An Investigation into a Combined Visual Servoing and Vision-Based Navigation System Robot for the Aerospace Manufacturing Industry

Lee Clift

7th March 2023

An Investigation into a Combined Visual Servoing and Vision-Based Navigation System Robot for the Aerospace Manufacturing Industry

BY

Lee Clift

A dissertation submitted to

The University of Sheffield

in partial fulfillment of the requirements for the degree

of

Doctor of Philosophy

7th March 2023

Lee Clift

An Investigation into a Combined Visual Servoing and Vision-Based Navigation System Robot for the Aerospace Manufacturing Industry

PhD Dissertation, 7th March 2023

Supervisors: Professor Ashutosh Tiwari
Mr Chris Scraggs

The University of Sheffield

Department of Automatic Control and Systems Engineering

Amy Johnson Building

Portobello Street

Sheffield, S1 3JD

Abstract

High-Value manufacturing, such as aerospace manufacturing, has been less impacted by the mainstream use of robotic automation compared to other manufacturing industries. This is due to the cost factor required when creating robotic systems which can successfully interact with such high tolerance workpieces. This research aims to investigate the gap of robotics within high-value aerospace manufacturing, with the goal of creating a generic robotic algorithm which can effectively and optimally detect and trace a variety of aerostructure inspired workpieces.

This goal was achieved by firstly developing a vision system for detecting and tracing particular features of partially-known workpieces. These workpieces varied in size and spatial profile, having both obtuse and acute edges. Once an effective vision system was developed, a variety of distribution-of-labour algorithms were developed, with the aim of dividing the task of tracing a workpiece between the kinematic arm and mobile base. The results showed that different distribution-of-labour algorithms performed differently, depending on the type of detected feature, specifically how vertically inclined the feature was. These results were used to develop an optimal distribution-of-labour algorithm, which could dynamically and optimally switch between different distribution-of-labour systems, to trace a workpiece both quickly and accurately.

Results showed that an optimal distribution-of-labour algorithm decreased tracing time and increased accuracy in realistic aerostructure-inspired workpieces compared to just using one major algorithm, and could dynamically trace workpieces regardless of previous knowledge or spatial profile.

Declaration

I, Lee Clift, declare that the work presented in this thesis is my own. All material in this thesis which is not of my own work, has been properly accredited and referenced.

Sheffield, 7th March 2023

Lee Clift

Acknowledgement

I would like to thank my two supervisors – Prof. Ashutosh Tiwari and Mr. Chris Scraggs for their invaluable supervision, support and tutelage during the course of my PhD degree. My gratitude extends to the Department of Automatic Control and Systems Engineering and Airbus UK for the funding of this project. Additionally, I would like to express gratitude to Dr. Divya Tiwari and Dr. Windo Hutabarat for their treasured support, regardless of the time of day, or triviality of the request; without their combined efforts, much of this work wouldn't have been possible. I also thank Dr. Jonathan Aitken for their mentorship, who without his expert knowledge, no physical experiments would have been completed.

Finally, I would like to express my gratitude to my parents, my friends and my fiancée, Rebecca. Without her tremendous understanding and encouragement over the past four years, it would have been impossible for me to complete this body of work.

List of Publications

Clift, L.; Tiwari, D.; Scraggs, C.; Hutabarat, W.; Tinkler, L.; Aitken, J.M.; Tiwari, A. Implementation of a Flexible and Lightweight Depth-Based Visual Servoing Solution for Feature Detection and Tracing of Large, Spatially-Varying Manufacturing Workpieces. Robotics 2022, 11, 25. https://doi.org/10.3390/robotics11010025

In Preparation:

Clift, L.; Tiwari, D.; Scraggs, C.; Hutabarat, W.; Ragunathan, G.; Aitken, J.M.; Tiwari, A. Development and Implementation of an Optimal Division of Labour algorithm for Feature Detection and Tracing of Large, Spatially-Varying Manufacturing Workpieces. International Journal of Computer Integrated Manufacturing

List of Figures

List of Tables

List of Equations

List of Acronyms

AGV	Automated Guided Vehicle.
AMRC	Advanced Manufacturing Research Centre.
ASM	Automated Spray Method.
CUDA	Compute Unified Device Architecture.
FPI	Fluorescent Penetrant Inspection.
IBVS	Image-Based Visual Servoing.
IMU	Inertial Measurement Unit.
MDF	Medium-Density Fibreboard.
PBVS	Pose-Based Visual Servoing.
ROS	Robot Operating System.
SLAM	Simultaneous Localisation and Mapping.
TF	Transform.

Contents

1

Introduction

1.1 Project Introduction

In the modern world, many aspects of human life are changing, being augmented by new technology. Although this has been continually happening for the past two millennia, and specifically within the last three centuries, we have experienced a substantial technological boom. Ever since the 19th century and the Industrial Revolution, manufacturing within the UK has represented a significant part of the workforce. As of March 2019, the manufacturing industry is one of the biggest within the UK; the Office for National Statistics recorded 2.737 million people employed within the manufacturing industry [12]. Therefore, there is a market for making these jobs safer and more economical to complete, and this is something which can be shown throughout multiple industries within the world.

Despite these possibilities, there are still many industries which have not fully embraced the use of automation, which is usually related to either task complexity or an unfavourable cost-to-benefit ratio. Therefore, more investigation is needed to find new cost-effective ways in order to solve this issue. One fundamental way of doing this is through the use of intelligent, flexible solutions compared to rigid robotic designs.

By introducing modern automation and robotics into the workplace, we can experience a paradigm shift in the way work can be completed. This can already be seen in industries such as the automobile industry, where kinematic servo robots complete much of the car assembly. The increase of robotics within the automobile industry has already increased productivity, but it can be predicted that, with further automation, this can be increased by another 20 to 25% [13]. By transferring this trend to other industries, the productivity benefits of automation could benefit a much wider audience.

Unlike the automobile industry, the aerospace industry has not experienced automation to the same extent. This is due to how the aerospace industry relies on flexibility within its manufacturing plants and the complex manufacturing which takes place there. Some automation has begun to appear within aerospace manufacturing, specifically the usage of AGVs, such as the Kuka OmniMove, at the Airbus Production Plant in Hamburg [14]. Airbus, in particular, are exploring more advanced manufacturing techniques, specifically within the Wing of Tomorrow programme. The programme's focus is on aircraft wing technology and how to manufacture them more economically [15]. To do this, automation and, more specifically, intelligent automation, could be a large part of it.

An example of the impact of automation on the wing manufacture/assembly process is the task of sealing and cleaning the aeroplane wing spars, which allows different parts within the wing to be sealed together. This is traditionally done by hand and is a long, complex task, sometimes lasting over six hours. Through the use of automation, this could be cut down to under two hours and potentially even quicker through the use of intelligent- planning robotic automation. An generic intelligent agent could dynamically plan its route to complete a task while adjusting its speed and technique to suit each section. By doing this, the agent would complete the task most optimally, ensuring it is done as quickly as possible while achieving a minimum set quality. By creating a generic algorithm which can trace the features of any type of workpiece, alongside tracing it at its optimal speed at each point of the task (while remaining within specific safety parameters), it could be able to be applied tot he Seal and Clean problem, as well as many other tasks within a wide range of industries.

1.2 Research Aims and Objectives

The main aim of this research is to develop a combined, visual servoing and vision-based navigation system which can integrate with an AGV and manipulation system for customisable, optimised movement.

In order to achieve these aims, four major objectives were planned, building on top of each other:

a) Develop and implement a vision system for the feature detection and tracing of spatially-varying, partially-known manufacturing workpieces

b) Create and evaluate various distribution-of-work algorithms for tracing a spatially-varying, partially-known manufacturing workpiece

c) Create a dynamic distribution-of-work algorithm to optimally trace a spatially-varying, partially-known manufacturing workpiece.

d) Validate and test the developed system and algorithms on test samples inspired by real-life engineering parts.

1.3 Research Methodology

By looking at each objective and analysing how it can be completed, an in-depth methodology was constructed, detailing how each objective will be completed. The first objective to be completed is the creation of the vision system, which the agent can use for the visual servo system, as well as using the vision system to trace manufacturing workpieces. This can be split into two parts: the vision system and the tracing system. Firstly, cameras were selected and thoroughly tested to decide which type of camera best suits the task at hand. Secondly, variables were chosen to determine how these cameras must be tuned to achieve optimal results. This was done through a factorial method to gain an understanding of each variable, its dependencies on other variables, and how they all combine to affect the image received. The factorial method was used to test a large variety of variables quickly and efficiently quickly. This data was used to create an optimal environment setup, the best possible environment for using a camera to detect workpiece features. Once completed, additional cameras were also tested using similar settings to form a comparison and to decide which camera gives the best outline of the workpiece. The output image was analysed and compared with others to see which camera gives the most consistent, clear images. The result was a list of scenarios and which camera would be best suited for each scenario, aiding in the choice of camera when detecting workpiece features.

The tracing part of the first objective was accomplished through experimentation with a robotic kinematic arm and the camera chosen from the previous set of camera experiments. By mounting the selected camera to the end-effector of a kinematic robot via eye-in-hand configuration, experiments were conducted to see how well the kinematic arm can trace the edge of a detected workpiece using the camera. The camera was used to create a PBVS system connected to the arm, providing the overall system with multiple sets of coordinates for the arm to move to. ROS was used to program the robot simply and quickly to perform these experiments, aiding in the creation of multiple techniques that can then be tested systematically. Experiments were then completed to see how accurate the arm was when moving to detected points via the camera system to ensure that the visual servoing system was accurate. This was done by recording the actual position of the robotic arm and comparing it against a known ground truth, providing an offset. The offset was recorded and compared per workpiece to determine the performance difference between different workpieces.

The kinematic arm first needed to be mobilised to complete the second objective. This could have been completed in one of two ways: either by adding the kinematic arm onto a pre-existing AMR or by porting over the vision system to a new mobile kinematic robot. Either option would have allowed the robot to become mobile and trace larger workpieces. Using this new mobile robot, algorithms were created to explore the distribution of labour when tracing a workpiece: how much work should be completed by the kinematic arm and how much should be completed by the mobile base. Two primary, opposing algorithms were developed and investigated. The first algorithm prioritised arm movement and only moved the base when needed (essentially keeping the base static for as long as possible). The second algorithm did the opposite, it moved the base as much as possible in an attempt to minimise arm movements. Experiments were carried out to compare the effectiveness of each algorithm to one another, to understand how optimal each algorithm would be when tracing a variety of workpieces. Each workpiece reflected different detectable features, a consistent non-inclined workpiece, a constant steadily inclining workpiece, and a spatially varying workpiece. This variety of workpieces was used to allow a direct comparison between the two algorithms when tracing the different workpieces. The results were compared by recording the accuracy of the arm (comparing the current arm positions to a known ground truth) and the time it took to trace the workpiece. This comparison was done to show the effectiveness of each algorithm, when tracing the different features of the different workpieces.

The results from these two algorithms were then used to inform the third objective: the creation of an optimised, dynamic algorithm. This algorithm used the previously-created vision system to analyse the workpiece and, depending upon the results from the previous objective, chooses how to distribute the work of tracing the workpiece. The result was an algorithm which could optimally trace a partially-known, spatially-varying workpiece faster than any previously-tested algorithm while still remaining accurate to a given precision. Once the algorithm was developed, it was tested against the previous algorithms. This was done to ensure that the newly created algorithm was a better alternative, either time wise or accuracy wise, than the previously created algorithms. This was done by measuring the accuracy and completion times of the new algorithm, and comparing it to the results of the previous algorithms.

The final objective was completed simultaneously to objectives 2 and 3. This was done through a high level of testing each algorithm went through. Each algorithm was tested using a variety of workpieces, ranging from basic shapes for fundamental testing, to aerostructure inspired workpieces, to receive more realistic results. This was done both in simulation and with physical workpieces:

simulated to provide a larger amount and variety of experiments, and physical to verify all the simulated experiments. By completing a variety of experiments for each algorithm, an accurate result for both speed and accuracy was obtained, and allowed the development of an optimal algorithm.

1.4 Outline of Thesis

This dissertation is presented as the following chapters:

- Chapter 1: Introduce the topic and research question, as well as provide aims and objectives for this research
- Chapter 2: A review of current, relevant literature regarding visual servoing and industrial automation, specifically within the manufacturing industry
- Chapter 3: Development of a visual servoing system for detecting and tracing manufacturing workpieces
- Chapter 4: Development and investigation of multiple distribution-of-work algorithms for tracing manufacturing workpieces
- Chapter 5: Development of an optimal distribution-of-work algorithm, for the tracing of manufacturing workpieces
- Chapter 6: Discussion and conclusion on the research completed, as well as providing information on the research's limitations and future work.

2

Literature Review

2.1 Introduction

The purpose of this literature review is to investigate the current research regarding robotics within the manufacturing industry, specifically their use in high-value aerospace manufacturing, and to understand why automation hasn't penetrated to the same level as that of other industries.

The aim of this literature review is to give a background to the most common techniques used in the fields of mobile robotics, kinematics and visual servoing. This literature review presents how these techniques are currently being used within the manufacturing industry.

To achieve these aims, the literature review will cover the following objectives systematically:

- Review the relevant technology and techniques for both mobile and kinematic robotics
- Review the current techniques used for robotic vision and visual servoing
- Review the relevant usages of these techniques with regards to the manufacturing industry – specifically feature detection and the aerospace industry

2.2 Methodology

An initial methodology was created for the literature review to aid in discovering and analysing related literature. This was done by using the aims and objectives as a guide and breaking them down into key concepts and points of interest. These points were then used to derive some searchable keywords which could be used to gather literature.

Figure 2.1 shows how each stage of the process was completed. The objectives (within the red area) were analysed and simplified, and the main points of interest for each objective were focused on (seen in the green area). Some objectives overlapped with the points of interest, such as the overarching theme of manufacturing, the idea of tracing features etc. By analysing and researching these points of interest, they helped aid in the creation of a list of searchable keywords (seen in the blue area). These keywords were used as a starting point to search for relevant literature and to help gather information. As additional research was completed, additional keywords were taken from literature where possible, to help expand what was being researched.

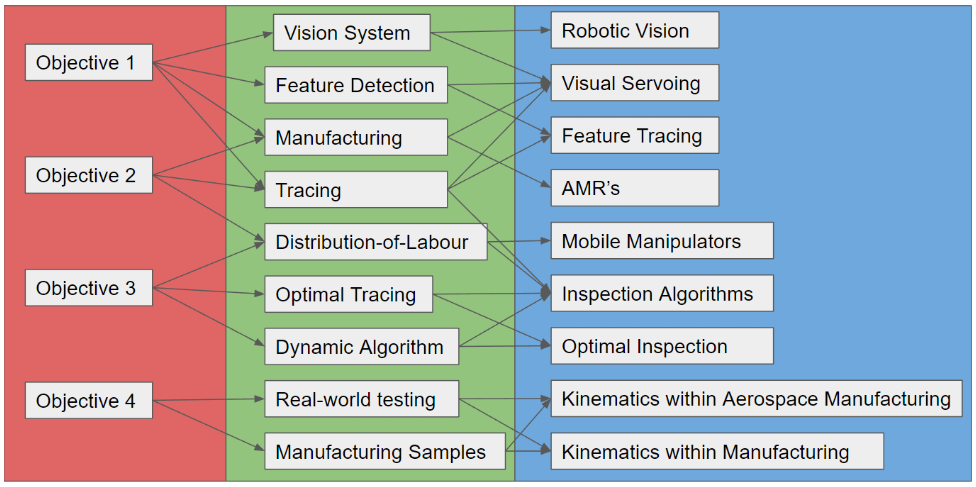


Figure 2.1: A Diagram showing how the literature review was composed. The red section represents the original four objectives, the green shows points of interest which were gathered from the objectives, and the blue section shows the searchable keywords which were derived from the points of interest.

Using these keywords, various websites were used to search for literature, such as journal papers, conference proceedings and books. These websites included:

- Scopus
- Google Scholar
- IEEE Xplore
- Science Direct

When searching with these keywords, the title of each publication was read initially, to get an estimation of how relevant the work was. If it didn't seem irrelevant, the abstract was then read, to help decide if the publication was highly relevant, and if so, the full text was downloaded and selected for analysis.

Additionally, as the literature review progressed, and publications were selected for analysis, a list of notable authors was made. Initially, a historical investigation of the relevant author's published work was completed, to check for any further relevant reading, but an additional systemic and regular check was done for any future publications, in case they continued to publish relevant materials.

This resulted in a literature review composed of two major parts: a discussion on the use of robotic techniques within manufacturing, and a discussion on examples of robots being used within the manufacturing industry, including the aerospace industry.

2.3 Robotic Technologies used within Manufacturing

In many cases, automation within a manufacturing environment can take one of two forms: either through an Automated Guided Vehicle (AGV) moving around an environment and moving equipment, tools and workpieces around (such as the Kuka OmniMove [16]); or a visual servo manipulator, created to repeat one job in a production line (commonly seen in both the automation and aerospace industries). As discussed, the future of manufacturing relies on automation, helping the manufacturing industry to become more flexible, and this cannot be achieved through the use of static, single-use machinery. A good way to meet this new demand for flexibility is to combine both of these systems thereby providing all the benefits of a static servo robot while still having the flexibility to move around an environment.

2.3.1 Mobile Robotics

AGVs are self-driven robots whose primary purpose is to move from one place to another, typically carrying goods (Figure 2.2). Ever since their inception in 1965 they have become increasingly prevalent in industry, increasing to over 20,000 individual AGVs by 2000 [17]. These AGVs can range in size from smaller robots which carry shelves in warehouses, to large AGVs which carry mineral ore in pit mines. In either case, how they move can be categorised in two ways: either they are intelligent, or they are not. Where typically an AGV will be restricted to a predetermined path, there are also Autonomous Mobile Robots or AMRs, which are capable of dynamically moving around an environment to complete their tasks. Typically, an AMRs will have more positional sensing than a typical AGV, as it needs to detect its environment and potential hazards within it, such as objects blocking its route. If an obstruction is detected, an AMR is then able to actively plan and reroute around these obstructions to complete its task [1], as shown in Figure 2.3. The AMR used by Liaqat et al is able to detect various obstacles via a LIDAR sensor, and would then generate boundaries in which it can travel. Using these boundaries, it can then plan and reroute itself around these new restrictions, ensuring it is able to avoid obstacles, both static and dynamic.

Figure 2.2: MiR 200 robot; an example AMR

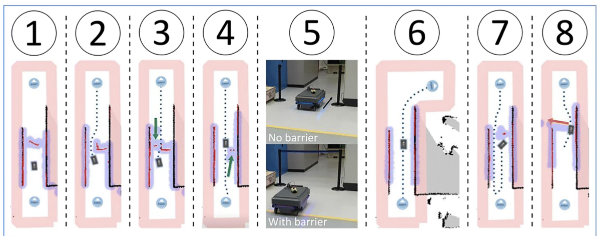


Figure 2.3: Diagram showing the different routes an AGV will make when traveling around various obstacles [1]

In many ways, this is similar to the two main types of AGV route types: static and dynamic [18]. While static-route AGVs will always travel the shortest distance from point A to point B, a dynamic-route AGV will change which route it takes using dynamic, real-time information. It is not mutually exclusive that only intelligent AGVs use dynamic path planning, as many AGVs will also be dynamic in a limited capacity – able to move out of blockages and overcome other similar barriers.

Line Following and Pre-Programmed Movement

There are a variety of ways in which an AGV can move around its environment in a controlled and pre-planned manner, ranging from having these routes already planned via a GPS system, to having physical markers in the environment that the AGV tracks.

In many ways, the line between intelligent and non-intelligent AGVs is being blurred, with many factors making set-path AGVs more intelligent than ever [19]. Many AGVs are programmed remotely with no significant modification required, meaning that if the workflow changes (ie the addition of a new charging station is implemented) the agents can be modified to incorporate that into their routes. Additionally, AGVs can make slight adjustments to their own paths dynamically, which allows them to learn how to best navigate from one point to the other while sticking to a dedicated route. The concept of this change can be explored further in the works by Junemann & Schmidt [20] and Tompkins et al. [21].

One of the more popular ways for an AGV to move is through the use of either line tracking, or object tracking. Both of these techniques rely on the agent being able to locate, find and follow particular objects in the environment. The line tracking technique uses a standard algorithm to follow a line on the floor whereas object tracking can be customised to find different objects within the environment. An experiment conducted by Martin et al. [10] discussed the differences between these two techniques, and which one would be most suitable for movement around a Bosch warehouse, explicitly focusing on the docking manoeuvres needed for loading and unloading. These manoeuvres require accuracy and robustness, both excellent qualities which translate well to other parts of an AGV system. Working alongside Bosch, the research aimed to find ways for an agent to traverse quickly and effectively without having to make extensive modifications to the factory floor – a key attribute which is a desirable factor in many industrial settings. To do this, they decided to create their own proprietary AGV and software, allowing it to fit the specifications required whilst also being low cost and not necessitating the modification of the agents' environment with things such as large physical markers. The authors explored three main solutions to the docking problem (Table 2.1):

- A line-following solution (which is accurate and low cost, but requires small amounts of modification)
- A box-finding algorithm (which looks for a common object already found in the environment, and uses that to localise and manoeuvre)
- A rail-tracking system (which uses rails commonly found within the environment to localise and manoeuvre)

Table 2.1: A comparison of the three proposed AGV tracking techniques [10]

	Pros	Cons
Line Following	Accurate Low Cost	Requires small amounts of modification to the environment
Box Finding	Pre-existing within the environment Low Cost	Required active searching by the robotic agent Could be easily interfered with by the environment
Rail Tracking	Pre-existing within the environment Low Cost Accurate	Required initial searching to an environmental feature

Overall, the rail system proved to be the best, given the lack of environmental modification required, although if environmental modification was not a problem, then line tracking resulted in a more constant localisation. In many ways the line- and rail-tracking systems serve a similar purpose, finding an object and following it consistently, while the box-tracking system actively required the agent to search for new instances of the object. Out of the three, the box tracking was the worst, given how easy it was for the technique to be influenced by environmental factors such as lighting (compared to the other two techniques where the objects it is looking for are distinct).

One of the primary reasons for using set-route AGVs is due to safety concerns relating to dynamic movement [22]. In many scenarios, operators and employees may be interacting in close proximity to the robots when they are working, and in these situations having an agent who has a specific endpoint, and a particular route, is much safer than having one which can dynamically change. Due to this preference, most AGVs within industrial settings have some set routes, although this may change as dynamic routes become more recognised, tested and safe.

Dynamic and SLAM Movement

Unlike static AGV techniques like line tracking, Simultaneous Localisation and Mapping (SLAM) and other dynamic-movement techniques are used less often, mainly due to safety and over-engineering concerns. In many situations line tracking is more suitable with little need for dynamic exploration algorithms. Regardless of this, it has not stopped research in this area, especially since different industries could take advantage of such dynamic movement for optimisation problems.

One key example of this is Sprunk et al. [23], who discussed the use of an extensive Active SLAM technique, in cooperation with Kuka, to travel around a factory floor and complete an inspection task on a wind turbine blade. The paper discussed using safety precautions to limit the robotics' speed and acceleration, and this is altered dynamically when it is closer to objects in its environment. In addition to this, it has custom navigation which can adapt dynamically to acceleration increases and decreases, as well as obstacles which may appear. This allows it to complete tasks accurately, precisely, but also quickly – all aspects which are appreciated within industry. This was all achieved on a Kuka OmniRob and more importantly, a Kuka Moiros, which is now being used within different industries and can have instruments mounted to the top of it, eg manipulators.

Schaefer et al. [24] have developed a technique for this, which takes advantage of polylines. Within factories, polylines are natural lines found within the architecture of the environment. This research work provides a method for extracting polylines through 2D laser range-finder scans. By extracting polylines, feature detection SLAM can occur with minimal memory footprint and a much faster computational time, making it more responsive. Unlike other techniques, this paper uses a probabilistic model which strives to find the set of polylines that maximise the measurement likelihood of the scan. By doing so, it outperforms many other state-of-the-art algorithms. Unfortunately, one major downside of this is the actual availability of polylines. Polylines are found in static environments – factories and environments which do not move and change. Some industries, such as the aerospace industry, have fluid environments which are continually shifting and evolving, reducing the utility of this new technique.

In many situations, an agent with a SLAM algorithm would need to fully explore an area before being able to start localising itself correctly. Research conducted by Boniardi et al. [2] looks at a unique way of supplementing current SLAM techniques, specifically for use within a factory setting. Here, the use of CAD-drawn architectural plans are used to help outline the key areas within a facility, and then SLAM is used to fill in the details (Figure 2.4). By providing the CAD drawing, the agent will already have a precise map of its surroundings, and also know where it starts. The generated SLAM map is then optimised to fit the generated pose-graph onto the floor plan. Once this is done, it is then able to estimate its relative pose with respect to the matching nodes of the pose-graph without a global optimisation process, which can be computationally heavy and time-consuming. This technique is specifically useful for industrial applications since it can overcome some of the traditional problems of SLAM (more precisely, problems such as when equipment blocks pathways or walls, or the generated map being easier to be read by non-experts).

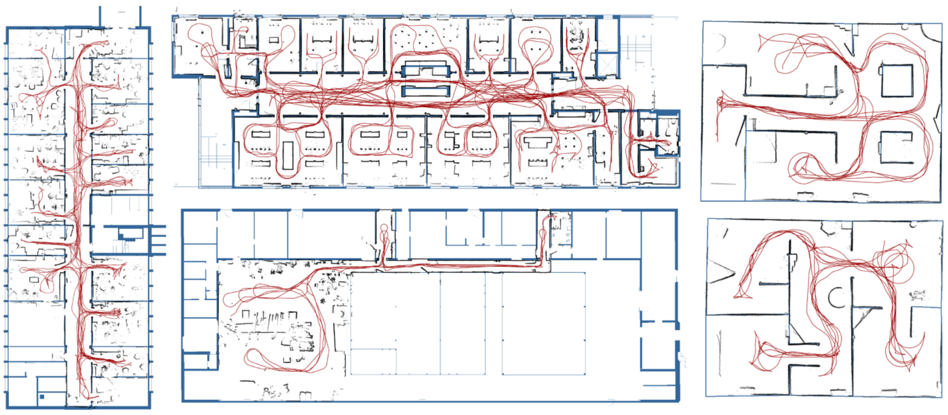


Figure 2.4: Five example CAD maps, overlaid with SLAM data [2]

Unfortunately, the CAD-assisted SLAM algorithm suffered from similar issues to that of the use of polylines – with how some environments which frequently change are not suited for the specific algorithm. Boniardi et al. [25] produced a follow-up paper about the possibility of using CAD drawings alongside SLAM, with a focus on long-term navigation. The new modified technique can use Bayesian Filtering Techniques to adapt to environmental changes, something which would be beneficial in an aerospace manufacturing environment or any area with frequent environmental changes. It does this by keeping an up-to-date Pose graph of the latest known environment, which can be changed and edited as new and conflicting information is found (similar in technique to graph SLAM). Bayes is used to track the previous possible features, and discards the least likely ones, allowing an increase in free memory and aiding long-term usage. This update to the CAD-assisted SLAM algorithm would be an extremely useful solution to the SLAM problem in a constantly changing environment, such as a high-value manufacturing plant, and could even be leveraged to incorporate a workpiece's location, and how it could traverse around it, while completing its task effectively and safely.

