whilst these grippers are usually used with articulated robots, the use of delta robots is also prolific within the industry because of their ability to pick, place and sort at extremely high speeds. These are particularly useful for small objects such as medication and food. Companies such as Ocado have pushed the usage of pick and place robots and have expanded their usage into the grocery sector [27]. Advances in pick and place robots in recent years has come from reduced motor size while maintaining or increasing payload capacity, as well as improvements in motion control software and hardware. An example of improved hardware can be seen where Ngadimin et al proposes a new design of pick and place robot to overcome issues rating to the size of components on surface mounted technology [28] and Zhang et al proposes a design of high speed sorting using vision systems [29]. Whereas He at al discusses optimisation of existing pick and place by optimising picking points by adjusting robot joint rotations and conveyor speed [30]. The company Amazon has been particularly keen in the pick and place sector. For

several years they run the Amazon Picking Challenge (APC) to promote development in the field and to obviously benefit their business. In 2017, what seems to have been the final year of the competition, an Australian team led by Peter Corke a renowned robotics expert, won the challenge by creating the Cartman robot, a Cartesian based robot that outperformed several other types of robot [31]. Despite the competition ending, work created through the project has gone on to influence further development. Zeng et al [32, p.] competed in APC in 2016, whilst they placed third and fourth in the competition, they developed a framework to estimate 6D poses which has gone on to be referenced by over 50 other papers. On top of this they have provided a high-quality dataset of images and scenes for future development.

2.4.4 Assembly Robots

Assembly robots are a staple type of robot in manufacturing. These robots position, fit, mates and assembles components or parts and are one of the most repetitive reducing devices in manufacturing. They increase productivity and quality while having a high return on investment. Despite assembly robots being so common they are among the most complicated type of robots to develop due to the extremely high accuracy required when positioning parts and the force of the parts being connected also needs to be considered. Recent trends show assembly robots moving more towards the collaborative robots rather than industrial robots. While industrial robots require a caged cell for safety, they offer greater payload weights and with the dual arm robots on the rise and stronger motors being utilised collaborative robots (Cobots) are making an ever-increasing contribution to the sector. Cobots have the advantage of not requiring a safety cell, in part because they do not have the payload compacity of their industrial counterparts but also because of the various force sensors built in which halt robot activity when a collision is detected. Developments have continued in this area to try and increase the safety of these cobots. Hakozaki et all proposed a robotic skin in 1999 by producing a sensor chip that detects electrical current, removing the need for wires which are embedded within a flexible skin like material [33]. Development continued and Yamada et al in 2005 discussed the development of a robot skin capable of accurately sensing the location of objects in area contact with the skins surface [34]. Due to the higher levels of safety, cobots often lower costs by not requiring the expensive cells which are typically required when operating articulated robots. The

removal of this cell means they can be used on assembly lines working side by side with humans and removing undesirable tasks such as screw fastening or box packing [35]–[37]

2.5 Future trends in robots

There are a myriad of developments happening in robots at an extremely fast pace. Just like modern televisions or mobile phones new models are being manufactured and marketed frequently. With such a large field that is continuously expanding there are a few emerging technologies that will be particularly impactful:

- Human Robot Interaction
- Underwater robots
- Multi-robot Coordination
- Legged Mobility

Each of these sections is elaborated on further in the proceeding sections.

2.5.1 Human Robot Interaction

Human Robot Interaction (HRI) is a field or robotics dedicated to the design, evaluation and understanding of interactions between a human and a robot. Generally, this study is split into two field, remote and proximate interactions. The former is where the robot and human are separated spatially or even temporally and the latter where the robot and human are co-located [38]. Examples of HRI have been mentioned in previous sections. Cobotics takes aim at the proximate interactions with humans, the task of making a robot operate in a safe manner that does not risk harming the human operator. As robotics appear into everyday life the ability to interact with these devices is paramount. Whilst society is not quite at the time when the four laws of robotics are required, the ability to work safely around robots is becoming ever more paramount. It is not only the safety of the operator that needs to be considered. Lamy et al discuss in their paper [39] a novel concept that has not been widely considered, this being how to control the amplified forces that a robot is capable of performing compared to a human. This being useful because whilst development towards cobotics is moving rapidly, humans are still much better equipped to evaluate and adapt to new unstructured condition. This work goes towards enabling operators to control robots that exert much greater force to specific tasks.

2.5.2 Underwater robots

Robots that do not often get much publicity are underwater robots or autonomous underwater vehicles (AUV). Over 363 square kilometres or 72% of the Earth's surface is covered by ocean. In 2017 nearly 2.4 billion people lived with 100 km of the ocean representing around 40 percent of the world's population [40]. Due climate change, rising sea levels and concern over the ecosystem of the oceans, AUVs have become an increasing interest due to their ability to withstand higher pressures that submarines and to be able to go where no one has explored before. The first AUV was developed by the Applied Physics Laboratory at the University of Washington known as the SPURV or Self-propelled underwater research vehicle. The SPURV was capable of diving over 10,000 feet and had an operation of time of about four hours [41]. In 2000, Yuh conducted a survey and found that at the time there was over 46 types of AUVs available for such purposes [42]. In recent years with the developments in autonomous navigation and sensor development AUVs have been used for mapping sea floors in 3D [43]–[45], monitoring sea ice [46] and monitoring sea life [47], [48].

2.5.3 Multi-robot Coordination

Multi-robot coordination or swarm robotics is a field of robotics that focuses on coordinating robot movements/actions to achieve joint goals or to enable individual tasks without collisions with other robots. This is a growing field in robotics because of the realisation that multiple robots can achieve a goal better than an individual robot. In addition to this there is a need to coordinate multiple robots working together. Examples of multi robot coordination can been seen in modern warehouses at companies such as Amazon and Ocado [49], [50]. These companies have leveraged the technology to stream line product picking and transportation to increase productivity. This sort of technology is by no means limited to warehouses, and is increasingly being looked at in the search and rescue fields [51], [52], mapping [53], [54] and construction [55], [56].

2.5.4 Legged Mobility

Whilst robots are replacing many tasks constrained to humans, humans still tend to have the advantage in many areas due to our flexibility and mobility. Robot arms have been

with us for many years but its usefulness is limited to its ability to reach its target. That is why a huge amount of effort has gone into developing biped and quadruped robots and in recent years a huge advancement has been made. When thinking of quadruped robots one cannot help but think of Boston Dynamics SPOT [57], a 4 legged robot "dog" that is capable of navigating an environment autonomously. There are many other actors coming to market that offer similar capabilities. Oxford University has been testing a quadruped robot for remote inspection of industrial sites [58]. Whilst a lot of progress has gone towards these types of robots, future developments are needed to develop their ability to navigate unstructured environments with rough terrain.

2.6 Trends in Automation

With Industry 4.0 being considered as the next industrial revolution, there are many areas of industrial automation that are going to see huge advancements in coming years. Some areas of particular interest are Machine learning/Artificial intelligence, smart factories, virtual/augmented reality, flexible manufacturing and cloud computing.

2.6.1 Machine learning

Machine learning is the study of computer algorithms that have the ability to improve itself based on experience it has gained whilst running. It in itself is a subset of artificial intelligence. Machine learning uses neural networks that takes training data and then tries to make predictions or decisions without being explicitly programmed to do so. Giving points to correct decisions a machine learning algorithm can be left to teach itself how to do a job better than previously. An example of machine learning can be found in email filtering, the algorithm will learn from user input how to identify spam emails, and then adapts itself to filter this content. Google has been using machine learning through bot identification verification to teach its AI how to identify common day to day objects in images, such as traffic lights, crossings, buildings. This training data has then gone into improving their self-driving cars, giving cars the ability to identify these common objects through the various array of sensors. Google have also been using robots to create their data sets. The TossingBot has been developed to create datasets on how to toss unknown objects in an unstructured environment outside of the robots kinematic range [59], by using multiple robots, repeating the same process over and over, the robots are able to improve their understanding of throwing objects and

gradually improve their accuracy over time. Another example is where Tan et al use machine learning to train quadruped robots to walk using reward signals in a simulation which can then be transferred to real applications [60].

2.6.2 Smart Factories

Smart factories are a response to ever changing demand and increased pressure on manufacturing to operate at lower costs and to be more flexible in their offerings. The term smart factory means a highly digitised and connected environment, consisting of modular machinery and equipment that is able to improve itself through the process of automation and self-optimisation. These modules or parts of production can be connected via Internet of Things (IoT) devices or other types of integrated circuits which enable sensing, measurement, control and communication of everything that is happening through the manufacturing process. These smart factories can include everything from production, information and communication technologies and includes the potential to integrate across entire supply chains [61]. With this integration and monitoring of everything happening within a factory, huge amounts of data can be generated meaning the information networks need to be capable of keeping up with a high rate of transmission. Migration from traditional factories to smart does however pose a problem for many businesses given the large scale and systematic changes required. Sjödin et al has tried to address these issues by gathering data from five in-depth studies of factories already making this transition to identify key steps needed during implementation, and from this have offered a preliminary model for future smart factories to work from. This model revolves around three key principles, the digitisation of people, introducing agile processes and configuring modular technologies [62]. Part of this trend is to take on the big challenge of data transfer and energy usage. Luo et al takes a look at these issues and proposed a mobile and hierarchical data transmission architecture to integrate wired/wireless field bus networks and wireless networks by taking advantages already present in existing mobile intelligence such as automatic guided vehicles (AGVs) in smart factories and by doing so propose a novel data and materials delivery scheme to overcome these issues [63]. With the virtual becoming every closer to the real, IoT within smart factories have played a large part in paving the way towards the full realisation of Industry 4.0. Needing to be highly flexible with volume and customisation proposes its own problems. Shrouf proposes an architecture

for IoT based smart factories which defines the main characteristics of these factories with a focus on sustainability [64].

2.6.3 Virtual Reality

In the last 5 years Virtual Reality (VR) and Augmented Reality (AR) have penetrated [65] various sectors other than computer games and this includes manufacturing and robotics. Augmented reality in particular has become widespread because the technology works perfectly in tandem with modern mobile phones which are omnipresent [66]. VR on the other hand has been a bit more limited in its impact partly due to the technological requirements. In particular the processing and graphical power required to operate more complicated VR applications as held it back. Companies such as Oculus have worked to try and bridge the gap between AR and VR with their more mobile VR headsets where as companies such as Valve have double down on higher specification hardware. AR has become particularly useful in industry. Areas such as remote maintenance [67] and quality assurance are prime examples [68].

2.6.4 Flexible Manufacturing

Flexible Manufacturing Systems (FMS) is another area where automation is seeing increased focus in recent years. FMS is a method of production that is designed to be easily adaptable in what it manufactures and at what quantities. Compared to typical manufacturing where mass production of a single item sees a lower production cost, FMS is set up to manufacture a range of products as demand sees fit. Having this level of flexibility has a cost so FMS typically has a higher upfront cost due to having to purchase and install specialised equipment but can return a lower production cost over time. FMS is a part of the make-to-order strategy that allows a level of customisation for customers. FMS was developed by Jerome H. Lemelson who was an American industrial engineer and inventor. Lemelson had patented the concept in 1954 [69] of a robot-based system that could weld, rivet, convey and inspect manufactured goods. By the 1970s systems based from Lemelson’s design started hitting factory floors in the US and in Europe [70]. Flexible manufacturing systems have their advantages and disadvantages. The advantages come in the form of reduced manufacturing cost, lower cost per unit, greater labour productivity and improved quality among others, whereas the disadvantages are the initial cost for systems and planning, requiring skilled labour

to operate the systems and they can be complicated to run and maintain [71]. Whilst FMS systems are not new, the renewed attention is down to quickly changing markets and world events. With advances in technology and artificial intelligence and with technology associated to smart factories, FMS are growing ever more intelligent [72].

2.7 Automation and Robot Sensors

Within industry, automation sensors have a very important role to play. They allow systems to detect, analyse, measure and process changes in measurements such as position, height, length, appearance or displacement. There are many types of sensors within automation such as vision, ultrasonic, position, proximity, photoelectric and temperature sensors. Robotic sensors are used to evaluate the robot's environment and condition. They connect to the robot controller which is then responsible for processing the information generated and to enable an appropriate behaviour of the robot. Sensors in robots are typically considered to be akin to human sensory organs as they perform the same sorts of functions. Robots require extensive information about their environment to operate effectively and sensors supply this. Robots also typically have internal sensors responsible for monitoring the robots state. They are used to measure current position, velocity and acceleration of the robot's joints and end effector. These sensors consist of position and velocity sensors. Position sensors include an encoder which is a digital optical device that converts the joints movements into digital pulses, a potentiometer which provides a variable resistance in terms of voltage of linear or angular displacements, synchros which transforms angular position into an electric signal [73] and resolvers which are rotatory transformer used to measure degrees of rotation which provide position and speed feedback [74]. Finally, velocity sensors measure position over time by taking measurements at known intervals and then computing the change in position values.

Sensors are analogues to human senses but they also extend further to sense things that humans are incapable of. There are typically four main classifications of sensors followed by two sub classes of each. There are simple and complex touch sensors, the former sensing an objects presence or absence while the latter sensing size, shape or hardness. Simple and complex force sensors, simply measuring forces along a single axis while complex sensing over multiple. In a similar fashion there are simple and complex vision sensors. Again, the former sensing edges, holes and corners and the

latter being capable of recognising objects. Last there are proximity sensors - in a class of its own which is a non-contact sensor simply capable of detecting objects in front of it. These sensors can measure many different properties, for example object proximity such as presence, bearing, colour or physical orientation and object's coordinates in space. There are also sensors capable of detecting the presence or concentration of chemicals and ones capable of detecting sound attributers such as frequency or intensity [75] among others.

Without sensors like these, advanced automation would not exist. It is these sensors which replicate or extend human capabilities further that enable tasks to be automated. More so it is sensors like these that enable robot capabilities to extend further and further and the development of newer more advanced sensors that is going to push the development of robotics and automation further. When Microsoft released the Xbox Kinect in 2010 [76] and new wave of vision sensing technology emerged as a result. The ability to detect RGB colours as well as depth (RGB-D sensors) allowed many advancements in robotic vision systems. Lai et al presented in 2011 a large dataset of classified images for researchers to develop better object recognition, manipulation, navigation and interaction capabilities [77]. Lai et al's work went on to contribute towards many further development in the field such as Boubou et al developing an adaptive filter for creating improved 3D data captured using the Kinect [78].

In the following section a more detailed review is undertaken of photo-electric sensors. Due to their uniqueness of their application in the system being developed, a dedicated sub-section is a provided detailing basic mechanics and application.

2.7.1 Photo-electric Sensors

A photoelectric sensor works by emitting a light beam which can be visible or infrared from a light-emitting element and a reflective-type photoelectric sensor is used in conjunction to detect the light being emitted. These can be used to discover the distance, absence or presence of an object [79] by using the transmitted light. Commonly used in industrial manufacturing there are three different types available. Opposed or through-beam, retro-reflective and proximity-sensing or defused. Both through-beam and retro-reflective work as partner sensors where one sensor emits the light and the other as the reflector which detects the light. Diffuse on the other hand is a single unit with both the emitter and reflector combined which then relies on the emitted light being reflected off

the target surface. There are advantages and disadvantages to each type of sensor, these being; Through-beam sensors are the most accurate of this type of sensor with the longest range and are considered very reliable, however the must be installed at two points on the system, one side being the emitter and the other being the receiver. Reflective offer slightly less accuracy compared to through-beam however, their range extends further than diffuse sensors. They too are also considered very reliable. The disadvantages come in a similar fashion as through-beam, they must be installed at two points, they are of a higher cost than diffuse and their range is lesser than through-beam. Diffuse sensors are slightly different to the former two sensors, they install at one point by combining the emitter and reflector in to a single unit and they also come at much lower costs. The disadvantage is that they are less accurate and require more time to setup as they require calibration to the target surface [80]. Photoelectric sensors are used in varying manufacturing environments, from the automotive industry to material handling to food and beverage, these sorts of sensors are used for object detection of parts or containers. When considering a type of sensor for a robot position adaption system, typically photo-electric diffuse sensors would not be considered because it is not their typical area of application. Currently there is no literature covering their use in the position adaption application which presents an opportunity to produce something novel. This thesis will go some way in answering whether these sensors are viable in this configuration.

2.8 KUKA Robotics

KUKA is a German manufacturer owned by the Chinese company Midea Group who are one of the world's leading suppliers of intelligent automation solutions. KUKA Systems GMbH is a division of KUKA whom are an international supplier of automated manufacturing solutions who have systems and equipment being used by many companies such as BMW, GM, Ford, Airbus, Siemens and many others [81]. KUKA Robotics offer a very broad range of different types of robots. Ranging from 3 kg to 1000 kg payloads and industrial robots to medical cobots. Their systems are based on an open architecture PC-based controller and as a result KUKA are the number one PC-controller robot manufacturer in the world. KUKA controllers are suitable for major customisation meaning they can integrate custom components or integrate with existing automation systems. Not limited to hardware they also offer various software systems

such as KUKA SIM which allows for virtual design and testing of complete robot systems. In addition KUKA offer a Systems Partner program giving availability of experts in key industries which mean they can offer technologies that can transform a generic KUKA robot into an application specific solution [82]. While predominantly in the automotive industry [83], KUKA operate in many other areas such as the medical field [84], manufacturing [85], construction and food and beverage industries [81]. KUKA also play a huge roll within the research and science fields assisting researchers and developing educational tools. The KUKA youBot is an example this [86], developed as an opensource educational tool that meets the requirements of industrial requirements as well as educational and research but most of all comes at an affordable price point.

2.8.1 Tool Centre Point

Within manufacturing there are many areas where manual labour outperforms robot systems, this is usually due to a human's exceptional ability to adapt to a change in circumstances. With a focus on specific tasks such as plastic welding industrial pipes, this research considers sensor aided positioning of a TCP. A robot's TCP is vital to any operation being completed by a robot. It provides information to the robot controller about where the tip of the tool attached to the end of a robot is located. As this part of the robot typically interacts with an object or surface, it is vital for the robot to know where it is in relation to the flange of the robot's arm. Figure 2-4 shows an example of a TCP, in this case the robot is configured with a gripper holding a pencil, and the tip of the pencil is configured as the TCP. Using this information, the position of the TCP is translated into rotary motions of the robot individual axis so the tip of the tool stays on the defined path specified in the robot program. Whilst this is the goal of the research, the following sections reviews literature where TCP control in manufacturing is critical.

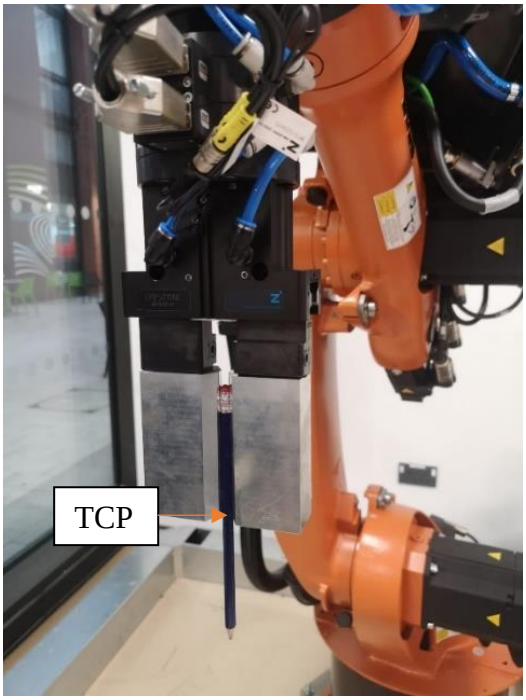


Figure 2-4 TCP Example

2.8.2 Type of KUKA robot movement

When programming an articulated robot there are a set of three different movement types available to the user as a base, Point-to-point (PTP), linear (LIN) and circular (CIRC). Point-to-point moves the tool centre point of the end effector from point A to point B by the shortest path possible (Figure 2-5). This does not mean the path is a straight line as it may be quicker for the robot to move between points by taking a more indirect route. The reason for this is because when moving between the two points, all 6 joints of the robot need to move as well, a PTP movement takes this in to consideration and tries to move each joint as little as possible by achieving the desired motion. A linear movement on the other hand is a straight-line movement between point A and point B (Figure 2-6). This movement takes longer because each joint needs to move further. Then there is the circular movement, this movement as described moves the TCP in an arc movement allowing a circular path to be drawn. A CIRC movement requires 3 points to be completed, a start and end point and an auxiliary point (Figure 2-7). The auxiliary point determines the size of the circular arc between the start and end points. Building off these movements there is the addition of spline movements. These movements are particularly useful for complex movements such as curved paths. Whilst curved paths can be completed using typical PTP, LIN and CIRC movements,

splines have their advantages. When generating the path with PTP, LIN and CIRC commands points are approximated, meaning not all defined points on the desired path are met. Spline movements do not have this issue, they follow generated points along the entire path and can work at a maintained velocity better than other movement types. The path always remains the same and is irrespective of any over settings used. When spline movements are generated, the robot controller takes the physical limits of the robot into considering and the robot moves as fast as possible within the constrains programmed by the operator. With the other basic movements, the physical limits are the robot are not considered until the movement is in motion which may result in the robot ceasing motion due to these limits breaching. In more recent iterations of KUKA robots they have introduced another set of commands to compliment spline movements, SPTP, SLIN and SCIRC commands. These commands work on the same principle as spline in the sense that they are more accurate and calculated ahead of time [87, pp. 159–160].

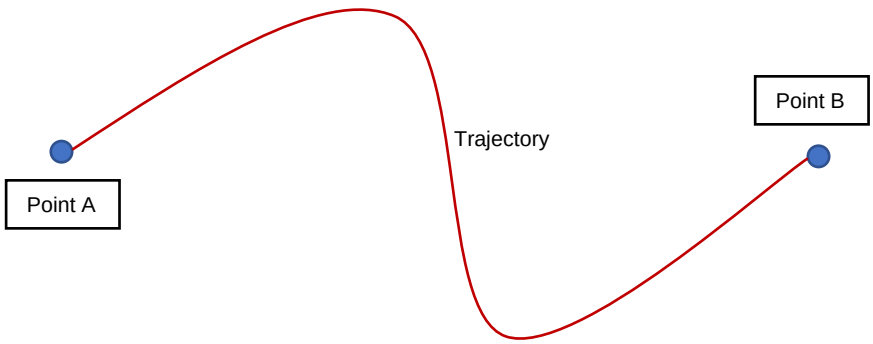


Figure 2-5 PTP Movement

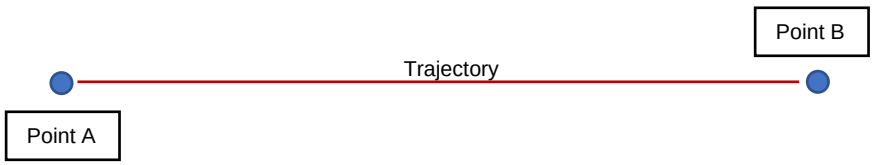


Figure 2-6 LIN Movement

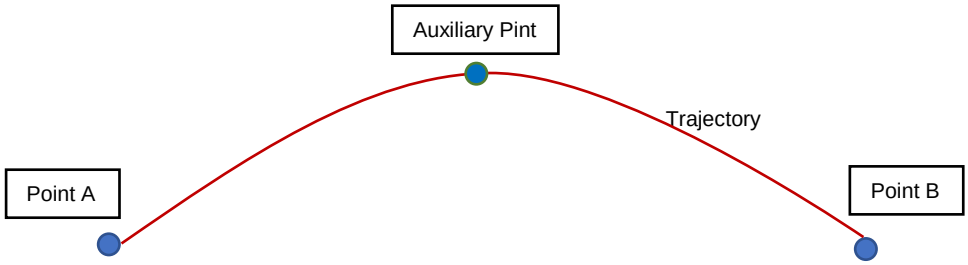


Figure 2-7 CIRC Movement

Even though robot movement types have been well established for many years, optimisation of these paths is very important to reduce cycle times in the industrial process. Even the smallest of time savings accumulate over time and can make a huge impact of production costs. Sven Severin and Juergen Rossmann discuss this optimisation in an environment where collisions are possible. When a robot needs to avoid an obstacle, intermediate points are needed, however when this is the case the robot will lose momentum moving between point A as it gets to point B and onward to point C. In their paper they compare three different metaheuristics to find where points B position should be located to reduce time lost and to maintain momentum [88]. Further examples of optimisation can be found in an in an article by Cooper et al. In their paper six degree of freedom robots have been considered from surface mount assembly of electronic components where these types of robot are not typically used [89]. The approach used here was using a genetic algorithm to optimise component placement ordering to decrease cycle times. In their paper they compare the various different movement types not just PTP movements. This shows that whilst the movement of the robot itself can be optimised, so can the positioning of objects the robot is interacting with.

2.8.3 KUKA Hardware Interfacing

Hardware interfaces involve connecting other systems or technologies to a robot system. For example, connecting a personal computer to a robot controller to expand control over the connecting robot. To do this KUKA offer a variety of different methods for interfacing such as Process Field Bus (PROFIBUS) and Ethernet. PROFIBUS is a standard for fieldbus communication for automation technology that was promoted by the German Department of Education and Research (BMBF) in 1989 [90]. It was from the combined work of the German government, twenty-one businesses and industry

leaders at the time which became the Central Association for the Electrical Industry (ZVEI) that created what still stands as leading solution for hardware interfacing particularly between European customers. It was only in 1993 that PROFIBUS was considered completed, it was when PROFIBUS DP (Decentralised Periphery) standard was introduced that allowed easier configuration and faster messaging [91]. The Ethernet interface on the other hand is a simpler interface in the sense that it is a standard communication interface used on the majority of computers all over the world. This software comes in different versions such as KUKA Ethernet KRL [92], KUKA Ethernet RSI XML [93] and KUKA Robot Sensor Interface [94]. Each of these software packages is an extension on the previous and represent the evolution of Ethernet control for KUKA robots. They work on the principle of exchange XML messages that can be configured on the controller via configuration files. The Ethernet interface of KUKA robots has allowed a variety of different research to be conducted using their devices, adding KUKA's commitment to development in the field of robotics.

2.8.4 Adaptive control

Adaptive control is a method where by a controller has the ability to adapt a controlled system with parameters that can very or ones that can be uncertain [95]. Adaptive control has for a long time been seen as an effective system for robot manipulator controller design due to its ability to deal with unknown certainties of robot dynamics models. The design of the controller for a serial manipulator contains two parts, an end effector path or trajectory that is first specified which generates a set of joint angles to achieve a desired path and then a second part to calculate the required torque that each joint needs to apply to achieve the required motion. The torque can be calculated based on inverse dynamic equations. Due to robot systems being extremely nonlinear, controlling the manipulator can be complex. When a robot mechanism is in motion the variables controlling each joint change and this is what makes the dynamic equations alter throughout the robot movement. This is why robots use Model Reference Adaptive Control (MRAC) as it has the ability to take these changes in to account. Traditional controller techniques generally are not used for robotics due to their poor performance at high speed, PID control systems may not provide stability or optimal control for the system either [96], this is why MRAC is one of the most prevalent methodologies used today [97]. KUKA robot controllers handle this adaptive control internally but expose

various options to build upon it. The KUKA RSI and Ethernet XML software packages expose certain parameters of the robot and allow manipulation to be extended from this.

2.9 Robot Data Acquisition

An important part of working with robots particularly as the world moves into Industry 4.0 is the ability to learn from experience. This can be in the form of neural networks and machine learning but also in the form of old fashion research. This means the ability to acquire data from robot and robot sensors in paramount. There are no real standards when it comes to this partly due to the sheer breadth of robot and sensor systems available. There has been some attempts to develop frameworks to achieve this [98]. There are two types of data acquisition that is typical seen, real time and offline. Real time usually relates to reading live sensory data as its generated such as GPS data and offline is where data is recorded on to a device and then analysed post completion of the desired task. A simple example of data acquisition can be found in Lego Mindstorms products [99] which is used when teaching data acquisition for a low cost. Other examples can be found from various research papers on the topic. Zhou discusses using C++ programming language to access sensory data of a mobile autonomous robot [100], Bryant and Gandhi discuss a real time data acquisition using LabVIEW software and Sarma and Bezboruah discuss using low cost Arduino UNO to interface with analogue sensors for data acquisition. Essentially data acquisition relies on the systems being used and their capability to be extended.