2.3.2 Kinematic Robotics

Kinematic Robotics, or kinematic manipulators, are generally defined as robots with multiple joints, who, through the use of a kinematic chain, can be manipulated into different poses to complete tasks [26]. These kinematic chains can normally be influenced by one of two algorithms: forward kinematics or inverse kinematics (Figure 2.5).

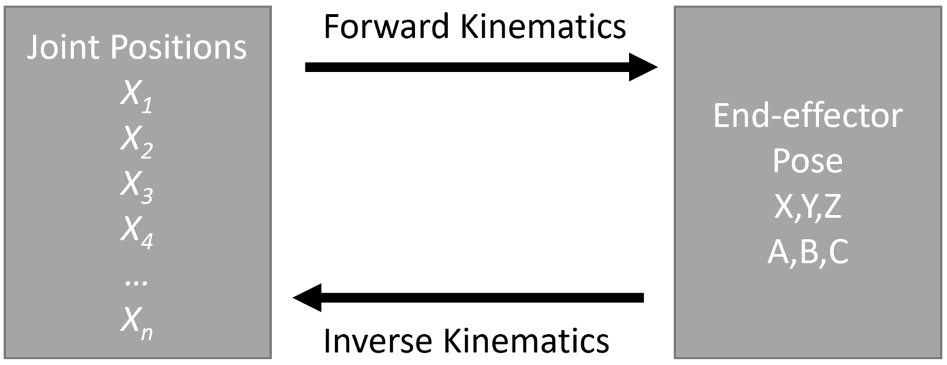


Figure 2.5: A diagram showing the inputs and outputs of various kinematic algorithms

- Forward kinematics is the use of a kinematic equation which allows the input of joint angles, and the output is an end-effector position. This is generally used for one-off positions, such as a setup pose, as it requires a very specific chain of inputs, which cannot easily be obtained at runtime for completing tasks.
- Inverse kinematics is the direct opposite of forward kinematics, where the input is an end-effector position in cartesian coordinates, alongside the pitch, roll and yaw (X, Y, Z, A, B, C). In many cases this is the more practical option, as inputting the desired end-effector position allows the task to be completed more easily, although this does come with a certain level of variance as the arm moves from its current pose to its desired pose.

When comparing the two algorithms, many of their pros and cons have resulted in them seeing usage in set roles. For example, forward kinematics allows a much higher level of control, ensuring that the robot's whole position is always known, and which doesn't deviate from cycle to cycle. This can be a required feature when operating in enclosed workspaces, such as surgery [27], or when handling objects which need to be kept in a particular orientation, such as lab work [28]. Additionally, forward kinematics has seen extensive use within the manufacturing industry, mainly for repetitive, single position pick-and-place tasks [29]. In these scenarios, where the arm has to move one object to another repeatedly, forward kinematics are more useful than inverse kinematics, since once the initial calibrations and calculations have been completed, the reduction in computational time thereon is quite noticeable.

In comparison to forward kinematics, inverse kinematics allows much easier variance in movement and positioning, as the major input is a desired cartesian position, compared to joint angles. The motion the arm then takes to reach the desired pose can be further controlled via motion planning, creating cartesian paths for the arm to follow, – although this does result in additional computational load. The use of inverse kinematics within industry has been extensive and ranges from painting and welding tasks within the manufacturing industry, to dynamic pick-and-place where multiple pick-up and drop-off poses are required [30]. It is common that when using inverse kinematics for a dynamic operation, some level of active sensing will be required to inform the arm of its desired end-effector pose, and, any motion planning if needed (i.e. to avoid hitting obstacles). In most modern cases, the best way to do this is through the use of an image sensor and incorporating robotic vision.

2.3.3 Robotic Vision & Visual Servoing

Robotic Vision and Camera Technology

When creating any automation system, specifically a robot, one of the core parts needed is the sensing equipment. This sensing equipment can come in many forms; from less reliable, but more economical sonar sensors, to very accurate, but also expensive, laser rangefinders. Another type of sensor, which has seen much more usage within the last two decades, is vision systems – specifically involving machine vision. Davies [31] describes machine vision as an automation problem, where image processing is manually conducted and machine vision must automate its tasks. The use of machine vision requires investigation into the choice of camera, ways to control the scene, and the variables and choice of algorithms etc.

One of the first important decisions is the choice of hardware: precisely what type of camera to use. This can be broadly classified into two categories: 2D or 3D. 2D cameras work like traditional commercial cameras, where they take a standard image. On the other hand, 3D cameras are able to take a three-dimensional image, getting depth data of objects within said image.

2D cameras operate in the same way as most commercial cameras and do not have any on-board depth-sensing capability. Because of this, much of the processing to work out object distance must be done remotely via a computer. Since all the processing must be done remotely, this increases the computational power needed to do automated machine vision, but the cost of many of these cameras offsets this, especially as the cost of computational power is constantly going down.

One key consideration while using a 2D camera is that the image is in RGB colour, or merely greyscale. Davies [31] discusses this decision and states that the choice of using colour is all about maintaining balance. Using a colour RGB sensor over a black and white sensor means that more data will need to be processed, requiring more processing power. Due to this increased processing power, the main consideration is if colour information is required for that particular application. In some situations, such as robotic vision where the agent may need to travel around an environment, the colour information is very useful as it can be used to help solve the data association problem by separating certain features by colour. On the other hand, some tasks, such as edge detection, are suited to natural contrast which black and white provide, and may not benefit from a full-colour sensor.

	Single Camera	Multiple Cameras
Passive vision	2D	Stereo vision Photogrammetry
Active vision	Time of flight Structured light Light coding Laser triangulation	Structured light Projected texture stereo vision

Table 2.2: A table showing different categories of 3D vision [11]

For 3D solutions, a review was conducted by Pérez et al. [11] on multiple different camera types, and where they were best suited. Five different camera types were tested (Table 2.2): Stereo, Time-of-Flight, Structured Light, Light Coding and Laser Triangulation.

- Stereo cameras work in a similar way to traditional cameras, but they use multiple sensors to detect the same feature within the environment, and, from this, the depth data can be computed (Figure 2.6). Due to their relative simplicity, they are relatively inexpensive, and therefore quite a popular solution for mobile robotics. They are good at detecting textured objects and environments but struggle to get accurate measurements in poorly or untextured environments. In addition to this, because they still use a traditional camera sensor, they struggle with poor lighting in the same way traditional cameras can.

- Time-of-Flight cameras use light sensors to determine the shape and distance of objects in the scene (Figure 2.7). The camera flashes beams of light at the object, which then get reflected, which the camera uses to determine an object's position [32]. Due to this, the data they return to the user can be quite noisy and distorted, although this is a trade-off as the sensors themselves are normally quite compact. Another major advantage of these types of cameras is that they are unaffected by traditional environmental factors, such as lighting.

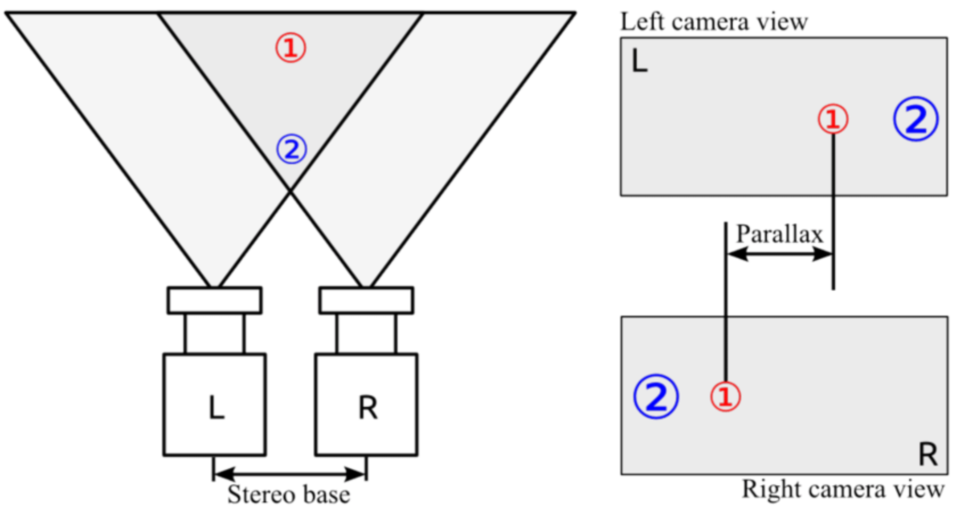


Figure 2.6: A diagram showing how a typical Stereo camera operates [3]

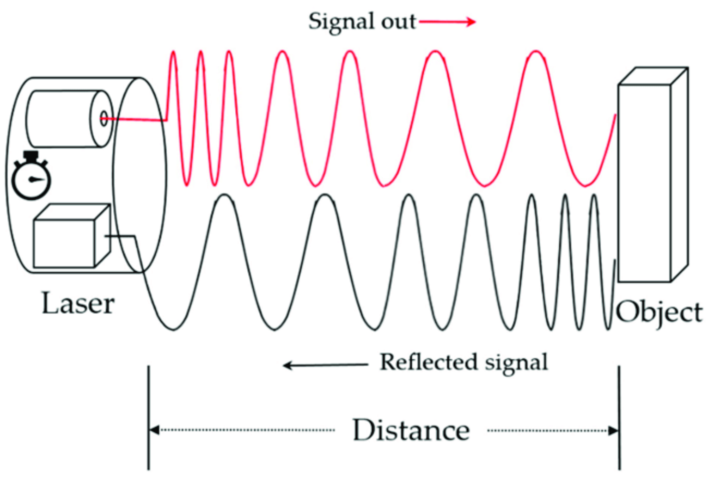


Figure 2.7: A diagram showing how a typical Time-of-Flight camera operates [4]

- Structured Light cameras work by using a projector to project light onto an object, and then cameras (typically at least two) are then used to detect how this light is dispersed on the object, and thus the dimensions of said object. There are two broad types of structured light cameras: ones which project one unique pattern of light, and ones which produce a series of patterns.

- Light Coding cameras are the newest of all camera types listed so far, and have gained a large following since their inception. Instead of using flashing lights like other cameras, this sensor sends out multiple points of light, similar to a point cloud. It then detects any reflections in the light when an object is visible within it (Figure 2.8). Microsoft Kinect is an extremely popular robotics camera which uses this technique – its popularity is mostly due to its affordability, availability and accuracy [33].

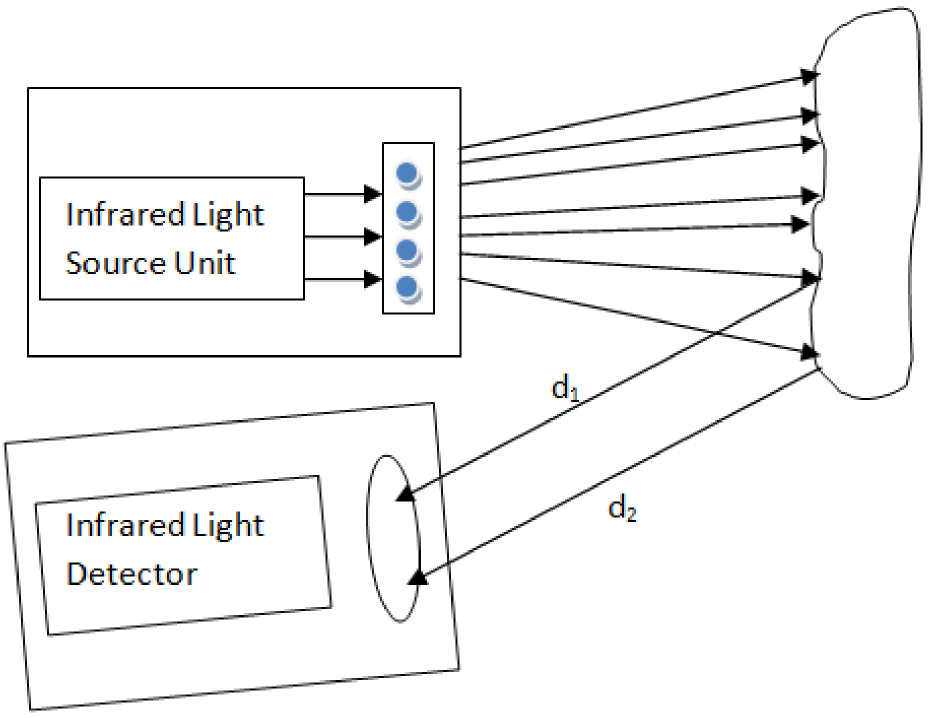


Figure 2.8: A diagram showing how a Microsoft Kinect, a typical Light Coding camera operates [34]

- Laser Triangulation is different from many of the other sensors discussed, but is most similar to Structured Light. It uses both a camera and a laser to work out the position of objects within a 3D space. The position of the laser is known to the camera, and therefore it can form a triangle between the camera, laser, and laser point. With this knowledge it can calculate how far the point is away, and therefore the distance of the object. Unfortunately, due to the use of lasers, this can sometimes be harmful to people working around them, dependent on the strength of the laser, and therefore precautions must be taken when using one.

Visual Servoing

Visual Servo robotics, also known as visual servoing is the use of a camera feed or similar vision system to inform the control system of a robot, typically a manipulator. The first instance of this was recorded in 1979 by SRI International [35]. There are typically two main types of visual servoing: Eye-in-Hand, or Eye-to-Hand [36] (Figure 2.9).

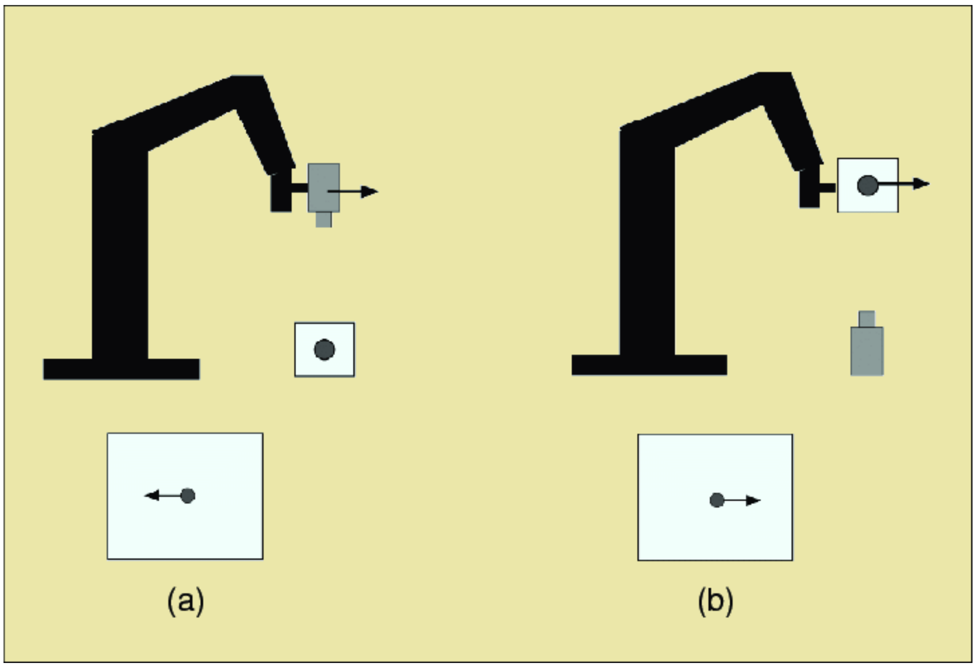


Figure 2.9: A diagram showing both eye-in-hand (a) and eye-to-hand (b) styles of visual servoing setups [37]

- Eye-in-Hand refers to the camera of the system being mounted onto the end-effector, giving a direct relationship between the robot's movement, and what the agent can see and perceive. Generally, the camera must be calibrated to the arm, converting any commands by the camera from its coordinate system into the arm's coordinate system, although research has been conducted that shows that this isn't strictly necessary [38].

- Eye-to-Hand refers to having the camera mounted elsewhere within the world, where it can view not only the robot and its movements, but also any other objects within the world it may be trying to interact with.

There is also a third variant, which takes the concept of Eye-to-Hand but which allows the camera to move independently on a gimble, such as panning and tilting to get different views of the environment.

Hutchinson, Hager and Corke [36] also go on to discuss the two main types of Visual Servoing control techniques: Image-Based Visual Servoing (IBVS) and Pose-Based Visual Servoing (PBVS).

- IBVS uses the image itself to create a feedback loop on how it should move [39].
- PBVS takes environmental features from the image and plots them using a cartesian coordinate system.

Naturally, PBVS has quite an advantage over raw IBVS, since it allows more precise control, although it requires more camera calibration, whereas IBVS requires less calibration, but is more computationally heavy since it needs to calibrate the Image Jacobian – the distance between the camera and an object it sees [40].

2.4 Robotic Roles within Manufacturing

Within the UK, manufacturing is one of the country's major industries which could benefit greatly from the use of automation. This can be defined as Smart Manufacturing [41], a field which specialises in automation and how it will affect the manufacturing industry. To automate a manufacturing process, digitalisation must come first. To digitalise a task is to make it compatible with modern technology, to gather data on the task, and make it computable. The result of this will be data which can be used to assist in the automation of a task, generally through reconstructing it. The gathering of data is usually done through the use of sensors which can monitor the task in action – typically cameras, movement sensors or a combination of other sensors. This large amount of data is then mined in a process known as Process Mining to find relevant data to reconstruct the actual process in a digital format [42].

2.4.1 Robotic technology in production lines

Automobile Manufacturing

One of the first major industries to adopt automation in a large capacity, specifically robotics, was the automobile industry. Ever since the production of the Ford Model T, the production line has been the predominant way to manufacture automobiles, and, due to the use of the production line, automation and robotics have always been a key target for this industry. Given that, in a traditional production line many operators would repeat one task many times, such as welding or painting, much of this work could be completed by the use of robotics. As early as 1984 automation was being used in the automotive industry, specifically to assist with jobs which human operators either found too skill-intensive to do, such as welding, or too heavy and unwieldy, such as painting [43]. This became more widespread in 1993 with the release of a patent by Mazda for an automobile assembly line which would weld body parts together with the aid of robotics [44]. Since then, many of the static tasks within the automotive industry have remained automated and have been expanded around the world to a higher degree.

Much of the work left to be automated is complex dexterous tasks, where having two arms which can work together would be vital to completing a task. Tsarouchi et al. [45] experimented with this concept, specifically assembling a car's dashboard computer and installing it, a task traditionally completed by humans. The result of this was the expected reliability and speed increases, as well as the general economic benefits of not using a human worker. Compared to a traditional one-armed robot, the dual-armed approach allowed it to do more tasks within the limited space and did not require any custom tooling for the movement of the dashboard computer as it could be grasped with both hands. Due to this, the dual-armed technique worked out cheaper, and it can also be assumed that a technique such as this could be used for different tasks, given the general purpose of its design. This means that, compared to the traditional one-armed technique, having two manipulators allows for a more flexible manufacturing process as each robot can be reassigned to a different task if needed.

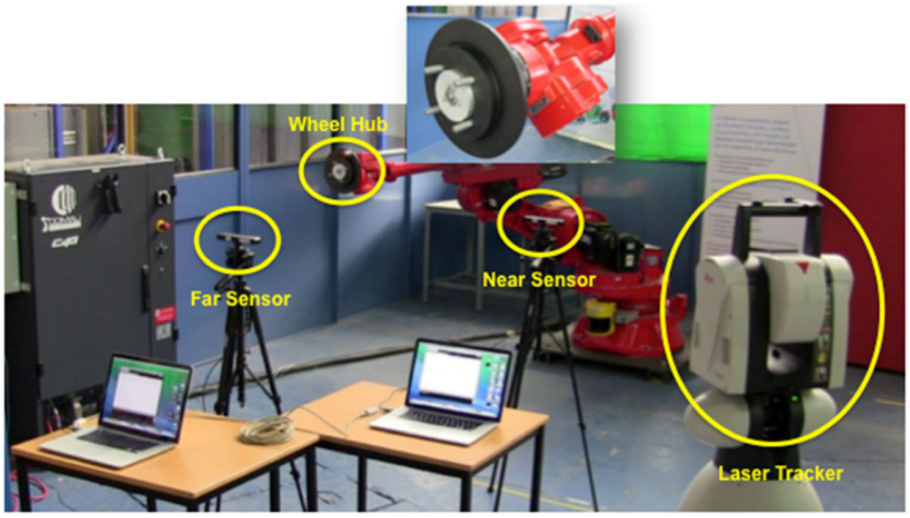


Figure 2.10: A lab setup for the tracking and placing of automobile wheels [5]

There has also been a recent push to expand into robotics which can assist with moving assembly operations, such as the fitting of wheels. Prabhu et al. [5] discuss how a complex task, such as the loading of wheels onto a moving car in production, could be digitalised and then automated. This was achieved through the use of multiple, depth-sensing cameras, which not only tracked the movement of the workpiece and where it was moving, but also tracked the random variations in how it moved, something which other similar works had not covered. A robot is then able to move the wheel into position, tracking both movements and synchronising them (as well as computing the angle of approach and the current random angle of the workpiece) to attach the wheel successfully (Figure 2.10). Two different Kinect sensors are used at two different distances to get both depth data and visual data of the workpiece. Although this work shows that mobile automation is possible, improvements could be made to the work – possibly by having the nearer sensor attached to the robotic arm which manipulates the wheel, thereby creating a visual servoing system. By doing this, not only would it free up more workspace, but it could also be used to gain more dynamic imagining information as the arm moves, adding to arm positioning.

Welding and Painting

Zhou et al. [46] provide an example of how a visual servoing system can perform a welding task. A PBVS is used in an Eye-in-Hand configuration to locate the weld line, identify its cartesian coordinates, and then use this information to get a successful weld on the seam. They also developed a series of robust image-processing techniques to ensure that the lighting conditions have minimal effect on the workpiece, as this can heavily affect the image quality. To test their system, they did test welds on both curved and straight edges, achieving a speed of 20mm per second while retaining an accuracy of within 5mm, which is appropriate for high-quality welding jobs. Unfortunately, these were the only tests conducted, and it would have been beneficial to run tests on more complex shapes and patterns, as this would provide more proof of its robustness and accuracy.

Xu et al. [47] also looked at a similar problem, but they focused more on creating a robust system, something Zhou, Le and Chen [46] did not. By using a multitude of image-processing techniques, they were able to create a PBVS welding system which could recognise weld seams and features, even in poor lighting conditions. In addition to this, they also built a second system which attempted to bridge the gap between PBVS and IBVS. It uses a PBVS to locate and move across the weld seam, and then also incorporates an IBVS to correct for any mistakes made. The robot moved at a much slower 3mm per second, and remained accurate, although the exact figure for this is not provided. The hybrid system produced was not only resistant to poor lighting conditions, but was also able to operate with camera calibration issues, although no quantitative data regarding the robot's accuracy was provided.

Unlike welding, painting tasks done by visual servo are a combination of a visual processing problem and path planning and control. To ensure that the automated spraying agent gets a good uniform paint coverage, the paint gun must always be perpendicular to the workpiece at a consistent distance. Any changes in this will result in an uneven coat being applied. Much of this is done off-line through the use of CAD models of the workpiece – which can be problematic if a workpiece is not in the exact right position, or if there are any engineering defects. Additionally, in many large workpieces, such as ship panelling and aircraft parts, deformation of the workpiece can occur, resulting in a similar problem.

This is already a popular option within the aerospace industry, specifically being used by Boeing [48]. One major drawback is its lack of flexibility when compared to traditional operators conducting the work. This concerns how the particular wing has to be positioned correctly, and the painting cell only works on one type of wing, in this case, the Boeing 777.

Chen et al. [49] discuss a method to make automatic painting agents work more robustly – which is through the use of an effective method of detection. The issue with using existing methods is how, before painting, many workpieces are textureless, and therefore cameras can struggle to detect features within them. Chen et al. suggest that a solution to this is to plan an initial route via CAD models and then use structured-light techniques to adjust the path if and when needed. In real-world tests, this system works well for straight and flat workpieces, but there has been no testing done on curved pieces, something which would be vital for certain industries such as aerospace.

2.4.2 Inspection Tasks within Manufacturing

One major way automation can be incorporated into current manufacturing workflows is through the use of machine imaging to digitalise a task. An example of this is that a large number of companies are using digitalisation as a way to automate quality control and defect detection. For over a decade this has been investigated as a possible way to streamline manufacturing, but the emergence of better technologies with time has helped this area of research.

In 2002, inspection tasks were attempted to be automated, but they had to be built and tuned specifically for each product. Prieto et al. [50] created an inspection system which was reliant on the use of CAD images and a point cloud to compare against, and then defects were evaluated according to how well these two matched. One main limitation of this is, not only how a new CAD image has to be obtained for each individual workpiece being inspected, but also a system like this may not be tolerant of manufacturing tolerances, and how individual workpieces may look ever so slightly different from each other. Even if this tolerance is accounted for, the opposite problem could then happen where some mistakes will not be noticed due to the allowance in manufacturing differences.