2.10 Summary

This literature review covers a wide range of topics with the purpose of giving the reader a complete introduction to robotics. Starting with its history the evolution of the robotics is discussed from the Vaucanson's duck to the modern articulated robot. Then the history of automation is reviewed culminating in an overview of Industry 4.0 – the latest industrial revolution. Once a firm foundation is built, a more specific investigation into industrial robotic automation is completed. An overview of four different types of robot systems prevalent in the manufacturing sector is discussed. These types of robots being welding pick and place, painting and assembly. A particular emphasis has been placed on hot gas welding in this section to highlight a lack of literature in the field. Next future trends are reviewed to understand where future robot

and automation development is heading to see if any topics intersects with research undertaken in this thesis.

The final three sections of the literature review focus on understanding the requirements of the hardware and software used in this project. A general overview of different types of sensors used within the manufacturing industry is presented with a dedicated sub-section for discussing the photo-electric sensors used in this project. The reason for having a dedicated sub-section is to communicate the uniqueness of the application of these sensors in this project. Where photo-electric sensors are typically used with pick and place robots, to use them as sensors for position adaption is novel and currently has no scientific research available addressing their use in the context of this thesis. Next an understanding of the KUKA robot systems is presented focusing on the KUKA KR16 used for experimentation. Certain aspects important to the project are discussed such us the movements types the robot is capable of completing and the hardware interfaces available. Finally, robot data acquisition is reviewed. The ability to accurately measure and analyse the robot movements during experimentation is important to the success of the research. Whilst there are many ways to record robot data, there is no standardised set of tools available and bespoke approaches are the typical norm. This presented an opportunity for development, specifically when working with KUKA robots. The next chapter takes the information garnered from this literature review and applies it to develop the experimental setup for this thesis.

3. EXPERIMENTAL SETUP

3.1 Introduction

This chapter discusses the lab setup, robot configuration and resources used during the formulation of this thesis. Initially the laboratory and workspace available is investigated, followed by and introduction to the robot being used along with its configuration. Then the Ethernet interface and robot communication are discussed detailing the configuration steps required. Next the Robot Sensor Interface software is detailed along with the RSI configuration used initially during experimentation. Finally, the sensor setup and communication is discussed along with the hardware components used to give the reader a complete guide for replicating the experiments discussed in later chapters.

3.2 Workspace configuration

During this research a robot cell equipped with a KUKA KR16-2 has been used. The cell is 3 x 3m in size and the robot is configured with software limits that are set just short of the surrounding cell walls (Figure 3-1). Multiple safety stops are located around the cell, two inside the cell, one external of the cell and one located directly on the tech pendent.

Figure 3-1 Robot Cell

3.3 KUKA KR16-2

The KUKA KR16-2 shown in Figure 3-2 is a versatile robot and well suited for the purposes of this thesis. An articulated robot with six axes, the KR16-2 has a payload of 16 kg which has been designed and manufactured by KUKA Robotics. Flexible with strength, the KR16 is constructed with light-weight alloys and designed to be space saving and cost effective. The KR16-2 is commonly found in the automotive components industry and various manufacturing sectors. The robot has a repeatability factor of ±0.05 mm making it well suited for position-oriented tasks. In Table 3-1 the main specification for the robot can be found followed by the range of motion available for each axis and each axis top motion speed. The robot has an affective reach of 2.412 meters making it ideal for the range of testing envisaged through this project. In Figure 3-3 the full working area of the robot is visualised, followed by the set of measurements for each position in Table 3-2. The KUKA KR16-2 is equipped with a gripper. Specifically, a two jaw parallel gripper manufactured by Zimmer [101] shown in Figure 3-4. The gripper is a useful tool, typically used for picking and placing but in the scope of this project it allows quick changes between sensor mounts and assemblies.

Figure 3-2 A KUKA KR 16-2 Robot

Table 3-1 KUKA KR16-2 Main specification

Maximum Robot Load	16 kg	
Number of axis	6	
Maximum horizontal reach	1611 mm	
Repeatability	±0.05 mm	
Controller	KR C4	
Axis information	**Range of motion**	**Robot motion speed in °/s (16 kg Payload)**
Axis 1	± 185°	156°
Axis 2	+35° / -155°	156°
Axis 3	+154° / -130°	156°
Axis 4	± 350°	330°
Axis 5	± 130°	330°
Axis 6	± 350°	615°

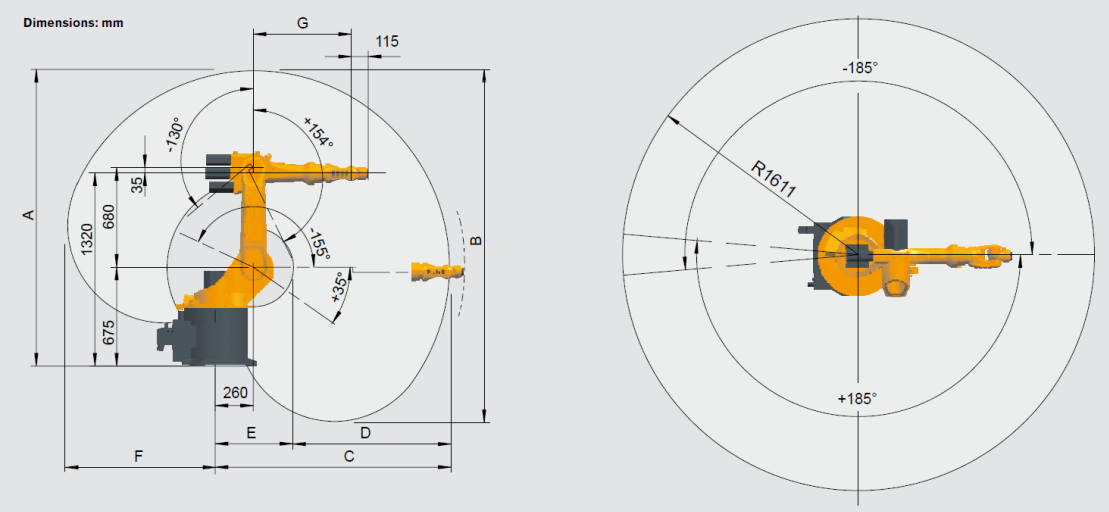


Figure 3-3 KR16-2 Work Envelope Visualised [102, p. 4]

Table 3-2 Work envelope figures

Work Envelope	KR16-2
A	2026 mm
B	2412 mm
C	1611 mm
D	1081 mm

E	530 mm
F	1027 mm
G	670 mm
Volume	14.5 m^3

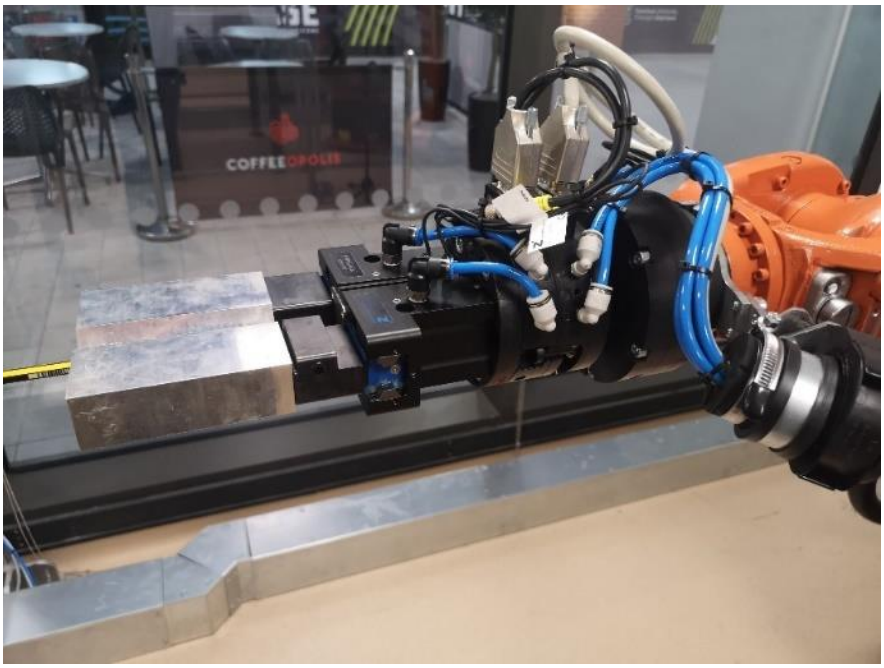


Figure 3-4 Zimmer Gripper

3.3.1 The KR16 Controller and Smart Pad Teach pendant

The KUKA C4 controller show in Figure 3-5 was a pioneering controller developed by KUKA robotics. Combining robot control, PLC control, motion control and safety control into a single unit means it is a common controller used within manufacturing where KUKA robots have been deployed. It has fast response times and options for expandability, coupled with the prolific use of KUKA robots in industry, mean it is the ideal candidate to be used in development of robot sensor systems [103]. The KUKA smartPAD Teach Pendant is a hand-held interface that gives the user complete control over the robot system. This pendant is used for the manual programming and adjustment of the KUKA robot. Its onscreen interface allows individual control over each axis along with the ability to switch between different coordinate frames. Coupled with a 3D mouse it gives a high level of control over the robot to the user. The smartPAD is quite intuitive by design and consists of an array of buttons and a touch screen display. Each of these buttons is described in Table 3-3 and the smartPAD itself can be seen in Figure 3-6.

Figure 3-5 KUKA C4 Robot Controller

Figure 3-6 KUKA Teach Pendant Front View

Table 3-3 KUKA smartPAD descriptions [87]

Item	**Description**
1	Button for disconnecting the smartPAD
2	Key switch for calling the connection manager. The switch can only be turned if the key is inserted. The operating mode can be changed by using the connection manager.
3	EMERGENCY STOP button. Stops the robot in hazardous situations. The EMERGENCY STOP button locks itself in place when it is pressed.
4	Space Mouse: For moving the robot manually.
5	Jog keys: For moving the robot manually.
6	Key for setting the program override
7	Key for setting the jog override
8	Main menu key: Shows the menu items on the smartHMI
9	Status keys. The status keys are used primarily for setting parameters in technology packages. Their exact function depends on the technology packages installed.
10	Start key: The Start key is used to start a program.
11	Start backwards key: The Start backwards key is used to start a program backwards. The program is executed step by step.
12	STOP key: The STOP key is used to stop a program that is running.

13	Keyboard key Displays the keyboard. It is generally not necessary to press this key to display the keyboard, as the smartHMI detects when keyboard input is required and displays the keyboard automatically.

3.3.2 KUKA Robot Language

The KUKA Robot Language (KRL) is a proprietary language developed for non-collaborative KUKA systems to enable easier programming of their robots via the KUKA pendant. The language exposes various degrees of complexity to suit different skill levels when programming, from simple inline forms used through the teach pendant to fully typed sections of code via its development environment. A typical KRL program is shown in Figure 3-7 and is structured with a leading DEF statement and the name of the KRL program, followed by INI which contains any internal variables or parameters that are initialised at the start of the program. Next the main content of the program is contained, in this instance a home position is specified followed by two LIN commands ending with the home position again. Finalising the program is the END statement, terminating execution of the code.

The language exposes seven movement types for use with a KUKA robot, PTP, LIN, CIRC, SPTP, SLIN, SCIRC and Spline blocks. The three formers are legacy movements whereas the latter are Spline movements which are newer motions designed to be more accurate and efficient. The PTP and SPTP movement moves the TCP along the fastest route possible to reach its endpoint. The fastest path not necessarily being the shortest. LIN and SLIN movements produce a straight-line path that the TCP follows at a desired velocity to reach its endpoint. CIRC and SCIRC movements produce circular paths that the TCP follows. This movement is made up of three parameters, a start points, auxiliary point and end point. A Spline block is simply a grouping of spline motions which is calculated and executed by the robot controller as a single motion.

```
DEF example_program( )
INI

PTP HOME  VEL= 100 % DEFAULT

LIN P1 VEL=2 M/S CPDAT1 TOOL[1]:PENCIL BASE[0]
LIN P2 VEL=2 M/S CPDAT2 TOOL[1]:PENCIL BASE[0]

PTP HOME  VEL= 100 % DEFAULT

END
```

Figure 3-7 Example KRL program

3.4 Robot configuration

3.4.1 End effector

The robot is equipped with a Zimmer GPP5010 gripper shown in Figure 3-4[101]. The decision to work with a gripper rather than a dedicated end effector mount was influenced by the intended use of the final design. To this end the mount design was a secondary issue, but the sensor positioning was in the important part. This means working with a gripper would be quicker in terms of designing new mounts and manufacturing them through 3D printing.

3.4.2 Sensor mounts

During development various sensor positions were considered. The initial design of the mounts for these sensors worked on the principle of two sensors of opposite sides angled to trigger at the same position. As this system is intended to work by integrating into existing systems, with tools already on the market such as the hot gas welding device in Figure 2-2 different mounting positions needed to be analysed to factor into any further research. Figure 3-8 shows the initial concept idea, as a comparison the setup in Figure 3-9 was considered, finally Figure 3-10 shows a hybrid of the previous two designs creating a more compact design. In Figure 3-9, the second sensor setup has target areas parallel to each other. The reason for this is to test whether overlapping targets points creates confusion between the two sensors as to which is being triggered at a given time.

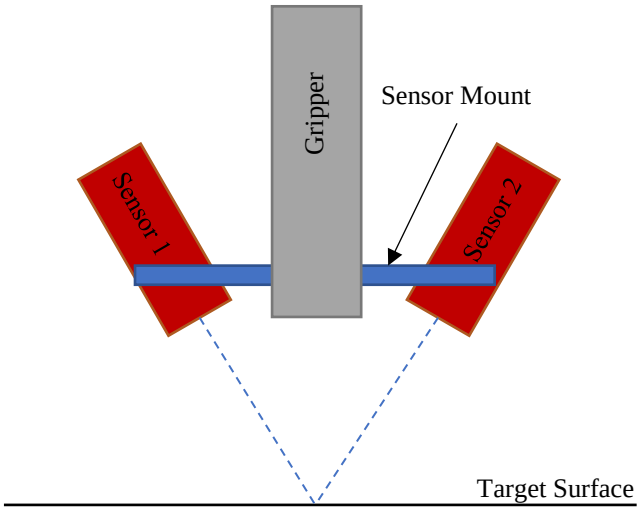


Figure 3-8 Proposed sensor configuration 1

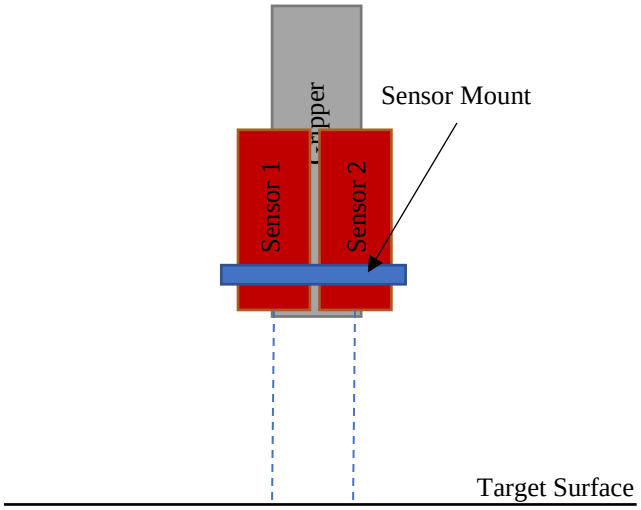


Figure 3-9 Proposed sensor configuration 2

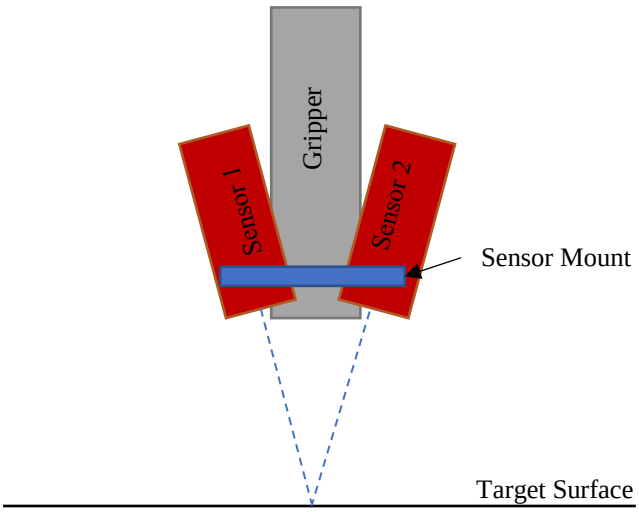


Figure 3-10 Proposed sensor configuration 3

3.5 Ethernet Communication

3.5.1 Controller Configuration

The RSI software relies on the User Datagram Protocol (UDP) over IP (Internet Protocol) communication to work. First the configuration of the RSI interface needs to be completed. An IP of 10.10.10.1 is assigned to this interface specifically to avoid any confusion with other interfaces on the robot as they maintain the 192.168 IP range. Warnings to this is present on the configuration screen. To configure this, the network settings of the device are accessed through the teach pendant. From here a new network interface is added, dedicated to RSI as shown in Figure 3-11.

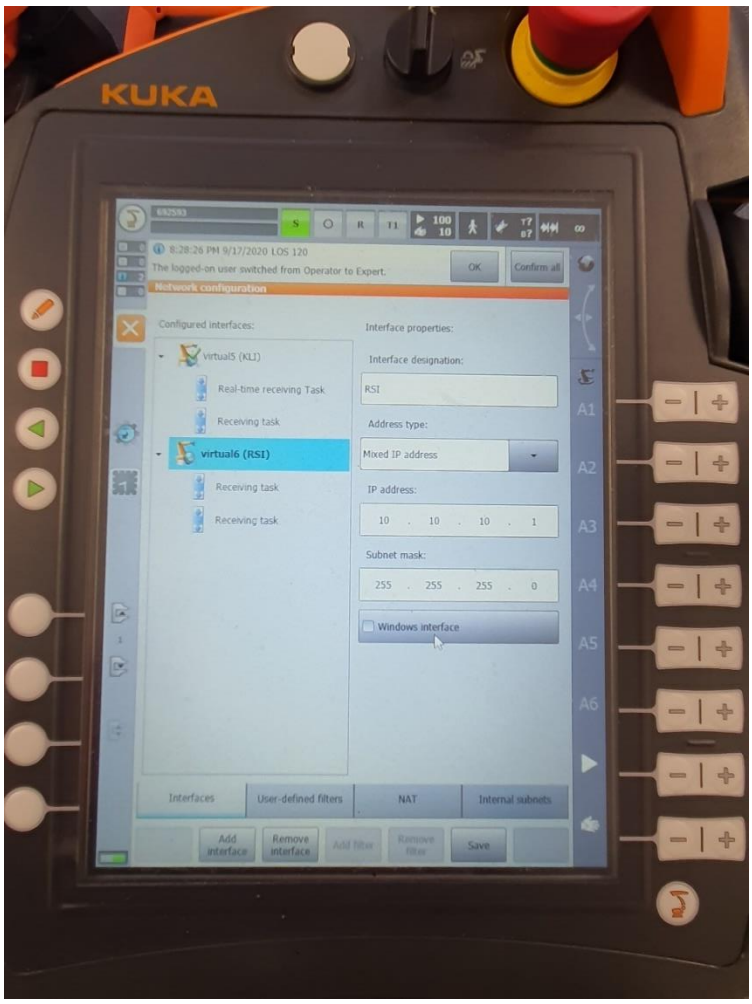


Figure 3-11 RSI network configuration

3.5.2 Client Configuration

Once the cabinet network interface has been configured, the client interface can be created. An CAT6 Ethernet cable is used to connect the laptop directly into the controller's X66 port shown in Figure 3-12, then a static IP address of "10.10.10.10" is assigned to the Ethernet interface of the laptop. A ping test is then used to test whether

communication has been established successfully. The RSI software's Ethernet objects works on the basis of sending a message and waiting for a reply. If a reply is not received within a specified timeframe the robot program will timeout resulting in the immediate stop of robot operation. If no change is expected in the robot then a reply of 0 values is returned by default. To test this configuration the RSI software comes with a test server interface shown in **Figure 3-13**. The server application has basic functionality, it allows configuration of the clients' network interface, allows the user to read incoming and outgoing XML messages and gives basic movements commands in the form of jogging the different robot axes.

Figure 3-12 KUKA C4 cabinet X66 port location

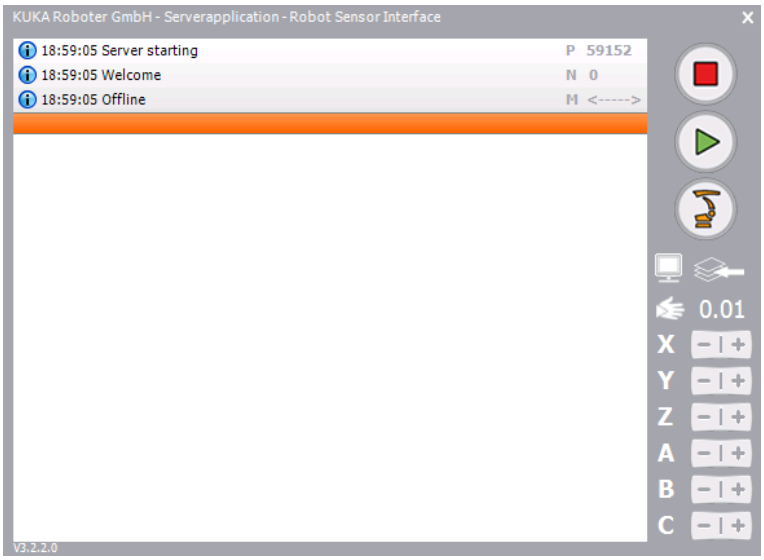


Figure 3-13 Robot Sensor Interface - Server Application

3.6 Robot Communication

One challenge to overcome is deciding on how to connect to the robot to influence its control. The following sections cover various products and research already completed into this field.

3.6.1 Robot Sensor Interface

KUKA Robot Sensor Interface (RSI) is a proprietary software package that expands the capabilities of a KUKA KR C4 Robot Controller giving it the ability to complete data exchange between a robot and sensor system via Ethernet or the Input/output system of the robot controller external factors to influence the motion of the robot or program execution. Configuration of the signal flow or RSI Context is done through the RSI Visual software package (Figure 3-14). RSI Visual provides access to a library of RSI Objects which are used to configure the signal flow.

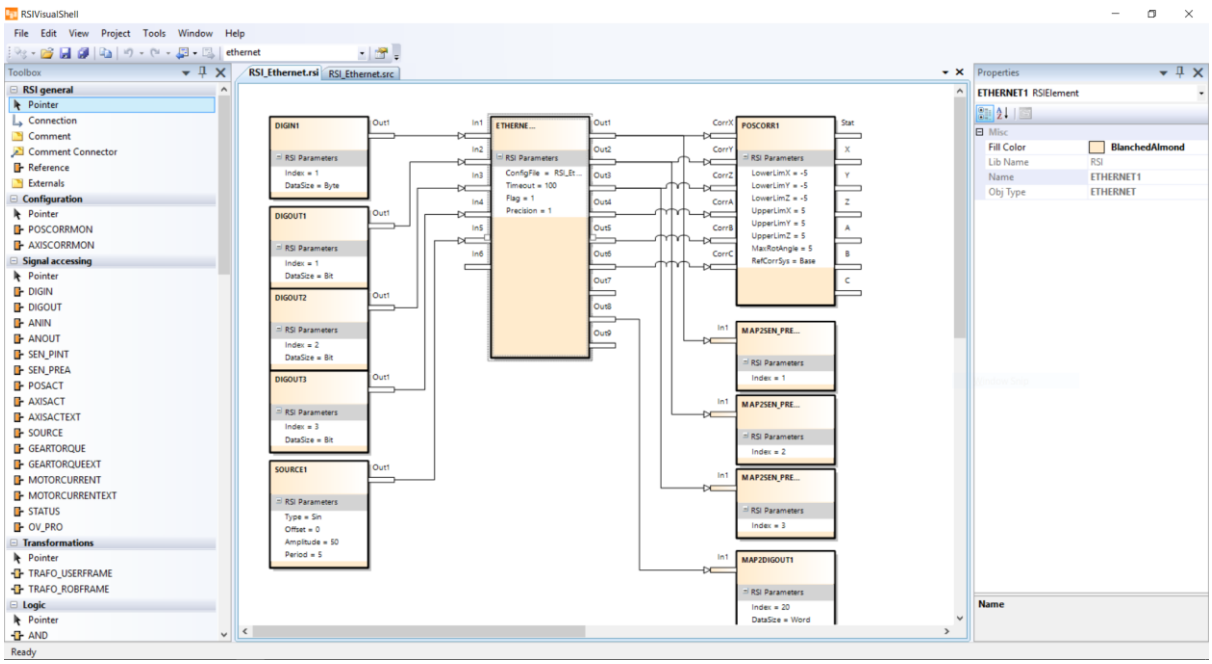


Figure 3-14 KUKA RSI Visual Software Package

3.6.1.1 Data communication

There are two options available for communicating with a robot controller adapted with KUKA RSI are via the I/O system or via Ethernet. Using the I/O system is a more direct method but working over Ethernet gives us much greater flexibility. Using a real-time capable network connection, data is transmitted via UDP/IP where no fixed data frame is specified. This must be configured in an XML file on the robot controller. Cyclical data transmissions from the robot controller to sensor system is run in parallel to the

robot program execution. For example, operating mode, position data and axis angles can be sent to the sensor system. The sensor system then again via cyclical data transmissions send information in parallel to program execution. UDP is a connectionless network protocol and is not reliable or secure. As it cannot be guaranteed that the packets sent arrive in a reliable manner it is up to the programmer to implement sufficient error correction measures to ensure proper operation. This could be checking that all packets have arrived correctly and re-requesting any that have failed.

3.6.1.2 Signal Processing

Signals are processed using RSI objects. An RSI object is essentially a function that has inputs and outputs like any other program (Figure 3-15). As talked about previously, RSI Visual provides a user with an extensive range of RSI objects via its library. A signal flow is constructed by stringing multiple RSI objects together via their inputs and outputs as demonstrated in Figure 3-16. Combined together these objects are known as the RSI context. Once an RSI context has been constructed using RSI Visual, the RSI context can be loaded in a KRL program and triggered to run in parallel. This allows the signals processed from sensory input and the KRL program to run concurrently and gives the user the ability to activate and deactivate as they see fit. The processing is calculated at the sensor cycle rate which depends of the selected mode. Either a rate of 12ms or 4ms where the former operates sensor mode #IPO and the latter operates sensor mode #IPO_FAST. This is shown in Figure 3-17.

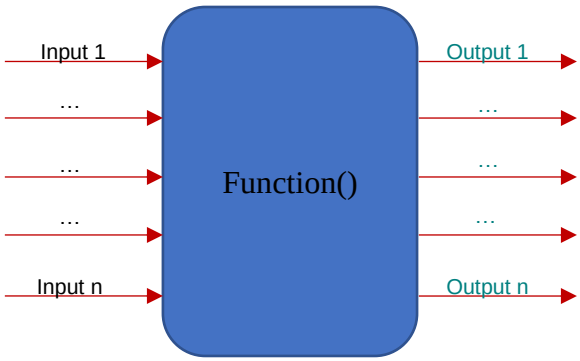


Figure 3-15 Structure of RSI object

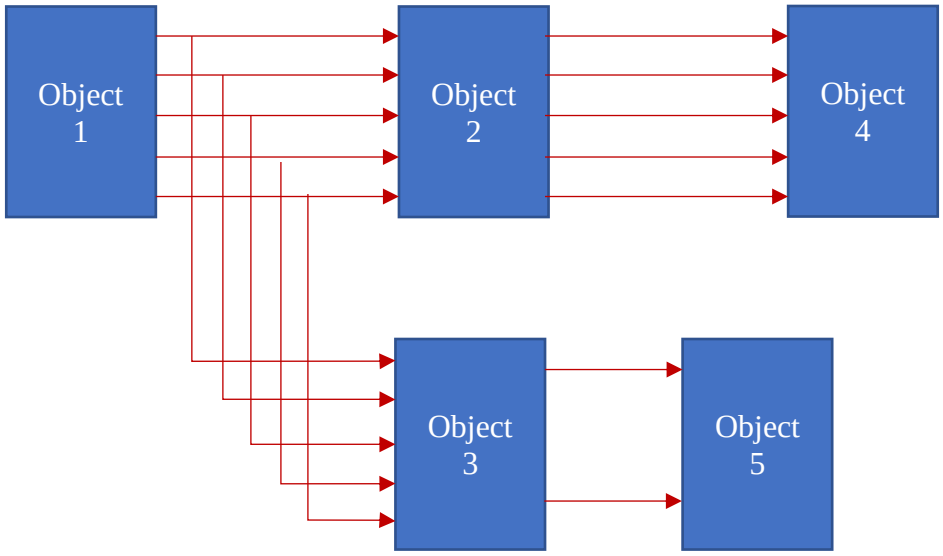


Figure 3-16 Structure of an RSI context

```
&ACCESS RVO
&REL 1
&PARAM EDITMASK = *
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
DEF RSI_Ethernet( )
; ===========================================
;
; RSI EXAMPLE: ETHERNET communication
; Realtime UDP data exchange with server application
;
; ===========================================

; Declaration of KRL variables
DECL INT ret    ; Return value for RSI commands
DECL INT CONTID   ; ContainerID

;FOLD INI
  ;FOLD BASISTECH INI
  BAS (#INITMOV,0 )
  ;ENDFOLD (BASISTECH INI)
  ;FOLD USER INI
    ;Make your modifications here
  ;ENDFOLD (USER INI)
;ENDFOLD (INI)

; Move to start position
PTP {A1 0, A2 -90, A3 90, A4 0, A5 45, A6 0}

; Create RSI Context
ret = RSI_CREATE("RSI_Ethernet.rsi",CONTID,TRUE)
IF (ret <> RSIOK) THEN
  HALT
ENDIF

; Start RSI execution
ret = RSI_ON(#ABSOLUTE)
IF (ret <> RSIOK) THEN
  HALT
ENDIF

; Sensor guided movement
RSI_MOVECORR()

; Turn off RSI
ret = RSI_OFF()
IF (ret <> RSIOK) THEN
  HALT
```

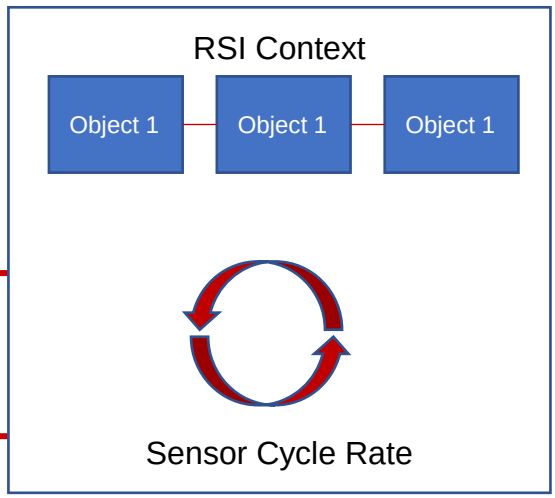


Figure 3-17 Relation between KRL program and RSI context.

3.6.1.3 Principle of data exchange via the I/O system

All data and sensor signals are read and written via the I/O system of the robot controller ($IN representing digital input and $ANIN representing analogue input). The signals are processed by the RSI context and then returned to the sensor system again via the I/O system ($OUT representing digital output and $ANOUT representing analogue outputs). These signals are read and written at the rate that the sensors cycle has been set to. In Figure 3-18, an RSI context is taking signals from a digital and analogue input of the I/O system, processed by the RSI context and written back to the I/O system via

the MAP2DIGOUT and MAP2ANOUT methods.

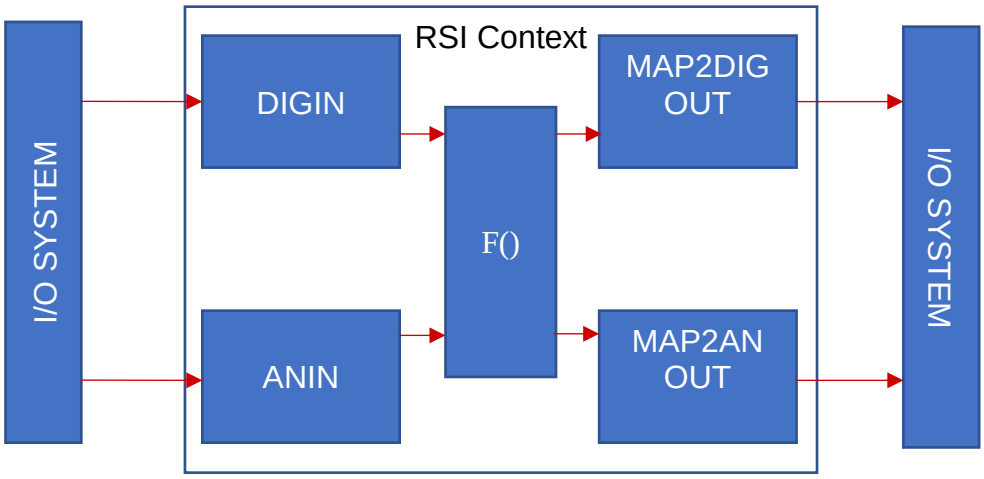


Figure 3-18 Data exchange via I/O system.

3.6.1.4 Principle of data exchange via the Ethernet system

Data exchange via the Ethernet system works in a similar fashion to the I/O system, however all communication to sensors is managed through an RSI object called ETHERNET within the RSI context. Shown in Figure 3-19, there are up to 64 inputs and outputs available for the ETHERNET RSI object where the signals at the inputs are sent to the sensor system and signals of the outputs are received by the robot controller. When signal processing is activated from within the KRL program a channel is prepared for sending data to the sensor system via the UDP protocol. The robot controller is responsible for initiating the data exchange. Once established it sends data packets to the sensor system at the sensor cycle rate specified previously. The sensor system is then responsible for responding to the data packets sent by the robot controller with data of its own. Data transmitted between the two systems is defined using a data set contained within an XML formatted file. This data set is transmitted at the sensor cycle rate. The XML file name is specified when created the ETHERNET RSI object. Figure 3-20 demonstrates the sequence of events when data is exchanged via Ethernet.

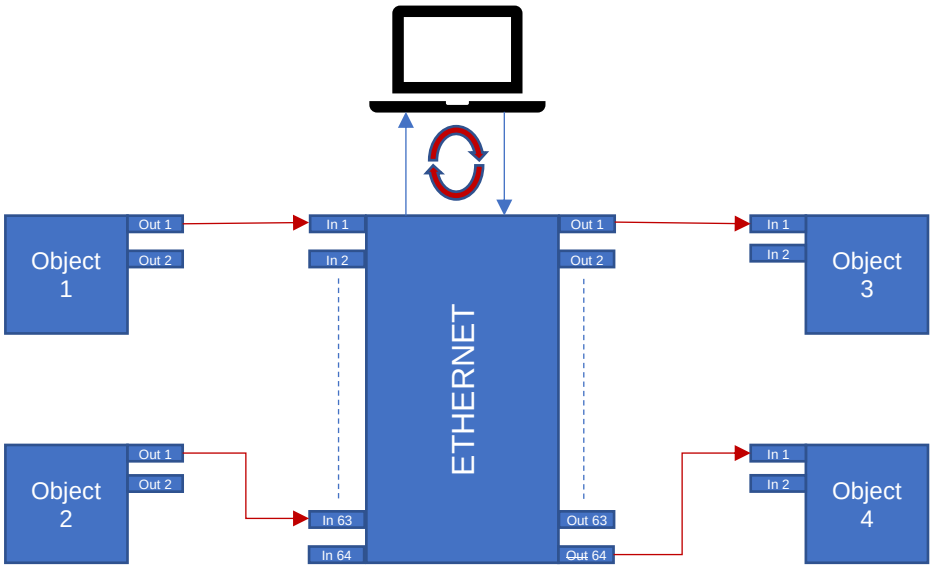


Figure 3-19 Example of data exchange via Ethernet

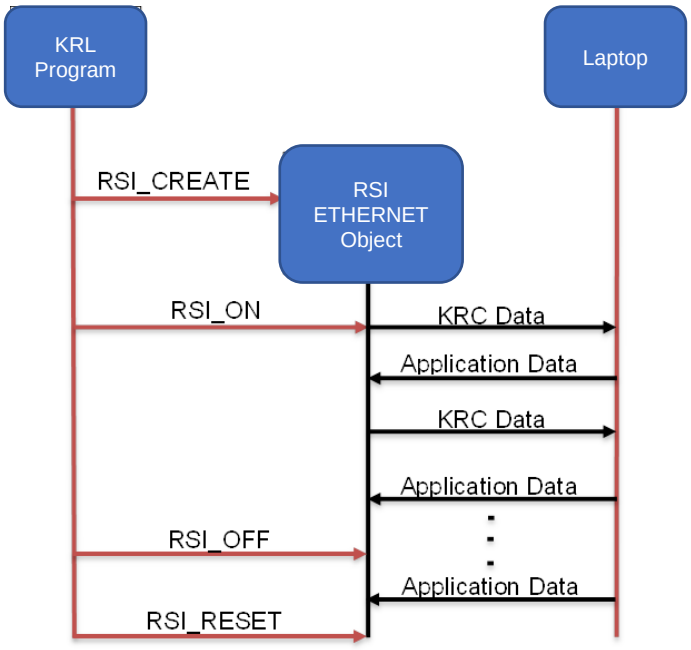


Figure 3-20 Data flow over Ethernet

3.6.1.5 Sensor based correction

The Robot Sensor Interface allows the user to exert continual influence over the motion of the robot by means of sensor data. There are two types of sensor correction available, either Cartesian or axis-specific. Cartesian creates a Correction Coordinate System in the TCP where the BASE, ROBROOT, TOOL, WORLD or Tool-based technologies system can be used a reference coordinate system. Axis specific corrections can influence axis A1-A6 or external axis E1-E6. There is however a limitation where

sensor correction cannot be used for asynchronous axes. There are also two correction modes, relative and absolute. Where relative correction values are added together and the new position results from the offset of the starting position by the previous correction and the current correction value combined, absolute correction results in an offset from the starting position by the correction value. Figure 3-21 and Figure 3-22 demonstrate these movements respectively. Finally, there are two sets of correction methods available; superposed sensors correction and sensor-guided motion. In the former, corrections are applied to existing programmed movement contained in a KRL program and merely adjusts pre-existing motions and the latter controls the entire movement of the robot system where no previous programmed trajectory has been configured.

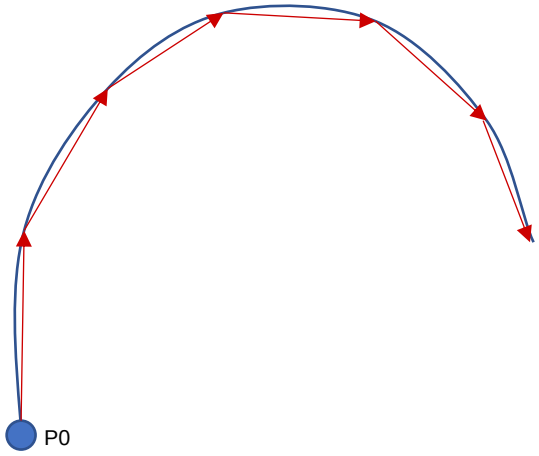


Figure 3-21 Sensor-guided motion based on relative values

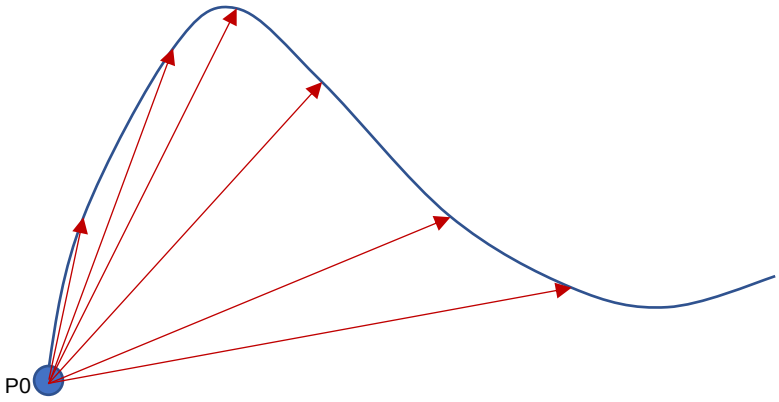


Figure 3-22 Sensor-guided motion based on absolute values

3.6.1.6 KUKA WorkVisual

KUKA WorkVisual (Figure 3-23) is a tool used for configuring and servicing KUKA Robot Controllers. A project within WorkVisual is a set of saved configurations for

each robot being configured. Whilst it is not intended to service the robot during this project, WorkVisual has the added benefit of having a KRL editor built into the package with the ability of transferring KRL programs to and from the robot controller. This can be a useful tool to have as it is faster and easier to construct KRL programs via a PC than on the teach pendant directly and also allows for offline editing of the robot programs. Within the editor there is validation to verify whether the program written has any errors in the code.

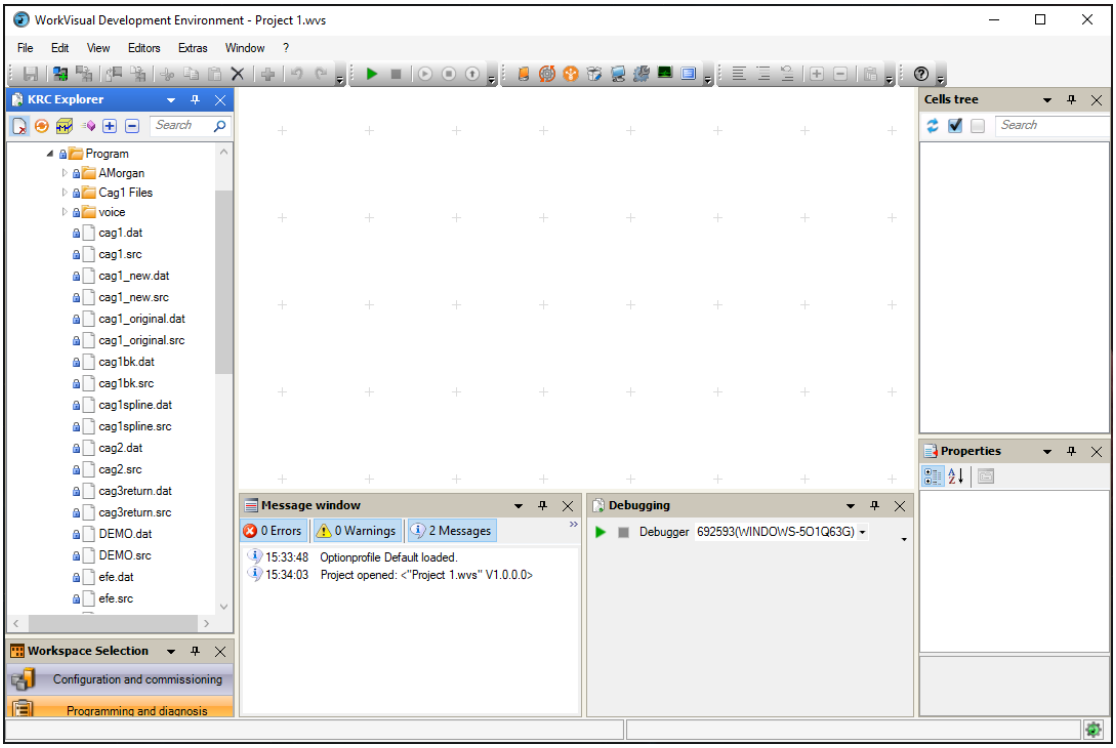


Figure 3-23 WorkVisual Development Environment Programming and Diagnosis Panel

3.7 RSI Configuration

Once the network configurations have been established, configuration of the RSI package is needed. The software itself relies on three configuration files and a single network configuration file. Different network configurations can be used with different sets on RSI configurations so project independent packages can be developed that do not require manual reconfiguration of the robot. Before the RSI Signal Flow can be created, a network configuration file is needed.

3.7.1 Network Configuration

The RSI network configuration file is an XML document with the following structure. Consisting of four main parameters shown in Figure 3-24 are what dictate how the

Ethernet interface of the robot will act. Here the IP address and port of the laptop is set. This is to reduce cross talk between multisensory systems. The <SENTYPE> element is a keyword that must be present in all replies from the laptop. This ensures the correct signal flow is replying to the right message. Finally, the <ONLYSEND> element tells the RSI interface whether to expect a reply. This is useful in situations where the Robot Sensor Interface is used for monitoring robot activity and not for influencing it.

```
<CONFIG>
<IP_NUMBER>10.10.10.10</IP_NUMBER>
<PORT>1337</PORT>
<SENTYPE>RSIMastersThesis</SENTYPE>
<ONLYSEND>FALSE</ONLYSEND>
</CONFIG>
```

Figure 3-24 RSI Network Configuration example

3.7.2 Send & Receive values

The secondary parameters relate to the configuration of the RSI signal flow and depends on what objects are used during the signal flow configuration. Within the RSI package are a set of system variables that allow basic information to be transmitted. In this instance the “RIst”, “RSol”, “AIPos”, “ASPos”, “Delay” provides Cartesian coordinates, axis positions setpoint positions, Cartesian and joint target positions and any delay acceptability (in milliseconds) respectively. System variables which are values provided by the controller are marked as “INTERNAL” in the INDX attribute. The remaining rows names and user dependant. They can be named anything as long as the INDX attribute reflects the output of the Ethernet object within the RSI Signal Flow. Any “.” Notation used with in a tag specifies a single RSI Attribute and in respect outputs an XML message such as <POSCor X=0 Y0 Z0 Z=0 A=0 B=0 C=0>. Any example of a complete Send section is shown in Figure 3-25. All XML values are processed by the RSI Ethernet object and messages and formed using the information provided by the configuration file. The above configuration will then translate to what can be seen in Figure 3-26. This information is converted into binary and is transmitted by the C4 controller to the client IP address that was configured.

In the same respect as the send values, the receive values configure the parameters the RSI Ethernet object expects to receive. Each INDX value is matched to

a corresponding output of the Ethernet Object in the RSI Signal flow. Any "." Tags translate to multi value elements and in this instance the RKorr values are communicating XYZABC Cartesian correction values back to the respective axis. The tag "EStr" gives the program an opportunity to send messages back to the KUKA Teach Pendant for any human operative using the system to see. Finally, in Figure 3-27 is an example of the expected configuration of the received XML message. Figure 3-28 then showing how the message actually looks when converted by the RSI Ethernet object.

```
<SEND>
    <ELEMENTS>
      <ELEMENT TAG="DEF_RIst" TYPE="DOUBLE" INDX="INTERNAL" />
      <ELEMENT TAG="DEF_RSol" TYPE="DOUBLE" INDX="INTERNAL" />
      <ELEMENT TAG="DEF_AIPos" TYPE="DOUBLE" INDX="INTERNAL" />
      <ELEMENT TAG="DEF_ASPos" TYPE="DOUBLE" INDX="INTERNAL" />
      <ELEMENT TAG="DEF_Delay" TYPE="LONG" INDX="INTERNAL" />
      <ELEMENT TAG="PosCorr.X" TYPE="DOUBLE" INDX="1" />
      <ELEMENT TAG="PosCorr.Y" TYPE="DOUBLE" INDX="2" />
      <ELEMENT TAG="PosCorr.Z" TYPE="DOUBLE" INDX="3" />
      <ELEMENT TAG="PosCorr.A" TYPE="DOUBLE" INDX="4" />
      <ELEMENT TAG="PosCorr.B" TYPE="DOUBLE" INDX="5" />
      <ELEMENT TAG="PosCorr.C" TYPE="DOUBLE" INDX="6" />
    </ELEMENTS>
  </SEND>
```

Figure 3-25 RSI Send XML example

```
<Rob Type="KUKA">
   <RIst X="1376.3" Y="0.9" Z="882.6" A="-180.0" B="45.0" C="-180.0"/>
   <RSol X="1376.3" Y="0.9" Z="882.6" A="-180.0" B="45.0" C="-180.0"/>
   <AIPos A1="0.0" A2="-90.0" A3="90.0" A4="0.0" A5="45.0" A6="0.0"/>
   <ASPos A1="0.0" A2="-90.0" A3="90.0" A4="0.0" A5="45.0" A6="0.0"/>
   <Delay D="0"/>
   <PosCorr X="0.0" Y="0.0" Z="0.0" A="0.0" B="0.0" C="0.0"/>
   <IPOC>2273640</IPOC>
</Rob>
```

Figure 3-26 Example of RSI XML Send Message

```
<RECEIVE>
  <ELEMENTS>
    <ELEMENT TAG="DEF_EStr" TYPE="STRING" INDX="INTERNAL" />
    <ELEMENT TAG="RKorr.X" TYPE="DOUBLE" INDX="1" HOLDON="1" />
    <ELEMENT TAG="RKorr.Y" TYPE="DOUBLE" INDX="2" HOLDON="1" />
    <ELEMENT TAG="RKorr.Z" TYPE="DOUBLE" INDX="3" HOLDON="1" />
    <ELEMENT TAG="RKorr.A" TYPE="DOUBLE" INDX="4" HOLDON="1" />
    <ELEMENT TAG="RKorr.B" TYPE="DOUBLE" INDX="5" HOLDON="1" />
    <ELEMENT TAG="RKorr.C" TYPE="DOUBLE" INDX="6" HOLDON="1" />
  </ELEMENTS>
</RECEIVE>
```

Figure 3-27 Example receive XML settings

```
<Sen Type='RSIMastersThesis'>
   <EStr>"RSI Connected "</EStr>
   <RKorr X="0" Y="0" Z="0" A="0" B="0" C="0"/>
   <IPOC>1212</IPOC>
</Sen>
```

Figure 3-28 Example XML send by KUKA RSI Software

3.7.3 RSI Signal Flow

Once the configuration file name is known, the RSI Signal flow is configured which will form the basis of communication to and from the robot. Shown in Figure 3-29, the signal flow is constructed of one or many RSI objects which are interlinked together. This project uses the following objects:

- POSCORRMON: Returns Cartesian values of the robot's current position, connected the inputs of the ETHERNET object.
- AXISCORRMON: Returns axis position values of the robot's current position, again connected to the inputs of the ETHERNET object.
- POSCORR: Receives correction values from the ETHERNET object's outputs, sets the upper and lower limits of corrections allowed. The outputs of this object are connected to the MONITOR object.
- MONITOR: This object creates an RSI Monitor instance which visualises the

changes made to the corresponding axes. The RSI Monitor interface is shown in Figure 3-30.

- ETHERNET: The Ethernet object as discussed in other chapters controls the information flow via the Ethernet interface of the device.

Once the signal flow is complete and saved, two files are created with the extensions ".rsi" and ".rsi.xml". This in conjunction with the network configuration file created in previous sections are then transferred via a USB device to the C4 controller into the "C:\KRC\ROBOTER\Config\User\Common\SensorInterface" directory.

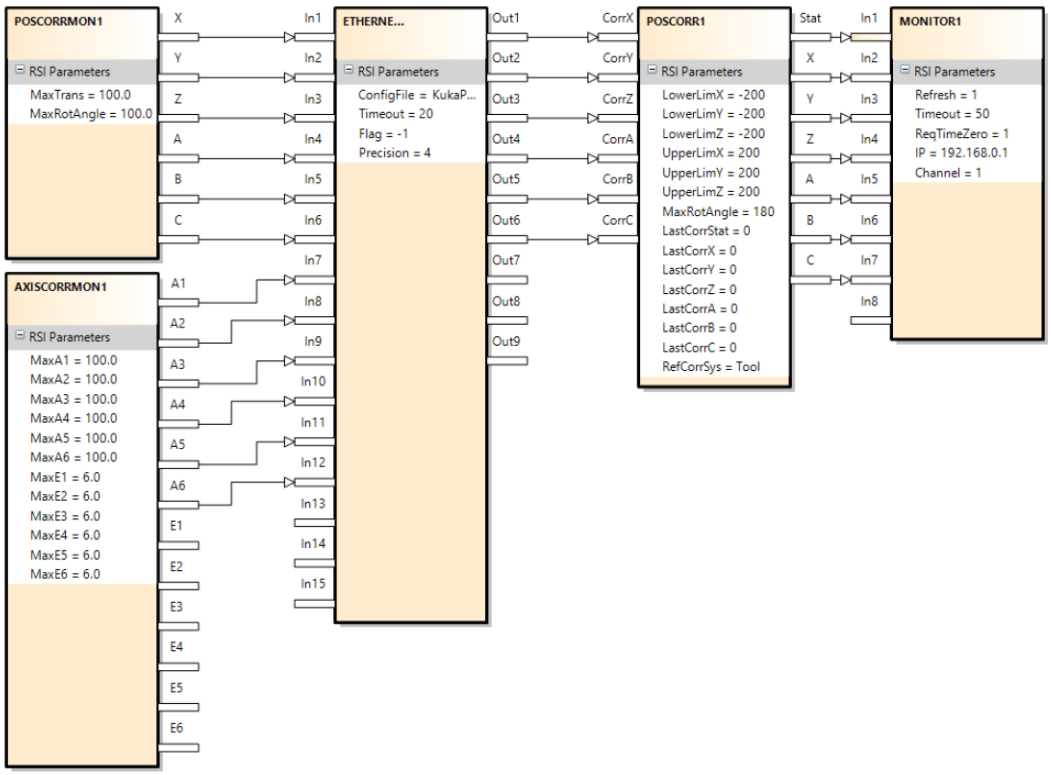


Figure 3-29 RSI signal flow example

Figure 3-30 RSI Monitor example

3.8 Sensor Configuration

3.8.1 Photoelectric diffuse sensors

Photoelectric sensors were chosen as they are relatively low-cost sensors that have not been used in this context before. The principal idea is to run two sensors in parallel with offset trigger distances. Sensor 1 is responsible for ensuring the target surface is within distance. This sensor is active by default, and if this value changes, it alerts the robot to move towards the target surface in the appropriate axis. Sensor 2 has its trigger distance reduced and is inactive by default. If the target surface fluctuates and raises higher than expected, sensor 2 triggers and informs the robot to move away from the target surface. When the TCP moves a sufficient distance from the target, sensor 2 deactivates, returning the application into its normal state. This is visualised in Figure 3-31.

As discussed in chapter 3.4.2, different sensor positions were tested. To accommodate these configurations Figure 3-32 shows the sensor mounts that are designed in SolidWorks 2020 and 3D printed using PLA plastic. The thought process behind these designs is to test whether the positioning of the sensors around the TCP of the robot effected the accuracy of the system. For example, initial experiments showed a lag in when between sensor activation and robot movement, so an additional sensor mount was developed to test whether this delay could be compensated for in the design of the sensor mount. Various designs were considered during the conceptualising stage but 3 designs were chosen to experiment with. These are shown in Figure 6-2.