Another example of an inspection task is how, in the aerospace industry, Fluorescent Penetrant Inspection (FPI) can be automated, compared to manually using and checking with this technique [51]. FPI is a technique where a surface, typically titanium within aerospace, is sprayed with a dye which then highlights and shows physical defects in the workpiece. This is generally checked by human technicians, although this can be automated through the use of a Random Forest algorithm. The algorithm successfully matched the number of errors found by a human inspector (76% of all errors), although it did also find twice as many false positives – this was, however, all done on a very small training set. It can be assumed that with more work and a larger training set, this could become more accurate than a human inspector, and much of this task could be automated.

2.4.3 Feature Detection and Tracing within Manufacturing

Some manufacturing tasks require the detection and interaction with specific features of a workpiece using a combination of sensors and robotics. In these cases, visual servoing can be very useful, as the gathered imaging data can be used to inform where in the workpiece the robot needs to interact. Unlike the previously-discussed cases, where the workpiece (or part of the workpiece) which needs to be interacted with can be clearly seen or marked, some work may require the system to find particular features of the workpiece (such as an edge), or continually interact with a feature (such as to trace it).

Research conducted into line- and edge-following via visual servoing has shown that using a visual servoing system to trace a 3D path can be completed, with high levels of success. A variety of algorithms were tested on a workpiece which varied in geometry, from being straight to angular and curved. The results showed that using a simple PID reacting to the immediately-needed movement from the camera was significantly less accurate than an algorithm which takes upcoming movements into account [52]. Additional experimentation also showed that a higher-speed camera, which was able to sample more frames more quickly, aided in accuracy in both straight and curved scenarios, although this isn't a surprising conclusion, given that more data generally aids to better accuracy [53].

In garment production, it would be necessary for a visual sevoing system to detect the edges of the fabric, and manipulate them ready for sewing [6]. When manipulating such a material, attention must be given to the initial and desired shapes of the fabric, as well as manipulating it into the correct shape, and applying enough tension to the fabric so that it is able to produce a good seam. The authors use fuzzy logic to deal with the majority of the variables, ensuring the robot can autonomously perform the tasks necessary. This is done by manipulating the workpiece so that an external force (the sewing needle) can trace the edges of the workpiece, successfully sowing it (Figure 2.11). Experiments completed by the authors show that the IBVS servoing, alongside the fuzzy logic, is able to complete the task successfully, and to an acceptable standard – although it is noted that the system is not able to deal with any anomalies, such as the fabric folding.

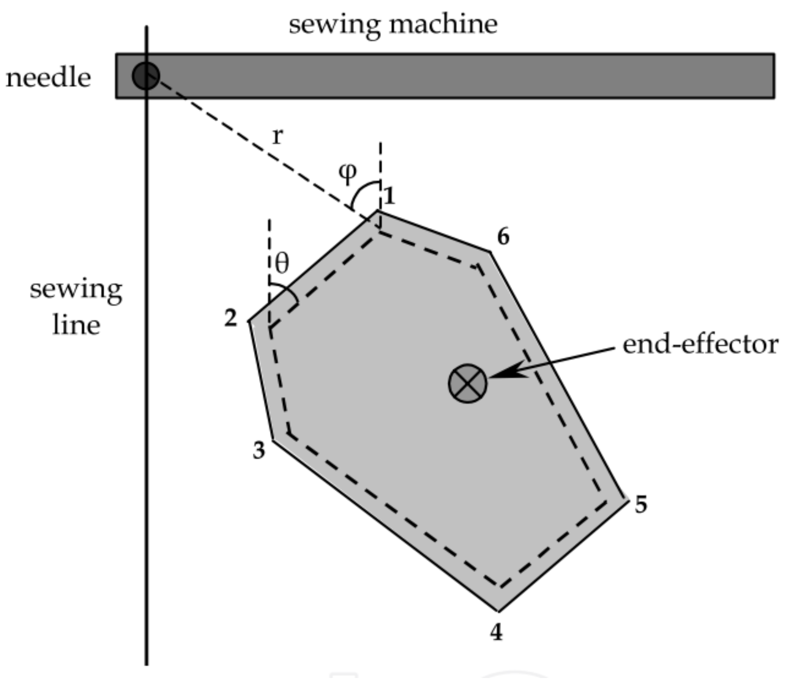


Figure 2.11: A diagram showing how a workpiece would be manipulated by the kinematic arm for the purposes of sewing [6]

Mechanical cutting is a common task done within manufacturing, which requires a contour to be followed, and material removed from the workpiece through the use of a rotary tool. Chen et al. [54] investigated the use of visual servoing to automate this task via edge detection and tracing. They used a two-camera setup in an eye-to-hand configuration, to create an IBVS system which could get the exterior contour of the workpiece, and then have the kinematic arm create a path around that workpiece. Their results were very promising, showing a high level of accuracy (less than 2.5px) while tracing the workpiece, and which could be used on any workpiece, as it was a generalised system. Unfortunately, this system relied on manual template matching, meaning it would have to be set up manually per workpiece, and the workpiece itself had to be completely in view of one of the cameras, limiting the size of the workpiece which could be worked on.

2.4.4 Aerospace Manufacturing

Unlike the automobile industry, which has had a long history of automation usage, the aerospace industry has not experienced anything like the same level of usage. It can be assumed that one of the leading factors of this has been the lack of production lines used within the aerospace industry. Due to how each component is made separately, and each component requires highly-skilled operators using custom tooling, the use of a production line is just not practical. For example, Airbus [55] have a multitude of different production plants around the world which are all responsible for different parts of the aircraft. Because of this, a single production line is not possible, as parts may take different lengths of time to create and ship around the world.

Instead of having an automated production line, where robots perform a repetitive task such as welding, the aerospace industry calls for highly specialised robotics which are capable of performing different high-skilled tasks. Even as far back as 1993, attempts by aerospace manufacturers were made at automating the process, such as Saab-Scania [56] and Airbus [57]. Both of these resulted in highly rigid robots which required high-precision positioning for drilling and riveting but which lacked flexibility, something which is crucial in the aerospace industry, hence the heavy dependence on human operators. We can see this trend continue with an automated wing construction cell implemented at Airbus UK [58]. While the robots used in the cell were able to construct a wing box through a series of drilling and riveting tasks, they still lacked flexibility, something which can be an issue considering that Airbus alone has a fleet of 12 current-generation planes in production.

Since the early 2000s, robotics within the aerospace industry has become more advanced, and there has been a push to find more flexible solutions as the aerospace industry moves toward industry 4.0. A key example of this is provided by Neumann [59] who discusses the creation of a drilling and machining robot, produced by the Advanced Manufacturing Research Centre (AMRC), and used by Airbus UK. This particular machine was created to be portable and inexpensive, in contrast to the robot created by Roche over two decades earlier, allowing it to be taken apart quickly so it can be moved around the manufacturing plant. This would allow the machine to access areas which could not be accessed by bigger, more static machines, while also providing all the advantages that automation duly provides. It also provides that vital flexibility required in aerospace, allowing the robot to be moved from one assembly to the other, so it is capable of working on different models of aircraft. It can also be assumed that, given that this robot is currently being used in one of Airbus's main manufacturing plants, and given what is known about future projects such as Wing of Tomorrow, this is the direction Airbus and possibly other aerospace companies would like to take in the future.

In addition to the automated construction of wing-boxes, there has also been research towards the automated construction of aircraft fuselages. Jayaweera Webb [60] investigated how the skin of an aircraft fuselage could be constructed through the use of a traditional industrial robotic (a Comau S2). By using a laser-guided arm, interchangeable end-effectors, and common mathematical and metrology techniques to locate and measure the components, the agent was able to construct the subcommand of the fuselage, with a tolerance of 0.6mm – much lower than the 1.6mm standard tolerance allowed. On visual inspection, the job done by the robot was considered good, as it passed all standard checks, which proves once again, that, even with decade-old technology, there is a precedent for some of the assembly tasks within aerospace to be automated.

Outside of drilling and riveting robots, there have been moves to use fixed robotic cells for different tasks, such as painting (Figure 2.12), which has also seen usage in the automobile industry [7]. Boeing currently uses robotics to paint their popular Boeing 777 aircraft; a technique referred to as the Automated Spray

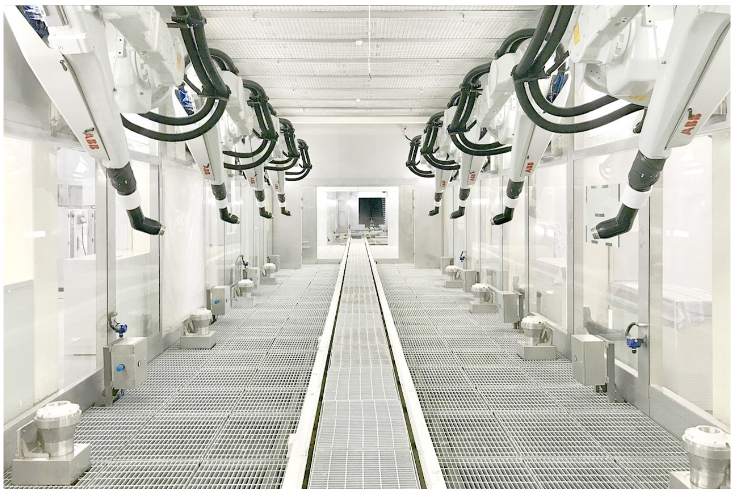


Figure 2.12: An example painting cell created by ABB [7]

Method (ASM) [48]. This new way of painting aeroplane wings has resulted in drastic reductions in painting times, reducing times from four and a half hours to under thirty minutes, while also increasing throughput by 100% [61]. The wing is set in place, and then the robots can move around the workpiece, spraying it as necessary. While this is not a flexible solution, the speed increase, as well as the reduction in human resources, is a worthwhile trade-off.

2.5 Research Gap

In much of the current literature, the vast majority of industrial automation is based on the continued research into single-use robotics, which are created to complete one single task, in one singular cell. There is a severe lack of flexible robotics which can do more than one single task, specifically within the manufacturing field. The types of AGVs used within an industrial setting generally err towards being controlled by either teleoperation [62], or use some level of preplanning [63] [64]. The main reason dynamic AGVs haven't seen a larger presence within an industrial setting is mostly a concern of safety and need. In the majority of cases, it is not required that the AGV travel around dynamically – it is much cheaper and safer for the robot to travel a set path, ensuring that any operators working alongside the robot remain safe, and are constantly aware of the robot's movements and position.

The same cannot be said for kinematic arms, where both set-position- and dynamic-movement robots have seen extensive use within industrial settings. The use of set-position robots for simple pick-and-place operations is commonplace within many industries, as is the use of a more dynamic pick-and-place, where a camera can be leveraged to help identify different pick-and-place locations. It is this use of visual servoing which shows the most relevance and promise, as the addition of the camera promised much-added data and utility which can be leveraged for more intelligent and dynamic decision-making.

There is even less work towards optimising robotic agents in an attempt to increase their own efficiency within manufacturing. Work which does investigate the optimisation of industrial robotics normally focuses on inspection tasks, as it is generally a matter of speed vs accuracy. There is little work discussing the optimisation of an agent which can use an end-effector to complete laborious manufacturing tasks, as well as agents which are capable of completing a variety of tasks. Additionally, much of the visual-servoing-focused manufacturing research being conducted investigates increasing either accuracy or time when interacting with small workpieces, things which can be fully viewed by a camera.

Research completed to complete each objective contains a mixture of foundational and novel work, some of which will add new novelty to the preexisting literature.

- The research completed to achieve objective 1 contributed to the literature by identifying and specifying a variety of environmental variables, and by providing suggestions on cameras for an edge detection and tracing visual servoing system. While many of the investigated environmental variables have been investigated in the past, the combination of variables investigated for this particular purpose is novel. Additionally, the types of workpieces being interacted with are novel, both the specific use case of them, but also the spatial profile and properties. Specifically, the feature detection and tracing of obtusely curved workpieces is novel, and was not previously published within the literature.

- The research completed to achieve objective 2 contributed to the literature via the creation and experimentation of various division of labour algorithms. All previous publications regarding the use of mobile manipulators in manufacturing generally focus on the end product - how effective a chosen solution is at solving a particular manufacturing task. This work focused more on the algorithm creation stage, and investigated the reasons behind certain algorithmic decisions, specifically how much movement was completed by different parts of the system.

- The research completed to achieve objective 3 contributed to the literature in a similar manner to objective 2, it focused on a rarely-published part of mobile manipulators: the distribution of labour. By taking the results of the previous algorithm into consideration, a novel algorithm was created which could optimally trace an partially-observed, spatially changing workpiece. This was a novel concept, not only as a research area, but also as an algorithm output. The use of such an algorithm for the use of aerospace manufacturing was also novel, expecilly since the development priorities were different to much of the related literature.

- The research completed to achieve objective 4 benefited both objective 2 and 3, and ensured their novelty in regards to the literature. This was done by putting all experiments in the reference of aerospace manufacturing, by using aerostructure inspired workpieces for testing. The field of aerospace manufacturing tends to shy away from generalised robotics due to the high value, low volume nature of the manufacturing, and the high precision needed. By specifically targeting the aerospace sector, this research has a continual level of novelty to it.

This research would add to existing knowledge by research groups such as the Autonomous Intelligent Systems group at the University of Freiburg. Much of their research focuses on the use of intelligent robotics within an industrial setting, and this research will build upon it, and attempt to make it more flexible and general-purpose. There will also be a focus on large-scale workpieces and developing dynamic visual serving systems which can optimally interact with any workpiece, regardless of its size or spatial variance.

This will contribute to the drive towards automation within industry by taking the current state-of-the-art technology and adapting it for more flexible roles in industries such as aerospace.

3

Implementation of a flexible solution for feature detection and tracing of manufacturing workpieces

3.1 Introduction

For a kinematic robot to interact with a manufacturing workpiece, data must be provided to the robot for it to know where and how to interact. This is primarily done via sensors, and no sensor is as flexible as vision. A high quality vision sensor can provide both raw data, as well as contextual information, and can be easily analysed and leveraged to aid the robot in making decisions when interacting with a workpiece. It is important to ensure that when using a vision sensor, it is being used in the best possible environment, else the data can be easily corrupted.

Experiments were conducted to establish major environmental factors, and the best way to manipulate them to mitigate any major ill effects they may have on image quality, ensuring the best possible data is gathered from the vision sensor. Additional experiments were also conducted to compare multiple cameras under ideal conditions, to ensure the most suitable sensor was used for feature detection and tracing. Once the camera was chosen, experiments were conducted with a robot to test the detect and trace capabilities of a visual servoing system against a variety of manufacturing workpieces.

3.2 Methods and Materials

3.2.1 Selection of Physical Cameras and Kinematic Robot

A camera was to be used as the primary sensor of the robot, allowing visual inspection of the workpiece. The camera was required to have the following features:

- Lightweight: the camera would have to be mounted onto the chosen robot for it to be considered an eye-in-hand visual servo system. Due to this, it was beneficial for the camera to be as light as possible, as that would allow a broader range of robots to be considered.

- Hardware Compatibility: the camera would have to interface with a PC, to allow its image stream to be accessed and manipulated. The easiest way of doing this would be to ensure the camera was able to send live video via a common connection protocol, such as USB.

The simplest type of cameras to meet these requirements were RGB cameras. They're typically lightweight due to the simplicity of their design and can normally interface easily via a standard image stream over USB. Three of the most commonly used RGB cameras for robotic vision were selected in order to find the most suitable camera for edge detection. The three cameras were:

a) Firstly, a Basler Ace camera was used as it is the industrial standard for RGB cameras. More specifically, it was a Basler acA2040-55um camera with a Moritex ML- M1616UR lens – this setup would give us suitable results for a reasonable price. The camera itself has a Sony CMOS sensor and can output 57fps at full resolution, which was deemed appropriate for its purpose. It also shoots in monochrome, which was preferred to full RGB colour, as it aids the detection of edges, and we have no particular use for RGB capabilities.

b) The second camera would be a standard webcam, something similar to a Logitech C920. The webcam would have to shoot in HD and could either be monochrome or full RGB. The reasons for opting to include such a camera are twofold: not only can it can be used as a direct comparison to the Basler Ace, but it can also be used to help justify the price of the Basler Ace. As the Basler is priced ten times higher than the Logitech C920 (or comparable webcams), it would be worthwhile testing out an alternative camera.

c) The final camera was an Intel RealSense D435i, a low-cost RGB camera which has depth-sensing and SLAM capabilities. The SLAM capabilities are provided by the camera's Inertial Measurement Unit (IMU), which measures the movement of the camera and which can be used alongside the agent's own odometry data (alongside any other sensors it may have). The depth sensor itself can detect objects up to ten metres away, but since the workpiece will be much closer, it can be expected to be more accurate. The camera sensor itself is high quality, on par with the HD webcam, being capable of 1080p at 30 frames per second.

In addition to cameras to test, a kinematic robotic arm is required to trace the detected features of the workpiece. Ideally, the kinematic arm would have the following two features:

- Lightweight: the robotic arm will need to be manoeuvred around a lab setting to be put into different environments. This is required to simulate tracing a large workpiece with a variety of edges and angles. By being physically lightweight, it makes it easier to be moved around, allowing a wider variety of angles to be tested.

- Software Compatibility: the robotic arm will need to be integrated into the existing workflow, so it can take data from the camera, and act on it to move. The easiest way to do this is to find an arm which is open source or has Robot Operating System (ROS) compatibility [65]. By being compatible with ROS, it would be able to accept instructions from both the camera and software sources such as MATLAB. After an investigation into kinematic arms, it transpired that the best option was a ROS-compatible robot, as they are generally lightweight, and useful for lab-based experimentation.

A Commonplace Robotics Mover6 was found to be suitable, as it was lightweight enough to move around (weighing 3.5kg) while also being ROS compatible. Additionally, the arm has a maximum payload of 400g which means that it would comfortably support the majority of cameras connected to the end-effector.

3.2.2 Selection of Software

OpenCV [66] was used to develop the overall system, providing an off-the-shelf edge detection algorithm. Additionally, the camera required its own dedicated software to function correctly, specifically pyrealsense2 [67].

ROS was chosen as the ideal robotics middleware as it is the most well-known and open source, allowing some cameras to interact directly with it and other software packages entirely [68]. Specifically, ROS Melodic, as it is the version targeted at Ubuntu 18.04.

By using ROS, the robot was easily controlled as it subscribed to joint demands, which were generated by MATLAB. The MATLAB robotics' toolkit published ROS messages, as well as providing a robust inverse kinematic function. Using both of these MATLAB features allowed a singular software solution for moving the arm to the desired location, which was both highly flexible and lightweight. The inverse kinematic function was able to calculate joint demands for the robot, and then these were published via ROS to the robot to allow it to move.

In addition to preparing software for real-world experiments, a simulator was used for more precise measurement experiments and for simulating curved surfaces for detection. Gazebo, a robotics simulation engine [69], was the primary and most obvious choice due to its capability to: integrate with ROS [70], simulate the selected cameras and the kinematic arm, and execute movement commands from MATLAB.

3.2.3 Algorithm Design

Acute Edge Detection

Once a camera was selected to do edge detection, and an optimal environmental setup was created, the next task was identifying the detected edge and enabling a robotic arm to move towards that location (Figure 3.1). Canny Edge

Detection [71], a longstanding edge detection algorithm, was used for all edge detection experiments, alongside an additional algorithm that automatically tuned the thresholds for edge detection [72]. While this algorithm did not produce results as perfectly as a well-tuned edge detector, it did produce results that were suitable for the purpose, and required no tuning while also increasing the overall completion speed of the solution. The camera detected the edges of the workpiece with Canny Edge and searched the edge for a pixel pair with the lowest X value – which would be a point furthest to the left of the frame. The depth to the workpiece was then obtained via various methods to provide the camera with the Z coordinates. The pixel pair and depth coordinates were then used to deproject them into 3D camera coordinates. These 3D coordinates were moved to MatLab and an offset and transformation matrix was added to turn them into world coordinates and input them into the inverse kinematic mover for the robot. While some cameras used were able to easily get the distance between themselves and the workpiece (such as the Intel RealSense), there are alternative methods to get the depth, such as using deep learning [73], or treating the video as a stereo camera system with a camera with a large FOV [74]. The local coordinates were transformed into global coordinates by offsetting the camera position and using a transformation matrix. Finally, the new global coordinates were fed into an inverse kinematic system which output the necessary joint demands for the robot arm to move to the specified point. This workflow was repeated with the arm moving from point to point until it had traced the whole edge.

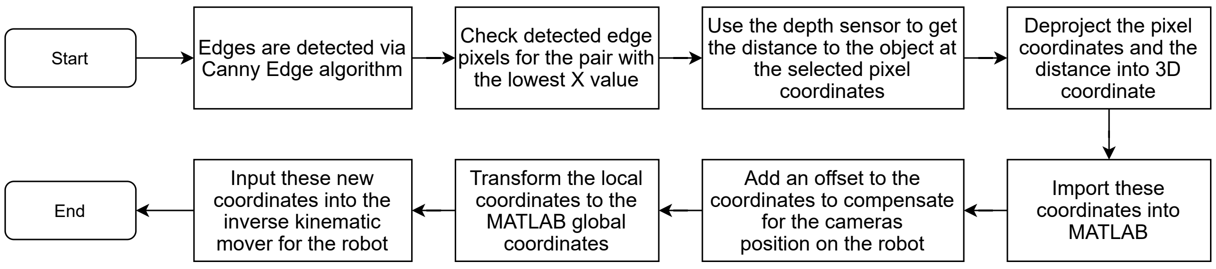
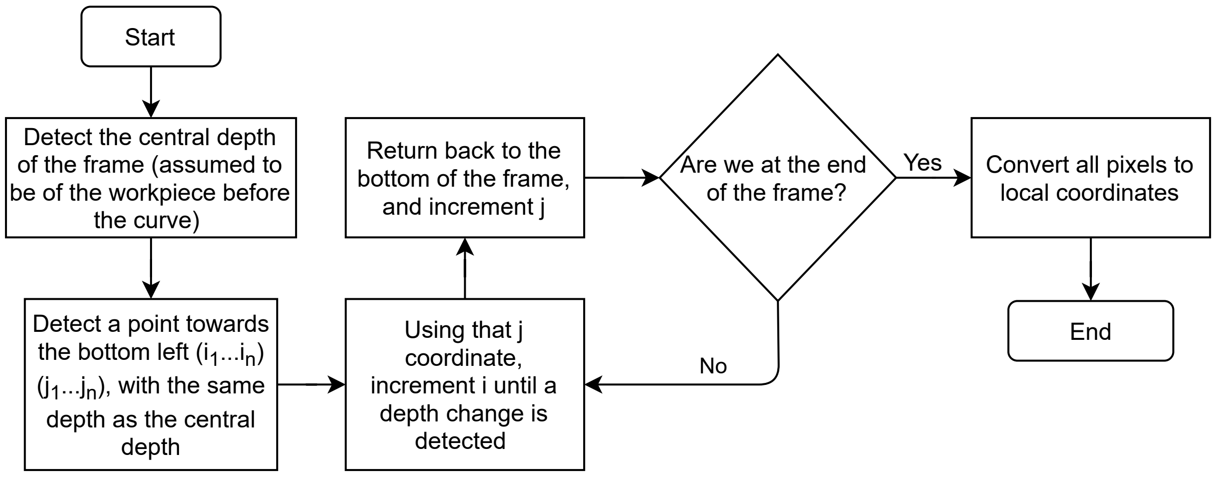

Start
Edges are detected via Canny Edge algorithm
Check detected edge pixels for the pair with the lowest X value
Use the depth sensor to get the distance to the object at the selected pixel coordinates
Deproject the pixel coordinates and the distance into 3D coordinate
Import these coordinates into MATLAB
Add an offset to the coordinates to compensate for the cameras position on the robot
Transform the local coordinates to the MATLAB global coordinates
Input these new coordinates into the inverse kinematic mover for the robot
End

Figure 3.1: The workflow used for detecting and moving towards edges with the Mover6

Obtuse Edge Detection

Some types of edges did not have a well-defined, acute edge – having edges with a low rate of curvature, similar to a cylinder. For features like this, the point at which the workpiece's face transformed into a curved edge needed to be detected. This was achieved by initially measuring the distance from the arm to the workpiece and then systematically moving across the workpiece, looking for the point at which the depth information subtly changes. This change was generally where the curve began, which is the feature that needed to be identified. Figure 3.2 demonstrate how the algorithm worked when the camera and robotic arm viewed the workpiece in a perpendicular position.



Start
Detect the central depth of the frame (assumed to be of the workpiece before the curve)
Detect a point towards the bottom left (i_1...i_n) (j_1...j_n), with the same depth as the central depth
Using that j coordinate, increment i until a depth change is detected
Return back to the bottom of the frame, and increment j
Are we at the end of the frame?
Yes
No
Convert all pixels to local coordinates
End

Figure 3.2: The workflow used for detecting and moving towards edges with the Mover6

Initially, the centre of the frame was used to provide a depth reading for the camera, to inform how far the camera was from the flat face of the workpiece. The algorithm then searched the face, initially from the bottom left pixel, incrementing the vertical coordinate until a depth change was detected. The point was saved in an array, the vertical coordinate resets, and the horizontal coordinates were incremented by one. This continued until the whole image was systematically searched, outputting an array of points where the flat surface starts to curve. These coordinates were then converted to local coordinates, in preparation for tracing by a kinematic arm.

Compared to the algorithm used for acute edge detection, this algorithm relied more heavily on the point cloud data provided by the stereo camera being used, as it was the primary sensor. Other works have also used a similar offline technique [75]-, allowing accurate, real-time information related to the structure at the time of operation.