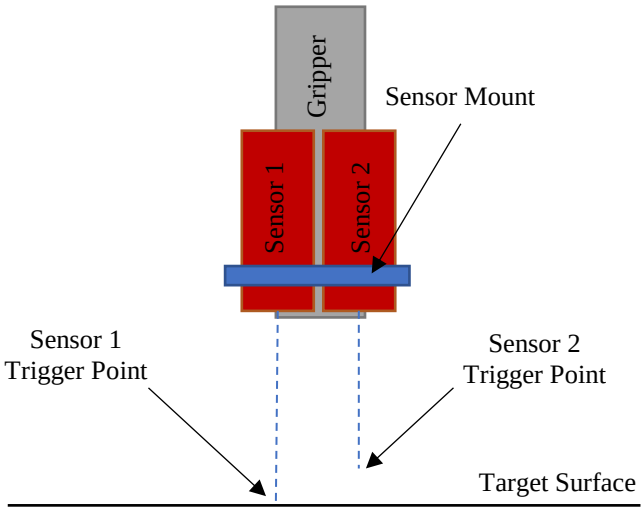


Figure 3-31 Sensor trigger distance configuration

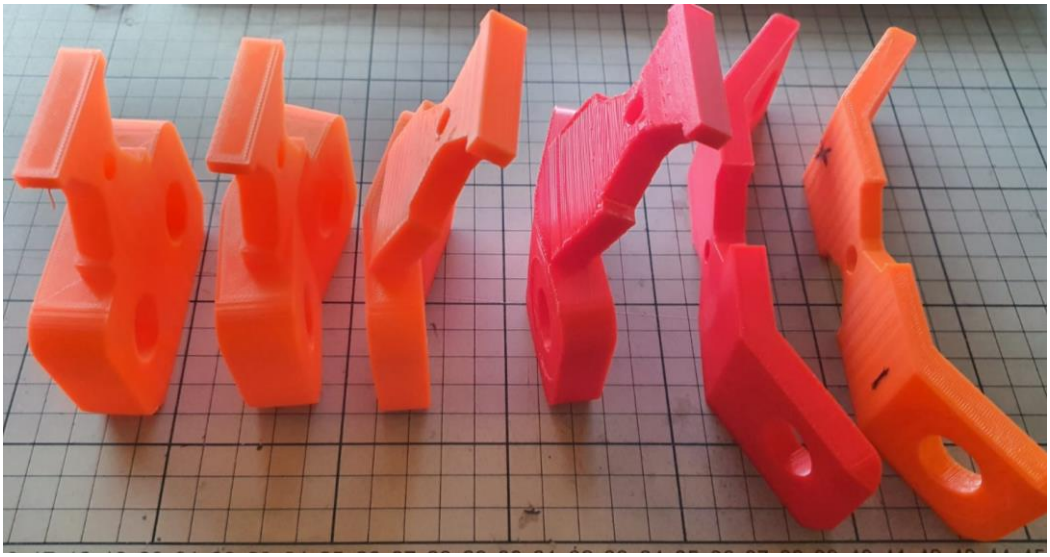


Figure 3-32 Example sensor mounts

3.8.2 Data Acquisition Device

These sensors are connected via a USB-1608G Data Acquisition device (DAQ) [104] manufactured by Measurement Computing (**Figure 3-33**). The USB-1608G DAQ was chosen because it is low cost and has digital inputs that the sensors which are to be used in this project are compatible with. The DAQ connects to a laptop via USB. By connecting the data terminal of sensor 1 to DIO1 terminal on the DAQ and sensor 2 to terminal DIO2. Negative terminal grounded and positive terminal to the 5v. Testing using MCC's company's software called DAQami confirms operation of the sensors. Utilising the Python libraries provided by MCC, sensor input is then incorporated into the main application. The two sensors are set up adjacent to the end effectors tool centre

point directed at a single location at the tip of the TCP.

Figure 3-33 USB-1608G DAQ

3.9 Investigatory Experiments

Initially a better understanding on how to use the RSI software was required. The following sets of experiments were designed to test basic movements of the robot and different settings of the RSI software. The results of these experiments were fed back into the design of the application and the final design was tested in chapter 6. Before moving on to more complex movements initial control over the robot needed to be proven. The following series of motions were tested to verify whether a sufficient understanding over the communication with the robot and the RSI package employing control over the robot was held. Each test was developed to incrementally more complex than the previous, culminating in a fully developed concept that can be taken forward to more advanced movements.

3.9.1 Experimental design

The first set of tests is to establish communication with the robot. Starting by establishing a connection whilst maintaining stability. Following successful communication, a set of tests were designed (Table 3-4) to establish how to move the robot successfully, and is aimed at testing the limits of the basic controls of the robot. Moving the robot in incrementally larger steps to ascertain the velocity and movement sizes allowed by the RSI system. The purpose of these sets of experiments is to establish the greatest movement allowed in a single correction. For movements under 1 mm in a single motion, the movement was smooth and without issue, however when moving 1 mm, the robot movements included a lot of vibration, and any greater movements resulted in a violent jerk or velocity errors from the system. Whilst almost identical to the previous set of experiments, the purpose of set 2 is to establish the difference in

relative and absolute movements within the RSI package to determine the more appropriate method of control for the application. As discussed in chapter 3.6.1.5 the difference between these movements types is how the robot received correction data. Whereas relative movements receive each correction individually, e.g. three correction values of 0.1, resulting in a total movement of 0.3 mm, absolute movements receive a total movement amount, e.g. three corrections of "0.1, 0.2, 0.3" for a total movement of 0.3 mm. This means that any movement larger than 1 mm requires additional calculation to send the correct correction value to the system. Experiments 3-1 and 3-2 were used to test larger movements of the robot. Experiment 3-1 was an addition to the previous tasks as the application is required to calculate the appropriate correction values at a speed the robot can handle. Whilst initially this resulted in jerky movements, lowering the correction value resulted in smoother movements. Moving to a particular coordinate was a more challenging task to test the robot's movement in more than one axis. The fifth set of tests is related to influencing motion of an existing KRL application. First to test how modifying the movement of the robot moving in a LIN movement and secondly testing the influence to the robot moving in a CIRC movement. As expected both movements worked correctly and the only difference between free movement and correctional movement are the RSI the values in the KRL program.

Table 3-4 Initial experiment plan

Test	Description
0-1	Establish connection with robot controller
0-2	Receive information from controller
0-3	Send information to controller
1-1	Move robot +0.1 mm in X axis using Relative movements
1-2	Move robot +0.2 mm in X axis using Relative movements
1-3	Move robot +1 mm in X axis using Relative movements
1-4	Move robot +2 mm in X axis using Relative movements
1-5	Move robot +5 mm in X axis using Relative movements
2-1	Move robot +0.1 mm in X axis using Absolute movements
2-2	Move robot +0.2 mm in X axis using Absolute movements
2-3	Move robot +1 mm in X axis using Absolute movements
2-4	Move robot +2 mm in X axis using Absolute movements
2-5	Move robot +5 mm in X axis using Absolute movements

3-1	Move robot +100 mm in X axis
3-2	Move robot to X Y Z Cartesian coordinates
4-1	Influence control over existing KRL program on a plane
4-2	Influence control over existing KRL program on an arc

3.9.2 Results

These sets of experiments, whilst do not look overly useful, they are critical in influencing the design of the Python application as they represent the fundamental motions required in this task and the limits of these motions. These results also give insights into further work needed to understand how the robot is moving versus what it is expected. A limitation encountered when analysing results was that working with such small movements and especially corrections to existing programs, it is often difficult to actually see the corrections being made when the corrections are of such a small value. The next chapter discusses a way of addressing this using the RSI software. Another application is developed for the express purpose of data analysis to overcome these issues going forward.

3.10 Summary

Chapter 3 starts by detailing the workspace used in all experiments, including the KUKA KR16 robot and its associated hardware. Included is all the configuration information along with a detailed overview of the KUKA Robot Sensor Interface software culminating in a set of pre-experiments to confirm appropriate understanding has been achieved. The next section discusses the application that is developed to utilise the KUKA hardware and software discussed in this chapter and the problems encountered when doing so.

4. APPLICATION DEVELOPMENT

4.1 Introduction

In the previous chapter the KUKA robot and its associated systems were set up and tested to confirm a sufficient understanding of the underlying systems had been achieved. This chapter takes the knowledge gathered and is applied to create a system that interlinks all the components discussed in chapter 3. First the initial concept of the software is discussed, followed by a breakdown of the main applications structure and how it intends to operate. The user interface is then presented along with all the unexpected problems encountered during development. Finalising in a summary of the final system that is used in all future experiments.

4.2 Initial Concept

KUKA's RSI software exposes access to exert control over the robot, so a communication interface and control system needed to be developed to interface with this system. Reviewing previous literature there were two options available for interfacing with the RSI software. The first being a Python interface designed for previous versions of RSI and the other being the Robot Operating System. The decision was made to write a custom piece of software to keep the software footprint low as ROS required large amounts of libraries to be installed on a system. Additionally, the RSI module for ROS is currently written in Python 2.7 whereas the libraries required for communication with the sensors requires Python 3, this means the supporting ROS libraries to interface with the RSI software would need to be re-written to support Python 3. Three Python threads are created when running this program, the first controlling the networking. The second running the user interface and the third controlling the sensor logic. Multiple threads are used to reduce processor delay, to stop the user interface from freezing and to allow the network communication run in the background while calculations are being calculated to avoid network timeouts.

4.3 Main application structure

The communication of the application operates in a closed loop where a UDP socket is polled at 4ms intervals looking for data sent by the robot system. Any information received is converted from binary into XML and the relevant variables are extracted

and stored. This reply may contain a default set of zero values or positive or negative correction values. The structure of the messages received is discussed in chapter 3.7.2.

Figure 4-1 shows the operating loop of the communication function. The process is broken down in to five steps. The program polls the network card and waits for data to be received. If no data appears, the network connection times out, resets and waits again. If information is detected the values is stored into a variable and step 2 continues.

- Process Message: The message received will be in binary format, this is converted to a string to form an XML message, from here values are extracted and validated for data integrity.
- Check Sensors: Any sensors attached to the system are probed for information. The resulting values stored in variables.
- Update Status: The UI is updated and all new joint positions and message values are refreshed.
- Formulate Reply: Values polled from the sensor system and from message received and then used to perform the calculations need to produce correction values to be sent to the robot.
- Send Reply: These values are then converted into binary and sent to the IP address identified from the message previously received.

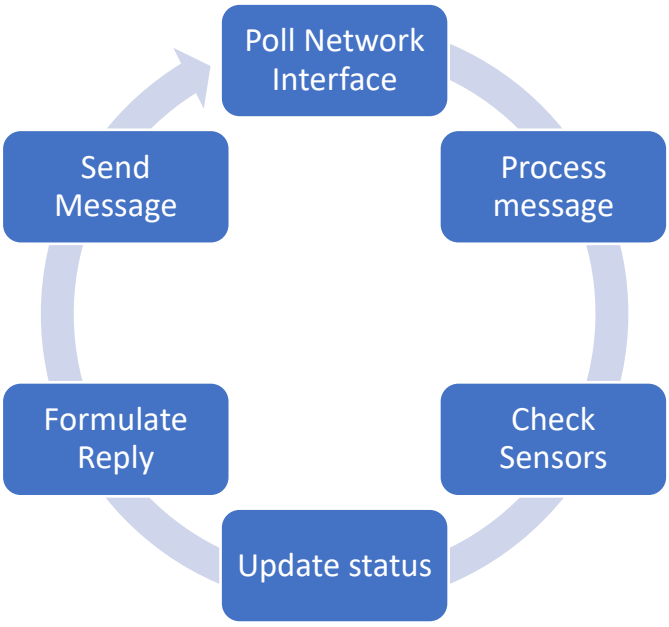


Figure 4-1 Python application communication process

4.4 User Interface

The initial user interface was designed using Python TK as a framework. Whilst bloated and over saturated with numbers it proved rather useful in the initial debugging of the application as well as testing of the RSI software. In Figure 4-2 the initial design of the UI is shown before a more advanced version was completed for the final experiments. The UI was heavily influenced by the RSI test server application. Through testing discussed in chapter 3.9, the UI was modified to include buttons to trigger and labels to view robot movements that were used when learning how to use RSI and influence the robot (Figure 4-3). This allowed for quicker testing and debugging, along with exposing sent and received information in a more presentable manner. The final interface shown in Figure 4-4 is far removed from its predecessors. More focussed and to task, the UI presents the basic information needed for operating the application by stripping out all unnecessary information. The IP address is presented to ensure correct connectivity, and then the Cartesian and axis values of the current robot position. Following this there are two visual indicators representing each sensor state and the centre value displays the current total correction value currently being used. Finally, a status box to display current application state, with a “Run” and “Stop” button to initialise and stop program operation.

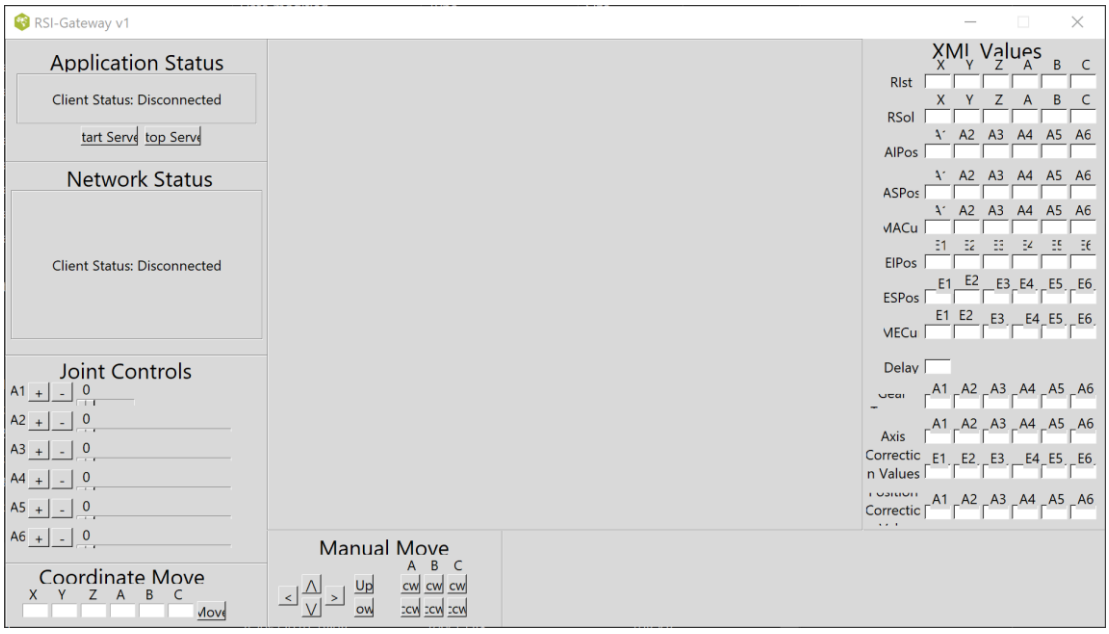


Figure 4-2 Initial UI Concept

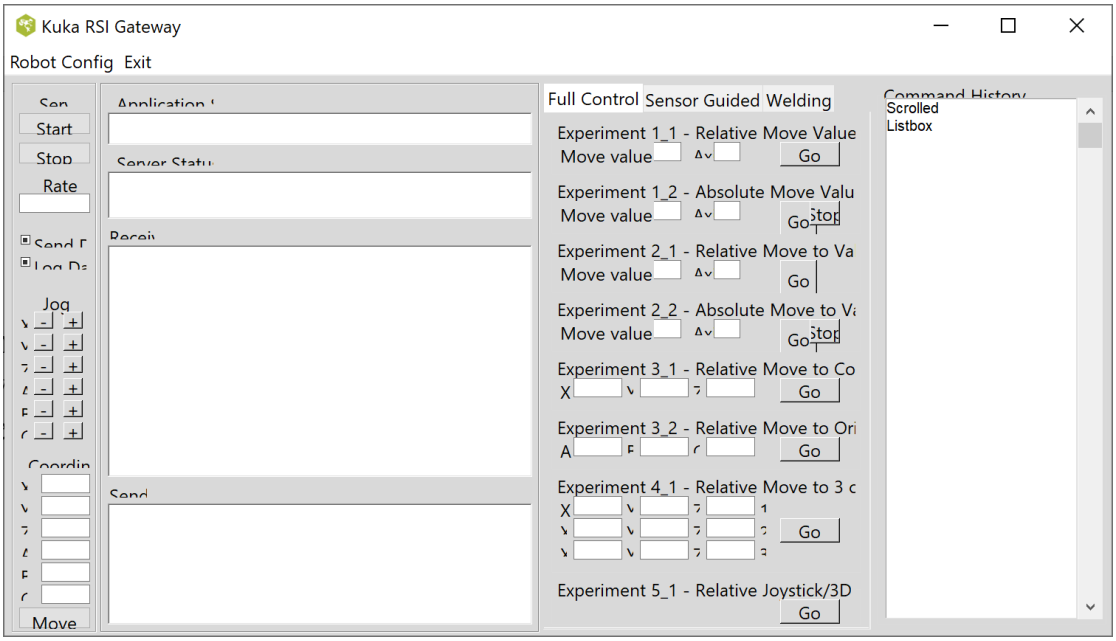


Figure 4-3 Updated UI Design

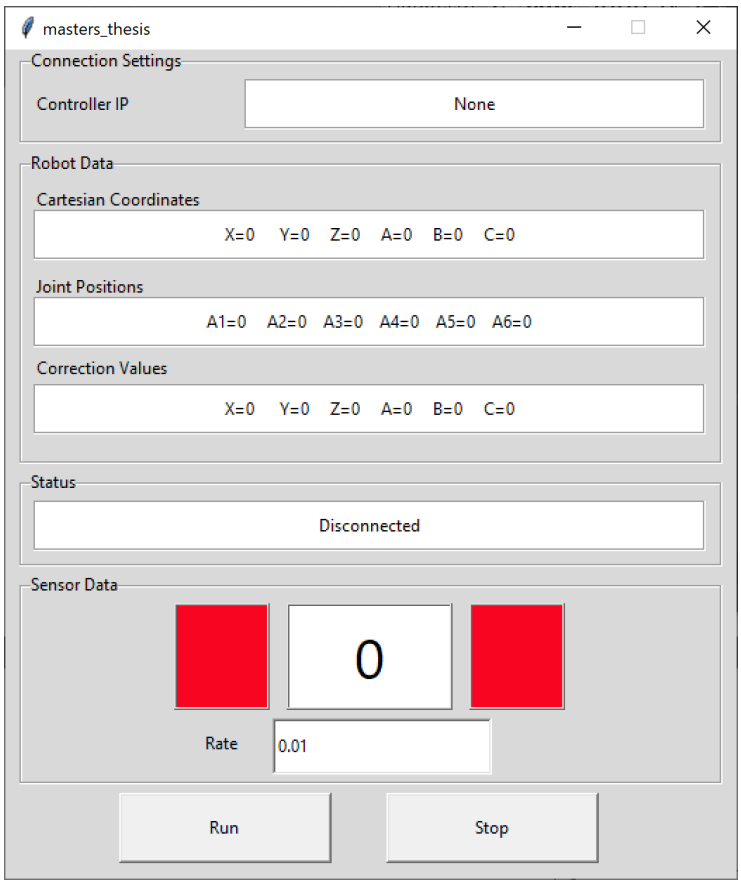


Figure 4-4 Final UI Design

4.5 Unexpected problems

4.5.1 KUKA Documentation

Documentation relating the RSI technology is rather sparse. The main Robot Sensor Interface manual [94] contained enough information for installation and configuration along with several example applications to learn from but the manual lacked critical information about the system variables that are available to be used when programming. This was not shipped with the product. System variables were eventually found in KUKA System Variables manual [105], whilst XML variables for the RSI network configuration was found in a previous KUKA product named KUKA Ethernet KRL [92], a previous product by KUKA and as a result they share a lot of the same system variables.

4.5.2 Singularities

One of the initial problems encountered during development was a singularity found in the robots starting position. A singularity was discovered when the robot axes were in the following position: A1 **0°,** A2 **-90°,** A3 **90°,** A4 **0°,** A5 **0°,** A6 **0°**. This resulted in errors on several axis, terminating the robots' movement. A singularity in terms or robotics is when multiple robot joints align in the same direction, the robot's end effector is unable to adopt certain poses. With the KUKA KR16-2, this results in the errors mentioned above rather than risking damage to the robot by moving in an unknown situation.

In the instance of this project, the problem was quickly overcome by moving the start position of the robot to another position, thus avoiding this singularity in particular. Whilst unlikely that the robot would hit this particular position during running of its main application, the error must be documented as a robot in an existing manufacturing environment would need to adjust its home position if this system is implemented.

4.5.3 Network timeouts

When initially testing the RSI software a constant network connection with the robot was difficult to maintain. This was because the program was not responding to the messages sent by the robot correctly. Due to a lack of documentation, excessive troubleshooting was required to determine why this was happening. Ultimately, it was

found to be caused by interpreting the values received by the robot controller incorrectly which resulted in incorrect values being sent back. The RSI system would catch this information and terminate the program due to the error. These problems were overcome when the modified user interface was used to display more information about what was being sent and received, allowing the error to be identified and from this a resolution to be created.

4.5.4 Multiple program threads

During initial development of the user interface a lock up in the user interface when trying to influence the robot was encountered. The problem occurred due to the way the program logic looped. During idle operating the program iterated through the user interface code and the logic code without issue but when influencing the robot, a secondary loop was created which did not allow any further user interface code to execute. The solution to this was running multiple threads with the user interface code and the program logic code separated into its own threads. A third thread was introduced to run any code influencing the robot system, allowing all other threads to run unimpeded. This had the additional benefit of speeding up execution, where the communication thread would reply with the correct values to the robot even if the logic thread had not completed processing. As long as the delay configured in the RSI configuration had not been breached, it allowed the system to process information with a built-in tolerance for a delay.

4.5.5 Erratic movement

During the first set of tests the robot experienced very sudden movements which resulted in velocity errors from the robot controller. These errors were the result of poor understanding and interpretation of the example RSI programs. For the robot to move smoothly, the flow of correction values had to be consistent. For example, to move in the X axis by a distance of 15 mm, a correction value of 0.1 needed to be sent to the robot continuously until the distance of 15 mm was met. However due to poor understanding, the resulting messages appeared: “0.1, 0.1, 0, 0, 0.1, 0” where as they should have looked like “0.1, 0.1, 0.1, 0.1, 0.1, 0.1”. This was fixed within the source code to smooth out the movement action.

4.5.6 Velocity errors

Other than the issues discussed in the previous sections, a high number of velocity errors were encounter, this time due to trying to move the robot quicker than it was possible to do so. After running through a set of experiments discussed in the following chapter more appropriate correction values were determined to use in further experiments.

4.6 Final System

Once all testing has completed, the final application operates as described in Figure 4-5. The control system operates on a PC or Laptop, connected via Ethernet to the KUKA C4 controller, the C4 controller then influences the robot. The sensors are connected to the DAQ which is then connected via USB to the laptop. The system uses two variables, to control the logic; the axis to move and a value called “data rate”. The data rate is the value that is sent as a “correction” to the robot. This value represents the distance to move the axis chosen in millimetres. When the sensors are triggered, a Boolean value is used to represent its state. Two Boolean values represent each sensor and when these values are received, the system communicates the data rate to the robot controller based on the logic detailed in Figure 4-6. This set of logic is referred to as the “simple algorithm”. A “smart algorithm” is introduced in chapter 6.2.

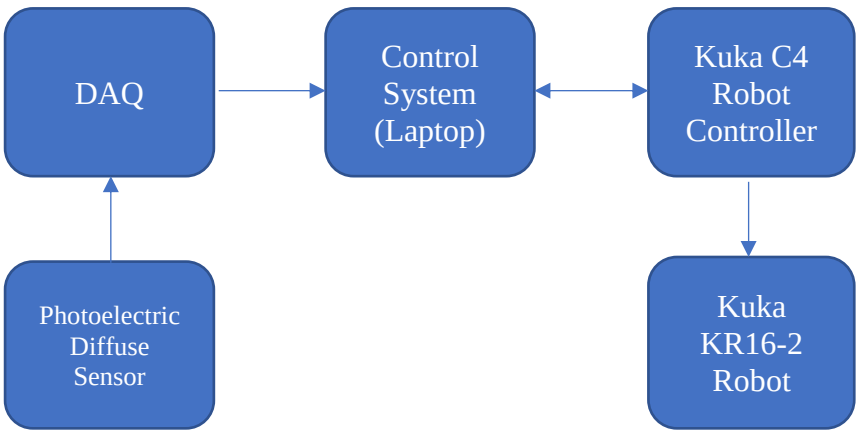


Figure 4-5 Overview of Control System

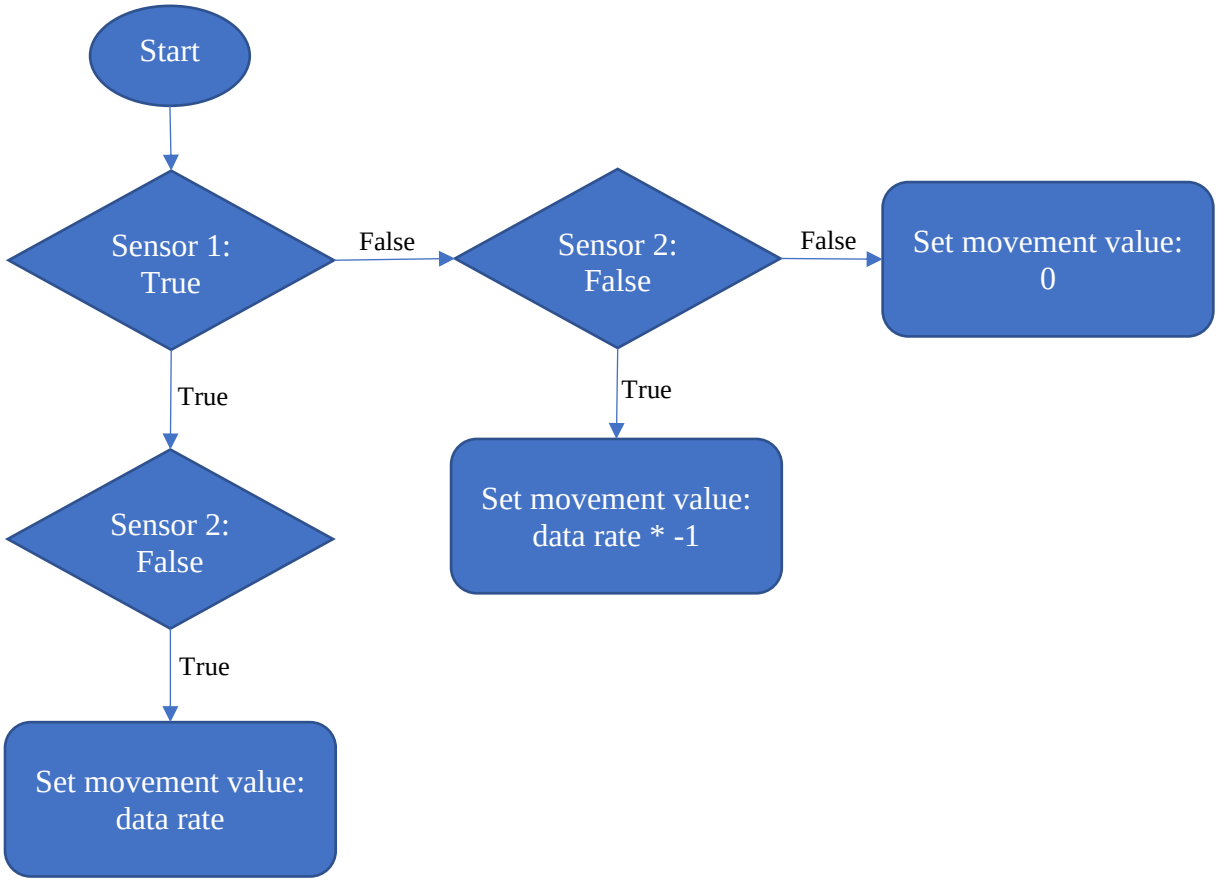


Figure 4-6 Sensor system logic flow

4.7 Summary

This chapter takes the reader through the development of the position adaption system. The concept, structure and interface of the application is discussed giving the reasoning behind all design choices and all of the problems encountered throughout is detailed, finalising in an overview of the final system. The next chapter takes a look at an application developed as an offshoot of the position adaption system that has been created for the sole purpose of gathering and analysing KUKA robot movement data.

5. ROBOT DATA LOGGER

5.1 Introduction

As discussed in chapter 3.9 a method for visualising robot movements was required for more accurate analysis of robot movements. The following chapter discusses how this was accomplished and how this method was further developed into a standalone application which can be used to analyse in post full KUKA KRL programs. The following chapter discusses the process of developing this tool and examples of what information can be produced.

5.2 Development

Whilst developing this system proposed in this thesis, it became apparent the information passed between the robot controller and the laptop can actually be quite useful for post analysis. Modifying the existing program to expose more values through the RSI Software, a subroutine was added for recording said data to a comma-separated value file (CSV). Later being moved into its own independent program, the system has the ability to add the relevant RSI code to existing KRL files to enable this functionality and then runs as on a third-party device connected to the controller via Ethernet. Whilst the main KRL program is operated, the system records all data being sent by the controller. When running with larger KRL programs a problem became apparent when the network connection would time out. Debugging the code to resolve the issue led to discovering that the process of writing to the CSV file become increasingly more time consuming as the file grew in size. This would delay the communication loop of the main program causing the timeout. Moving the CSV writing process to separate thread resolved the issue as the two processes now run concurrently without one process delaying another. The resulting CSV file contains 25 columns of data which can then be interpreted in programs such as Microsoft Excel or Matlab. The headings listed in Table 5-1 summarise each column, each RIst, RSol, AIPos and ASPos have 6 columns each representing either the XYZABC coordinates of axis 1-6 and Figure 5-1 shows an example of a typical log file created after a KRL program has been run.