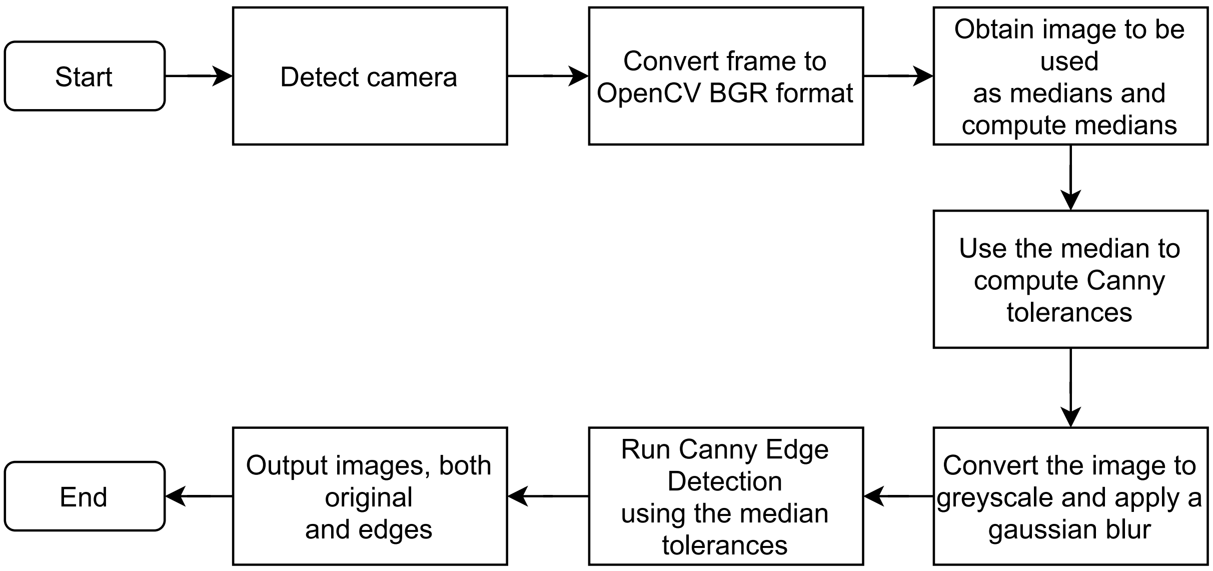
3.3 Experimental Setup

3.3.1 Environmental Variable and Camera Comparison Experimentation

A set of environmental experiments were designed, using the factorial design technique to understand the effect and relationship between significant environmental variables [76]. The four environmental variables used were: Camera Focus, Camera Aperture, Environmental Lighting and Distances from Target. These variables and their factorial maximum and minimum settings can be seen in Table 3.1. The use of the factorial design of experiments resulted in an increased speed of experimentation due to variables being limited to the minimum and maximum setting, and allowing the calculation of both dependency and interdependency scores for each variable and combination of variables. Each variable combination was recorded, and a test image was taken with an industrial RGB camera using the workflow shown in Figure 3.3. Once the camera was detected, and an image frame was grabbed, proprietary software was used to convert the image frame to an OpenCV compatible BGR colour formatted image. Tolerances were then generated from this image using the median pixel density, limiting the amount of tuning needed. Finally, Canny Edge was run with these settings, and an output image was exported and analysed. These images were systematically scored against a ground truth test image to show the effects that a combination of variables had on image quality and feature detection.

Table 3.1: A table showing the different variables being used in the experiments, and what the maximum and minimum settings are.

Variable	Maximum (+)	Minimum (−)
Focus	Maximum	Minimum
Lighting	862 Lux	478 Lux
Aperture	1.8 f	16 f
Distance	143 cm	75 cm



Start
Detect camera
Convert frame to OpenCV BGR format
Obtain image to be used as medians and compute medians
Use the median to compute Canny tolerances
Convert the image to greyscale and apply a gaussian blur
Run Canny Edge Detection using the median tolerances
Output images, both original and edges
End

Figure 3.3: The workflow used for detecting the dependency values of different environmental variables

Once the dependency and interdependencies of variables were established, an optimal environmental setup was created. With this optimal setup, the three cameras were taken and compared against each other as well as against a ground truth image. After grading each output image, it was then established which camera would be the most suitable for edge tracing.

A workpiece was cut from 3.6mm plywood into a sawtooth design with 0.5mm vertices on one side. The workpiece was placed 95cm off the ground at varying distances, as shown in Figure 3.4. This setup allowed easy modification of the different variables to understand the different environmental factors impact-ing edge detection quality. This setup was then used again in order to compare three different cameras against each other with an optimal environmental setup.

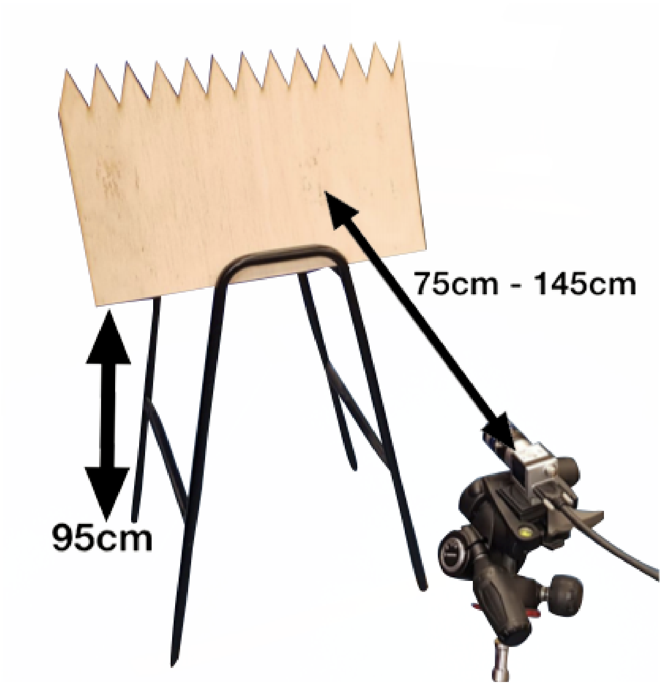


Figure 3.4: A photo of the environmental setup, showing the environmental parameter experiments

3.3.2 Acute Feature Detection and Tracking

Experiments were conducted that tested the algorithm's edge detection and tracing ability when parallel to a face with a 90° acute edge. A setup was created in Gazebo, to allow easier and quicker changes to the setup. The workpiece was suspended 90cm off the ground, with a camera an optimal distance away. The algorithm detected the edge and attempted to trace it moving from point to point. These experiments aimed to ensure that the algorithm could detect the correct edge and corresponding local coordinates, ready for tracing.

Once the pixels were identified, they were converted from the camera's local coordinate system into 3D coordinates and into the robot's global coordinate system. Once completed by deprojecting the 2D coordinates into 3D coordin-

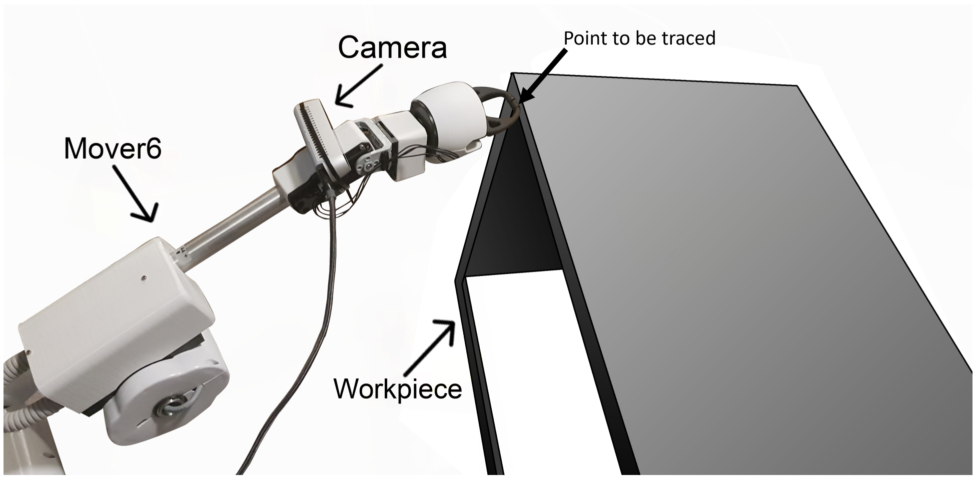


Figure 3.5: The tracing experimental setup with a hollow cardboard workpiece measuring 500mm by 350mm by 350mm

ates and applying a transformation matrix and offset, an experiment was conducted to test the agent's accuracy. The transformation matrix represented the difference between the current state of the robotic joint the camera is mounted to, and the robot's base, as this contained both the distance and the rotation of the camera compared to the bottom of the robot. The physical robot was given a hollow 3-dimensional workpiece to trace, and the initial point was identified (Figure 3.5). A physical marker was then placed on the workpiece where it would be expected to move, allowing a comparison between the predicted and actual positions. The hollow workpiece was 1m by 1m and had 10cm edges to be traced. The edges were made thick enough to support the structure while also ensuring the workpiece tracing was hollow on the inside to measure the offset and ensure no damage to the kinematic arm.

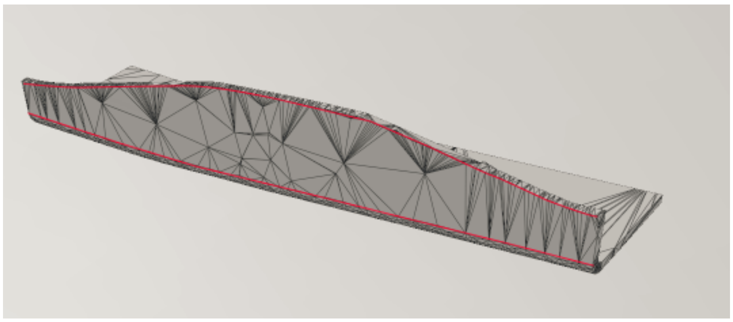


Figure 3.6: An example aerostructure workpiece, digitally marked with the detected edge

In order to verify the simulated experiments, some final, simulated, acute edge detection experiments were conducted. An example aerostructure was imported into the simulator (similar to the one seen in Figure 3.6), and various amounts of Gaussian noise were added to the camera feed. The aerospace structure was roughly 1 metre long, and 10 centimetres high, and had a variety of spatial changes along its prominent edge. As Table 3.2, 16 different levels of Gaussian blur were added by modifying the variance (amount of noise) and mean (saturation of the noise) variables. These were calculated by experimenting until the output image was unreadable, therefore getting a broad range of variables. The Gaussian blur results were compared against an averaged ground truth result, which had no blur. The ground truth was obtained by calculating the average of 10 different non-blurred results to ensure that it was completely accurate.

Table 3.2: A table showing the experiments completed, and the corresponding amounts of Mean and Variance

		Mean			
		0	**4**	**8**	**18**
Var	**0.3**	1	2	3	4
	6	5	6	7	8
	20	9	10	11	12
	50	13	14	15	16

3.3.3 Obtuse Feature Detection and Tracking

Multiple experiments were planned to test this algorithm and its effectiveness in finding the point at which curvature starts. A simulated setup was created where a 1x0.5x0.5m cuboid with an equally-sized semi-cylinder on top was spawned. The camera was spawned facing this object at the height of 30cm off the ground (Figure 3.7). All absolute positions of the workpiece were recorded, and multiple runs of the simulation were performed. The aim was to see how accurately the curved edge was detected compared to the known absolute values. Throughout these tests, the distance and angle of the workpiece varied, ensuring that the algorithm was robust at a range of distances and workpiece inclines.

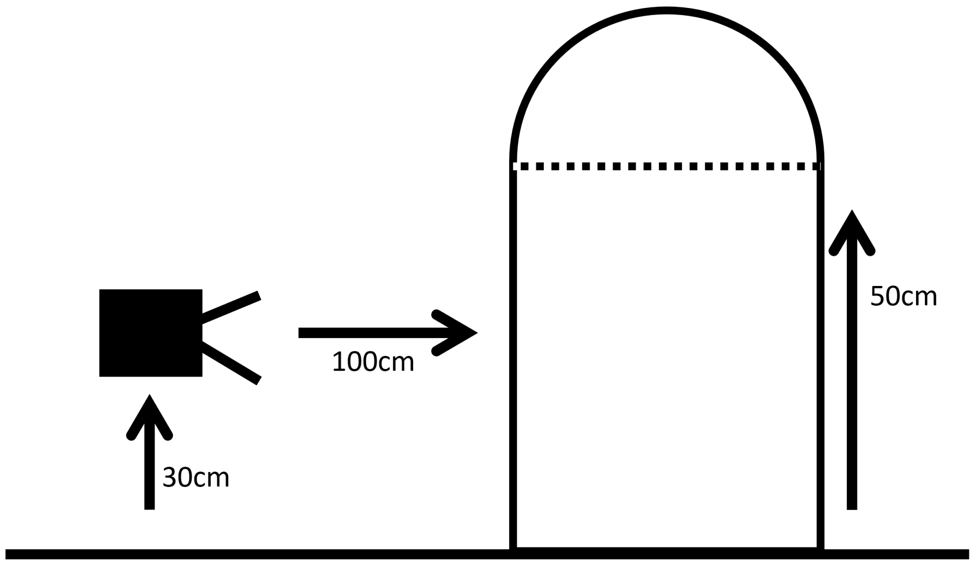


Figure 3.7: The setup for simulated obtuse edge detection

3.3.4 Physical Detection and Tracing

Once the algorithms were tested and proven in simulation, physical experiments were completed to show the algorithms used with a physical robot, camera and workpiece. A set of experiments were conducted to test the feature-tracing capabilities of the arm, initially using a hollow 3D workpiece (Figure 3.5) to trace multiple different points independently, and then three further reference workpieces for the arm to trace, inspired by existing aerostructures (Figures 3.8, 3.9 and 3.10).

The workpieces were all inspired by existing aerospace structures and incorporated specific abstract features which could be used to test the algorithm. The initial workpiece was hollow and constructed out of cardboard, measuring 500mm by 350mm by 350mm. This was initially used to safely test feature accuracy, to ensure that camera-detected features could be reached. Due to

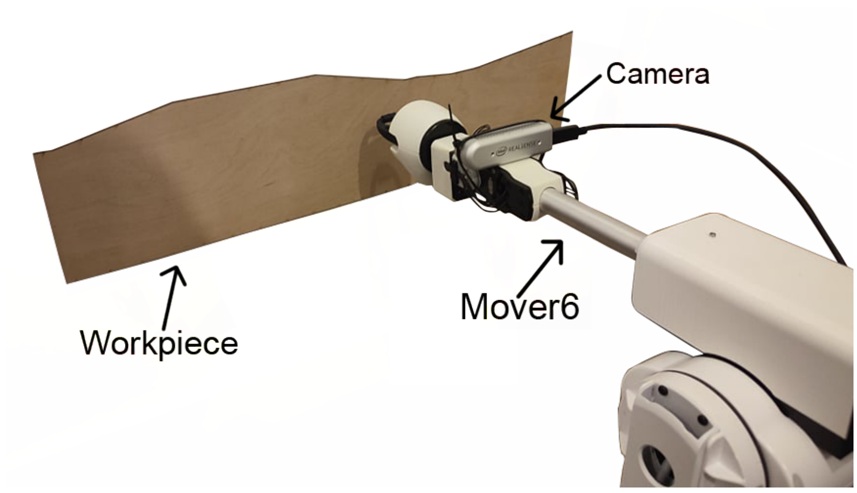


Figure 3.8: The tracing experimental setup with an aerospace-inspired plywood workpiece measuring 600mm by 170mm by 3mm

the workpiece's narrow edges, the arm's 3-dimensional accuracy could also be tested quickly.

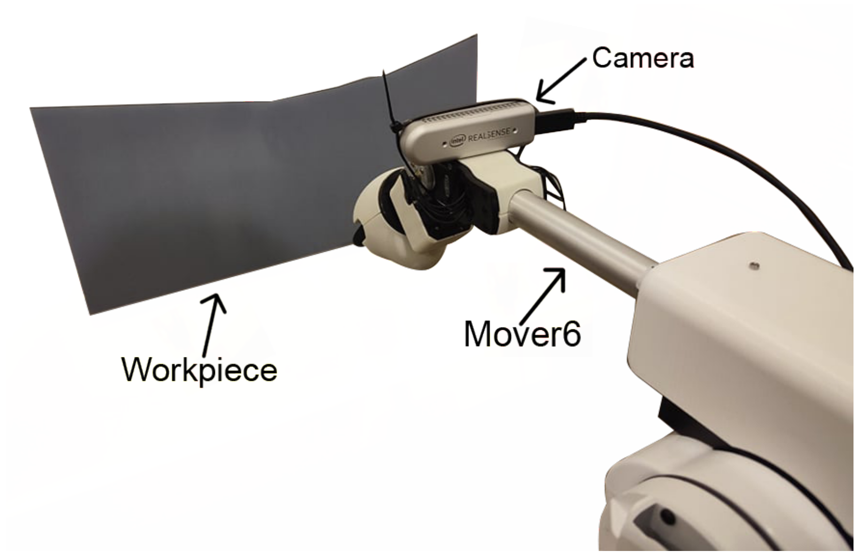


Figure 3.9: The tracing experimental setup with an aerospace-inspired concave Perspex workpiece measuring 400mm by 200mm by 3mm

Another three workpieces were also constructed, more closely mimicking typical aerostructure design and were created from a range of materials. The initial workpiece was a scaled-down aerostructure made from 3mm plywood, measuring 600mm by 170mm. Using a more abstract aerostructure design, a second workpiece was made out of 3mm Perspex and measured 400mm by 200mm.

With both of these workpieces in conjunction, the algorithm could be tested against various realistic workpieces, ranging in dimension and material.

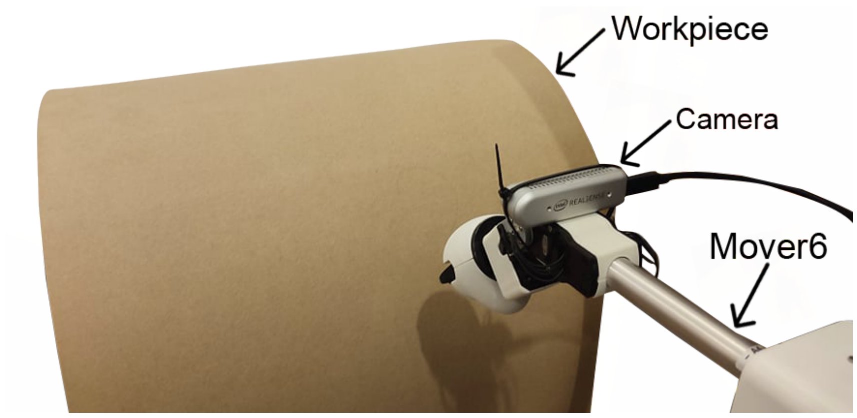


Figure 3.10: The tracing experimental setup with a curved MDF workpiece measuring 1220mm by 607mm by 6mm

A final workpiece was constructed to mimic the curved part of an aerostructure, to test the obtuse edge detection and tracing aspects of the system. It was a piece of flexible 6mm Medium-Density Fibreboard (MDF) measuring 1220mm by 607mm and was curved, creating a workpiece measuring 607mm by 470mm by 520mm.

3.4 Results and Discussion

3.4.1 Camera Experimentation

Experiments were completed where the camera would be pointing directly at a highly detailed and complex workpiece. The workpiece created was made from 3.6mm thick plywood laser-cut to size to test the robustness of the cameras and algorithms

Identification of Best Environmental Parameters

For each test case, an environmental variable was changed to give a different resulting image. The obtained standard image and an image showing the outlines were recorded. From these experiments, the effects certain variables had on the image quality were calculated and an optimal setup was identified. Table 3.3 shows the effect each variable and combination of variables had on image quality. The effect each variable had was determined by scoring all the final images out of 100 (by comparing them to a ground truth image) and evaluating the strength and clarity of the outline of the photos.

Table 3.3: A table listing all the dependency values of each experiment in numerical order

Experiment	**Dependency Value**
BC	−357
AD	−224.3
C	−39.4
AC	−16.6
ABCD	12.3
ABC	10.9
A	10.1
BD	7.4
BCD	5.6
CD	5.2
AB	−5.0
D	−3.6
ACD	−3.6
ABD	−3.6
B	−2.6

By analysing Table 3.3, conclusions can be drawn regarding the variables used and their effects on each other.

- Taken on its own, the aperture had a significant effect on the final image, as expected, as it controlled how much light was let into the lens as well as the lens' depth of focus. When combined with lighting, these became the two variables with the strongest interdependencies. This was expected as the two variables complemented each other and directly affected how the other one would affect the image.

- Similarly, the second strongest interdependency was focus and distance. Individually both of these variables had little effect on the final image but combined, they interacted to either create a focused image or one which was entirely out of focus.

- Unsurprisingly, the variable with the smallest impact was the environment's lighting, due to how the minimal lighting setting selected was not total darkness, but having one of the possible three lights on, producing a value of 478 lux. This was done to consistently provide usable images and not cause anomalies with the other variables.

These experimental results had some similarities with the work reported in literature, eg by Manish and Denis Ashok [77], where a Canny edge detection method was used – at 5.66 lux, edges became visible and detected, but at anything over 87 lux, the interior of the object was detected as well. This was reflected in many of the tests conducted, where the image's interior is fully detected. This was counterbalanced by adjusting the aperture, which aided in creating an optimal setup.

Using the experimental results and other knowledge within the literature, an optimal environmental setup was created. Figure 3.11a shows the result of such a setup. Where the workpiece was closer to the camera (within one metre), the focus was set to match this on a case-by-case basis (so the sharp features of the workpiece were clear) by setting a high aperture and a lower environmental lighting condition.

Camera Comparison Experimentation

Once an optimal environmental setup was created, the three cameras were easily compared to each other. An image was taken with each camera from

75cm away and set to the most optimal settings when available. Each image was then compared against the others and a ground truth image.

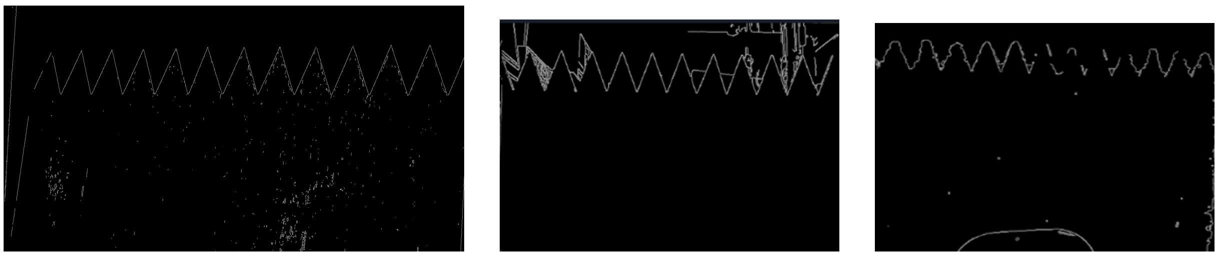

(a) Basler Ace (b) Logitech C920 (c) Intel RealSense

Figure 3.11: Examples of each cameras edge detection output with optimal settings applied

Figure 3.11 shows the three images taken on the three cameras, Basler, Logitech and Intel, respectively. By analysing these images, conclusions can be drawn about the quality and suitability of each camera.

- **Field of View:** Immediately, it can be seen that the Basler Ace and Intel Realsense had a larger field of view compared to the other camera, having at least two more fully-formed vertices in view when compared to the other cameras. While this was a benefit compared to the other two, it did not result in any tangible benefits and provided more data for the algorithm to sort through.

- **Resolution:** Both the Basler and Logitech cameras provided a higher resolution image, 2048x1536 and 1920x1080 respectively, whereas the Intel camera only provided 720x480. This, alongside the disadvantage of stereoscopic cameras, due to the possible misalignment of images, resulted in the fine points of the workpiece not being as sharply defined in the Intel image.

- **Background Visibility:** The background was seen in the Basler and Logitech images whilst not being visible at all in the Intel image. Given the technique chosen for line tracing, having the background visible meant that there would be additional data for the algorithm to sort through, increasing computation time. This could later be adjusted via post-processing to remove the background if desired, although this would slow down the process and require extra computational power.

- **Hardware Customisability:** The Basler gave the most customisability out of the three cameras, allowing both focus and aperture to be changed manually. The other two cameras adapted this dynamically, making it much harder to customise certain aspects of the cameras' settings.
- **Ease of Use:** Workflow-wise, the Logitech was the easiest to work with the Canny Edge Algorithm, as it didn't need any additional libraries, and worked directly with OpenCV. In contrast, the Intel RealSense and Basler Ace required additional libraries to work within OpenCV (PyRealSense2 and PyPylon, respectively).

In conclusion, all three cameras were suitable for edge detection, although certain situations would benefit from the use of specific cameras. If customisability was needed, the Basler Ace (Figure 3.11a) was found to be the best, but if ease of use and OpenCV compatibility were required, then the Logitech (Figure 3.11b) was the most suitable. Finally, the Intel RealSense (Figure 3.11c) was recommended if depth-sensing was needed, or workflow would benefit from lightweight background removal.

The Intel RealSense d435i was chosen to detect and trace the edges of the workpiece due to the depth-sensing capabilities, allowing the distance of the workpiece to be obtained without any intrinsic calibrations needing to be completed. Additionally, the edges of the workpiece were detected clearly, and the solution was lightweight and required minimal expert setup. Using this camera, the various edges were identified, extracted and then finally traced.

3.4.2 Feature Identification and Tracing

To show the feature-detection capabilities of the method, experiments were conducted individually on both the acute and obtuse detection systems, which built on top of each other.

Two sets of analyses were conducted:

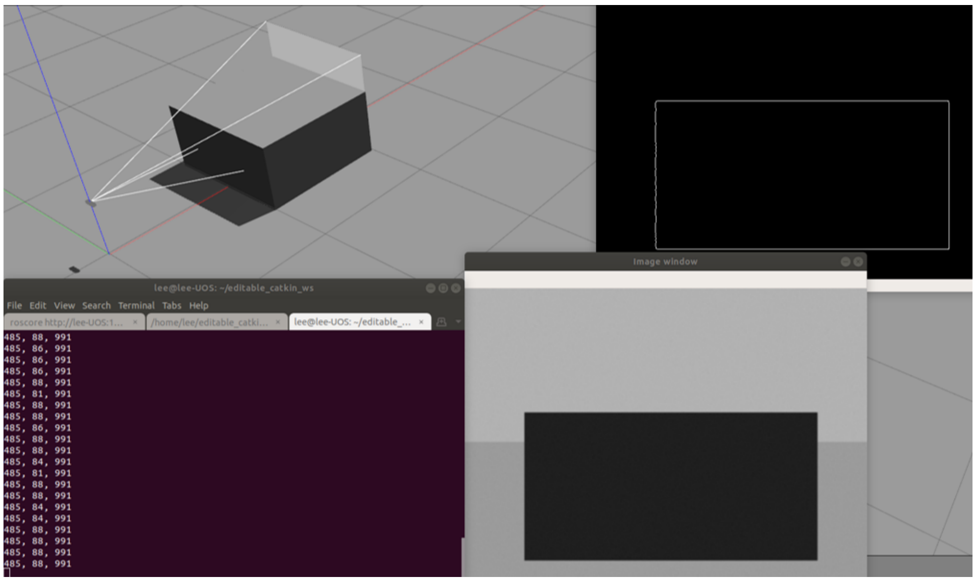
a) To check the robustness of the acute detection, a workpiece was positioned in front of a virtual camera, and two checks were made. Firstly, to ensure that the camera could successfully detect the outline of the workpiece, and secondly, to ascertain that the initial detected pixel of the traced line was the expected, correct pair.

b) To check the capabilities of the obtuse edge detection, a workpiece was placed 1 metre away from a simulated Intel Realsense camera positioned 30cm up in the air. The camera was then tasked with finding the point at which the curve began, and then tracing this point across the whole shape. The resulting 'edge' would then be traced by a kinematic arm, a task needed in specific manufacturing tasks.

Acute Feature Identification and Extraction

The first condition was checked purely by viewing the output image, whereas the second condition was checked by having the specified pixel printed out on screen and checked against a known value. By testing in this way, both the accuracy of the edge and initial pixel detection was established. This is important since the camera's edge detection, and specifically, initial pixel detection, was the primary source of information for where the kinematic arm needed to move to for workpiece tracing.

A simulated workspace was created, where a 1m x 0.5m rectangular workpiece was placed 0.6 metres away from an Intel Realsense D435 camera at various locations within the world (Figure 3.12). These locations varied by moving the object from left to right across the camera's frame and increasing or decreasing the distance to the camera by 10cm either way. In eight out of the ten tests conducted, the camera successfully detected the rectangular workpiece and identified the first pixel of that edge. The two exceptions were anomalies where more than one face could be seen in the frame, resulting in an extra edge being detected initially. This shows the importance of the workpiece needing to be parallel to the camera and, ideally, as central to the image frame as possible.

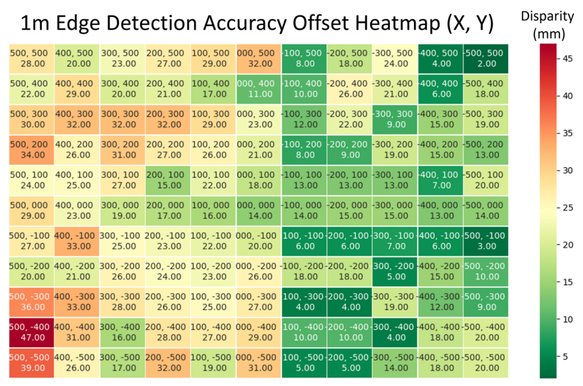

lee@lee-UOS: ~/editable_catkin_ws
File Edit View Search Terminal Tabs Help
roscore http://lee-UOS:1...
/home/lee/editable_catki...
lee@lee-UOS: ~/editable_...
Image window

```
485, 88, 991
485, 86, 991
485, 86, 991
485, 86, 991
485, 88, 991
485, 81, 991
485, 88, 991
485, 88, 991
485, 86, 991
485, 88, 991
485, 88, 991
485, 84, 991
485, 81, 991
485, 88, 991
485, 88, 991
485, 84, 991
485, 84, 991
485, 88, 991
485, 88, 991
485, 88, 991
485, 88, 991
```

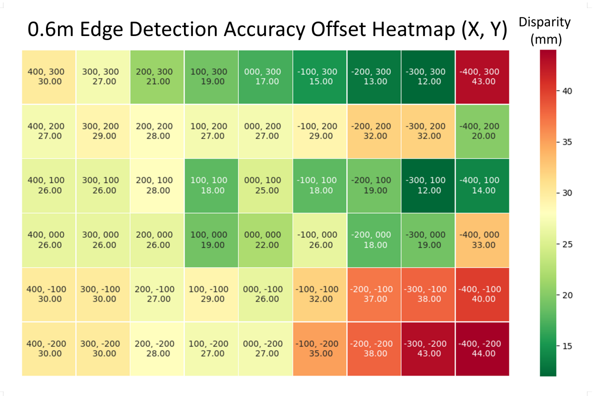
Figure 3.12: A simulated setup for the camera to detect the edges of the workpiece. **Top Left:** The Gazebo simulation. **Top Right:** The Canny Edge output. **Bottom Left:** The pixel detection output. **Bottom Right:** The raw camera output

Once it was established that the outline could be detected and the correct pixel set could be found, a test was run to ensure that the correct local 3D coordinates could be deprojected. By deprojecting the coordinates, this allows a local 3D view of the camera and workpiece to be established, an essential step in allowing the arm to trace the workpiece without needing preprogrammed information relating to the position of the workpiece. A test was run where a small workpiece was moved systematically across the camera's frame of a simulated Intel Realsense D435i. The output coordinates of the first detected pixel were then checked for accuracy by geometrically measuring the distance from the camera to the initial pixel. This data was then used to inform the accuracy of the deprojection and data related to the accuracy across the frame.

As seen in Figure 3.13, the results were mapped to a heat map – Figure 3.13a



(a) The workpiece edge detection disparity heatmap from 1m distance



(b) The workpiece edge detection disparity heatmap from 0.6m distance.

Figure 3.13: The workpiece detection heatmaps, displaying disparity between detected results to a known ground truth

was 1m, while Figure 3.13b was at a distance of 0.6m. By looking at the heat maps and the discrepancy between what was recorded and the ground truth, patterns began to emerge. While the overall accuracy of the deprojection showed a discrepancy of 50mm or less, a large amount of the detected pixels were within 20mm or less, an acceptable tolerance given the tracing task requirements. Additionally, there was a clear accuracy difference between the left and right sides of the frame, broadly favouring the right side at all distances. This is related to the Intel Realsense's lens setup and how deprojection is achieved. Since the depth-sensing lens was positioned on the device's right side, and deprojection relies on depth data to inform its Z coordinate, this difference wasn't unexpected. This holds less true as the camera gets closer, showing a pattern of poor accuracy towards the bottom-right of the frame as the camera gets closer. This is due to the algorithm incorrectly interpreting the environment's Z value as the workpiece's, something which can be corrected via making multiple readings before assigning a Z value. By identifying these eccentricities, this information could be leveraged for later experimentation when designing control patterns for the inspection and observation of workpieces, prioritising the more accurate side of the frame. Specifically, if using a probabilistic model of control, more certainty could be applied to the upper-right side of the lens to decrease the uncertainty of the world state when compared to just observing with the left.

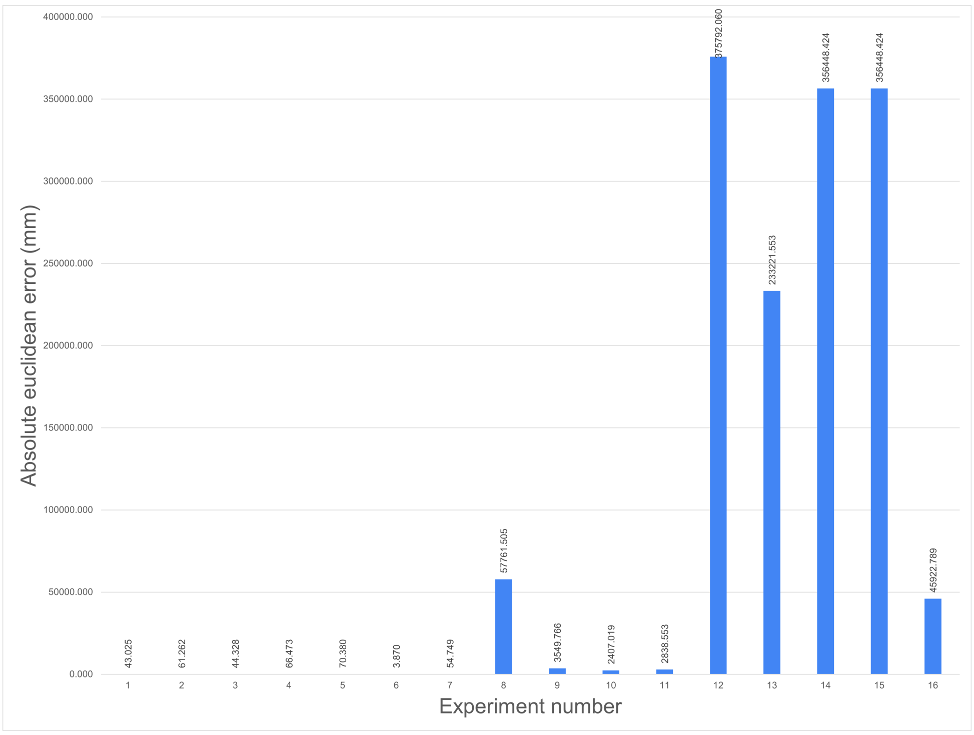


Figure 3.14: A bar chart showing the Euclidean distance between the ground truth and each experiment

Once the algorithm detected a simple workpiece successfully, it was tested on a complex workpiece with additional noise for verification. Figure 3.14 shows the absolute disparity between the Gaussian blur and the averaged ground truth output. The algorithm is robust enough not to be affected by blur with a variance of 6 or less until the mean is increased to its maximum setting of 18, at which point it deviates from the ground truth. This is because the edge-detection algorithm is sensitive to light, so once the noise gets heavily saturated, the algorithm will start detecting false positives more frequently. This pattern can be seen again for experiments 9 through 12, where the high level of variance has little effect on the algorithm until the mean is increased. The final four experiments have such a large amount of variance that the edge detector was producing many false positives. In most cases, the edge detector failed to get a precise reading. The only time it was successful was when the mean was increased to its maximum, which produced such a bright amount of noise that the algorithm ignored the noise entirely and successfully traced the edge with only a few false positives.

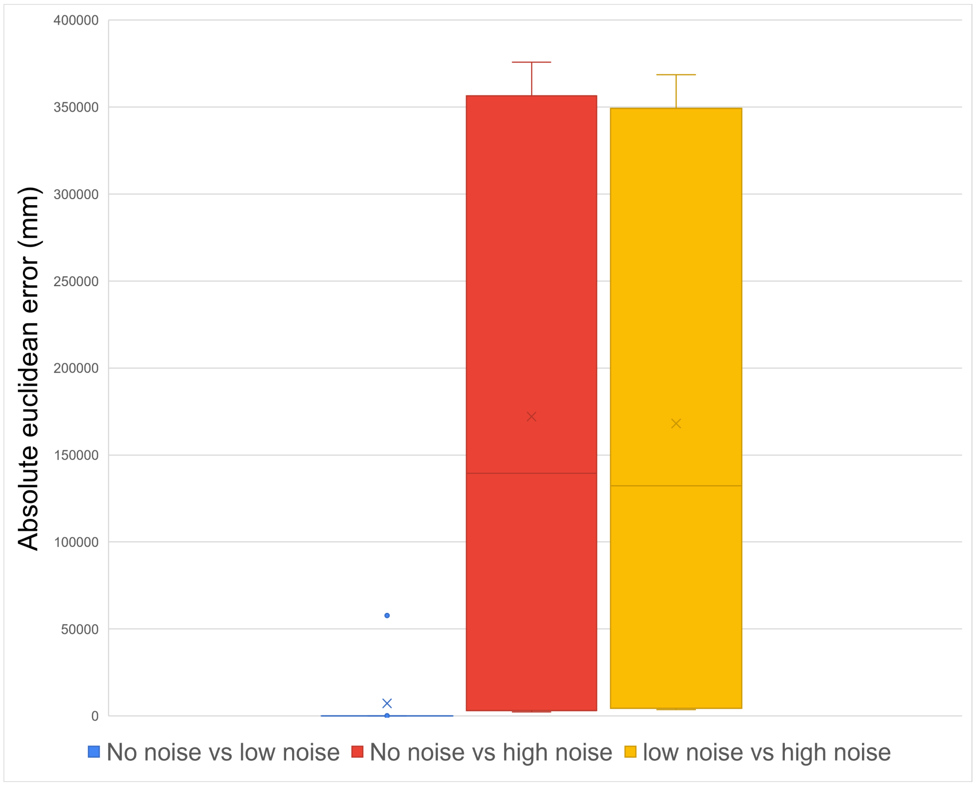


Figure 3.15: A box plot showing the relationships between the amount of noise and the absolute Euclidean error

Further analysis of the results (via Figure 3.15 and Figure 3.16) shows how the differing results compared with each other. The experiments were put into two groups: low noise (Experiments 1 through 8) and high noise (Experiments 9 through 16). By comparing these groups to each other, a better effect of the noise overall can be examined. The error of low noise compared to no noise is very low, almost having no effect, excluding the outlier with an error of 50000 mm. This can be seen again when comparing the results of the high noise group to both the no noise and low noise groups, with them only having a difference of less than 1000mm.

By adding such a varying amount of noise to these experiments, and requiring the system to trace a significantly more complex shape than in previous experiments, the system's robustness is proven. The real-world cameras previously

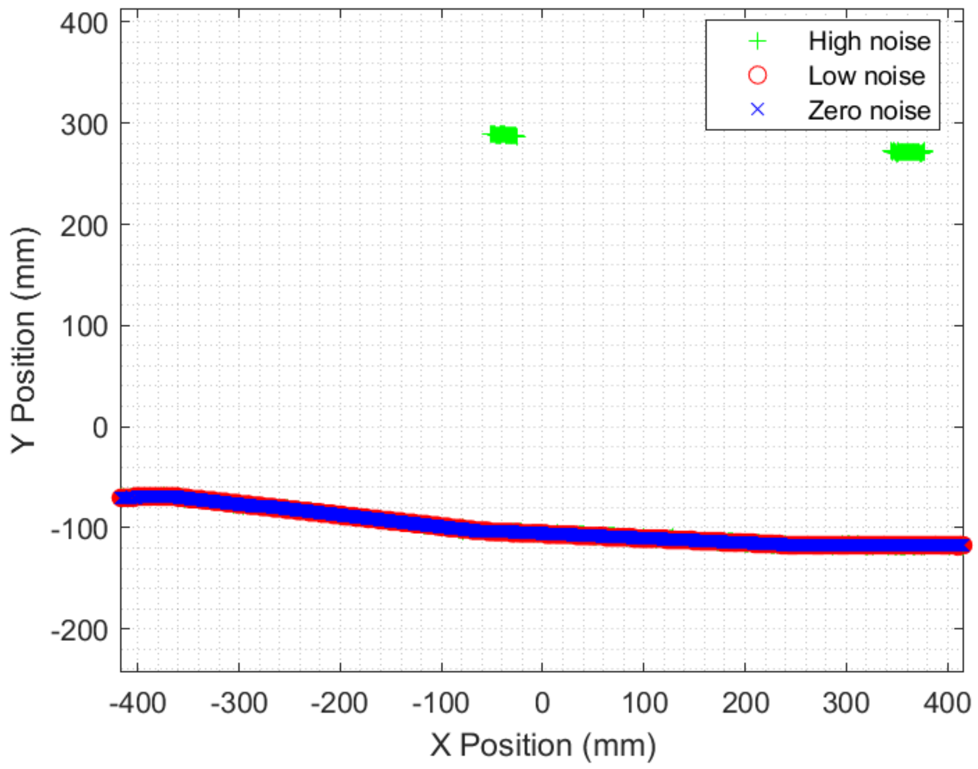


Figure 3.16: A scatter graph showing all detected points for each experiment

suggested that edge tracing would produce a noise level far less than anything used in this experiment, showing that whilst many of the experiments performed in this paper are simulated, the results they output are still valid.

Obtuse Feature Identification and Extraction

An initial test was conducted, with the workpiece and camera 1 metre away and the camera 30cm off the ground plane. The expected height of where the flat surface of the workpiece transitions into a curve was 50cm off the ground plane; therefore, the expected local height coordinate would be 20cm. The resulting detected height was a consistent 24.3cm across the whole workpiece (Figure 3.17). The resulting absolute difference between the expected and actual result was 4.3cm when viewed from 1 metre away. Other measured factors of the experiment, such as computation time, produced acceptable results – detecting the whole workpiece in under 2 seconds.

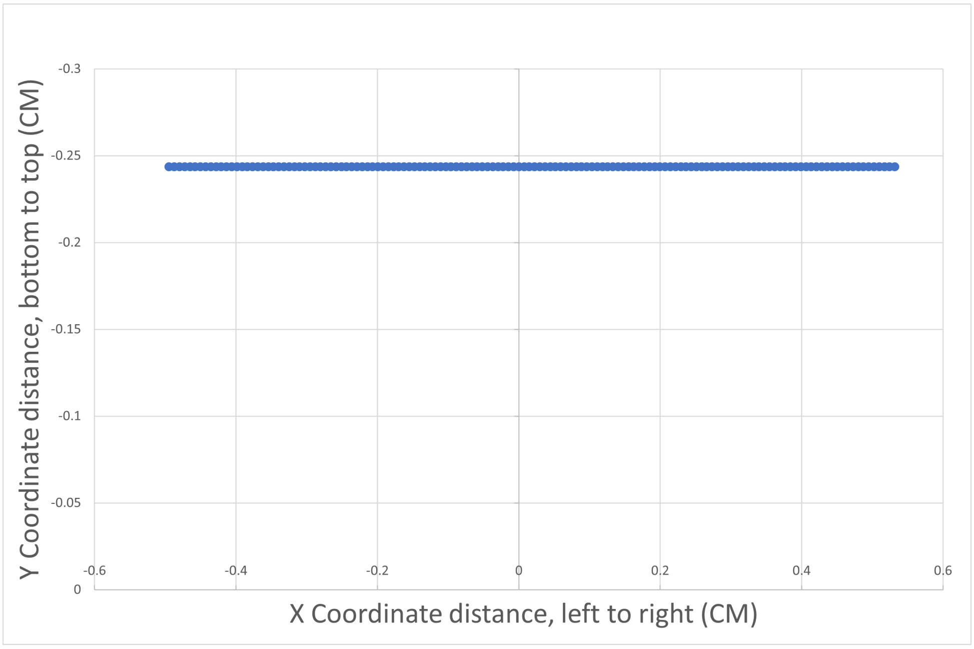


Figure 3.17: Detected points of a curved workpiece edge at a 1m distance

An additional experiment was done to verify the previous results and to understand the offset better. A comparable simulated setup was created, but this time the camera was 0.75m away from the workpiece, as opposed to 1m. The results were very similar (Figure 3.18), with an offset of 4.58cm, compared to the previous offset of 4.36cm. Given that the algorithm checked for a curvature increase every 5mm, the offset could be related to a combination of factors, such as; the rate of curvature of the workpiece (10mm ±5mm), the accuracy of the simulated depth cloud, or the position of the lens of the Realsense D435 compared to the camera's origin position.

One final experiment using a non-parallel workpiece was conducted, positioned 1 metre away from the camera at a 30° angle (Figure 3.19). The detected edge deviated by 2cm from the ground truth at its maximum height, with a maximum detected height of 34.7cm. Across the whole detected edge, the difference between the detected edge and the ground truth fluctuated between the expected 2cm and 4cm. Given this slight deviation, a line of best fit could be drawn through all these points to better-define the boundary between flat surface and curvature to provide a more accurate edge for an arm to trace.

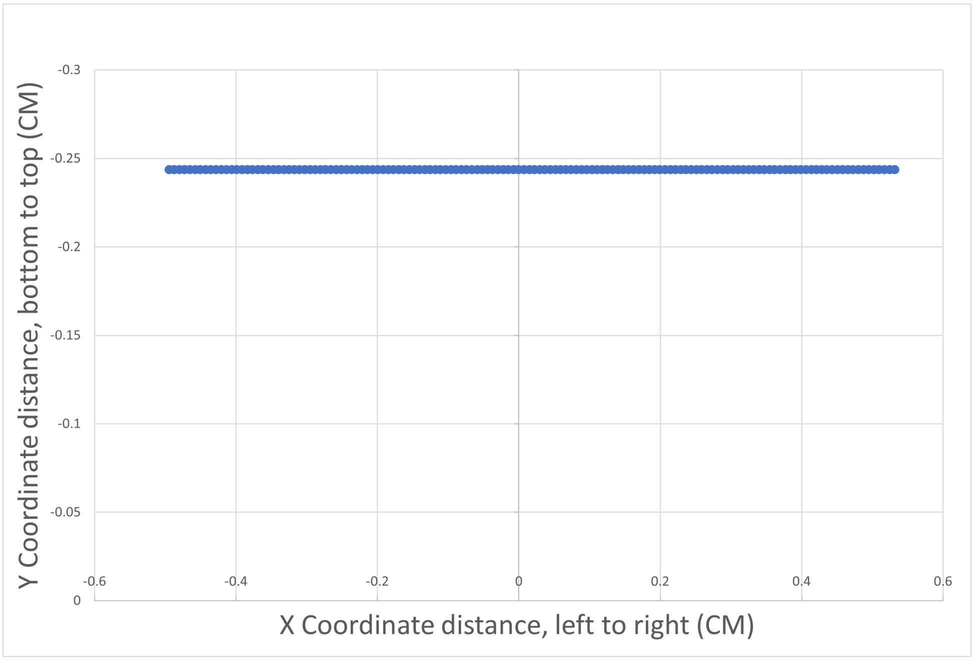


Figure 3.18: Detected points of a curved workpiece edge at a 0.75m distance

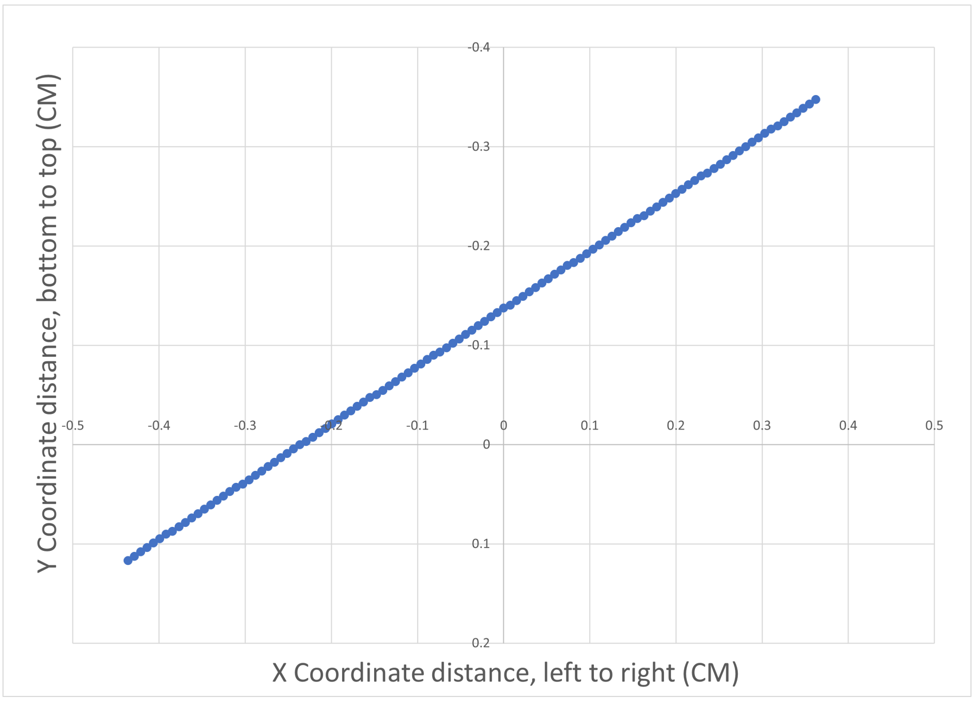


Figure 3.19: Detected points of a curved workpiece edge at 30°, at 1m distance

In all cases, the computational time for finding the points at which flat surfaces become curved was low – usually under 2 seconds from program execution. Since the algorithm relied on an organised point cloud, this allowed quicker indexing of pixel coordinates. Additionally, only the non-curved part of the workpiece is analysed (and not the whole curve), since the goal is to find the point at which the curve starts.. By leveraging the information we had, such as always having an organised point cloud, the technique shown was fast and efficient and reasonably lightweight, not requiring any heavy graphics-processing such as Compute Unified Device Architecture (CUDA).

Feature Tracing

A hollow 3-dimensional workpiece was placed in various positions alongside the workpiece, alongside the kinematic arm, to allow the kinematic arm to trace and detect different features in different positions. By doing this, a better understanding of how accurate the arm was at different distances was realised. While the robot's maximum reach was 550mm [78], this was vague in practical applications, as the height, width or distance of what the agent would be tracing is unknown until the time of detection. This is best demonstrated in Table 3.4 and Figure 3.20, where four different features were detected and traced in a variety of positions and distances, and the accuracy of the robot's final position compared to its expected position was recorded.

Table 3.4: A table showing kinematic movement test cases, the local coordinates of the detected feature, and the robot's offset once moved to the position

Test Case	Feature Position in mm (X,Y,Z)	Offset in mm (X,Y,Z)
Test 1	$-146, -15, 276$	$50, -115, 35$
Test 2	$88, -45, 34$	$-3, -125, -30$
Test 3	$-153, 94, 289$	$1, -2, 5$
Test 4	$-111, -22, 216$	$0, 3, 14$

Tests 1 and 2 were unsuccessful, due to the reach limitations of the robotic arm being used, with the arm prioritising movement in the X-axis, then Z, and finally Y. This explains why the offsets for the first two tests were minor on the X and Z axes (within 5cm) but were over 10cm on the Y-axis. Once the workpiece was positioned closer to the arm, as seen in Tests 3 and 4, the arm had much greater success in moving to the detected feature, having almost no offset – especially in Test 3.