Table 5-1 RSI Datalogger Column Headings

Column Header	**Description**
IPOC	Timestamp of movement

RIst	Setpoint position of robot TCP
RSol	Actual position of robot TCP
AIPos	Setpoint position of robot axis
ASPos	Actual position of robot axis

The Figure 5-2 to Figure 5-4 show information collected when using this RSI data logger in conjunction with a KRL program used to place components on a PCB. Using this data, individual target points that the robot is moving to can be visualised. In Figure 5-2 Matlab was used to plot the coordinates to visualise the positions of a KRL program. Each position would represent a component to be placed on the PCB. Each point represents either the start location, part feeders, or target component locations and is useful for analysing theoretical vs actual and for validating a KRL program is correctly programmed. Additionally, the robot TCP trajectory can be plotted. This is useful when optimising robot movements because when watching a robot in operation it is difficult pinpointing a singular movement when there are multiple rapid movements in succession. Figure 5-3 gives an example of how this can be visualised. Not only can the robot movement in a three-dimensional space be visualised but it can also be viewed in a two-dimensional context. In Figure 5-4, joint 1 to 6 of the robot over the total time period of the application running is compared. This is useful to view how each of the 6 joints is moving during the total operation of the program and to test to see if there is any correlation between their movements.

IPOC	RIst X	RIst Y	RIst Z	RIst A	RIst B	RIst C	RSol X	RSol Y	RSol Z	RSol A	RSol B	RSol C	AIPos A	AIPos A	AIPos A	AIPos A	AIPos A	AIPos A	ASPos A	ASPos A	ASPos A	ASPos A	ASPos A	ASPos A
119107	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119111	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119115	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119119	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119123	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119127	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119131	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119135	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119139	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119143	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119147	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119151	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119155	947.1	13.7	904.4	-179.5	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119159	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119163	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119167	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119171	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119175	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119179	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119183	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119187	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.7	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119191	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119195	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119199	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119203	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119207	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119211	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119215	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119219	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119223	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119227	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119231	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119235	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119239	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119243	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119247	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3
119251	947.1	13.7	904.4	-179.6	2.6	178.4	947.1	13.7	904.4	-179.5	2.6	178.4	0.3	-89.6	71.6	-1.7	105.4	0.3	0.3	-89.6	71.7	-1.7	105.4	0.3

Figure 5-1 Example RSI data

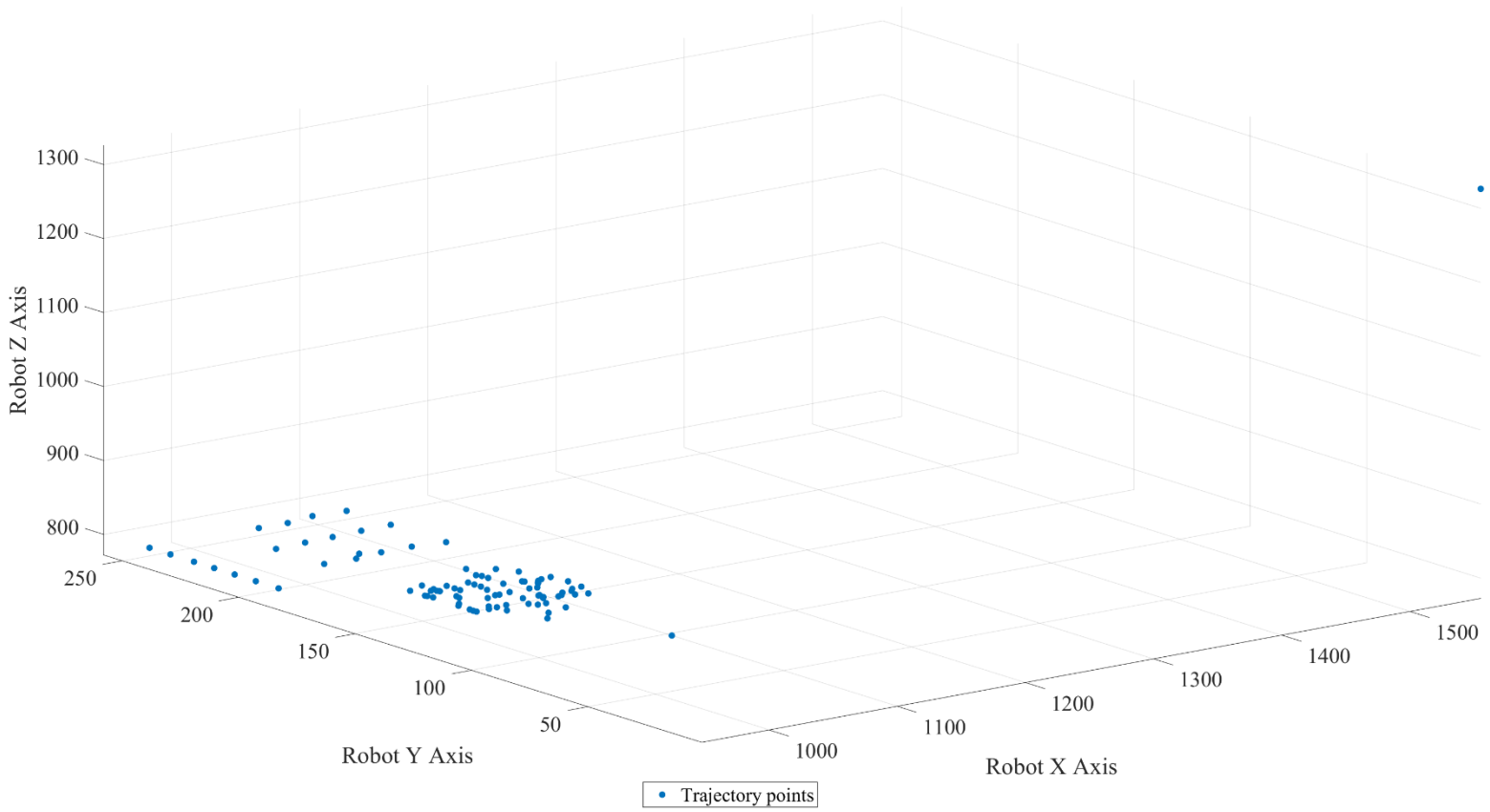


Figure 5-2 RSI Datalogger Point Data

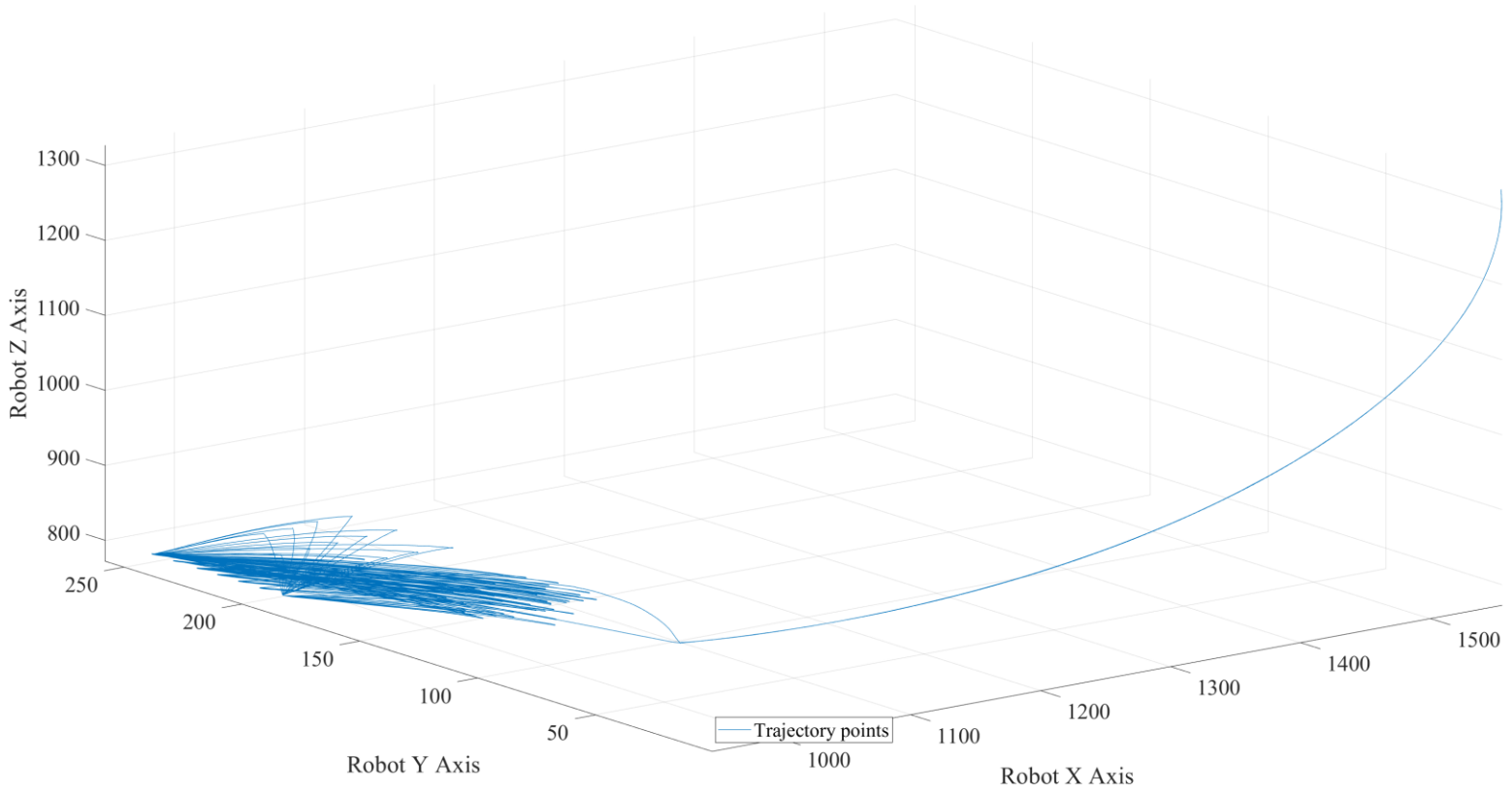


Figure 5-3 RSI Logger Actual Movements

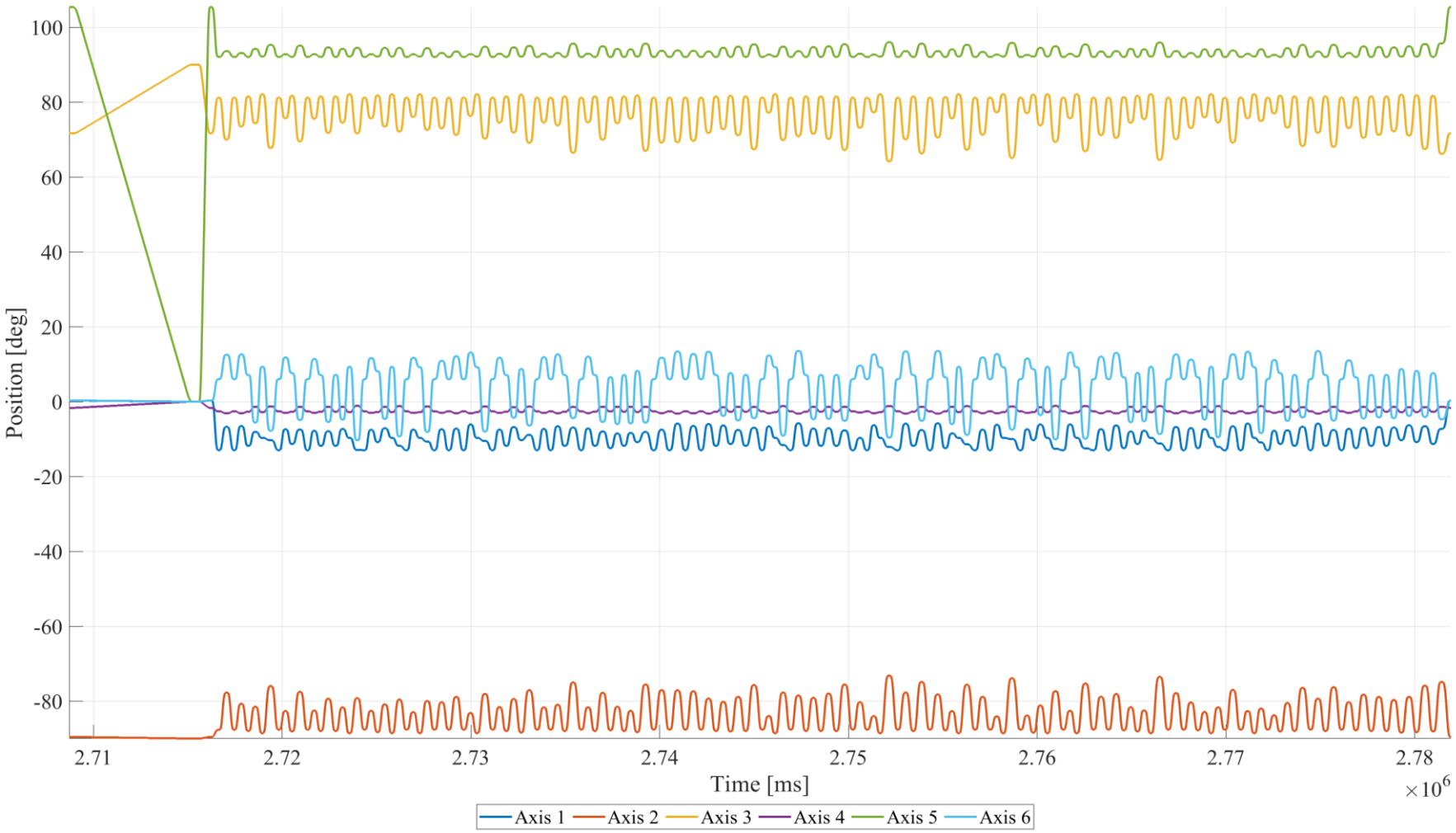


Figure 5-4 RSI Logger Axis Position Comparison

5.3 Standalone application

A standalone package was developed with the aim of being easy to use. A user interface was constructed shown in **Figure 5-5**. In this UI, two sets of options are presented, the primary being to initiate the data logging process and the second being a method to modify the existing KRL code to include the required RSI configurations files needed for the logging software to work. The output of this file is a duplicate of the original KRL program with the appendage "_rsi". This appendage was to not overwrite the original application and allow for additional analysis rather than a replacement. Once the RSI code if added to a file, that program will no longer operate on the KUKA robot without the RSI logger software active. Accompanying RSI configuration files and instructions are packaged with the program. For the remainder of this project the functions presented in this chapter are built in to the final application for testing and verification purposes.

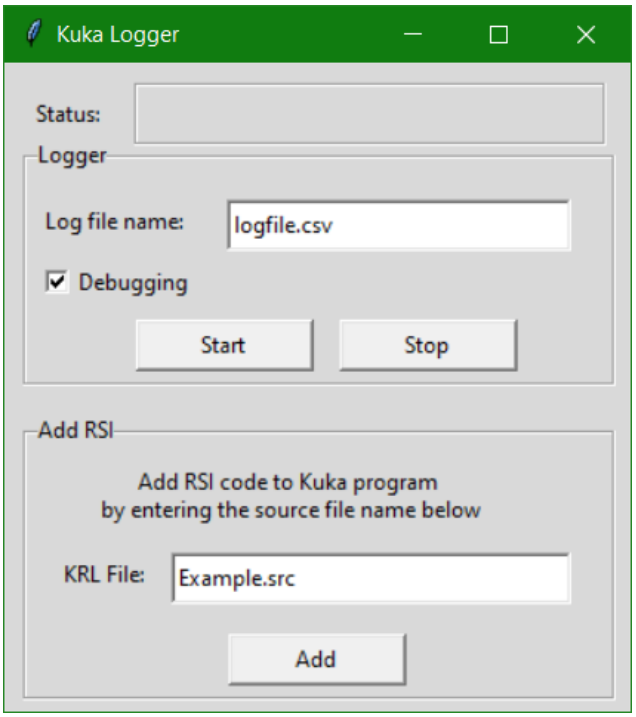


Figure 5-5 KUKA Logger UI

5.4 Summary

This chapter demonstrates the KUKA data logger application which has been developed in response to requiring a more accurate way of analysing the experimental data within this thesis. The next section goes back to focusing on the position adaption system and outlines the optimisation phase of development which in part utilises methods discussed in this chapter.

6. OPTIMISATION EXPERIMENTS

6.1 Introduction

Previous chapters have covered the conception, testing and development of the position adaption system created in this thesis. This chapter takes the application and runs it through a series of experiments to investigate whether further optimisation is possible. A set of different parameters such as speed and sensor position are tested and an algorithm derived to improve system accuracy is introduced. The results are then collated and analysed to introduce further improvements.

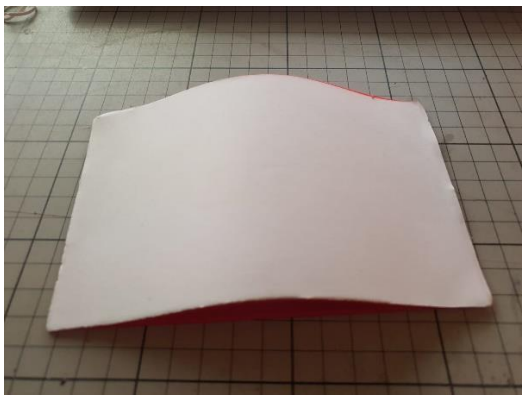
6.2 Experimental design

Any adaptations to robot movement needed to be visually analysed because simply observing these changes while the robot was moving proved difficult. From the previous experiments the need to improve upon the data analysis of the experiments was identified. Whilst having the ability to record this data is beneficial, it required a means to visualise it. The initial concept for these optimisation experiments was to use folded paper shapes to act as obstacles for the robot to avoid. However, it proved difficult to visualise these shapes on a graph, the paper would flex and fold under the its own weight and if the robot TCP collided it would result in an obstacle being permanently disfigured. This meant any experiments undertaken with this type of obstacle would yield inaccurate results. To resolve this a series of shapes was developed where their geometric data is structured by an equation or set of graph coordinates which can then be overlaid onto a plot along with the robot data recorded to analyse the effectiveness of the overall system and to identify improvements.

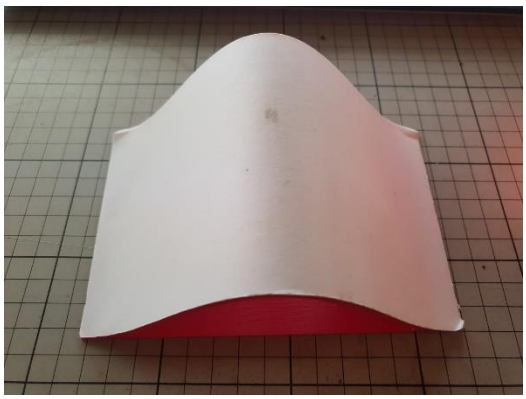
Each object would be 150 mm long and have varying degrees of change in the shape presented. The first object shown in Figure 6-1a is 150 mm long section with a small inclination at the centre. This a simple object to test basic functionality of the program to adapt to a small variation. The second object demonstrated in Figure 6-1b has a large inclination at the centre to test the upper limits of its ability to adapt. The third object (Figure 6-1c) is a sine wave with a smaller amplitude repeated to test the applications ability to adapt to repeated variations in surface materials over a short distance. These objects were derived from mimicking the types of deformation likely to be encountered during the process of polymer welding industrial pipes. In this sort of application, you would encounter gradual change in surface shape rather than

encountering a random shaped object. To test both minor change in surface and large changes object 1 and 2 was developed. Object 3 was an extension of this thinking and designed to replicate a large number of variations rather than a singular change.

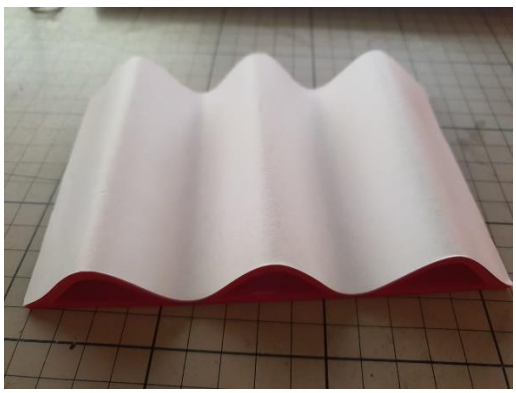
Figure 6-1a to c are created with equations 1-3 respectively. Creating the objects in this manner allows for easy calculation of the expected trajectory of the robot. The base KRL program used for these experiments is a simple LIN motion between two points, the XYZ coordinates generated from this movement is used as inputs for three equations to plot the expected trajectory of the robots TCP when the sensor system is used in conjunction with each object. In chapter 6.3 the expected trajectory is compared with actual to visualise how the robot is moving against what is expected.



a



b



c

Figure 6-1 Example obstacle objects; (a) Object 1 (b) Object 2 (c) Object 3

Equation 1 Object 1 equation

$$y = 10 * \sin\left(\frac{x - 37}{24.3}\right) + 11$$

Equation 2 Object 2 equation

$$y = 30 * \sin\left(\frac{x - 37}{24.3}\right) + 31$$

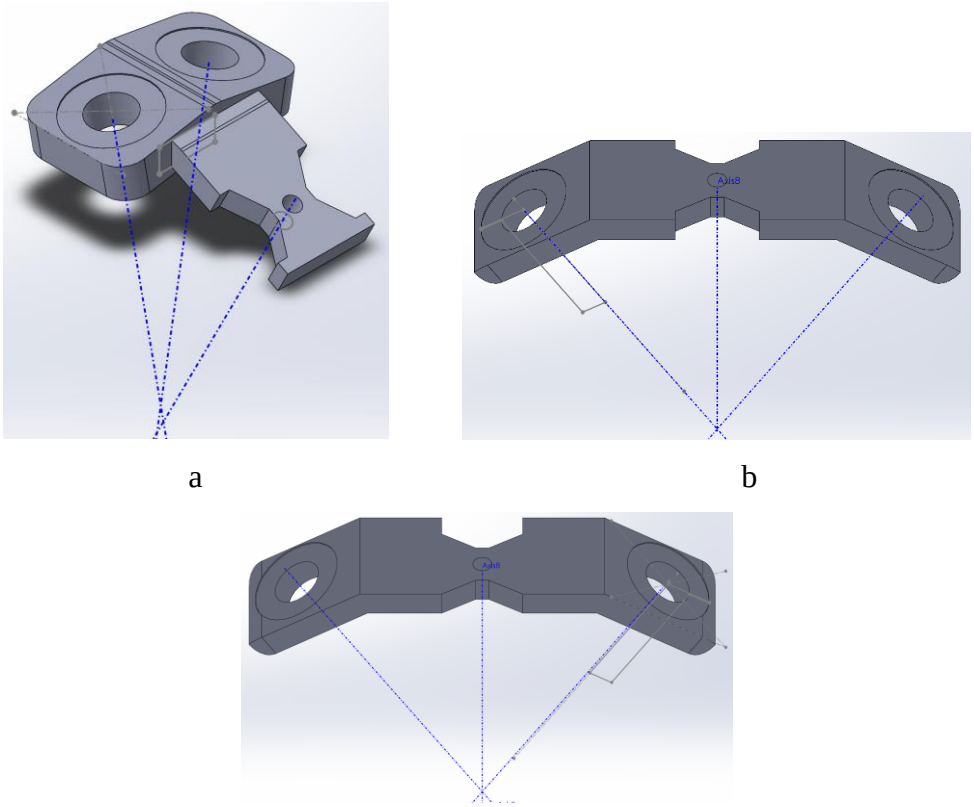
Equation 3 Object 3 equation

$$y = 10 * \sin\left(x - \frac{12.7}{7.95}\right) - 12$$

Additional to experimenting with different obstacles, a set of three different sensor mounts was tested to analyse the effectiveness of photoelectric diffuse sensors positioned in different orientations to see whether the position of the sensor affects the response of the system. The first mount shown in Figure 6-2a positions the sensors next to each other angled at 10 degrees so the target locations converge at the same point. This is then mounted to the robot using the gripper and orientated at a 27-degree angle to focus the sensors at the TCP. Figure 6-2b shows a mount where the sensors are positioned either side of the end effector, angled at 32 degrees to converge on the TCP. These two designs represent possible mounting points of these sensors onto existing equipment. When considering robot plastic welding for example, the TCP is going to be subject to very high temperatures, which means keeping the sensors away from this point was important. The mount shown in Figure 6-2c is similar to Figure 6-2b but the sensor positions are offset by 5.7 mm to target ahead of the robot TCP. The purpose of this was to test to see whether sensing changes in the target surface ahead of time is adequate as the cycle time of the KUKA RSI software may not be quick enough to compensate for changes at high speed. Each of these sensor mounts were designed and 3D printed so that quick replacements with optimised modifications if needed.

In Appendix A, Table 9-1 details the list of parameters used for each experiment conducted. The purpose of these tests was to analyse how the application developed in chapter 4 responds against different obstacles using different data rate values, robot speeds and sensor mounts. In addition to testing these parameters, two more parameters of a more advanced algorithm were tested. This “smart algorithm” was developed to control the data rate value being sent to the robot unlike the simple algorithm which used a fixed data rate value. This “smart algorithm” has two parameters which need to be optimised. The “smart delay” which represents the number iterations the algorithm logic loops through before incrementing the data rate with the “smart increment” value.

The number of loops is referred to as “cycles” through the rest of this thesis. The smart increment is the value actually added to the data rate. Each experiment was conducted in the YZ frame. A total of 240 tests were completed. Figure 6-3 illustrates the expected outcome when the robot is operated with the sensor system active and Figure 6-4 shows the KRL program trajectory that is used in our experiments.



a b c

Figure 6-2 Sensor mount designs **(a)** Adjacent Sensors **(b)** Parallel sensor mount **(c)** Parallel offset sensor mount

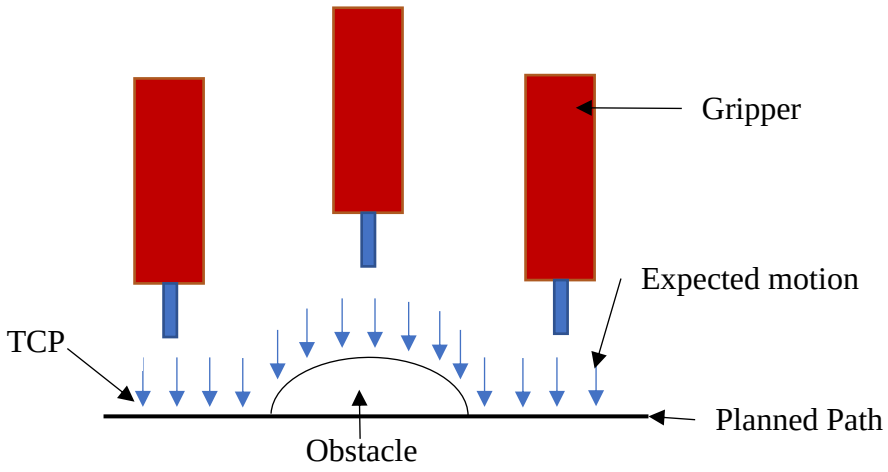


Figure 6-3 Linear Experiment

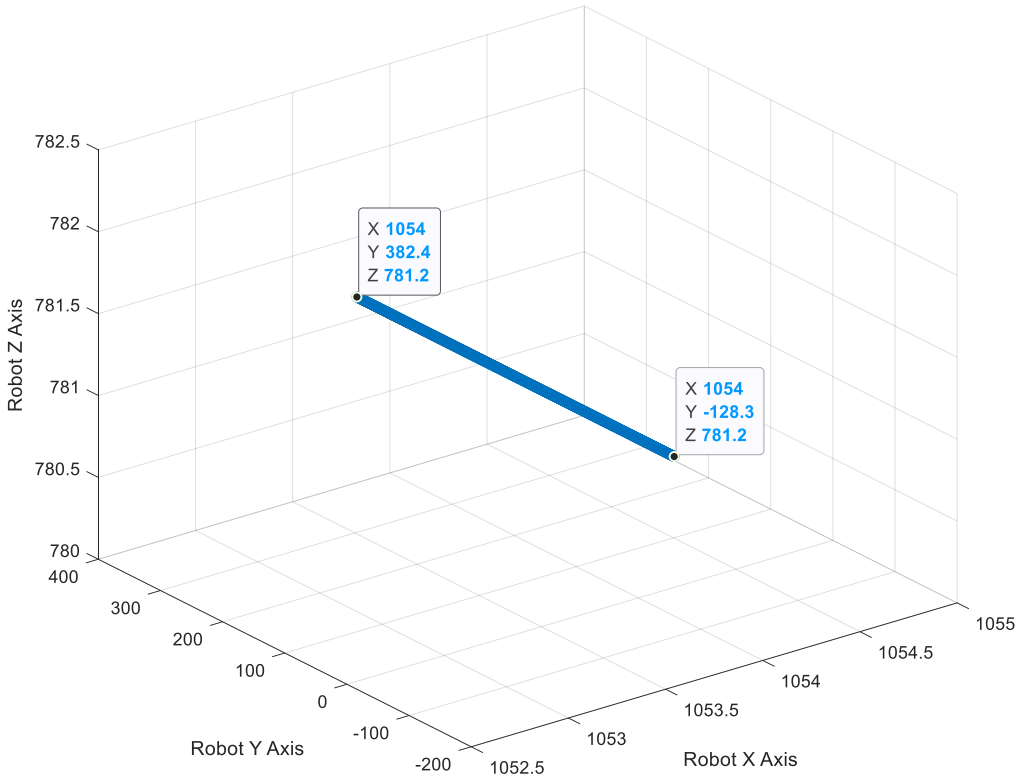


Figure 6-4 Baseline LIN results

6.3 Results

In the following section the results from the experiments conducted are discussed. The aim of these experiments is to establish how capable the system is in maintaining a fixed distance from the target surface when an obstacle of unknown geometry is encountered in the robot TCPs path. Additionally, to identify optimum values for the smart algorithms' parameters. First, a comparison of the different objects is used to test the system using the simple algorithm. Then how different robot speeds and data rates affect the responsiveness of the sensor system. This is then followed by testing different sensor mounts and validating them. Next the same tests are repeated but now using the smart algorithm. The results are evaluated and compared to the simple algorithm to establish whether the smart algorithm performs any differently to the simple. Next a set of experiments to test different parameters of the smart algorithm is performed to see if any optimisation can be made. Finally, the optimum values are selected from these results and a last set of tests is performed. Tests were conducted using speeds of 1%, 3%, 5%, 10% and 30% of 2 m/s.

To help evaluate these results a scoring system was devised to analyse the effectiveness of each experimental run. To calculate the score the actual trajectory of

the robot is compared to the position of the obstacle. Each obstacle was designed to be 150 mm in length which means equation 4 below can be used to calculate a value where x_1 represents the X coordinate of the obstacle and x_2 represents the X coordinate of the actual robot trajectory. The sum of x_2 value deducted from x_1 value at each millimetre point on the X axis is divided by 100 to give a score. This demonstrated in Figure 6-5, the yellow line represents the object, the orange line represents the KRL programmed path and the blue line represents the actual trajectory. If the program results in a run that is largely under the object line the score becomes positive, if the robot is completely avoiding the obstacle the score becomes negative, the closer to zero the value gets the more accurate the system is.

Equation 4 Accuracy Score Equation

$$\frac{\sum_{n=1}^{150}(x_{1n} - x_{2n})}{100}$$

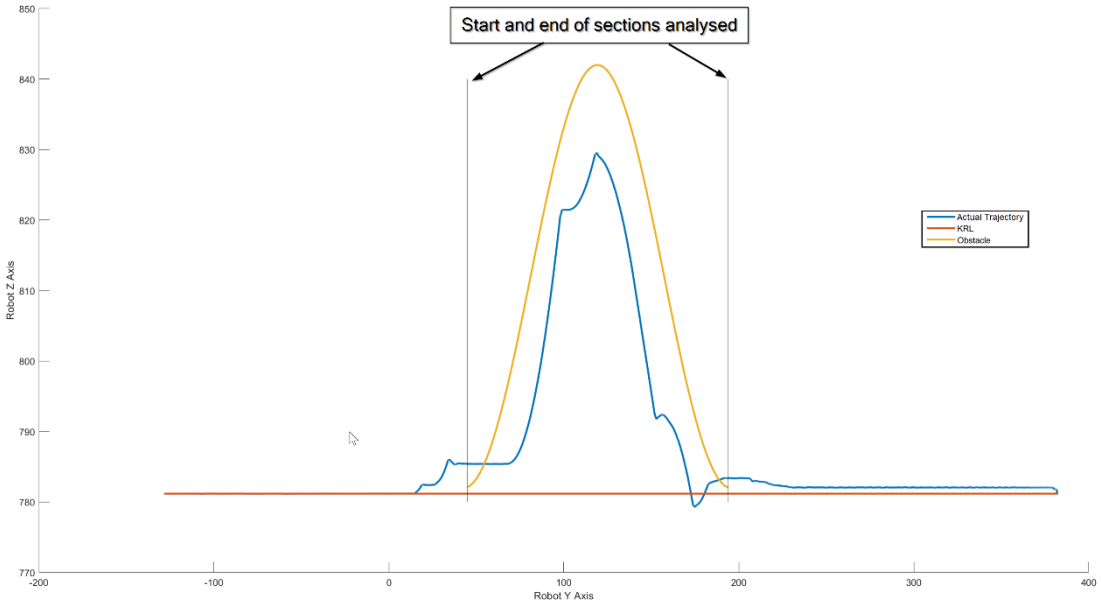


Figure 6-5 Example trajectory comparison

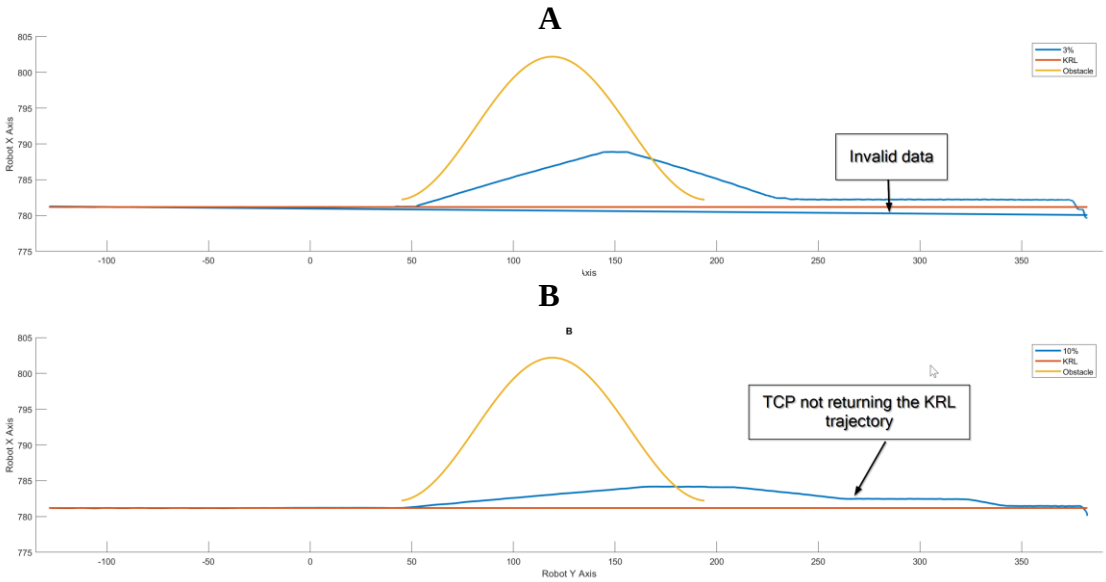


Figure 6-6 Invalid Data Example

The first set of experiments completed highlighted a few inaccuracies in the experimental design. The first being a slope in the Z axis between the start and end points of the LIN movement found in the KRL program which was used as a base to these experiments. Whilst a minor issue, it made visualising the experiments difficult because it did not accurately represent the surface of the podium used. Next a problem with the data logger was found, information from previous runs was carrying over to the next which contaminated the results. An example of this is highlighted in Figure 6-6a. Figure 6-6b also shows there was a problem with the calibration of the sensors. Too large of a gap between the sensor 1 and 2 triggering height resulted in the robot positioning itself too close to the surface at the initial run of the experiment and then too far away at the end of the movement. To resolve this problem, the start and finish positions of the run were adjusted to be closer to the centre of the table and performing dry runs of the experiment without any objects in between when calibrating the sensors to ensure they trigger where expected.

As discussed in chapter 6.2, three objects were designed for these experiments. Figure 6-5a shows a subset of results displaying the typical response from each obstacle. While these varied slightly through other parameters discussed later on they are largely representative of all results relating to the objects used as seen in Figure 6-7. Each experiment run resulted in the robot TCP being unable to maintain its fixed distance from the object. Despite this result in Figure 6-8, object 1 and 2 have movements

resembling the objects being tested even though they did not manage to fully avoid the obstacle. Object 3 on the other hand did not respond well, Figure 6-7c shows an example of a completed run. All remaining results are very similar which suggests that the system is incapable of repeat variations in a defined space or that the photoelectric sensors are simply not capable of responding in time. When analysing the results in Table 6-1, object one - the smallest object, scored the lowest, whereas these scores jumped significantly when testing the larger object 2, and then reduced for object 3. This suggests there is an upper limit to the size of corrections capable of being made by the adaptive system developed. Object 3 on the other hand produced low scores because the system tends to cut through the middle of the object. This shows that whilst these scores are useful in analysing each configuration used, without a visual understanding of the movements the scoring would produce inaccurate results.

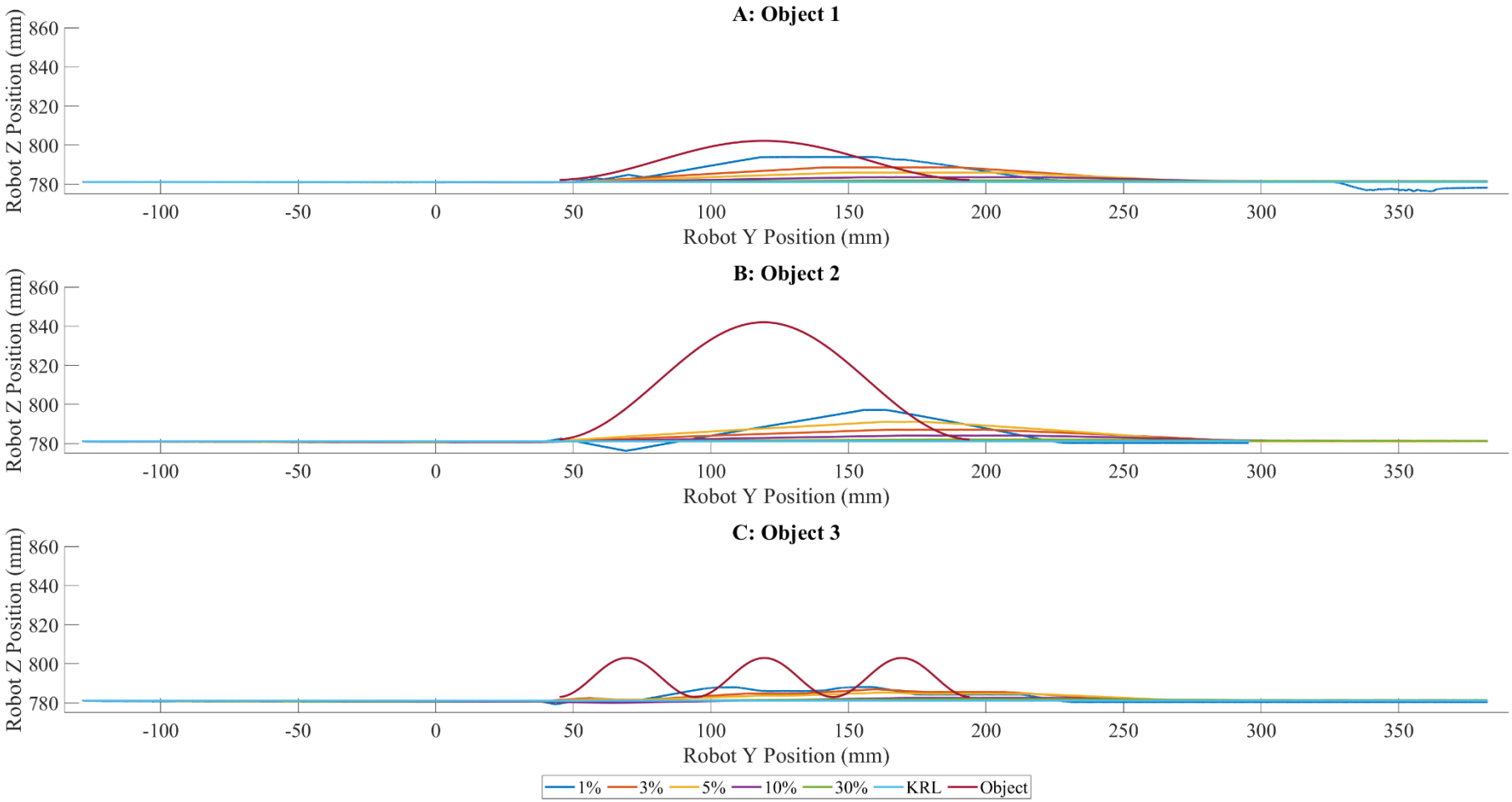


Figure 6-7 Example results of all objects using 0.01mm corrections

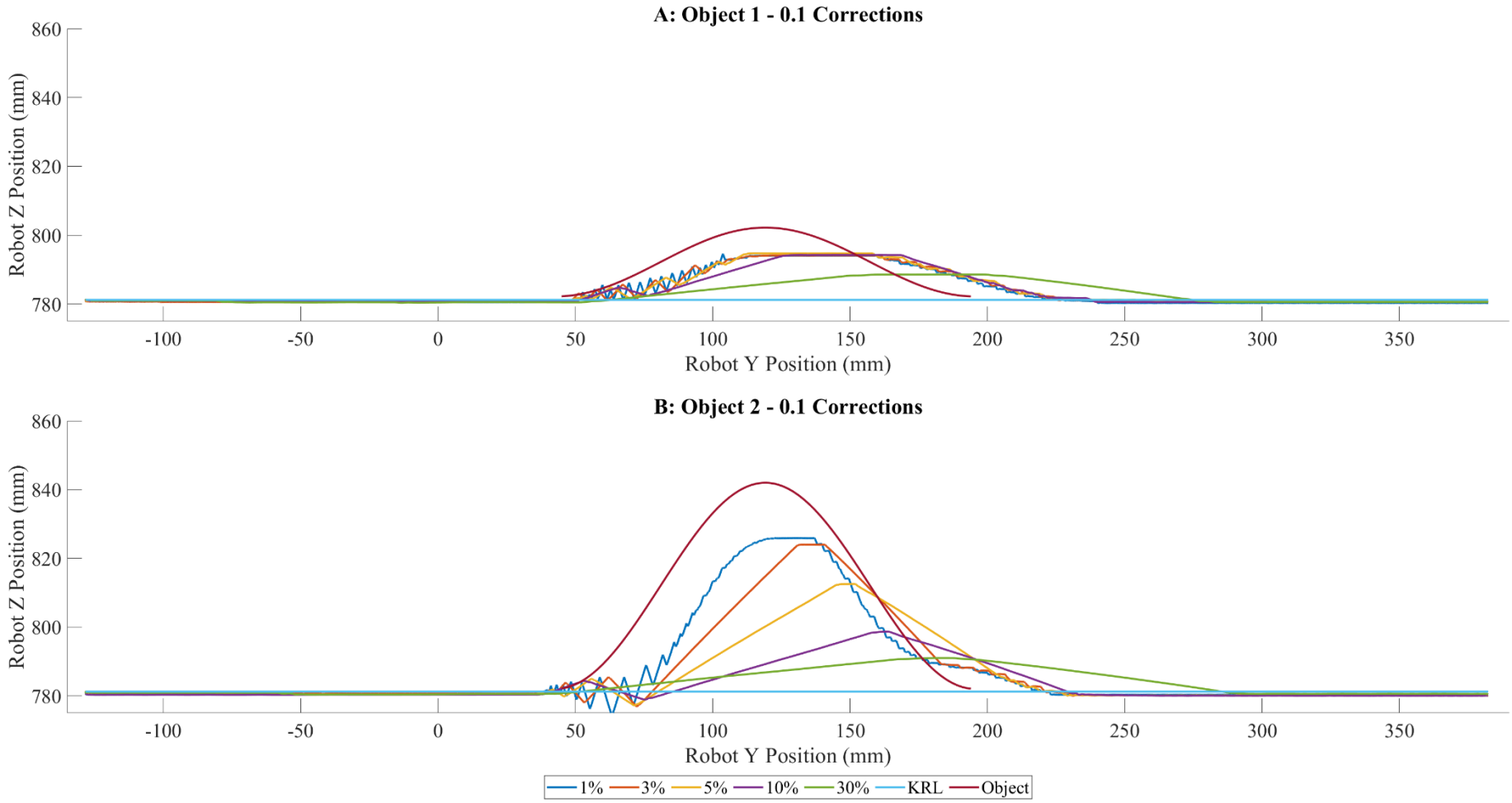


Figure 6-8 Example of object 1 and 2 - 0.1 corrections

The first observation made when comparing the various speeds tested is a trend between the speed of the robot and smoothness of the correction. At the lowest speed of 1% the robot produces a large number of fluctuations between the two sensors trying to correct its position. However, when increasing the speed of the robot these jagged lines smooth out but as a consequence the system is unable to move the robot TCP the same distances as it can at slower speeds. Figure 6-9 shows a comparison of using mount 1 with object 2 with a data rate of 0.01 mm, 0.05 mm and 0.1 mm. 8-9b and c both show that the system is reacting to the change in surface and does go some way into reacting effectively however there is still a gap of 10 mm between where the object is and where the TCP should be. When reviewing the scores of the experiments derived from equation 4 shown in Appendix A Table 9-4, overall 1% speed scored highest in 9 out of 15 groups of tests followed by 3% which scored highest in 3 out of the 15 tests.

In all experiments a data rate of 0.01mm provided too little of a change and does not allow the robot to compensate at a speed that matches the speed of the KRL program but provides a smoother trajectory. 0.05mm changes produced much better results as seen in Figure 8-9, however the higher speeds result in the trajectory curved flattening off. The 0.1 mm corrections provided the more accurate movements, however this in turn introduces a lot of instability in the movement at lower speeds. In particular, it was observed that movements become sharp and overcompensation is common.

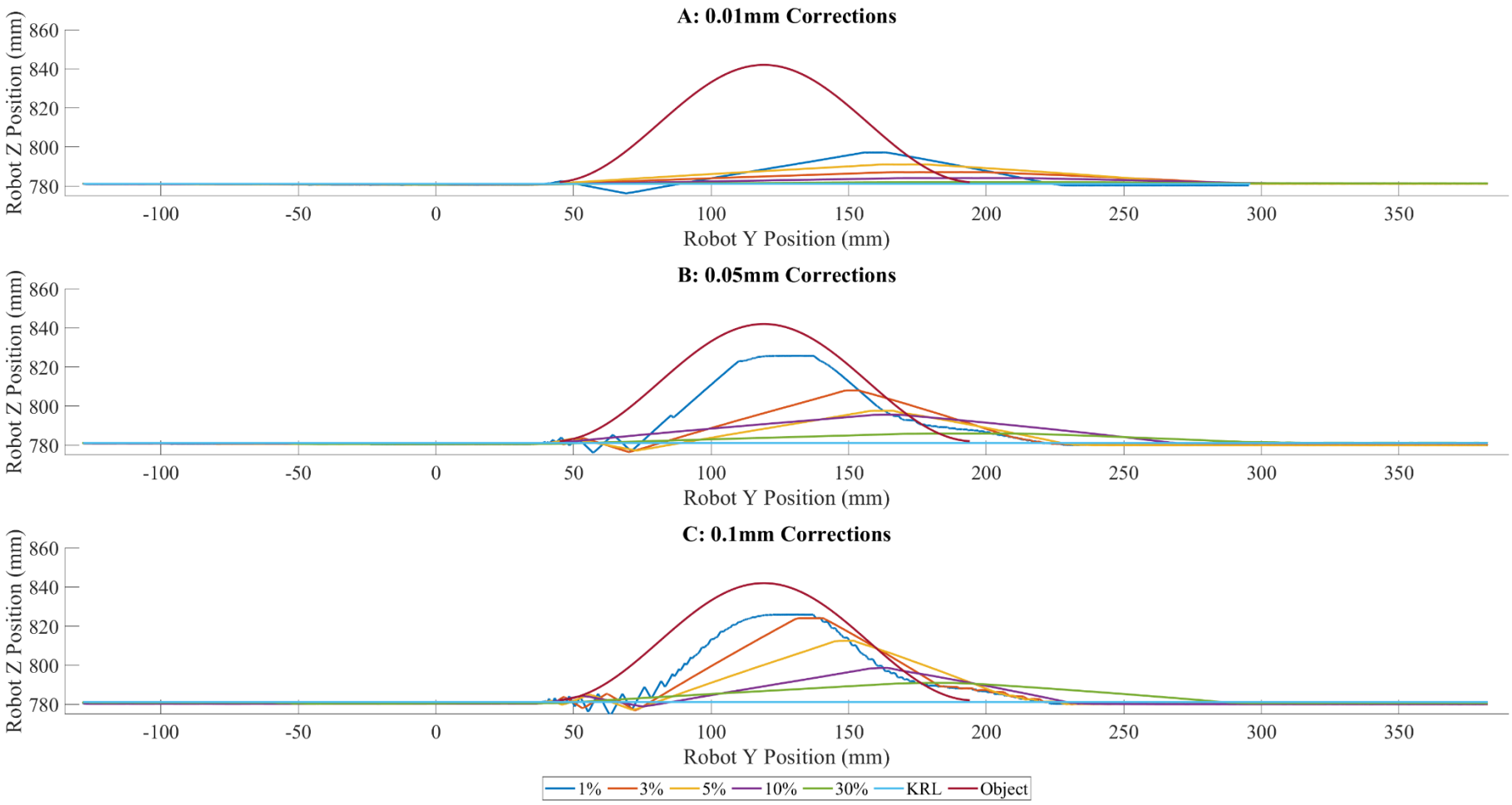


Figure 6-9 Mount 1 - Object 2 Speed Comparisons

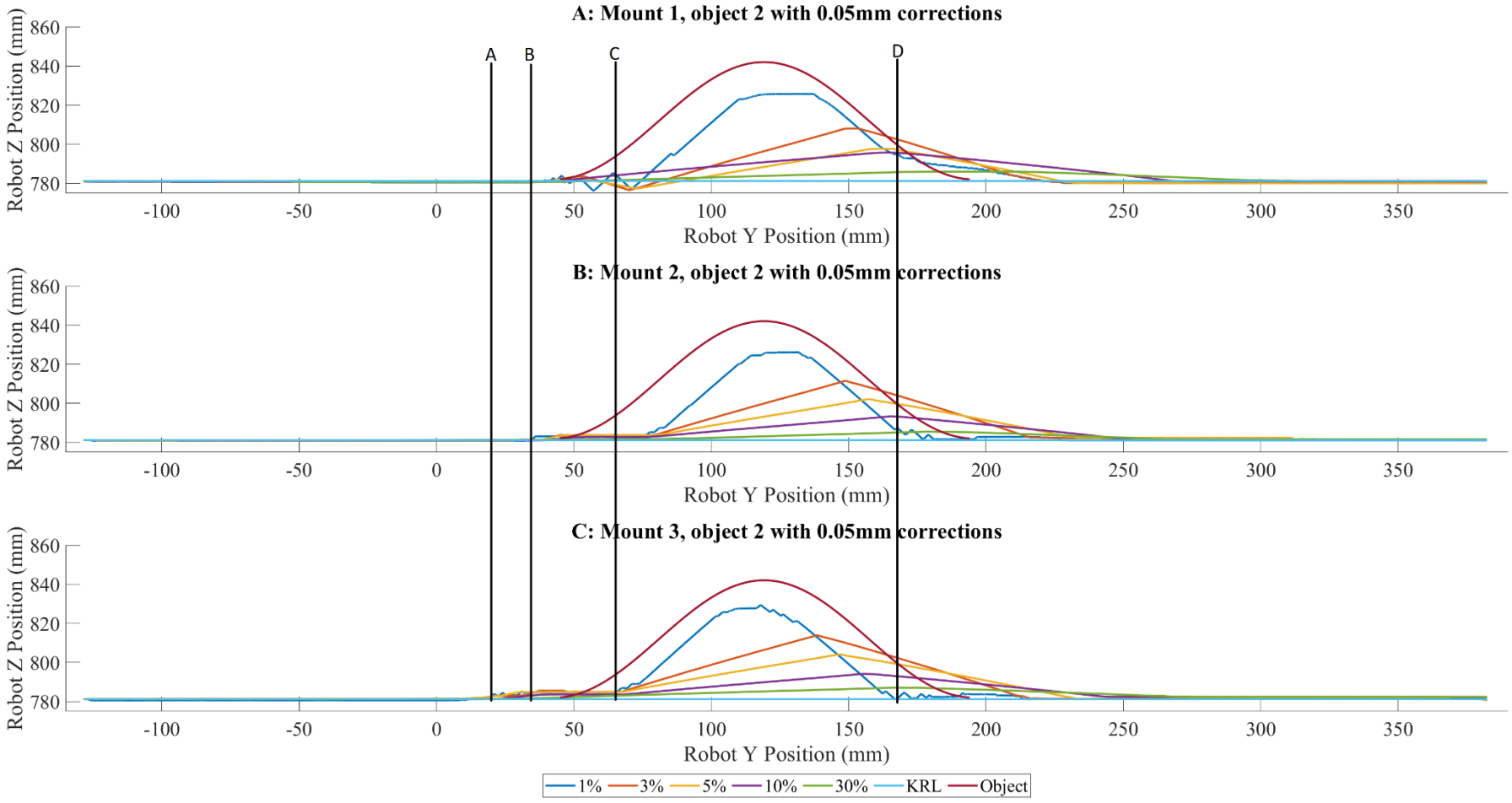


Figure 6-10 Mount 1, 2 and 3 compared

When comparing the three mounts, mount 3 seems to perform the best when observed visually but also when comparing the accuracy scores. In Figure 6-10c looking at line A, mount 3 responds before mounts 1 and 2. Again at line B you can see the sensors moving and the robot reacting to objects and finally at line C the motion taking place where as mounts 1 and 2 shown in Figure **6-10**a and b respond at a later point. The draw back here is at line D the sensors stop responding sooner than the other two mounts. Overall, sensor mount 3 provided consistently better accuracy scores which can be seen in Table 6-2. This suggests that there is a delay between sensing and the robot responding, so compensation needs to be factored in to get a better result.

Table 6-1 Mount comparison results

Obstacle	**Robot Speed**	**Algorithm**	**Data Rate**	**Accuracy Scores**		
				Adjacent	**Parallel**	**Parallel Offset**
2	1%	Simple	0.05	16.69	20.60	18.17
2	3%	Simple	0.05	29.51	25.03	20.91
2	5%	Simple	0.05	36.88	30.98	27.28
2	10%	Simple	0.05	33.90	37.44	35.46
2	30%	Simple	0.05	42.38	42.92	40.627

After completing the experiments using the simple algorithm it became apparent that a speed of 1% was needed to make any significant change in robot trajectory due to the fixed data rate limiting the maximum attainable change in distance. Any speed greater than 1% reduced this maximum value further. The smart algorithm was derived to attempt to overcome this issue, its function is to scale up the smart increment value based on the length of time the sensors were triggered. It also filters out any minor fluctuations in the sensor readings to smooth out any movements made, this is done by implementing a correction if the sensor is triggered two cycles consecutively. Early tests show the accuracy value is slightly lower with all mounts, but mount 3 again provided better results overall, this can be seen in Figure 6-11. This algorithm operates on two values, these being the delay value and change increment value. The delay value operates to slow the increment value changing. If this was not used the change value would almost instantly get to its maximum value before the sensor system cycled. The change increment value is simply the value added to the value sent to the robot on every

cycle the sensors are active. Whereas originally a fixed value was sent every time the sensors were triggered, when consecutively triggered this would raise in 0.01 steps. A start value of 0.01 was used to provoke a smoother response with a maximum value of 0.1.