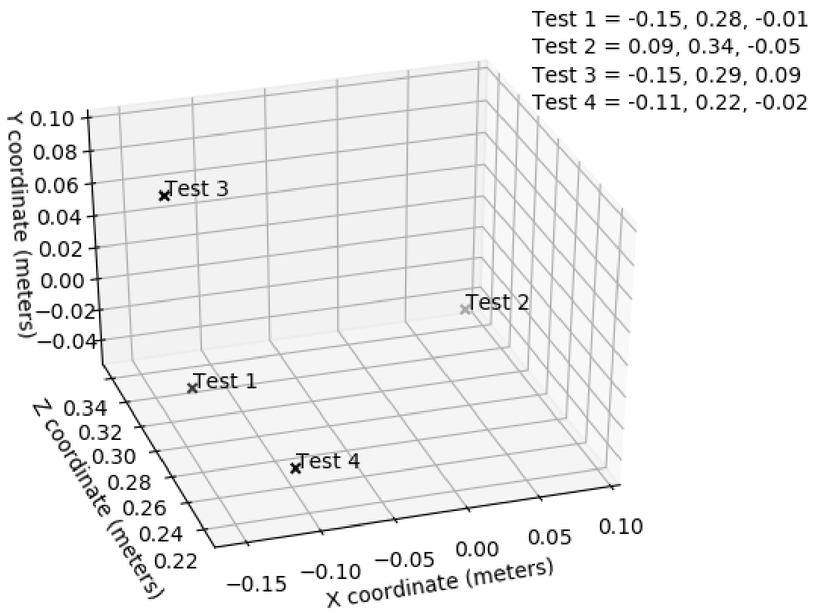


Figure 3.20: A scatter graph showing the local coordinate positions of the four test cases

$$D_t = \sqrt{(x_t + o_t^x)^2 + (y_t + o_t^y)^2 + (z_t + o_t^z)^2}. \tag{3.1}$$

Equation (3.1) takes the euclidean distance in 3D, incorporating the offset of the camera to the base of the robot. The D distance of the robot to the specified location at t time can be approximated through the use of the provided x, y and z coordinates, as well as there relative offsets, all at current t time [79]. Using the knowledge provided by an accurate depth-sensing camera, it can be estimated if the point can be reached from the robots current position, or if the robot would need to move first to avoid an unreachable target.

A final set of experiments was completed, combining all aspects of the previous experiments. Three aerospace-inspired workpieces were created out of various materials in various sizes and were all traced using a physical kinematic arm and camera. The workpiece was placed 500mm away from the robot, central to the workpiece, and in its home position, ensuring the workpiece was reachable by the arm. Its task was then to detect the visible edge of the workpiece within the frame and trace it.

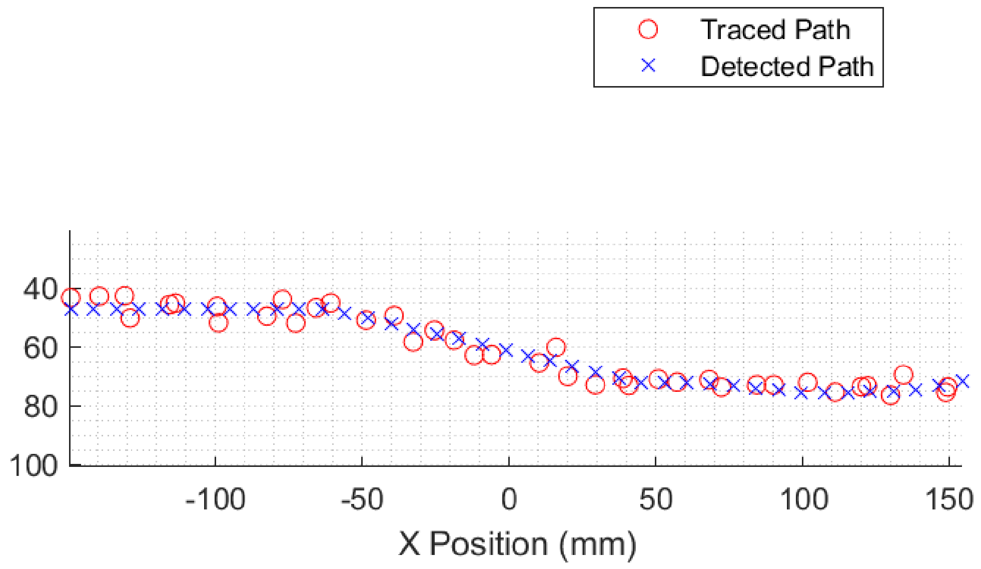


Figure 3.21: Detected points of an aerospace inspired workpiece

The three workpieces being traced were all influenced by existing aerospace structures to provide a realistic interpretation for tracing actual workpieces. The workpieces were: a 600mm cut-out of the aerostructure seen in Figure 3.6 made out of 3mm plywood; a 400mm concave cut-out made from 3mm grey-tinted Perspex; and a 600mm curved edge made from 6mm flexible MDF.

The algorithm was lightweight and fast, identifying the edge and beginning tracing within 5 seconds of activation, and detecting various edges of varying size, material and style. As seen in Figures 3.21, 3.22 and 3.23, the resulting traced patterns are accurate to 5mm on both the X and Y axes, similar to previous tracing experiments. The accuracy was within a consistent range across all three workpieces and did not fluctuate regardless of the arm's cardinal direction.

Compared with other techniques, the discrepancies experienced here using

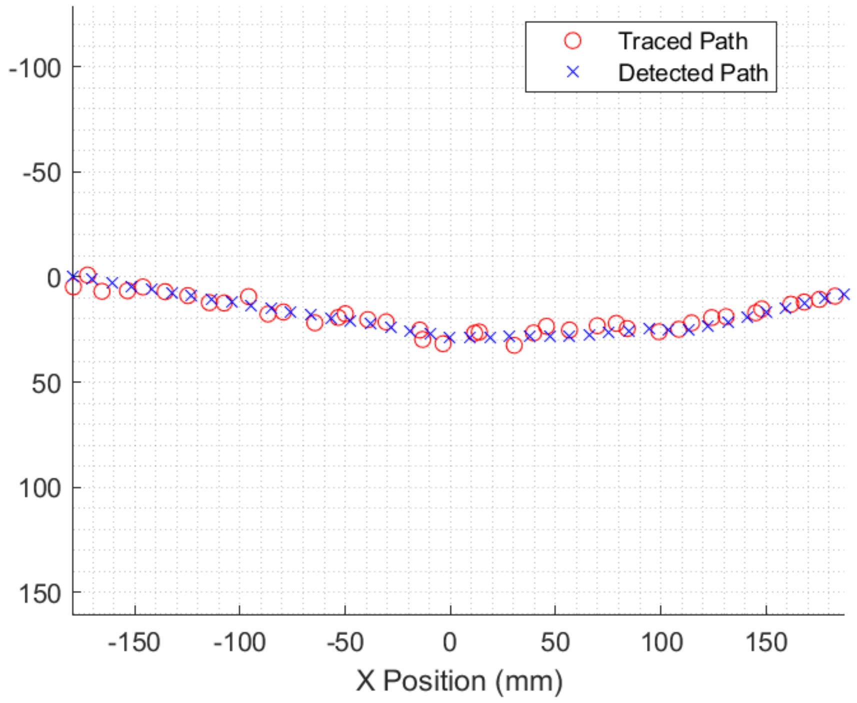


Figure 3.22: Detected points of an aerospace inspired concave workpiece

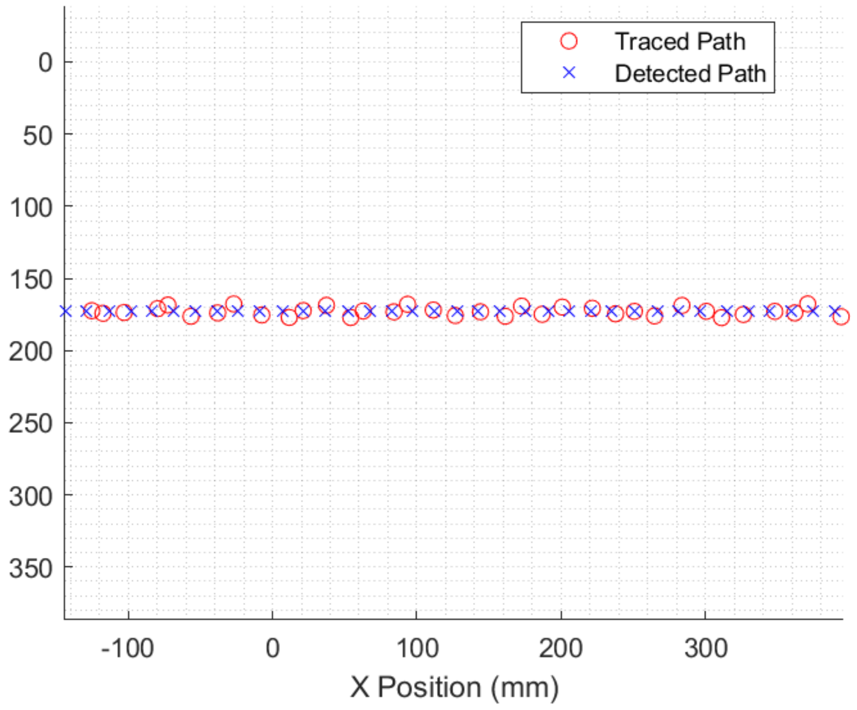


Figure 3.23: Detected points of an aerospace inspired curved workpiece

an Intel Realsense and Mover6 Arm are more significant than in other works – such as Santos et al. [80]. Whilst their techniques for 'camera only' visual servoing are similar, their discrepancy is lower when tracing an edge, likely due to the higher precision of instruments used such as the Kuka iiwa and Cognex 7402C.

3.5 Chapter Summary

In conclusion, the implementation provided a lightweight solution for identifying and tracing workpiece edges regardless of size, spatial profile or previous knowledge. Experiments were conducted on a variety of edges to test feature extraction and the accuracy of the output data. The experiments showed that the technique provided satisfactory results, which, while they do not beat current industrial or literature standards when tracing acute edges, are performed on a much more lightweight system and can trace a broader range of spatially varying workpieces, specifically pieces without acute, sharply defined edges. Experiments demonstrated the robustness of the developed algorithm for tracing an aerospace structure by adding high levels of artificial noise to the image frame, in an attempt to mimic and exceed the noise present in the real world. Additional experiments were completed with a physical robot and camera, to verify the simulated experiments, and show the overall goal of the algorithm. Furthermore, the technique provided can be used in multiple cells, in quick secession, without any need for a specific setup, normalisation, or pre-planning. Additional experiments demonstrated the key environmental factors that impact edge quality and how the edge detection of the workpiece can be optimised. This allowed the creation of an optimal environment for comparing cameras for edge tracing, which showed that the majority of high quality cameras are capable of detecting workpiece edges, although leveraging extra data such as 3D vision can be beneficial for deprojection.

4

Evaluation of various distributions of labour algorithms for tracing a partially-known workpiece

4.1 Introduction

To expand on the visual servoing system created in section 3.2.3, a mobile robot platform was introduced, allowing the kinematic arm to interact and trace significantly larger manufacturing workpieces. With the introduction of mobile base, it became necessary to investigate the distribution of labour: how much work would be completed by the kinematic arm, and how much work would be completed by the mobile base. Two fundamentally opposed algorithms were created to investigate the distribution of labour when tracing manufacturing workpiece: an algorithm which prioritised base movement, and an algorithm which prioritised kinematic arm movement.

Each algorithm was tested against a variety of workpieces, monitoring its accuracy and completion speed, to test how each algorithm performed when tracing different workpiece features. It was hypothesised that each algorithm would perform better and worse when tracing different features: the algorithm prioritising base movement would be quicker, but generally less accurate, and the algorithm prioritising arm movement would be slower but more accurate.

4.2 Methods and Materials

4.2.1 Selection of Camera and Robot

The results from section 3.4 showed that using a depth-sensing camera provided a high enough level of accuracy whilst also giving a precise depth measurement suitable for single-camera, eye-in-hand visual servoing. Due to this, an Intel RealSense D435i continued to be the primary camera used, as it offered all the major advantages of an RGB camera, with easily retrievable depth data. An additional benefit of the Intel Realsense was its ability to integrate seamlessly with the ROS system.

The use of the official Intel ROS Wrapper [67] allowed the different camera sensors to be published by ROS as ROS Services. This allowed the data streams from each individual sensor to be easily separated, and the data could be leveraged as needed. The pure image data was separated from the depth data and was interpreted as a standard OpenCV RGB image, while the depth data was interpreted as both a Depth Image and a Point Cloud, and both sets of data could be leveraged separately.

Once a camera had been selected, a robot had to be chosen for the next set of experiments and development cycle. Unlike the initial set of robot experiments, which required the robot to be stationary for experimentation, the follow-up experiments required the robot to move freely to interact with a larger percentage of the workpiece. The robot would need to possess the following properties:

- Movement: Unlike the Commonplace Robotics Mover6 used previously, the robot needed for the second development cycle was required to have a mobile base that could move independently of the kinematic arm. This could be achieved in one of two ways: either the robot would be an AMR with a separately attached kinematic arm, or a fully-integrated mobile manipulator.
- Availability: The robot had to be easily accessible in a physical format and easily simulated via Gazebo. This was important as access to a simulated environment would allow much easier and quicker experimentation due to the lack of logistically using physical workpieces. Having a physical counterpart would also allow verification of any simulated results.
- Compatibility: The robot would have to be compatible with the previously-completed development environment and be ROS compatible. Due to the nature of ROS, this included the majority of robots on the market, outside of a few industrial robotics which primarily use their own priority software (although they can be modified to allow limited ROS integra-

tion [81]).

- Payload: The robot's kinematic arm was required to hold a camera in an eye-in-hand configuration, so a minimum payload weight of 350 grams was required. This was to minimise discrepancies between the kinematic solver moving the arm and any external forces working against the arm when moving.

The initial goal was to simply mobilise the Mover6, by attaching it to an education-focused AMR, such as a Turtlebot Waffle or Pioneer 3D-X, although integrating the two systems together would cause extra complications and take up a large percentage of development time. An investigation was then done on possible mobile manipulators to ascertain if any of those would be suitable. Unlike the initial goal, they wouldn't need to be integrated, and many of these would simply run on Gazebo as intended.

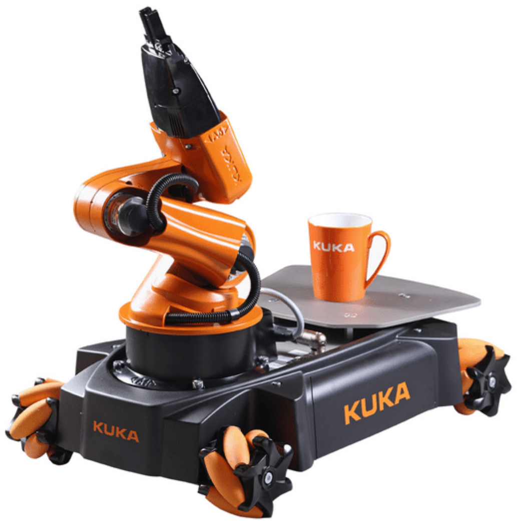


Figure 4.1: A Kuka Youbot [8]

One example of a popular mobile manipulator is the Kuka Youbot [82], created by Kuka (Figure 4.1). It has a maximum payload of 500 grams [9], exceeding what was needed for the camera to be held efficiently. Additionally, this robot was easily accessible physically due to its commonality and was built to be compatible with ROS.

4.2.2 Selection of Software

Since the proposed second wave of development and experimentation directly follows the author's previous work(Section 3), much of the software being used was expected to remain similar. For example, Gazebo was still used as the main tool to simulate the robot, camera and workpiece, and all interactions between the robot and the workpiece. Gazebo, being open-source, allowed a variety of community developers to contribute to the software, ensuring a wide variety of tools were available for it. In this case, it allowed access to a simulated Kuka Youbot [83] and a simulated Intel RealSense D435i [84]. While both of these projects were originally created by the companies' official developers (Kuka and Intel respectively), it was the open-source nature of Gazebo that the wider ROS community took them over and made them compatible with later versions of ROS and Gazebo.

In addition to simulating the robot and camera, a control solution was needed to interpret Cartesian coordinates from the camera, and convert them to joint angles and velocities, creating a Cartesian path for the arm to follow. In previous experiments, this had been completed by manually interacting with external software such as MATLAB, but seeing as these experiments required the robot to be simultaneously moving, and continually tracing multiple positions of the workpiece, a more automated solution was required. MoveIt [85] was designed as a complete package to be used with any ROS-compatible kinematic arm, allowing it to send generic instructions for complex motions. Whilst it is capable of accepting various sensors and intelligently moving around an environment, the main component used for the experiments were the Cartesian controller parts. This allowed the camera to provide MoveIt with a list of feature-detected coordinates, which were then converted into a list of joint angles and velocities, which, when acted on sequentially, would result in a smooth Cartesian path, and the robot tracing the workpiece.

Due to all the different systems running simultaneously, ROS was used once again as middleware, to ensure that all the different components of the system could communicate with each other effectively. This is done through a series of publishers and subscribers, where one component can publish data (such as a camera), and another component can listen for a particular type of data (such as MoveIt). As the rqt graph shows (Figure 4.2), Gazebo (/gazebo) is publishing both joint state data (/joint_states), as well as camera data (/camera) – this is the robot and camera existing and working within the simulation. The camera is then subscribed by MoveIt (/move_group_python_interface. . .), which converts the subscribed coordinates into a list of joint velocities and angles (/move_group). Finally, the move_group publishes these joint velocities and angles in a format which is more generic for ROS to handle, and it's subscribed to by the kinematic arm's controller (/arm_1), which executes the movement at a lower level. This movement is then subscribed to by Gazebo, which displays the movement, and the cycle continues from the beginning.

4.2.3 Algorithm Design

When evaluating the possible distribution of labour when using a mobile manipulator to interact with a large workpiece, there are two main parts which need to be considered: the kinematic arm, and the base. The arm has a limited amount of precise movement by itself, due to its restrictive workspace (Figure 4.3), but when combined with the base, this movement becomes much more flexible, with greater width, but with an assumed loss in accuracy.

This problem can be viewed as a matter of percentage of labour: how much of the labour of the task (tracing a workpiece) can be done by the different components (arm or base), and what is the best possible contribution to receive the best possible outcome – high accuracy with low competition time. To test this, two fundamentally-opposing algorithms were developed. One algorithm would prioritise base movement when tracing workpieces, while the other would prioritise kinematic arm movement. A similar approach to a factorial design

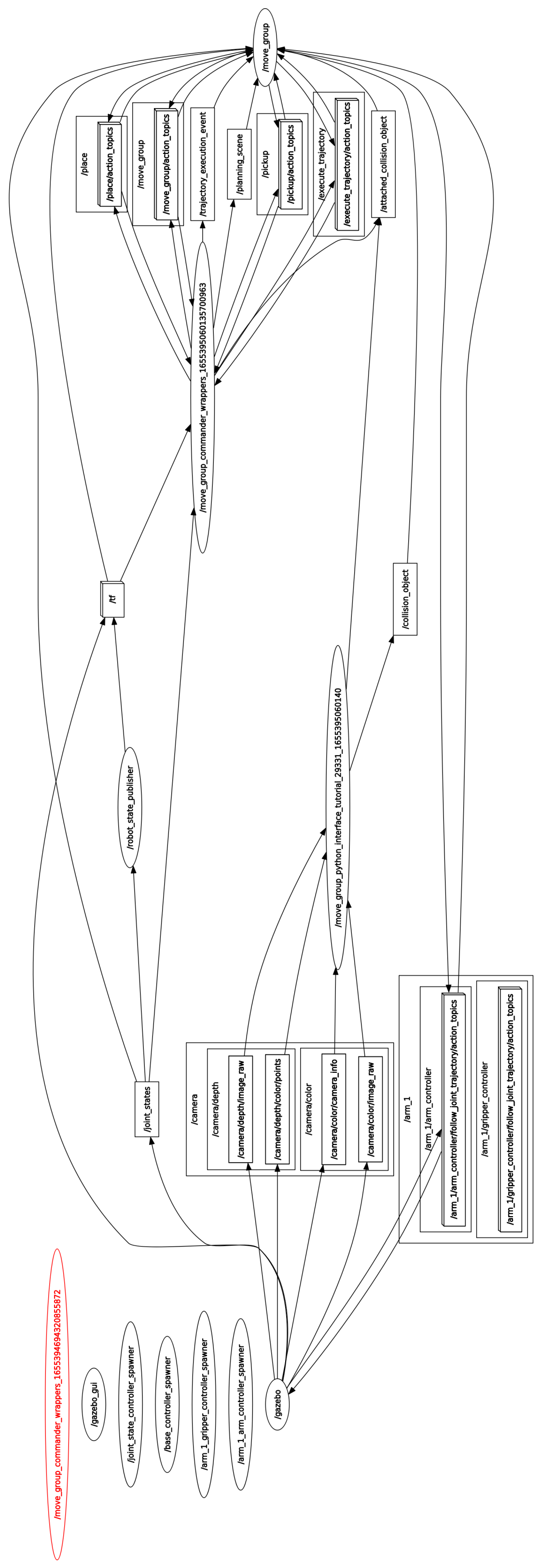

/move_group
/place
/place/action_topics
/move_group
/move_group/action_topics
/trajectory_execution_event
/planning_scene
/pickup
/pickup/action_topics
/execute_trajectory
/execute_trajectory/action_topics
/attached_collision_object
/move_group_commander_wrappers_165539506013570096
/tf
/collision_object
/move_group_python_interface_tutorial_29331_1655395060140
/robot_state_publisher
/joint_states
/camera
/camera/depth
/camera/depth/image_raw
/camera/depth/color/points
/camera/color
/camera/color/camera_info
/camera/color/image_raw
/arm_1
/arm_1/arm_controller
/arm_1/arm_controller/follow_joint_trajectory/action_topics
/arm_1/gripper_controller
/arm_1/gripper_controller/follow_joint_trajectory/action_topics
/move_group_commander_wrappers_165539469432085587
/gazebo_gui
/joint_state_controller_spawner
/base_controller_spawner
/arm_1_gripper_controller_spawner
/arm_1_arm_controller_spawner
/gazebo

Figure 4.2: An rqt graph showing the ROS publishers and subscribers for all the different system components

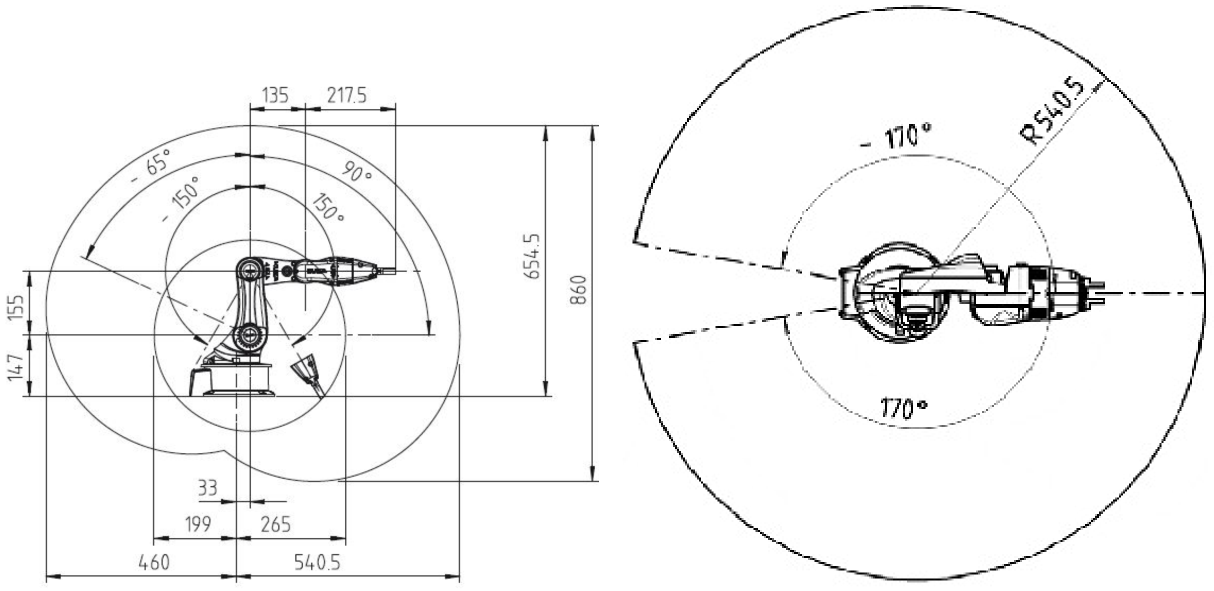


Figure 4.3: The Kuka Youbot Workspace from a side and top view [9]

was taken when designing the algorithms; each algorithm used the maximum values for its variables, ensuring that the algorithms clearly represented that style of movement. By comparing the results of the two, the amount of accuracy lost when actively moving and tracing can be deduced (and the time benefits of this can be examined), and the ensuing results can be used to develop a more intelligent and dynamic system of movement.

Static Base Movement

The first algorithm, referred to as the Static Base Movement Algorithm, prioritised the kinematic arm movements over the base's possible movement, ensuring that the maximum amount of kinematic workspace was used up before repositioning the base. This would ensure that the arm was able to do as much of the movement across the workpiece as possible, generally tracing from the absolute left position of its workspace to the absolute right position (while staying within reach of the workpiece), and then repositioning to repeat this movement.

Figure 4.4 demonstrates how the robot would trace a workpiece while using the Static Base Movement Algorithm.