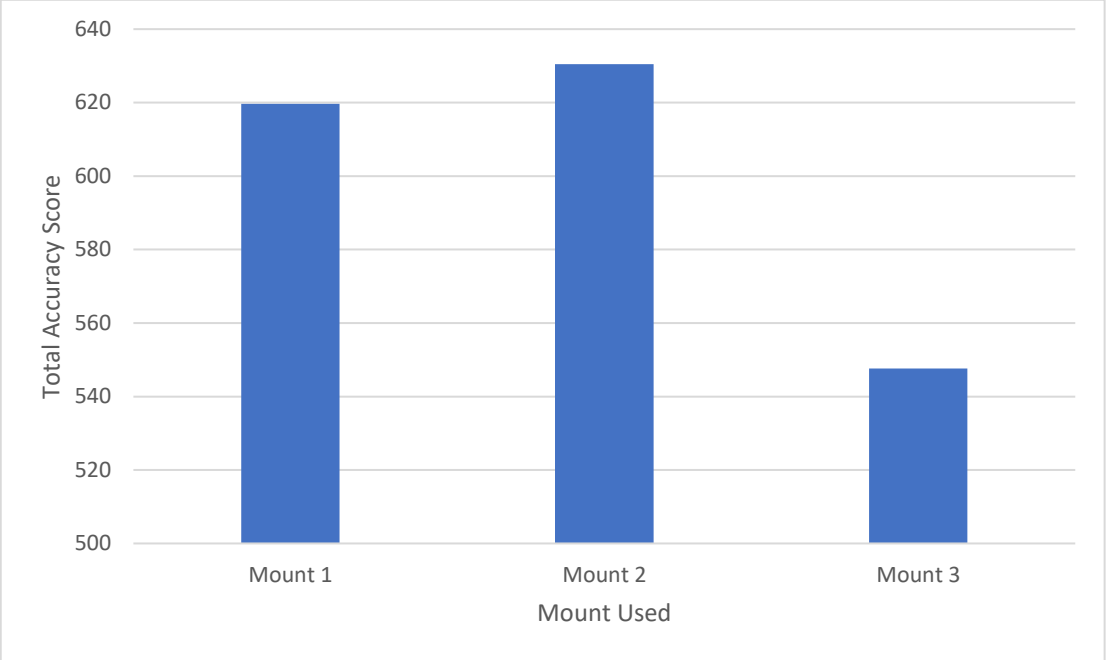


Figure 6-11 Smart algorithm results compared

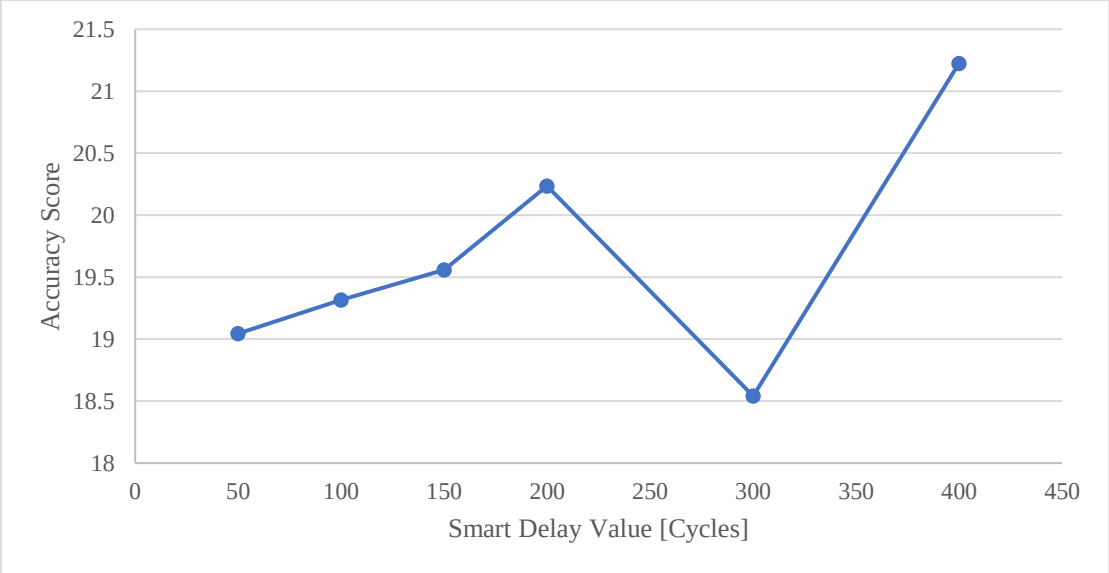


Figure 6-12 Further smart delay experiment results using mount 3 [Cycles]

When looking at the accuracy results in Appendix A Table 9-2, a delay value of 100 cycles slightly improves upon the results of the 500-cycle delay. A cycle being the

number of iterations in the algorithm loop. This suggested that this variable could be fine-tuned to offer better results. Further tests were then run comparing additional delays of 50, 100, 150, 200, 300 and 400 using a small subset of parameters such as a single mount and object. The results shown in Table 9-3 and Figure 6-12 show an odd pattern. As the delay value gets greater the result value gets higher meaning further deviation from the desired trajectory, except for the value of 300 which sees a partial drop in its results value suggesting this is the optimum value. When looking visually studying the results shown in Figure 6-13, whilst 300 may offer a more accurate movement, the shape of the curve is pointed. Visually a value of 150 appears to be closer to the shape of the object and a smoother trajectory. Figure 6-13 shows that the higher the delay meant the robot was responding to the object too slow resulting in the case of the 400 value as almost missing the object.

Table 6-2 shows the results of testing individual increment values. The results show that the larger the increment value is the better the accuracy value, however the larger the number the higher frequency of fluctuations can be seen in the trajectory, a trend as seen in experiments previously discussed. When studying the Figure 6-14 the value of 0.02 seemed to offer the smoothest movement resulting in a movement closest to the object being avoided.

Table 6-2 Increment values results

Obstacle	Robot Speed	Algorithm	Data Rate	Smart Delay	Smart Increment	Score
2	3	Smart	0.01	100	0.01	19.16
2	3	Smart	0.01	100	0.02	18.66
2	3	Smart	0.01	100	0.05	18.84
2	3	Smart	0.01	100	0.1	18.20

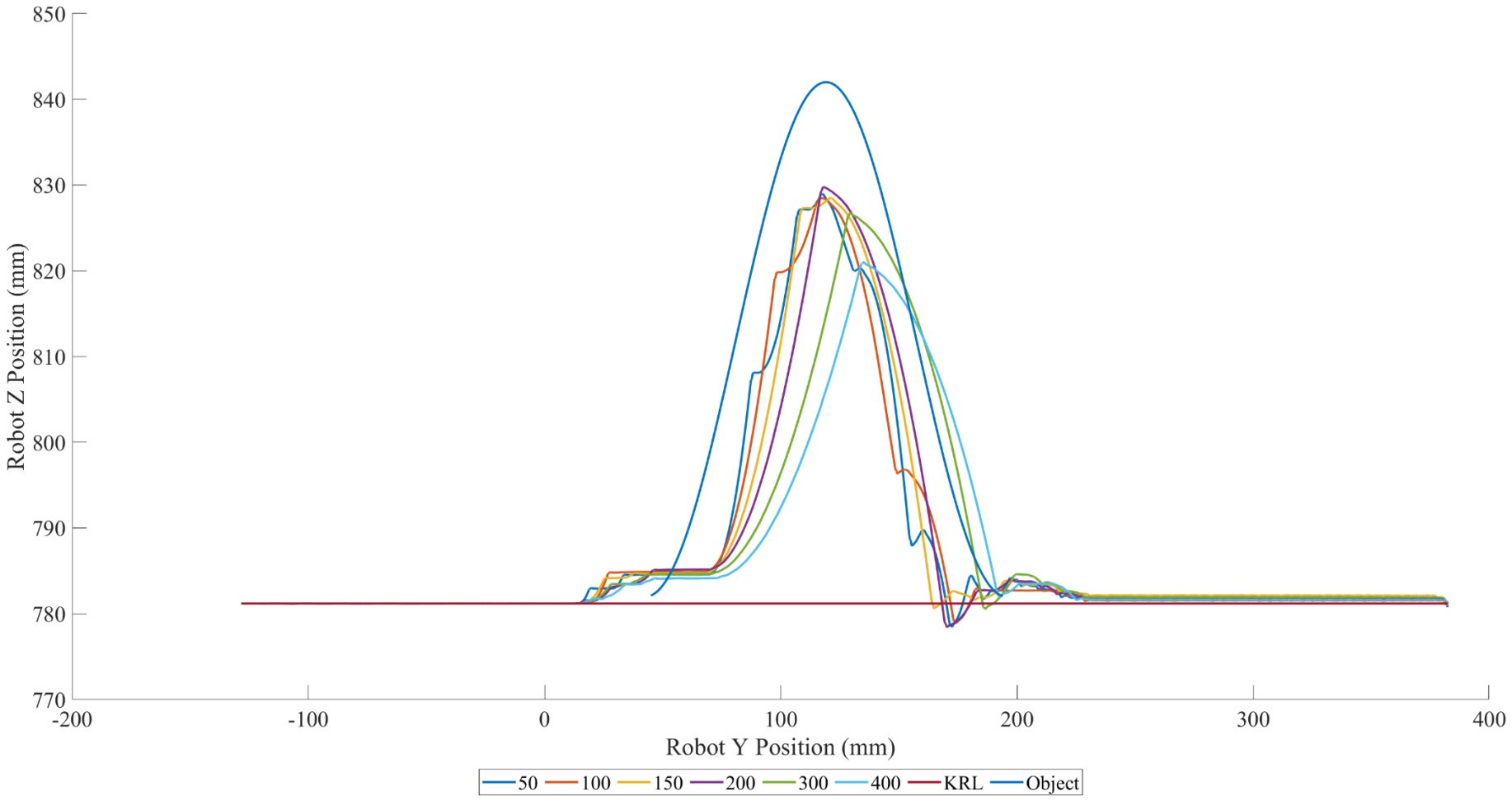


Figure 6-13 Delay Values Compared

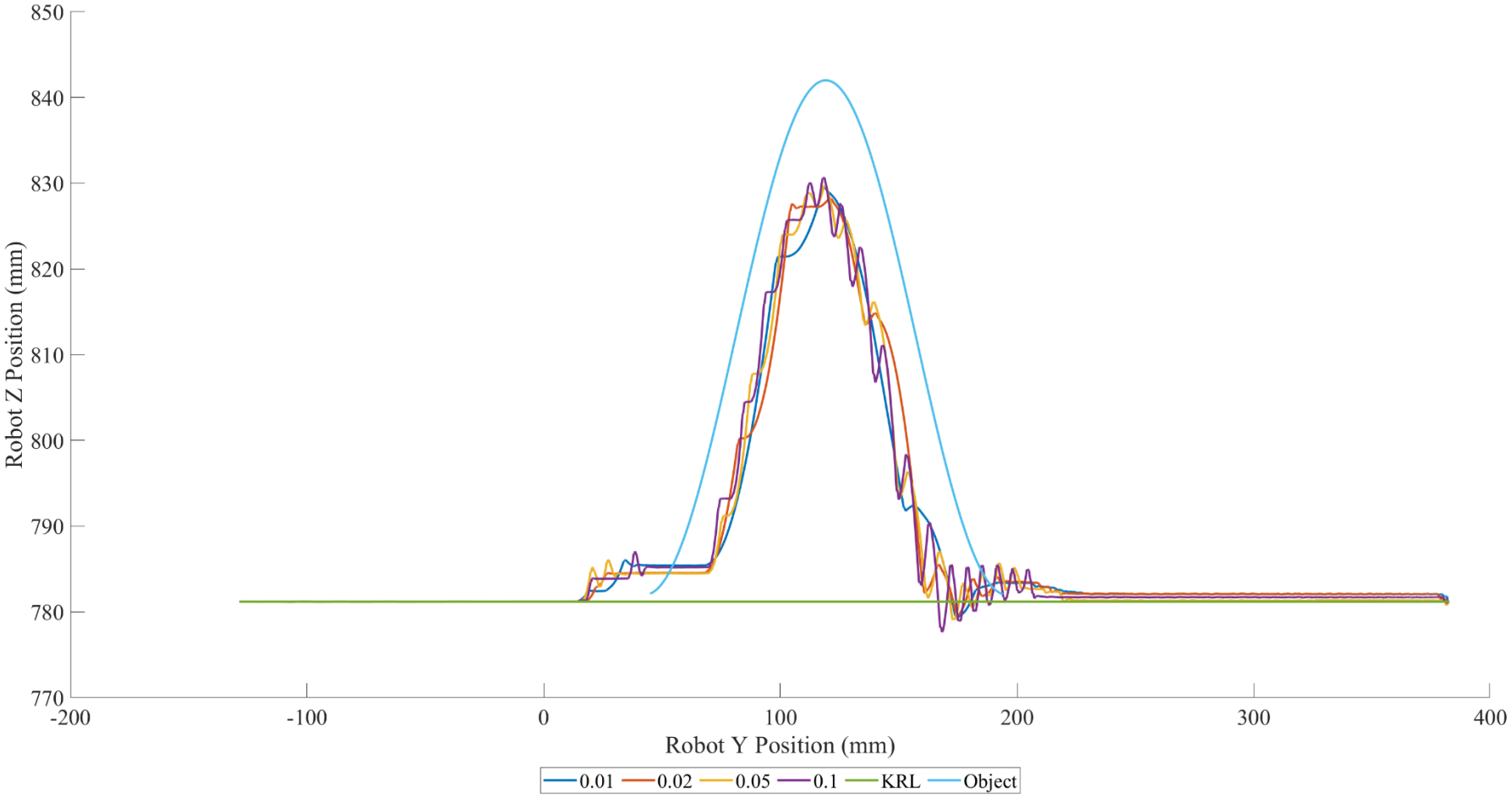


Figure 6-14 Increment Values Compared

6.4 Summary

This section of optimisation has largely been successful and has given some insights into the ideal parameters for the smart delay, smart increment and data rate. Two common trends were identified when reviewing the results of the parameter tested during experimentation, the greater the value sent in a single cycle to move the robot the closer to the desired coordinates the robot moved. However, the greater these values the greater the instability in the associated movement. When consecutive movements are made any change in axis direction results in a sharp movement often resulting in a torque error on the robot. A balance is needed between the robots overall operating speed and the data rate value used to influence the robot's motion. Introducing the smart algorithm mitigates some of these issues by scaling the data rate based on movement duration. Whilst the smart algorithm has improved the sensor system responses, it is unlikely to solve the main problem where the corrected trajectory of the robot is not capable of navigating around the object. The peak of each object is typically where the issue is observed. Figure 6-15 shows a 11.1 mm gap between the object and the trajectory. Further tests need to be conducted to try and isolate this issue. Additional materials as the target surface need to be tested as infrared absorption could be what is preventing the sensors from reacting appropriately. Aluminium is the most reflective surface for the IR sensors so a round of experiments will be conducted using this as a surface. If this can be removed then a selection of experiments will be re-run to validate previous findings. Then a final set of experiments will be conducted used the optimum values found during this phase of optimisation.

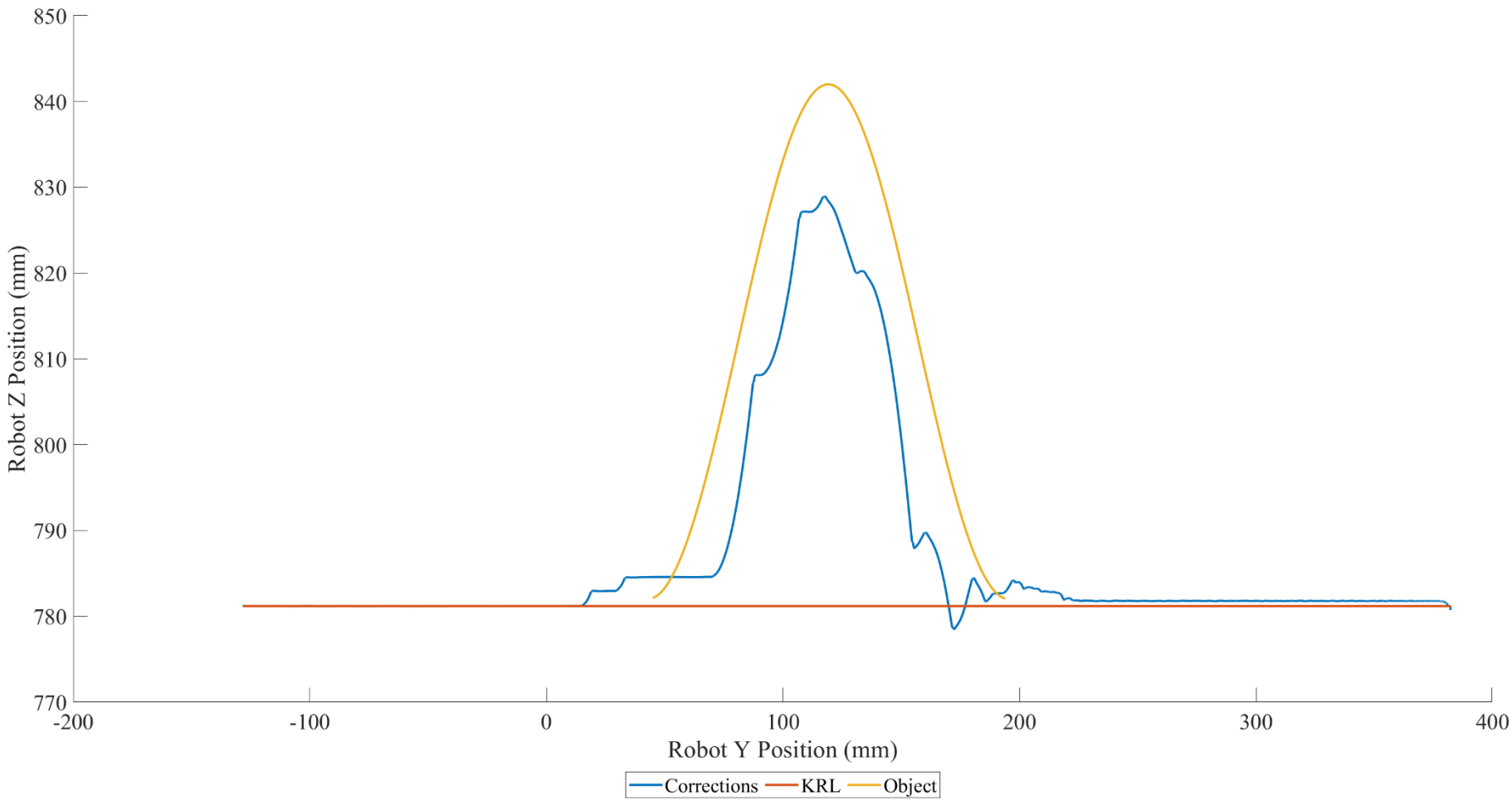


Figure 6-15 Trajectory Gap

7. CONCLUSIONS

This thesis is the culmination of study and development towards the goal of creating a robot end effector position varying system that is capable of maintaining a specified distance between the robot TCP and a target work surface. The aim of this research was to develop a novel concept to allow an industrial robot to adapt to deformations in a surface material to maintain a specific distance between a robot end effector using photoelectric diffuse sensors. The system needed to be able to adapt to an unknown geometry or surface that has inconsistent surface levels. For a better understanding of industrial robotics an in-depth study was taken into its history with particular focus on articulated robots and robot applications. Then to gain familiarity with the research laboratory resources, a study of KUKA robots and established robot operations was collated. Existing methods of interfacing peripheral technology with a KUKA robot was investigated, and research into software and hardware requirements with an overview of different types of sensors applicable to this research given was completed. Using a KUKA KR16-2, a six-axis articulated robot; a system was designed and tested utilising a developed Python program that can interface two photoelectric diffuse sensors and a KUKA C4 controller via a computer utilising the KUKA Robot Sensor Interface software.

To test the viability of this system a series of experiments was completed using the KUKA KR16-2 robot paired with a KUKA C4 controller. This system was taken and applied in a laboratory setting. Results of these experiments was recorded using another developed Python program that was written to read the values received from the KUKA Robot Sensor Interface and record them to a file. This file then contains robot movement data captured whilst the robot is in full operation. This data was then evaluated using a set of custom functions created in Matlab to visualise robot movements. A scoring system was developed using the robot positional data to work in combination with these visualisations in order to identify which experiments performed best and closest to the expected outcome.

The key findings from the experiments undertaken were:

- A software package was developed with capabilities to influence a KUKA KR16-2 robots movements based off sensory input from two photoelectric diffuse sensors. The package was written in Python and designed to be light weight and compatible with low cost computer systems such as a Raspberry Pi. This system is designed to be integrated into existing manufacturing environments with ease

and can be operated with digital sensors.

- An algorithm was developed to dynamically calculate values being communicated to the KUKA robot. These values are determined by the combination and duration of the sensor values being supplied. Two key values are used to configure this algorithm, a delay value which represents the number of iterations the system logic loops through before outputting and an increment value which is a float value that is added to the correction value that is then sent to the robot. This algorithm was tested against a simpler version that contained static correction values to determine effectiveness and was determined to provide a higher rate of accuracy when moving the robot to its required position to avoid an obstacle.
- An optimum increment value of 0.05 mm was identified through 240 iterations of experiments when compares against values of 0.01, 0.02 and 0.1 mm. Additionally, an optimum delay value of 200 algorithm cycle iterations was applied in the same set of experiments where values of 50, 100, 150, 200, 300 and 400 were tested. The increment and delay value results were evaluated by means of visual inspection and a scoring system that was created for this project.
- Comparing the KUKA robot operating speed, percentages of 1, 3, 5,10 and 30% of 2 m/s was tested. A speed of 0.1 m/s was identified as the optimum speed for the robot to operate at when using the proposed positioning system presented in this thesis. 0.1 m/s offered a balance between speed and accuracy when compared to the other values.
- Through testing three different sensor mounts it was identified that positioning the systems sensors at a distance of 5.7 mm ahead of the tool centre point resulted in corrections closer to the desired trajectory of the robot over sensors targeted directly at the TCP.
- To further understand the robot movements, a program to record robot path operations for the use with KUKA robots supporting the Robot Sensor Interface software was developed. The application is capable of producing detailed value files and the Matlab functions can create plots from these files to view the robot's movement and to compare joint positions from multiple programs.

The results of these experiments showed the position adaption system worked as expected despite the system underperforming due to the sensors utilised. Using photoelectric

diffuse sensors proved to not provide enough accuracy when detecting surfaces for the system to work at its optimum, they demonstrated the system works as intended and showed it would still find a place for a wide range of industrial applications. Photoelectric sensors had been used as a low-cost solution in an environment where resources were limited. There are however sensors on the market that would offer better accuracy and could work very well in conjunction with the system designed. Further work could be undertaken to investigate this. With the existing sensor system additional research can be conducted to provide a more defined specification as to which materials the photoelectric sensors work best with. Additional obstacle such as cylindrical objects can be tested to further verify the ability of this. Finally, the system can be tested in conjunction with one of its intended applications such as plastic welding or spray-painting surfaces. This would be to verify the system's ability to successfully compensate when actively welding or painting and object.

7.1 Future work

The KUKA robot logging software developed in this thesis is extremely valuable for assisting current and future research that utilise a KUKA robot. Its ability to record KUKA robot movements in conjunction with the functions in Matlab for analysis will complement future work well by giving researchers a better way of visualising robot movements. This software lends itself to aiding research in many other areas of robotics. Future development of the position adaption system would focus around applying the system to real world applications such as hot gas welding or spray painting to assess the viability of it and to further refine the system and algorithm. There is scope to apply different sensors more suited to the application intended and other industrial applications outside of the initial areas considered. Once a real-world application is validated future work in machine learning can be applied. With the data logging features developed in this thesis, data sets containing information on industrial processes could be built and analysed. This information can then be applied using machine learning and neural net algorithms which could be integrated in to the adaption system, enabling a degree of artificial intelligence. Further following the Industry4.0 trends, smart factory functionality could be added including real-time data communication.

8. BILBIOGRAPHY

[1] D. M. Fryer and J. C. Marshall, 'The Motives of Jacques de Vaucanson', *Technology and Culture*, vol. 20, no. 2, pp. 257–269, 1979, doi: 10.2307/3103866.

[2] J. Riskin, 'The Defecating Duck, or, the Ambiguous Origins of Artificial Life', *Critical Inquiry*, vol. 29, no. 4, pp. 599–633, Jun. 2003, doi: 10.1086/377722.

[3] E. J. Hobsbawm, *The age of revolution 1789-1848*, 1st Vintage Books ed. New York: Vintage Books, 1996.

[4] P. Corke, *Robotics, Vision and Control*, 2nd ed. Springer International Publishing.

[5] J. G. C. Devol, 'Programmed article transfer', US2988237A, Jun. 13, 1961.

[6] 'Japanese robots at Expo 2005 Aichi', Mar. 06, 2006. https://web.archive.org/web/20060306224452/http://int.kateigaho.com/spr05/robots.html (accessed May 15, 2020).

[7] 'The Stanford Arm'. http://infolab.stanford.edu/pub/voy/museum/pictures/display/1-Robot.htm (accessed May 15, 2020).

[8] 'History of KUKA: Automation then and now', *KUKA AG*. https://www.kuka.com/en-gb/about-kuka/history (accessed May 17, 2020).

[9] 'What is Automation?- ISA', *International Standards of Automation - What is automation?* https://www.isa.org/about-isa/what-is-automation/ (accessed May 18, 2020).

[10] R. O. M. T. POSTED 06/12/2018, 'Top 6 Future Trends in Robotic Automation', *Robotics Online*. https://www.robotics.org/blog-article.cfm/Top-6-Future-Trends-in-Robotic-Automation/101 (accessed May 18, 2020).

[11] IFR, 'Top Trends Robotics 2020', *IFR International Federation of Robotics*. https://ifr.org/ifr-press-releases/news/top-trends-robotics-2020 (accessed May 18, 2020).

[12] 'Automation - Manufacturing applications of automation and robotics', *Encyclopedia Britannica*. https://www.britannica.com/technology/automation (accessed May 31, 2020).

[13] 'What is Industrial Automation | Types of Industrial Automation', *ELECTRICAL TECHNOLOGY*, Sep. 26, 2015. https://www.electricaltechnology.org/2015/09/what-is-industrial-automation.html (accessed May 31, 2020).

[14] R. O. M. T. POSTED 11/28/2017, '7 Common Types of Robotic Welding Processes and When They're Used', *Robotics Online*. https://www.robotics.org/blog-article.cfm/7-Common-Types-of-Robotic-Welding-Processes-and-When-They-re-Used/72 (accessed May 18, 2020).

[15] Y. Li, Y. F. Li, Q. L. Wang, D. Xu, and M. Tan, 'Measurement and Defect Detection of the Weld Bead Based on Online Vision Inspection', *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 7, pp. 1841–1849, Jul. 2010, doi: 10.1109/TIM.2009.2028222.

[16] K. Suwanratchatamanee, M. Matsumoto, and S. Hashimoto, 'Robotic Tactile Sensor System and Applications', *IEEE Transactions on Industrial Electronics*, vol. 57, no. 3, pp. 1074–1087, Mar. 2010, doi: 10.1109/TIE.2009.2031195.

[17] Y. Feng, Z. Chen, D. Wang, J. Chen, and Z. Feng, 'DeepWelding: A Deep Learning Enhanced Approach to GTAW Using Multisource Sensing Images', *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 465–474, Jan.