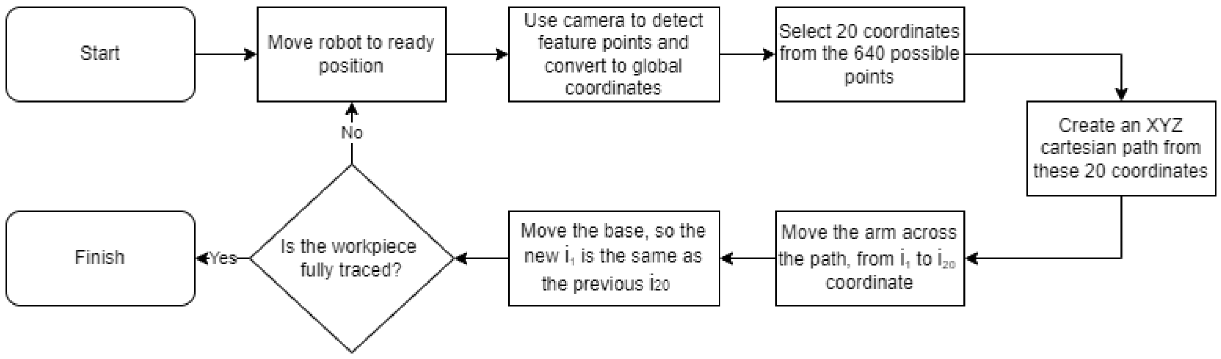

Start
Move robot to ready position
Use camera to detect feature points and convert to global coordinates
Select 20 coordinates from the 640 possible points
Create an XYZ cartesian path from these 20 coordinates
Move the arm across the path, from i1 to i20 coordinate
Move the base, so the new i1 is the same as the previous i20
Is the workpiece fully traced?
No
Yes
Finish

Figure 4.4: A flowchart showing the Static Base Movement Algorithm

a) Initially, the robot gets into its ready position: the base moves to the beginning of the workpiece which needs to be traced, so it is perpendicular to the workpiece, leaving a 25cm (±5cm) gap between itself and the workpiece. The base was positioned sufficiently far enough away from the beginning of the workpiece so there was no wasted movement, while also ensuring that the beginning of the workpiece was traced. The kinematic arm was also rotated 270 degrees, so it was facing the workpiece, and the camera clearly has the desired edge in frame (Figure 4.5).

b) The Intel RealSense used the technique described in section 3.3.2 to detect the required feature of the workpiece, obtaining a list of pixels, and deprojecting them into global coordinates. Unlike the previous chapter, there is no external software needed for converting the pixels into global robot coordinates, as this is now done via a combination of different sensors within Python.

c) Out of the initial list of 640 global coordinates (one for each x pixel), 1 in every 32 is added to a new list, ready to be converted to a Cartesian path, resulting in a list of 20 XYZ global points. These points are given to MoveIt, which is able to create a Cartesian path (a list of joint velocities and angles for the arm to execute), which will move the end-effector to each of the desired points. Only one thirty-second of the initial pixels are used, as this aids with the creation of a smoother Cartesian path, with a quicker calculation time, due to having fewer original points.

d) The kinematic arm now executes the Cartesian plan, moving from position i_1 to i_{20}, in effect tracing the workpiece. Once i_{20} has been reached, the arm moves back to its initial position, and the base begins to move. The base moves the amount of distance needed to make the previous position of i_{20}, the new position of i_1.

e) Finally, the algorithm checks if there is any remaining workpiece which hasn't been traced, by checking the image frame for any additional edge or features past the initial 30% of the frame. If there is no remaining workpiece, then the algorithm declares the workpiece is finished and closes (otherwise the workflow repeats until the whole workpiece has been traced).

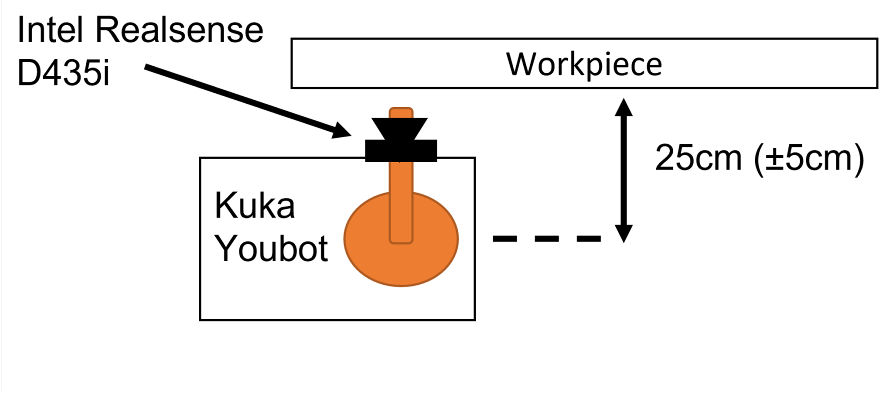


Figure 4.5: The Kuka Youbot's initial position when tracing a workpiece

The resulting algorithm worked similarly to a standard static manipulator, with the mobile base only moving when needed. It was predicted that this would result in a slow algorithm, but with a high level of accuracy due to the lack of interference from the base. As seen in section 3.4, it was also predicted that the accuracy of the algorithm might be bottlenecked due to some of the equipment used, specifically the camera. Since the camera is the only form of sensor being used, any errors resulting from its RGB or depth sensor would be compounded when running these algorithms.

Moving Base Movement

The second algorithm, referred to as the Moving Base Movement Algorithm, does the opposite of the Static Base Movement Algorithm: it prioritises base movement over the kinematic arm movements. Specifically, it uses the base to move across the workpiece while using the arm to control the depth and height. This ensures that the base does the majority of the work, as it is controlling the majority of the movement, while the arm makes adjustments while moving.

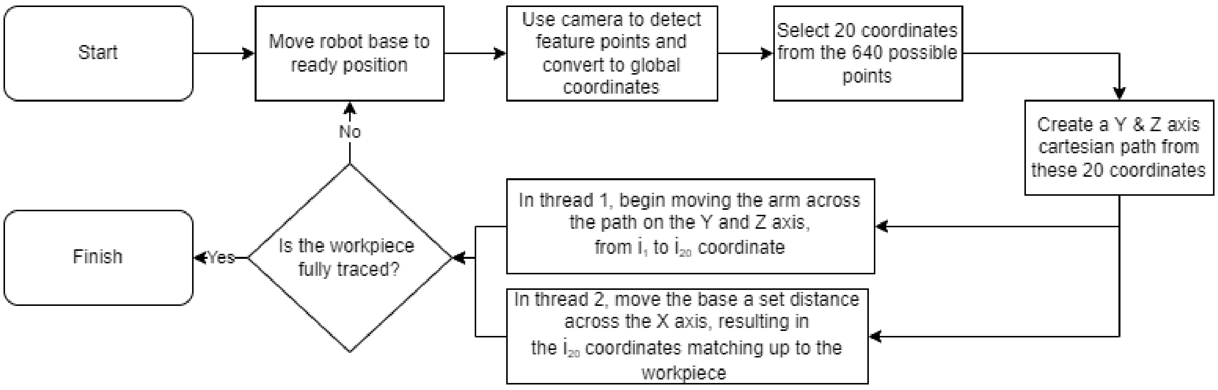

Start
Move robot base to ready position
Use camera to detect feature points and convert to global coordinates
Select 20 coordinates from the 640 possible points
Create a Y & Z axis cartesian path from these 20 coordinates
In thread 1, begin moving the arm across the path on the Y and Z axis, from i_1 to i_{20} coordinate
In thread 2, move the base a set distance across the X axis, resulting in the i_{20} coordinates matching up to the workpiece
Is the workpiece fully traced?
No
Yes
Finish

Figure 4.6: A flowchart showing the Moving Base Movement Algorithm

Figure 4.6 demonstrates how the Moving Base Movement Algorithm is completed on a robot, and how it differs from the Static Base Movement Algorithm.

a) The robot firstly moves into position, similar to the position seen in Figure 4.5. It ensures it is parallel to the workpiece, 25cm ($\pm$5cm) away from the workpiece, and that the arm is in the same position as shown (rotated 270 degrees).

b) Similar to the Static Base Movement Algorithm, the Intel RealSense camera gets up to 640-pixel points from the detected workpiece feature and uses Python to convert them into global coordinates.

c) One in every 32 of the global coordinates is taken and added to a new list, and then converted into a Cartesian path via MoveIt. Unlike the Static Base Movement Algorithm, where each point within the Cartesian path contains information on X, Y and Z information, only the

Y and Z information was recorded into the Cartesian path. This was done to ensure that the arm would only approach the workpiece and move vertically, not horizontally (as this is done via the base movement).

d) Two Python threads are now created.

   a) In thread one, the arm begins executing its Cartesian path, moving towards the workpiece, and horizontally up and down as needed, from position i_1 to i_{20}.

   b) In thread two, the base of the robot moves a set distance, matching up with the movement completed by the arm. In principle, this results in the base being responsible for the X movement when tracing the workpiece.

e) Both these threads are activated, with thread two activating one second later than thread one, to allow time for the arm to reach the workpiece. This results in the arm moving towards the workpiece, and, as it starts to move horizontally, the base simultaneously moves vertically, synchronising the movements, resulting in the arm tracing the workpiece in the combined X, Y and Z directions.

f) Similar to the previous algorithm, the camera now checks if there is any more workpiece to be traced. If there is, then the workflow is repeated, otherwise, the algorithm stops running.

While this algorithm shares a lot of similarities with the Moving Base Movement Algorithm, especially the detection workflow at the beginning, it was predicted that this algorithm would offer opposing results. While the static algorithm was predicted to have a high level of accuracy, but low overall completion time, this algorithm was predicted to have a substantially quicker completion time, but its accuracy would suffer due to the inclusion of the base movement while tracing the workpiece. Additionally, any errors compounded by the camera would result in a worse outcome due to the additional accuracy of movement.

4.3 Experimental Setup

4.3.1 Simulated Algorithm Experimentation

After creating the two opposing algorithms, simulated experiments were planned to test the algorithms' performance on various workpieces. The goal was to test the hypothesis that a moving algorithm would perform much quicker than a static algorithm, but that the performance would be less accurate. Specifically, it was hypothesised that this lower accuracy would be worsened when tracing a workpiece with either an angled feature or features which changed in angle rapidly.

The robot was placed at a set distance from the workpiece, and the algorithm was run. It would then repeat the workflow until the whole workpiece had been traced. In many cases, this required at least three iterations, but it varied per algorithm, per workpiece. Three different workpieces were modelled and imported into the simulator, ranging in complexity, to give a diverse set of results. The most simplistic workpiece was a 100cm by 40cm rectangle, which was used as a baseline for the algorithms. Due to the simplistic nature of this workpiece, it would show if there were any fundamental issues with the algorithms (Figure 4.7). The second workpiece was a 100cm long, angled rectangle, measuring 386mm at its lowest end, and 421mm at its highest end. This workpiece was mainly used to test the angled capabilities of the algorithms, and to understand how they would perform when tracing a workpiece with an inclined feature (Figure 4.8). The final workpiece used was an aerostructure-inspired workpiece, measuring 118cm long, and ranging in height from 384mm to 406mm. This was the most important workpiece results-wise, as it represented a realistic workpiece the robot may have to trace in an industrial setting and had a variety of features (Figure 4.9).

Once the kinematic arm made contact with the workpiece, a Transform (TF) service was called [86] which simultaneously started a timer and began tracking the end-effector's position. This positional data was given every 1 second after

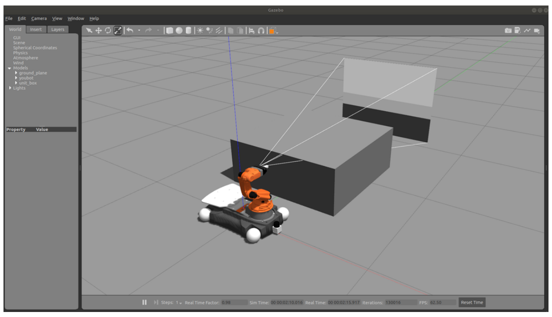


Figure 4.7: The rectangular workpiece in the simulator

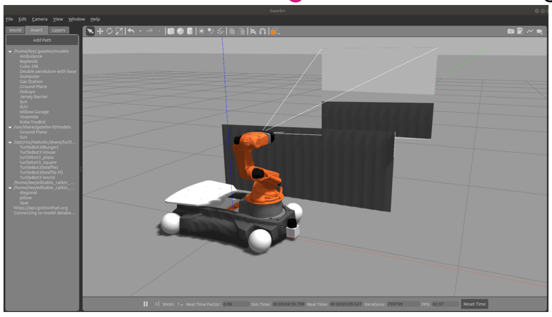


Figure 4.8: The angled workpiece in the simulator

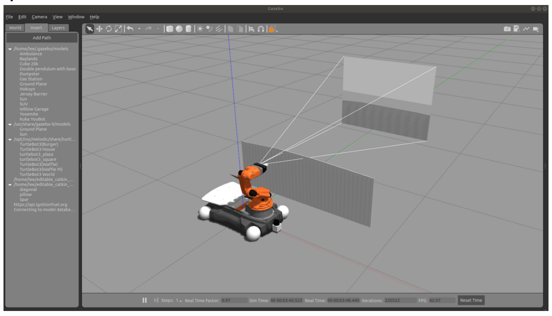


Figure 4.9: The aerostructure inspired workpiece in the simulator

the service was called, allowing for consistent tracking of the arm. The position of the end-effector was tracked in reference to the base of the robot, while the robot's overall global position was tracked by the simulation software. These two pieces of data were then summed together, providing the overall global position of the kinematic arm each second during tracing. This overall position could then be recorded and compared to the ground truth of the workpiece's absolute position, giving information on how accurate each algorithm was.

Table 4.1: A table of experiments for the simulated experiments

	Static Base Algorithm	**Moving Base Algorithm**
Rectangular Workpiece	30 Trials	30 Trials
Angled Workpiece	30 Trials	30 Trials
Aerostructure inspired Workpiece	30 Trials	30 Trials

Each algorithm completed 30 trials on each workpiece, to ensure that the results gathered were accurate and consistent (Table 4.1). For each trial, the workpiece and robot started in the same place, and all variables were kept the same. The only variables which changed were the ones measured, such as; where the arm moved to when it was tracing the workpiece, and where the base moved to when needed.

4.3.2 Physical Algorithm Experimentation

A limited set of physical experiments were also completed in a lab setting, in an attempt to verify a large number of simulated experiments. An identical robot and identical workpiece were used, along with the same camera and setup (Figure 4.10). The robot was then tasked with tracing the workpiece, and the results were recorded and compared against the known position of the workpiece. The robot's movements (base and kinematic arm) were tracked via a Leica Absolute Tracker ATS600 (Figure 4.11), giving an accurate global Cartesian position each second.

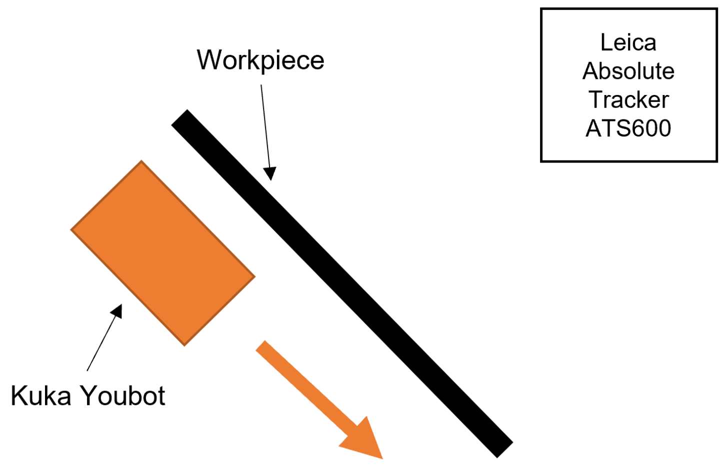


Figure 4.10: A diagram of the physical experimental setup

Figure 4.11: The Leica Absolute Tracker ATS600

The identical rectangular workpiece measured 100cm by 40cm and was created out of fibreboard which was painted in white gloss (Figure 4.12).

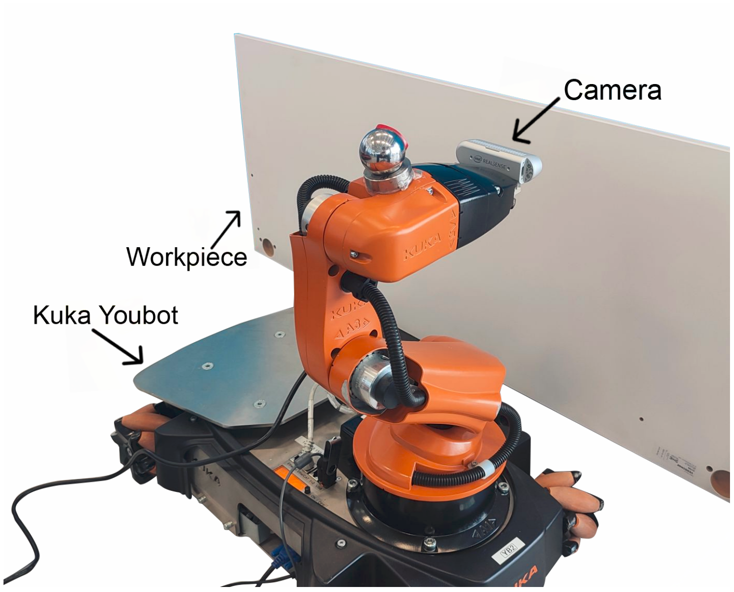


Figure 4.12: The rectangular workpiece during physical experiments

Similar to the simulated experiments, both the time and accuracy of the robot were measured through the use of the Leica tracking software. The results of these experiments didn't need to be conclusive by themselves, their primary purpose being to provide similar and equivalent results to the simulation's average result, as a form of verification.

By investigating the results of the physical experiments, and comparing them to the simulated experiments, an understanding was gathered of how accurate the simulator had been in a realistic situation, and if the simulation experiments could be verified. Due to this, only three trials were completed, compared to the 30 completed in simulation. This was mostly due to time constraints, as physical trials take substantially longer and cost more than a simulated trial.

4.4 Results and Discussion

When tracing the workpieces (both simulated and physical), two main variables were monitored and recorded: end-effector position and completion time. When running the experiments in simulation, they were recorded by calling a TF ROS service, which would provide the position of the end-effector in reference to the base of the robot every second. When running experiments physically, the accuracy and time required to trace the workpieces were recorded via the Leica Absolute Tracker, giving positional data, alongside time stamps for each data point.

It was predicted that, in all cases, the Moving Base Algorithm would be quicker to trace a workpiece but would generally be less accurate. Specifically, as the workpiece gets more complex, and requires more synchronised movements from the base and the arm, the Moving Base Algorithm would become increasingly less accurate, but still remain faster.

The goal of these experiments was to gather data on the uses of each algorithm in different situations and discover when it is best to use either technique. This data could then be leveraged to create a more dynamic algorithm which could trace a workpiece at its most optimal speed and accuracy.

4.4.1 Simulated Algorithm Experimentation

Accuracy Results

When analysing the results from the simulation experiments, it can immediately be seen that the accuracy of the static algorithm is consistently high across all workpieces on the X & Y axes (Figures 4.13, 4.14 and 4.15), clustering within 1cm or less of the workpiece's ground truth. Once the base starts to move, there is some discrepancy on where the robot starts, best demonstrated in Figure 4.13. This is due to the lack of a dedicated sensor on the robotic base and it is therefore determining the distance moved via pure odometry readings. This could potentially be improved through the addition of a sensor, or by reducing the amount of movement undertaken. This would result in the robot retracing some of the workpieces and thereby adding to the overall completion time, but it would also ensure that the entire workpiece is fully traced.

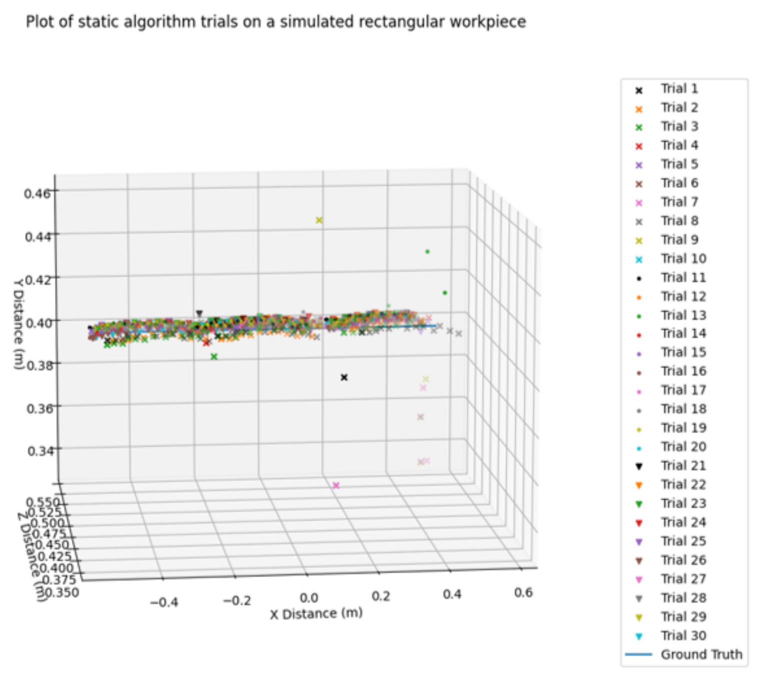


Figure 4.13: A graph showing the accuracy of the Static Base Algorithm when tracing a rectangular workpiece in simulation

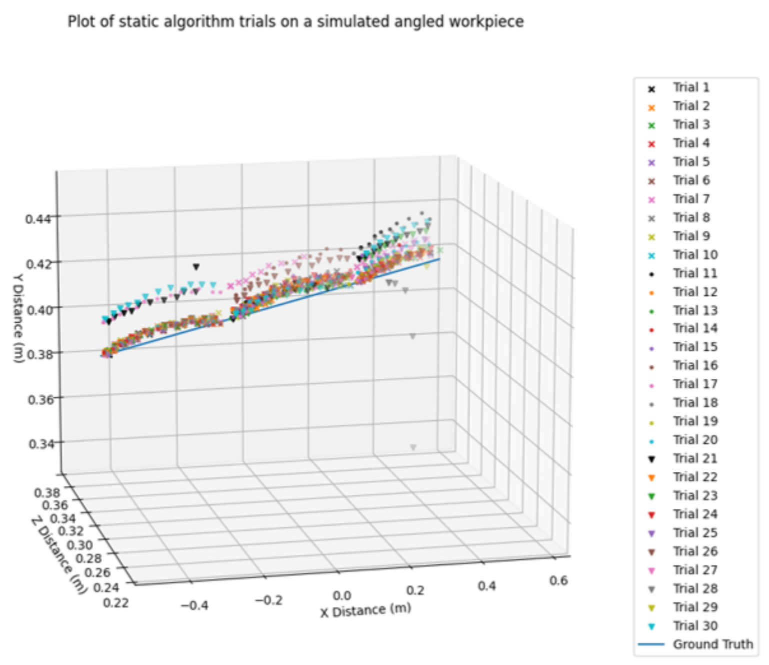


Figure 4.14: A graph showing the accuracy of the Static Base Algorithm when tracing an angled workpiece in simulation

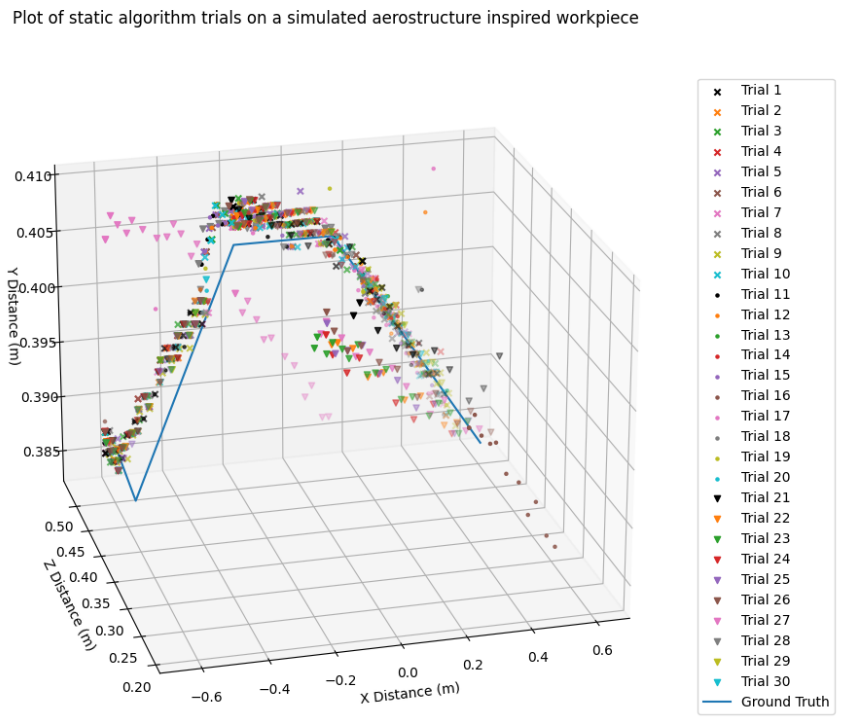


Figure 4.15: A graph showing the accuracy of the Static Base Algorithm when tracing an aerostructure-inspired workpiece in simulation

There is also a noticeable discrepancy in accuracy when the robot is tasked to trace a feature at an angle for a consistent amount of time, such as at the beginning of Figure 4.14 and Figure 4.15. It can clearly be seen that the robot does not trace the workpiece at the same consistent angle, and eventually falls off. This may be a restriction of the camera, or the angle at which the camera is held in the robot's hand, due to its limited rotational capabilities.

Compared to the Static Base Algorithm, the Moving Base Algorithm proved to be generally less accurate across all workpieces. When tracing the rectangular workpiece (Figure 4.16), the accuracy was comparable to the Static Base Algorithm at times, although the consistency across all 30 trials shows that the accuracy worsened overall. While many trials had a comparable accuracy of 1cm, outliers showed an accuracy of 10cm or more, showing that larger variations occur when in motion compared to being static. The accuracy was worsened in the Z-axis when compared to the X and Y axes, implying that the accuracy could be increased with better offset tuning.

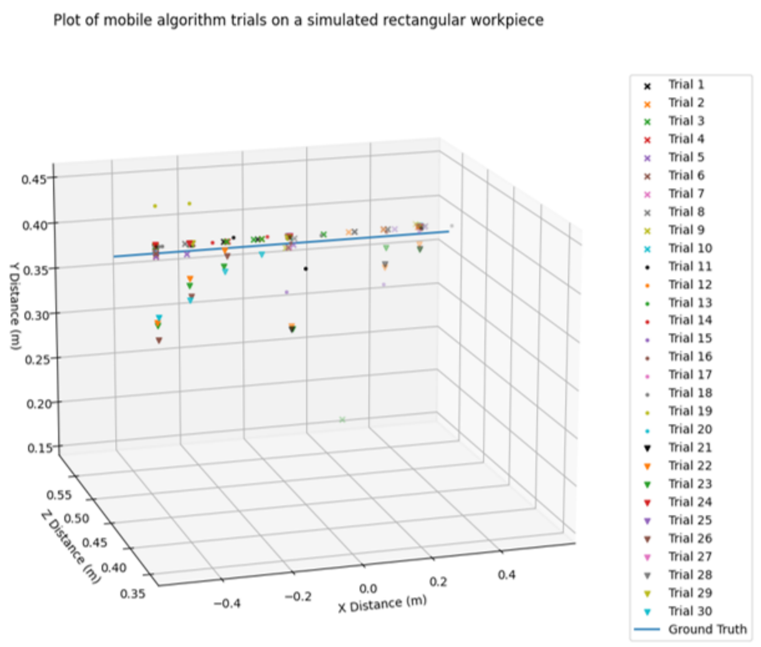


Figure 4.16: A graph showing the accuracy of the Moving Base Algorithm when tracing a rectangular workpiece in simulation

The results from the angled workpiece (Figure 4.17) were much the same, with some results having an offset of 1cm, although generally the results deviated much further. Another important observation with the angled workpiece is how the points clustered away from the ground truth, such as towards the end of the tracing task, where clustering could be observed below the line. This is likely due to the way the arm only moves on the Y and Z axes, and therefore may not reach the intended position before the base starts moving.