2020, doi: 10.1109/TII.2019.2937563.
[18] P. Kushwah, ‘Plastic Welding Technique’, 3, 2019. [Online]. Available: www.ijtre.com.
[19] ‘Hot gas end effector’, *Valk Welding Website*, 2018. https://www.valkwelding.com/en/news/welding-of-plastic (accessed Jan. 16, 2020).
[20] ‘Eugen Riexinger GmbH & Co. KG Robot Milling and Welding’, *YouTube*. https://www.youtube.com/channel/UCQqMA-Ml822uwJdYx1L81Ww (accessed Jan. 16, 2020).
[21] A. S. Vempati *et al.*, ‘PaintCopter: An Autonomous UAV for Spray Painting on Three-Dimensional Surfaces’, *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2862–2869, Oct. 2018, doi: 10.1109/LRA.2018.2846278.
[22] Z. Meng, C. Li, G. Li, J. Zhao, and J. Yan, ‘Research of positioning method for automatic spraying on large ship block surfaces’, in *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, Jun. 2016, pp. 431–436, doi: 10.1109/RCAR.2016.7784068.
[23] R. Chen, G. Wang, J. Zhao, J. Xu, and K. Chen, ‘Fringe Pattern Based Plane-to-Plane Visual Servoing for Robotic Spray Path Planning’, *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 3, pp. 1083–1091, Jun. 2018, doi: 10.1109/TMECH.2017.2747084.
[24] M. E. Helou, S. Mandt, A. Krause, and P. Beardsley, ‘Mobile Robotic Painting of Texture’, in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 640–647, doi: 10.1109/ICRA.2019.8793947.
[25] P. J. From, J. Gunnar, and J. T. Gravdahl, ‘Optimal Paint Gun Orientation in Spray Paint Applications—Experimental Results’, *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, pp. 438–442, Apr. 2011, doi: 10.1109/TASE.2010.2089450.
[26] Q. Yu, G. Wang, and K. Chen, ‘A robotic spraying path generation algorithm for free-form surface based on constant coating overlapping width’, in *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, Jun. 2015, pp. 1045–1049, doi: 10.1109/CYBER.2015.7288089.
[27] ‘Experimenting with robots for grocery picking and packing’, *Ocado Technology*. https://www.ocadotechnology.com/blog/2019/1/14/experimenting-with-robots-for-grocery-picking-and-packing (accessed May 25, 2020).
[28] J. I. Ngadimin, F. I. Hariadi, and M. I. Arsyad, ‘Design and implementation of 3D motion control of small scale pick and place surface-mount technology machine’, in *2017 International Symposium on Electronics and Smart Devices (ISESD)*, Oct. 2017, pp. 95–100, doi: 10.1109/ISESD.2017.8253312.
[29] W. Zhang, J. Mei, and Y. Ding, ‘Design and Development of a High Speed Sorting System Based on Machine Vision Guiding’, *Physics Procedia*, vol. 25, pp. 1955–1965, Jan. 2012, doi: 10.1016/j.phpro.2012.03.335.
[30] Z. He, Z. Li, and J. Ma, ‘Research on a high-speed picking-placing motion of the sorting robot based on the optimal picking point’, in *2016 Chinese Control and Decision Conference (CCDC)*, May 2016, pp. 5148–5153, doi: 10.1109/CCDC.2016.7531917.
[31] ‘Aussies Win Amazon Robotics Challenge - IEEE Spectrum’, *IEEE Spectrum: Technology, Engineering, and Science News*. https://spectrum.ieee.org/automaton/robotics/industrial-robots/aussies-win-amazon-robotics-challenge (accessed May 25, 2020).

[32] A. Zeng *et al.*, ‘Multi-view self-supervised deep learning for 6D pose estimation in the Amazon Picking Challenge’, in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1386–1383, doi: 10.1109/ICRA.2017.7989165.
[33] M. Hakozaki, H. Oasa, and H. Shinoda, ‘Telemetric robot skin’, in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, May 1999, vol. 2, pp. 957–961 vol.2, doi: 10.1109/ROBOT.1999.772431.
[34] Y. Yamada, T. Morizono, Y. Umetani, and H. Takahashi, ‘Highly soft viscoelastic robot skin with a contact object-location-sensing capability’, *IEEE Transactions on Industrial Electronics*, vol. 52, no. 4, pp. 960–968, Aug. 2005, doi: 10.1109/TIE.2005.851654.
[35] ‘Cobots Take Over Undesirable Tasks, Optimize Assembly and Packaging by 30%’. https://www.youtube.com/watch?v=8e76BjH9ez4 (accessed Jun. 02, 2020).
[36] ‘Techman Robot Automatic Packing and Boxing’. https://www.youtube.com/watch?v=X-WmAatNyrA (accessed Jun. 02, 2020).
[37] A. Owen-Hill, ‘Cobots in Packaging: The State of the Industry in 2018’. https://blog.robotiq.com/cobots-in-packaging-the-state-of-the-industry-2018 (accessed Jun. 02, 2020).
[38] ‘1: Introduction | Human-Robot Interaction’. https://humanrobotinteraction.org/1-introduction/ (accessed May 26, 2020).
[39] X. Lamy, F. Collédani, F. Geffard, Y. Measson, and G. Morel, ‘Overcoming human force amplification limitations in comanipulation tasks with industrial robot’, in *2010 8th World Congress on Intelligent Control and Automation*, Jul. 2010, pp. 592–598, doi: 10.1109/WCICA.2010.5553839.
[40] ‘Ocean-fact-sheet-package.pdf’. Accessed: May 26, 2020. [Online]. Available: https://www.un.org/sustainabledevelopment/wp-content/uploads/2017/05/Ocean-fact-sheet-package.pdf.
[41] ‘Special Purpose Underwater Research Vehicle (SPURV)’. http://www.navaldrones.com/SPURV.html (accessed May 26, 2020).
[42] J. Yuh, ‘Design and Control of Autonomous Underwater Robots: A Survey’, *Autonomous Robots*, vol. 8, no. 1, pp. 7–24, Jan. 2000, doi: 10.1023/A:1008984701078.
[43] S. Negahdaripour, X. Xu, A. Khamene, and Z. Awan, ‘3-D motion and depth estimation from sea-floor images for mosaic-based station-keeping and navigation of ROVs/AUVs and high-resolution sea-floor mapping’, in *Proceedings of the 1998 Workshop on Autonomous Underwater Vehicles (Cat. No.98CH36290)*, Aug. 1998, pp. 191–200, doi: 10.1109/AUV.1998.744455.
[44] ‘Seafloor mapping AUV’, *MBARI*, Dec. 02, 2015. https://www.mbari.org/at-sea/vehicles/autonomous-underwater-vehicles/seafloor-mapping-auv/ (accessed May 26, 2020).
[45] K. Zwolak *et al.*, ‘An unmanned seafloor mapping system: The concept of an AUV integrated with the newly designed USV SEA-KIT’, in *OCEANS 2017 - Aberdeen*, Jun. 2017, pp. 1–6, doi: 10.1109/OCEANSE.2017.8084899.
[46] P. Norgren and R. Skjetne, ‘Using Autonomous Underwater Vehicles as Sensor Platforms for Ice-Monitoring’, *MIC*, vol. 35, no. 4, pp. 263–277, 2014, doi: 10.4173/mic.2014.4.4.
[47] N. Barrett, J. Seiler, T. Anderson, S. Williams, S. Nichol, and N. Hill, ‘Autonomous Underwater Vehicle (AUV) for mapping marine biodiversity in

coastal and shelf waters: Implications for marine management', Jun. 2010, pp. 1–6, doi: 10.1109/OCEANSSYD.2010.5603860.
[48] D. Haulsee, M. Breece, D. Miller, B. Wetherbee, D. Fox, and M. Oliver, 'Habitat selection of a coastal shark species estimated from an autonomous underwater vehicle', *Mar. Ecol. Prog. Ser.*, vol. 528, pp. 277–288, May 2015, doi: 10.3354/meps11259.
[49] J. Vincent, 'Welcome to the automated warehouse of the future', *The Verge*, May 08, 2018. https://www.theverge.com/2018/5/8/17331250/automated-warehouses-jobs-ocado-andover-amazon (accessed May 31, 2020).
[50] 'Your First Look Inside Amazon's Robot Warehouse of Tomorrow', *Wired*.
[51] N. C. Hou, L. W. Han, and L. M. Kuan, 'Optimising Search Operations with Swarm Intelligence', in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Nov. 2019, pp. 1993–1997, doi: 10.1109/APSIPAASC47483.2019.9023211.
[52] E. R. Magsino, F. A. V. Beltran, H. A. P. Cruzat, and G. N. M. De Sagun, 'Simulation of search-and-rescue and target surrounding algorithm techniques using Kilobots', in *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*, Apr. 2016, pp. 70–74, doi: 10.1109/ICCAR.2016.7486701.
[53] S. M. Thayer and S. P. N. Singh, 'Development of an immunology-based multi-robot coordination algorithm for exploration and mapping domains', in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2002, vol. 3, pp. 2735–2739 vol.3, doi: 10.1109/IRDS.2002.1041683.
[54] P. Pirjanian and M. Mataric, 'Multi-robot target acquisition using multiple objective behavior coordination', in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, Apr. 2000, vol. 3, pp. 2696–2702 vol.3, doi: 10.1109/ROBOT.2000.846435.
[55] Y. Meng and J. Gan, 'A distributed swarm intelligence based algorithm for a cooperative multi-robot construction task', in *2008 IEEE Swarm Intelligence Symposium*, Sep. 2008, pp. 1–6, doi: 10.1109/SIS.2008.4668296.
[56] R. L. Stewart and R. A. Russell, 'A Distributed Feedback Mechanism to Regulate Wall Construction by a Robotic Swarm', *Adaptive Behavior*, vol. 14, no. 1, pp. 21–51, Mar. 2006, doi: 10.1177/105971230601400104.
[57] 'Spot® | Boston Dynamics'. https://www.bostondynamics.com/spot (accessed May 31, 2020).
[58] O. R. Institute, 'Legged Robots', *Oxford Robotics Institute*. https://ori.ox.ac.uk/theme/legged-robots/ (accessed May 31, 2020).
[59] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, 'TossingBot: Learning to Throw Arbitrary Objects with Residual Physics', 2019, Accessed: May 27, 2020. [Online]. Available: https://tossingbot.cs.princeton.edu/.
[60] J. Tan *et al.*, 'Sim-to-Real: Learning Agile Locomotion For Quadruped Robots', *arXiv:1804.10332 [cs]*, May 2018, Accessed: May 27, 2020. [Online]. Available: http://arxiv.org/abs/1804.10332.
[61] O. Motors, 'What is the Smart Factory and its Impact on Manufacturing?', *OTTO Motors*, May 29, 2020. https://www.ottomotors.com/blog/what-is-the-smart-factory-manufacturing (accessed May 30, 2020).
[62] D. R. Sjödin, V. Parida, M. Leksell, and A. Petrovic, 'Smart Factory Implementation and Process Innovation', *Research-Technology Management*, vol. 61, no. 5, pp. 22–31, Sep. 2018, doi: 10.1080/08956308.2018.1471277.

[63] Y. Luo, Y. Duan, W. Li, P. Pace, and G. Fortino, ‘A Novel Mobile and Hierarchical Data Transmission Architecture for Smart Factories’, *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3534–3546, Aug. 2018, doi: 10.1109/TII.2018.2824324.
[64] F. Shrouf, J. Ordieres, and G. Miragliotta, ‘Smart factories in Industry 4.0: A review of the concept and of energy management approached in production based on the Internet of Things paradigm’, in *2014 IEEE International Conference on Industrial Engineering and Engineering Management*, Dec. 2014, pp. 697–701, doi: 10.1109/IEEM.2014.7058728.
[65] A. Sanna and F. Manuri, ‘A Survey on Applications of Augmented Reality’, *Advances in Computer Science : an International Journal*, vol. 5, no. 1, Art. no. 1, Jan. 2016.
[66] M. Filipenko, A. Angerer, A. Hoffmann, and W. Reif, *Opportunities and Limitations of Mixed Reality Holograms in Industrial Robotics*. 2020.
[67] R. Masoni *et al.*, ‘Supporting Remote Maintenance in Industry 4.0 through Augmented Reality’, *Procedia Manufacturing*, vol. 11, pp. 1296–1302, Jan. 2017, doi: 10.1016/j.promfg.2017.07.257.
[68] Production Engineering, Centro Universitário de Araraquara – UNIARA, Araraquara, Brazil, M. A. Frigo, E. C. C. da Silva, and G. F. Barbosa, ‘Augmented Reality in Aerospace Manufacturing: A Review’, *JIII*, 2016, doi: 10.18178/jiii.4.2.125-130.
[69] J. H. Lemelson, ‘Flexible manufacturing systems and methods’, US6708385B1, Mar. 23, 2004.
[70] A. Hayes, ‘How a Flexible Manufacturing System (FMS) Works’, *Investopedia*. https://www.investopedia.com/terms/f/flexible-manufacturing-system.asp (accessed May 31, 2020).
[71] ‘Flexible manufacturing system’, *Wikipedia*. Apr. 23, 2020, Accessed: May 31, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Flexible_manufacturing_system&oldid=952645940.
[72] P. Kostal, A. Mudrikova, and D. Michal, ‘Possibilities of intelligent flexible manufacturing systems’, *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 659, p. 012035, Oct. 2019, doi: 10.1088/1757-899X/659/1/012035.
[73] ‘Synchro’, *Circuit Globe*, Sep. 12, 2017. https://circuitglobe.com/synchro.html (accessed May 27, 2020).
[74] ‘What is a Resolver? - A Galco TV Tech Tip’. https://www.youtube.com/watch?v=7X7KBtx3D7o (accessed May 27, 2020).
[75] ‘Robot Platform | Knowledge | Types of Robot Sensors’. http://www.robotplatform.com/knowledge/sensors/types_of_robot_sensors.html (accessed May 27, 2020).
[76] ‘Kinect’, *Wikipedia*. May 26, 2020, Accessed: May 27, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Kinect&oldid=958970348.
[77] K. Lai, L. Bo, X. Ren, and D. Fox, ‘A large-scale hierarchical multi-view RGB-D object dataset’, in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1817–1824, doi: 10.1109/ICRA.2011.5980382.
[78] S. Boubou, T. Narikiyo, and M. Kawanishi, ‘Adaptive filter for denoising 3D data captured by depth sensors’, in *2017 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, Jun. 2017, pp. 1–4, doi: 10.1109/3DTV.2017.8280401.
[79] A. Javed, H. Tariq, and A. Khalid, ‘Implementation of IR sensors in thru-beam

and diffuse-reflective modes for obstacle detection', in *2017 International Symposium on Wireless Systems and Networks (ISWSN)*, Nov. 2017, pp. 1–5, doi: 10.1109/ISWSN.2017.8250013.
[80] 'What is a photoelectric sensor? | Sensor Basics: Principle-based Guide to Factory Sensors | KEYENCE'. https://www.keyence.co.uk/ss/products/sensor/sensorbasics/photoelectric/info/ (accessed May 27, 2020).
[81] 'KUKA', *Wikipedia*. Apr. 29, 2020, Accessed: May 28, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=KUKA&oldid=953909414.
[82] 'KUKA Robotics Corporation on Robotics Online', *Robotics Online*. https://www.robotics.org/company-profile-detail.cfm/Supplier/KUKA-Robotics-Corporation/company/378 (accessed May 28, 2020).
[83] 'Robot-based friction stir welding for the production of electric car batteries', *KUKA AG*. https://www.kuka.com/en-gb/industries/solutions-database/2018/12/friction-stir-welding-emobility (accessed May 28, 2020).
[84] 'Robot-assisted rehabilitation with ROBERT® and KUKA', *KUKA AG*. https://www.kuka.com/en-gb/industries/solutions-database/2019/08/robert-from-life-science-robotics (accessed May 28, 2020).
[85] 'Pick and place: KUKA robot loads tube laser', *KUKA AG*. https://www.kuka.com/en-gb/industries/solutions-database/2019/09/trafoe (accessed May 28, 2020).
[86] R. Bischoff, U. Huggenberger, and E. Prassler, 'KUKA youBot - a mobile manipulator for research and education', in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1–4, doi: 10.1109/ICRA.2011.5980575.
[87] K. R. Gmbh, *Kuka System Software 8.3 Operating Instructions*. 2013.
[88] S. Severin and J. Rossmann, 'A Comparison of Different Metaheuristic Algorithms for Optimizing Blended PTP Movements for Industrial Robots', in *Intelligent Robotics and Applications*, Berlin, Heidelberg, 2012, pp. 321–330, doi: 10.1007/978-3-642-33503-7_32.
[89] M. P. Cooper, C. A. Griffiths, K. T. Andrzejewski, C. Giannetti, and College of Engineering, Swansea University, Swansea, UK, 'Motion optimisation for improved cycle time and reduced vibration in robotic assembly of electronic components', *AIMS Electronics and Electrical Engineering*, vol. 3, no. 3, pp. 274–289, 2019, doi: 10.3934/ElectrEng.2019.3.274.
[90] 'Profibus', *Wikipedia*. Jan. 06, 2020, Accessed: May 29, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Profibus&oldid=934510812.
[91] 'PROFIBUS Protocol Overview', *Real Time Automation, Inc.* https://www.rtautomation.com/technologies/profibus/ (accessed May 29, 2020).
[92] KUKA Roboter GmbH, 'KUKA System Technology - Kuka Ethernet KRL 2.1'. .
[93] KUKA Robot Group, 'Kuka Ethernet RSI XML 1.1'. .
[94] KUKA Roboter GmbH, 'KUKA System Technology - Kuka RobotSensorInterface 3.3', 2016.
[95] 'Adaptive control', *Wikipedia*. May 15, 2020, Accessed: May 31, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Adaptive_control&oldid=956730179.
[96] A. Visioli, *Practical PID Control*. Springer Science & Business Media, 2006.
[97] T. Hsia, 'Adaptive control of robot manipulators - A review', in *1986 IEEE International Conference on Robotics and Automation Proceedings*, Apr. 1986,

vol. 3, pp. 183–189, doi: 10.1109/ROBOT.1986.1087696.
[98] S. Youssefi, S. Denei, F. Mastrogiovanni, and G. Cannata, ‘A real-time data acquisition and processing framework for large-scale robot skin’, *Robotics and Autonomous Systems*, vol. 68, pp. 86–103, Jun. 2015, doi: 10.1016/j.robot.2015.01.009.
[99] M. Caputo, J. T. Lyles, M. S. Salazar, and C. L. Quave, ‘LEGO MINDSTORMS Fraction Collector: A Low-Cost Tool for a Preparative High-Performance Liquid Chromatography System’, *Anal. Chem.*, vol. 92, no. 2, pp. 1687–1690, Jan. 2020, doi: 10.1021/acs.analchem.9b04299.
[100] M. Zhou and W. Gao, ‘Multi-sensor Data Acquisition for an Autonomous Mobile Outdoor Robot’, in *2011 Fourth International Symposium on Computational Intelligence and Design*, Oct. 2011, vol. 2, pp. 351–354, doi: 10.1109/ISCID.2011.190.
[101] Zimmer, ‘Zimmer Gripper GPP5010’. https://www.g4.com.tw/userfiles/files/Datasheet/zimmer_gripper_gpp5010.pdf (accessed Jan. 17, 2020).
[102] Kuka Roboter GmbH, ‘Kuka KR 6-2, KR 16-2’, 2012, [Online]. Available: http://www.kuka-robotics.com/en/products/industrial_robots/low/kr6_2/start.htm.
[103] Kuka Roboter GmbH, ‘Kuka KR C4’, *KUKA AG*. https://www.kuka.com/en-gb/products/robotics-systems/robot-controllers/kr-c4 (accessed Jan. 17, 2020).
[104] ‘USB-1608G DAQ’. https://www.measurementsystems.co.uk/data-acquisition-solutions/usb_data_acquisition/usb-1608g (accessed Jan. 20, 2020).
[105] KUKA Roboter GmbH, ‘Kuka System Variables 8.1 - 8.3’. .

9. APPENDIX A – Results Tables

Table 9-1 Optimisation Experiment Plan

Obstacle	Robot Speed	Algorithm	Data Rate	Smart Delay
1	1%	Simple	0.01	N/A
1	3%	Simple	0.01	N/A
1	5%	Simple	0.01	N/A
1	10%	Simple	0.01	N/A
1	30%	Simple	0.01	N/A
1	1%	Simple	0.05	N/A
1	3%	Simple	0.05	N/A
1	5%	Simple	0.05	N/A
1	10%	Simple	0.05	N/A
1	30%	Simple	0.05	N/A
1	1%	Simple	0.1	N/A
1	3%	Simple	0.1	N/A
1	5%	Simple	0.1	N/A
1	10%	Simple	0.1	N/A
1	30%	Simple	0.1	N/A
2	1%	Simple	0.01	N/A
2	3%	Simple	0.01	N/A
2	5%	Simple	0.01	N/A
2	10%	Simple	0.01	N/A
2	30%	Simple	0.01	N/A
2	1%	Simple	0.05	N/A
2	3%	Simple	0.05	N/A
2	5%	Simple	0.05	N/A
2	10%	Simple	0.05	N/A
2	30%	Simple	0.05	N/A
2	1%	Simple	0.1	N/A
2	3%	Simple	0.1	N/A
2	5%	Simple	0.1	N/A
2	10%	Simple	0.1	N/A
2	30%	Simple	0.1	N/A
3	1%	Simple	0.01	N/A
3	3%	Simple	0.01	N/A
3	5%	Simple	0.01	N/A
3	10%	Simple	0.01	N/A
3	30%	Simple	0.01	N/A
3	1%	Simple	0.05	N/A
3	3%	Simple	0.05	N/A
3	5%	Simple	0.05	N/A
3	10%	Simple	0.05	N/A
3	30%	Simple	0.05	N/A
3	1%	Simple	0.1	N/A
3	3%	Simple	0.1	N/A
3	5%	Simple	0.1	N/A

3	10%	Simple	0.1	N/A
3	30%	Simple	0.1	N/A
1	1%	Smart	0.01	500
1	3%	Smart	0.01	500
1	5%	Smart	0.01	500
1	10%	Smart	0.01	500
1	30%	Smart	0.01	500
1	1%	Smart	0.01	100
1	3%	Smart	0.01	100
1	5%	Smart	0.01	100
1	10%	Smart	0.01	100
1	30%	Smart	0.01	100
2	1%	Smart	0.01	500
2	3%	Smart	0.01	500
2	5%	Smart	0.01	500
2	10%	Smart	0.01	500
2	30%	Smart	0.01	500
2	1%	Smart	0.01	100
2	3%	Smart	0.01	100
2	5%	Smart	0.01	100
2	10%	Smart	0.01	100
2	30%	Smart	0.01	100
3	1%	Smart	0.01	500
3	3%	Smart	0.01	500
3	5%	Smart	0.01	500
3	10%	Smart	0.01	500
3	30%	Smart	0.01	500
3	1%	Smart	0.01	100
3	3%	Smart	0.01	100
3	5%	Smart	0.01	100
3	10%	Smart	0.01	100
3	30%	Smart	0.01	100

Table 9-2 Smart Algorithm Results

Number	**Obstacle**	**Robot Speed**	**Data Rate**	**Smart Delay**	**Accuracy Scores**		
					Mount 1	**Mount 2**	**Mount 3**
1	1	1%	0.01	500	4.33	6.22	3.14
2	1	3%	0.01	500	9.43	10.11	6.48
3	1	5%	0.01	500	12.22	12.56	10.06
4	1	10%	0.01	500	14.68	14.61	13.29
5	1	30%	0.01	500	16.10	15.94	15.12
6	1	1%	0.01	100	3.92	6.00	3.31
7	1	3%	0.01	100	4.28	6.17	3.19
8	1	5%	0.01	100	3.98	6.70	2.68
9	1	10%	0.01	100	7.48	7.94	3.10
10	1	30%	0.01	100	13.47	12.49	11.52

11	2	1%	0.01	500	4.04	6.39	3.41
12	2	3%	0.01	500	3.97	5.48	3.00
13	2	5%	0.01	500	3.69	6.47	2.99
14	2	10%	0.01	500	4.74	5.39	2.36
15	2	30%	0.01	500	10.86	10.03	7.88
16	2	1%	0.01	100	37.52	32.16	27.95
17	2	3%	0.01	100	37.90	40.94	36.84
18	2	5%	0.01	100	41.66	42.46	39.93
19	2	10%	0.01	100	44.60	44.85	43.08
20	2	30%	0.01	100	45.05	44.59	44.29
21	3	1%	0.01	500	16.69	20.60	18.17
22	3	3%	0.01	500	29.51	25.03	20.91
23	3	5%	0.01	500	36.88	30.98	27.28
24	3	10%	0.01	500	33.90	37.44	35.46
25	3	30%	0.01	500	42.38	42.92	40.62
26	3	1%	0.01	100	16.51	20.07	18.33
27	3	3%	0.01	100	19.58	21.02	18.79
28	3	5%	0.01	100	25.92	23.98	18.59
29	3	10%	0.01	100	35.45	32.58	28.60
30	3	30%	0.01	100	38.73	38.19	37.03

Table 9-3 Further smart delay experiment results using mount 3

Obstacle	Robot Speed	Algorithm	Data Rate	Smart Delay	Smart Increment	Accuracy Score
2	3	Smart	0.01	50	0.01	19.04
2	3	Smart	0.01	100	0.01	19.31
2	3	Smart	0.01	150	0.01	19.55
2	3	Smart	0.01	200	0.01	20.23
2	3	Smart	0.01	300	0.01	18.54
2	3	Smart	0.01	400	0.01	21.22

Table 9-4 Simple Algorithm Results

Object	Robot Speed	Algorithm	Data Rate	Accuracy Scores		
				Mount 1	Mount 2	Mount 3
1	1%	Simple	0.01	4.33	6.22	3.14
1	3%	Simple	0.01	9.43	10.11	6.48
1	5%	Simple	0.01	12.22	12.56	10.06
1	10%	Simple	0.01	14.68	14.61	13.29
1	30%	Simple	0.01	16.10	15.34	15.12
1	1%	Simple	0.05	3.92	6.00	3.31
1	3%	Simple	0.05	4.28	6.17	3.19
1	5%	Simple	0.05	3.98	6.70	2.68
1	10%	Simple	0.05	7.48	7.94	3.10
1	30%	Simple	0.05	13.47	12.34	11.52
1	1%	Simple	0.1	4.04	6.39	3.41

1	3%	Simple	0.1	3.97	5.48	3.00
1	5%	Simple	0.1	3.69	6.47	2.99
1	10%	Simple	0.1	4.74	5.39	2.36
1	30%	Simple	0.1	10.86	10.03	7.88
2	1%	Simple	0.01	37.52	32.16	27.95
2	3%	Simple	0.01	37.90	40.94	36.84
2	5%	Simple	0.01	41.66	42.46	39.97
2	10%	Simple	0.01	44.60	44.85	43.08
2	30%	Simple	0.01	45.05	44.59	44.29
2	1%	Simple	0.05	16.69	20.60	18.17
2	3%	Simple	0.05	29.51	25.03	20.91
2	5%	Simple	0.05	36.88	30.98	27.28
2	10%	Simple	0.05	33.90	37.44	35.46
2	30%	Simple	0.05	42.38	42.92	40.62
2	1%	Simple	0.1	16.51	20.07	18.33
2	3%	Simple	0.1	19.58	21.02	18.79
2	5%	Simple	0.1	25.92	23.98	18.59
2	10%	Simple	0.1	35.45	32.58	28.60
2	30%	Simple	0.1	38.73	38.03	37.03
3	1%	Simple	0.01	12.02	10.58	8.59
3	3%	Simple	0.01	13.10	13.41	10.11
3	5%	Simple	0.01	14.39	14.41	12.31
3	10%	Simple	0.01	17.46	16.11	14.58
3	30%	Simple	0.01	14.60	14.49	13.85
3	1%	Simple	0.05	12.06	9.58	8.41
3	3%	Simple	0.05	12.06	9.80	8.27
3	5%	Simple	0.05	11.80	10.09	9.32
3	10%	Simple	0.05	13.09	11.65	8.70
3	30%	Simple	0.05	14.65	12.90	10.28
3	1%	Simple	0.1	12.31	9.76	8.77
3	3%	Simple	0.1	12.35	9.791	8.54
3	5%	Simple	0.1	11.84	10.55	8.84
3	10%	Simple	0.1	11.26	9.82	7.82
3	30%	Simple	0.1	10.79	1.28	8.78