The aerostructure-inspired workpiece (Figure 4.18) had the biggest consistency issues when being traced by the Moving Base Algorithm, showing almost no clustering when tracing. While the quantitative accuracy of the workpiece was almost equivalent to the Static Base Algorithm, investigating the graph qualitatively shows that the accuracy was substantially worse, as the tracing points by themselves show very little consistency to the ground truth.

Figure 4.19 shows a nearest-neighbour analysis of the rectangular workpiece, showing the closest traced points to each point of the ground truth. The nearest-

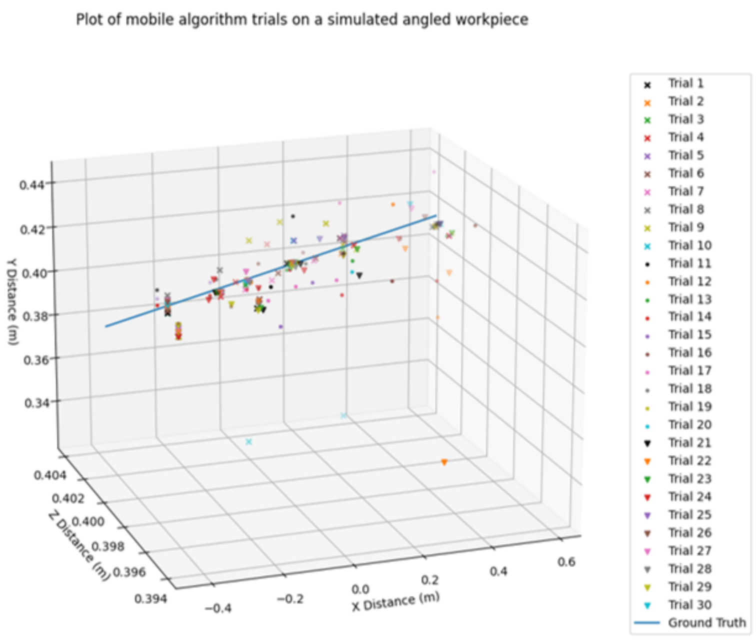


Figure 4.17: A graph showing the accuracy of the Moving Base Algorithm when tracing an angled workpiece in simulation

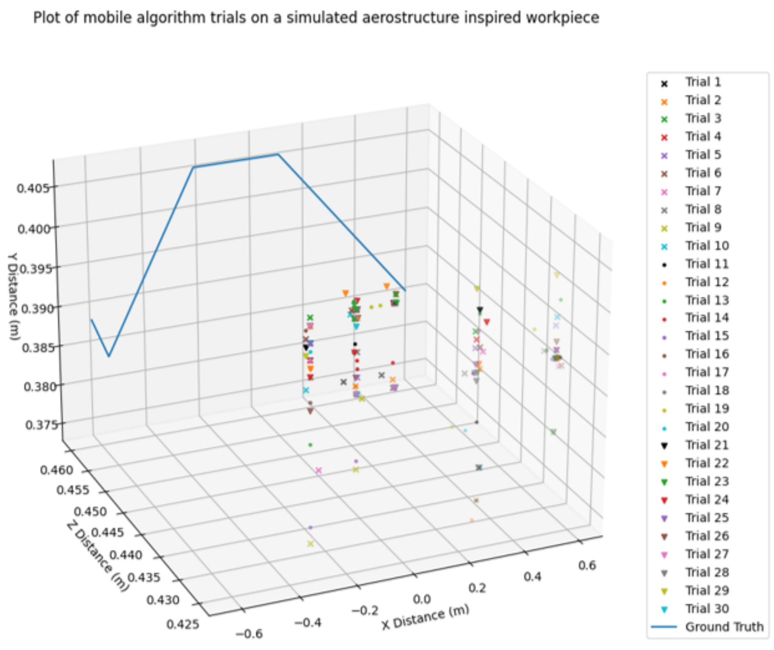


Figure 4.18: A graph showing the accuracy of the Moving Base Algorithm when tracing an aerostructure-inspired workpiece in simulation

neighbour analysis only considers the X and Y axes, as it wasn't practical to compare the algorithms via the Z-axis due to varying ground truth Z positions. After completing all 90 trials (30 with each algorithm) it can be seen that the static algorithm performed much better, accuracy-wise, with only two nearest neighbours being created via the moving algorithm. A similar effect is seen in Figure 4.20, where the majority of the nearest neighbours of the angled workpiece are also from the static algorithm compared to the moving algorithm. It can be hypothesised that this is due to the consistency of the workpiece features, since there is no dynamic change in the workpiece. While the accuracy of the static algorithm is higher generally, there are still points during the tracing experiments where the mobile workpiece was equally as accurate and wasn't rendered ineffective due to quickly-changing workpiece features.

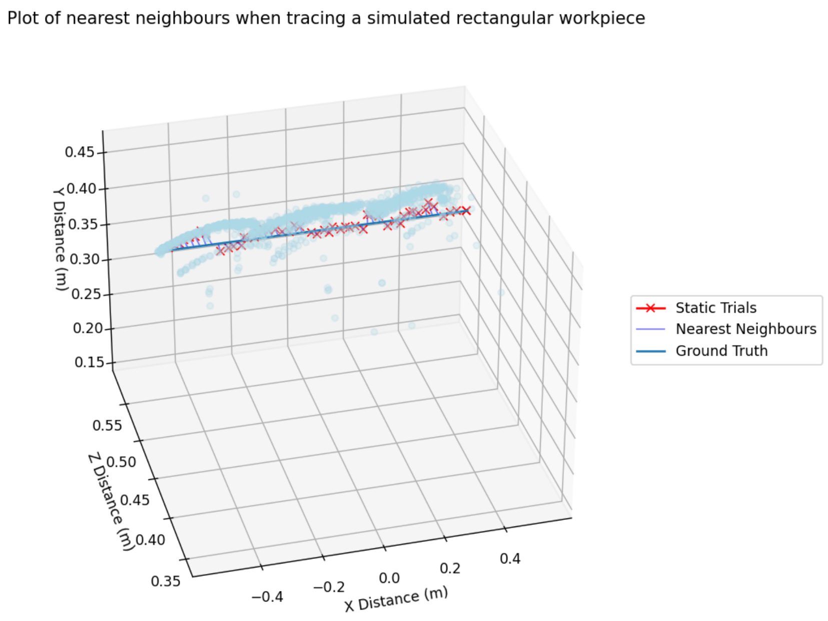


Figure 4.19: A graph showing the nearest neighbours of both algorithms when tracing the rectangular workpiece in simulation

Figure 4.21 proves this hypothesis, as the nearest-neighbour analysis of the aerostructure-inspired workpiece has no nearest neighbours from the Moving Base Algorithm experiments – unlike the previous two workpieces, which had a majority of Static Base nearest neighbours and one or two Moving Base Algorithm nearest neighbours. This is likely due to the differences in feature consist-

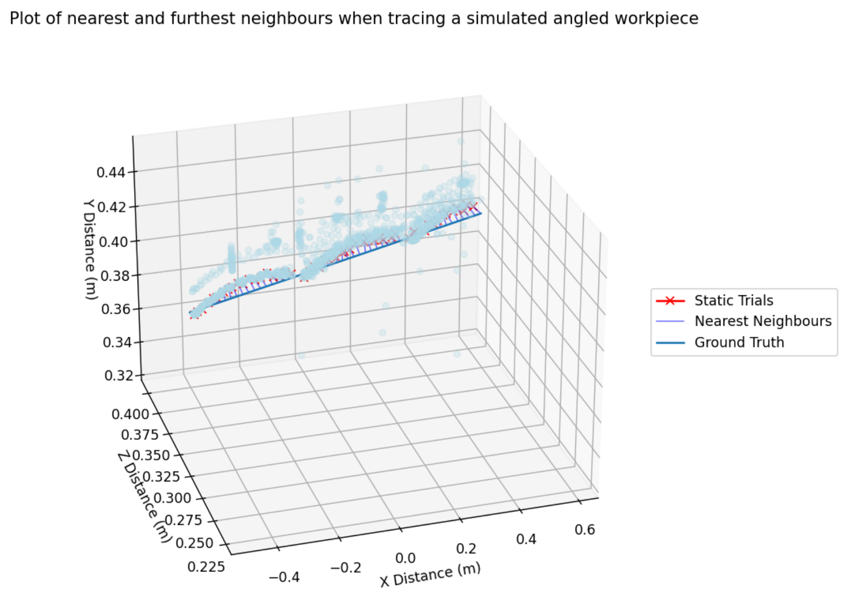


Figure 4.20: A graph showing the nearest neighbours of both algorithms when tracing the angled workpiece in simulation

ency: how consistent and long a feature stays present in a workpiece. Looking at the rectangular and angled workpieces, they both possess one dominant workpiece feature: either a straight edge or an angled edge. Compared to the aerostructure-inspired workpiece, which has a combination of 4 features interconnected, a clear distinction can be seen.

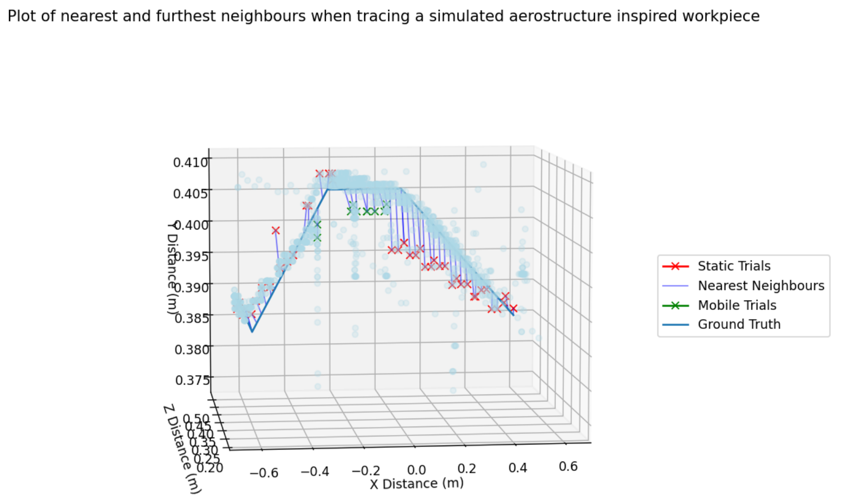


Figure 4.21: A graph showing the nearest neighbours of both algorithms when tracing the aerostructure-inspired workpiece in simulation

It is this distinction which made the Moving Base Algorithm that much worse when tracing the aerostructure-inspired workpiece in comparison to the Static Base Algorithm. If the Moving Base Algorithm begins tracing the workpiece halfway through a feature (such as at -0.4 in Figure 4.21), it had to trace that feature and then adjust to tracing a new feature. While, inherently, this isn't a difficult task, the Static Base Algorithm does it well. It is the added complexity of synchronicity between the base and arm movements which makes this much more inaccurate.

Time Results

The other variable which needed to be compared was completion time – how long did it take for an algorithm to completely trace a workpiece. The recorded time only took contact time into account: it started timing when the agent made contact with the workpiece and stopped the timer when it stopped. This was done to remove any possible computational variables from influencing the results, ensuring that the data gathered would be valid. It was consistently much quicker tracing the workpiece using the Moving Base Algorithm compared to the Static Base Algorithm – generally being 10 times quicker (Figures 4.22, 4.23 and 4.24). This was expected, and was due to how the moving algorithm didn't have to stay static while tracing and could therefore generally trace the workpiece much faster.

There was a wider time variance when using the Static Base Algorithm, having a range of almost 20 seconds in the case of Figure 4.24. This was due to the variance in movement when the base moved from position to position while doing a Static Base trial, resulting in the robot needing to complete another workflow cycle. On average, the robot needed to complete three cycles, but if the robot consistently underestimated the preset 30cm movement throughout a trial, it would result in being required to do another workflow cycle, thereby extending the time of the trial.

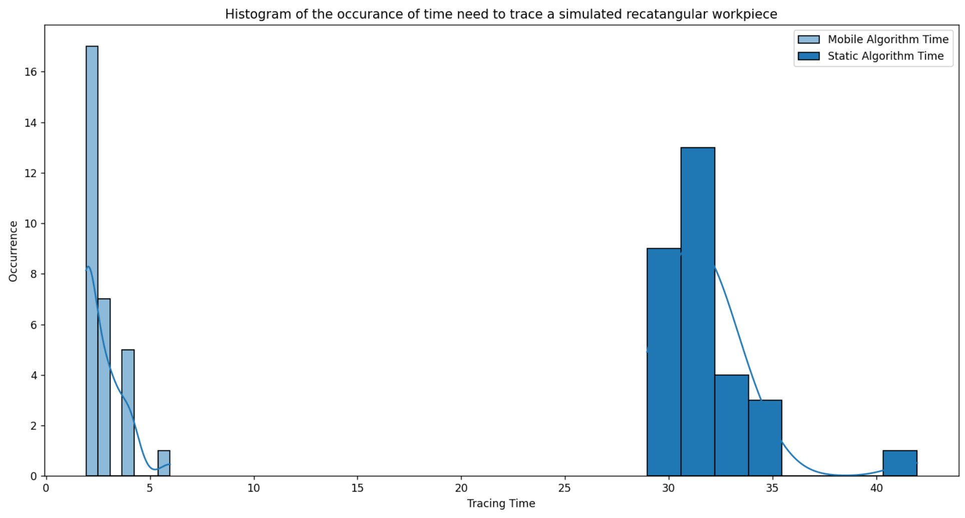


Figure 4.22: A histogram showing the times needed to trace the rectangular workpiece with both algorithms in simulation

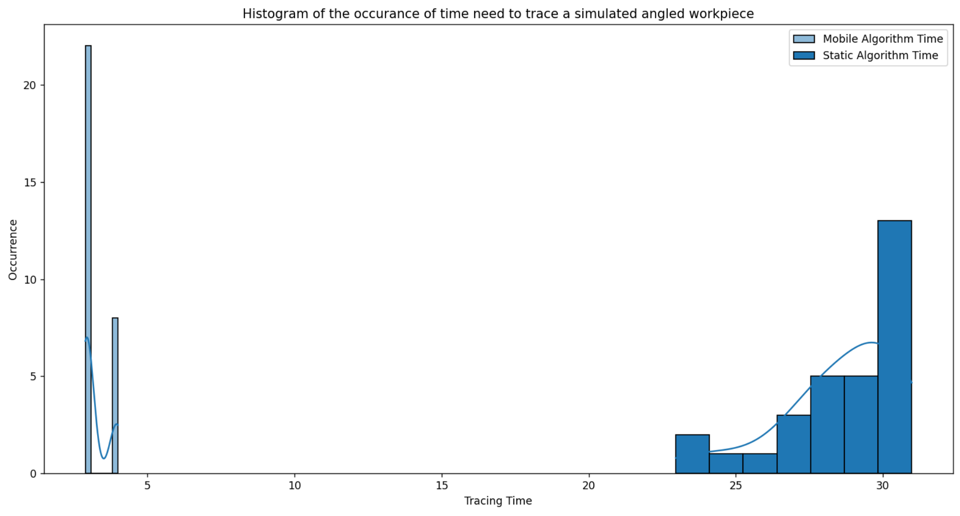


Figure 4.23: A histogram showing the times needed to trace the angled workpiece with both algorithms in simulation

Accuracy over Time Results

One final comparison was completed, which investigated trials on overall accuracy compared to completion time. This was done by resampling the workpiece's ground truth to match the sample size of the trial [87] and then using SciKit Learn [88] to get the Means Square Error of the trial compared to the ground truth. This was then compared against the recorded completion time. This gave an indication of how accuracy is reflected in completion time and

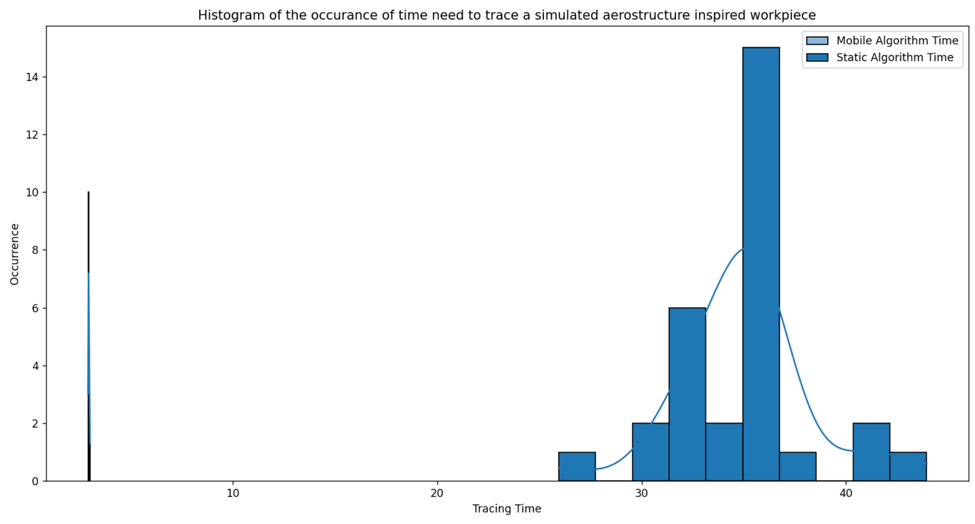


Figure 4.24: A histogram showing the times needed to trace the aerostructure-inspired workpiece with both algorithms in simulation

would allow data to be gathered regarding the initial hypothesis: that the moving algorithm would be quicker but with relatively lower accuracy compared to the static algorithm.

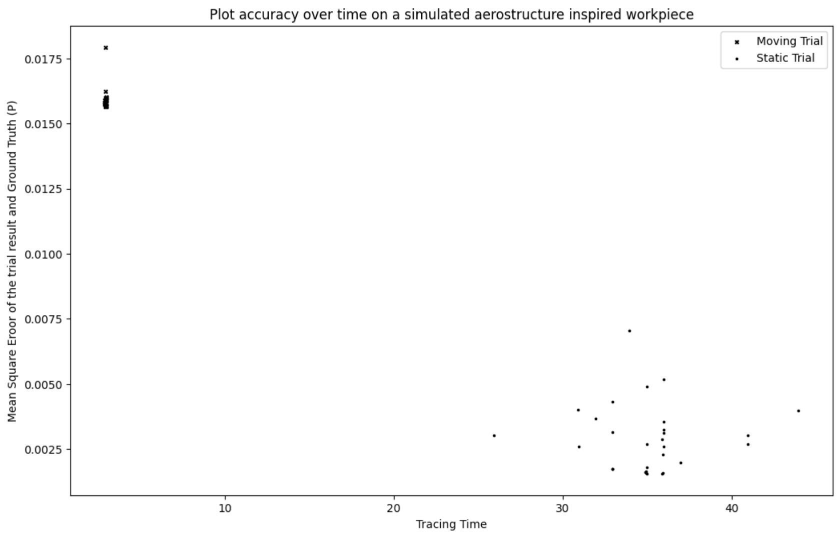


Figure 4.25: A scatter graph showing accuracy over time for both algorithms when tracing a rectangular workpiece in simulation

When tracing the rectangular workpiece, the time difference between both algorithms is large, the Static Base Algorithm being a magnitude slower with only a small decrease in P (Figure 4.25). Therefore, it is proposed that when a workpiece with a straight feature needs to be traced, using a constantly moving algorithm will save a lot of time while only sacrificing a minimal amount of accuracy. This accuracy could further be improved by tuning the offset more finely in the Moving Base Algorithm since any discrepancy in the camera-to-end-effector offset gets amplified due to uncertainty in the wheel movements.

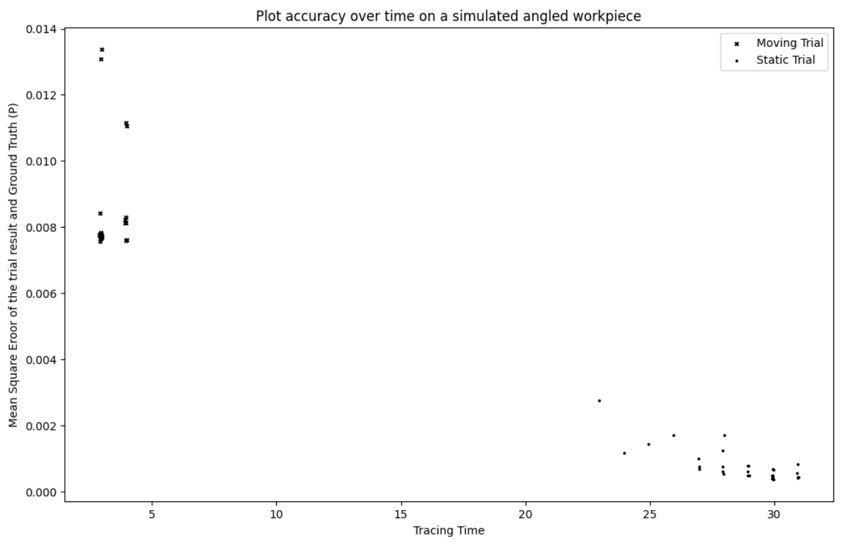


Figure 4.26: A scatter graph showing accuracy over time for both algorithms when tracing an angled workpiece in simulation

Compared to the rectangular workpiece, the angled workpiece (Figure 4.26) had a similar difference in accuracy between the two algorithms and a slightly shorter difference in time. The Static Base Algorithm performed the same as when tracing the rectangular workpiece, significantly slower, but the Moving Base Algorithm's accuracy is more consistent when tracing the angled workpiece – as seen by the clustering, although it is less accurate in some trials. This greater gap in accuracy, when compared to the rectangular workpiece, is likely due to the angled feature of the workpiece. Due to the complex nature of synchronising the base movement and the arm movement together for tracing a workpiece feature, any mistakes made in synchronisation will result in a large decrease in accuracy. While this matters less when tracing a workpiece with a straight feature, tracing a workpiece such as the angled workpiece, which required a change in verticality by the arm, decreased accuracy.

Due to the possibility of desynchronisation (and therefore a loss of accuracy), it is proposed that when the robot needs to trace a workpiece whose primary feature has an angle or requires vertical changes by the arm, the Static Base Algorithm is used. While time is saved by using the Moving Base Algorithm, the potential loss in accuracy means the algorithm may not always produce satisfactory results.

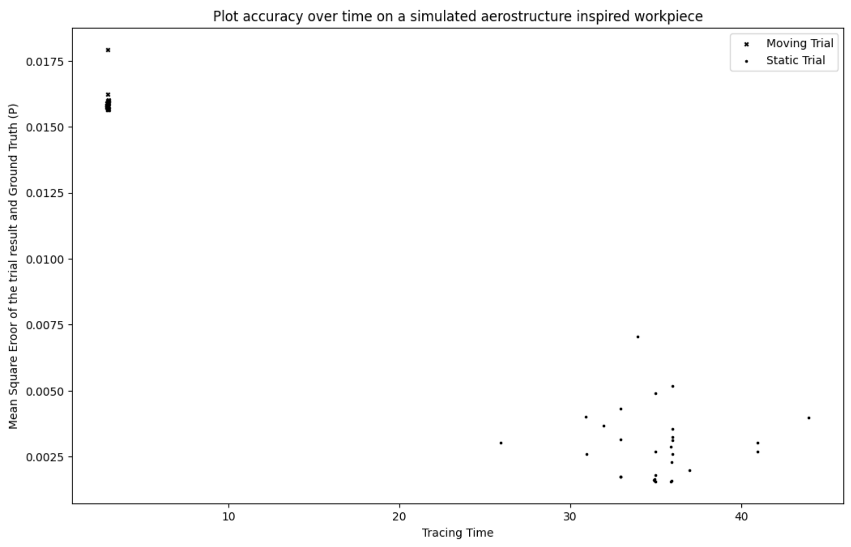


Figure 4.27: A scatter graph showing accuracy over time for both algorithms when tracing an aerostructure-inspired workpiece in simulation

Compared to the more simplistic rectangular and angled workpieces when tracing the aerostructure-inspired workpiece (Figure 4.27), both algorithms see a substantial decrease in accuracy, with the Static Base Algorithm being as inaccurate as previous Moving Base Algorithms, and the Moving Base Algorithm itself becoming substantially less accurate than any previous workpiece. The difference in time between the two workpieces stays consistent with the results from the previous workpieces, with the Moving Base Algorithm being substantially quicker.

Unlike the previous workpieces, the level of disparity of accuracy between the two algorithms is huge, showing a lack of accuracy when using a Moving Base Algorithm. This can be qualitatively seen in Figure 4.18, where the recorded robot positions have little to no immediate correlation with the workpiece's ground truth. Therefore, it can be concluded that, without a major overhaul, the use of the Moving Base Algorithm isn't viable for workpieces with a variety of rapidly changing shorter features – such as the aerostructure-inspired workpiece.

4.4.2 Physical Algorithm Experimentation

An additional set of physical experiments was conducted to verify the large number of simulated experiments completed. This was done by using two identical workpieces and placing them into similar situations, as seen in simulation, to see how the results compared.

Accuracy Results

When tracing a rectangular, physical workpiece using the Static Base Algorithm (Figure 4.28), there is a clear similarity in the pattern shown. The path made in the XY axes mimics the one seen in previous simulated experiments, showing that the arm can trace a simple physical workpiece. When the Z axis is also investigated, it can be seen that the traced path 'bows' at the beginning and end of each cycle. This is likely due to the robot getting ready to interact with the workpiece in each workflow and reaching towards and away from the workpiece. While this 'bowing' is something which wasn't present in the simulated experiments, it is worth noting that this only adds 3cm to the Z offset and could, in future, be eliminated by more dedicated tuning of the algorithm to the robot.

Similar to the Static Base Algorithm, a clearly-defined relationship between the simulated and physical trials can be seen when using the Moving Base Algorithm (Figure 4.29), as a clear movement pattern can be seen. The general motions

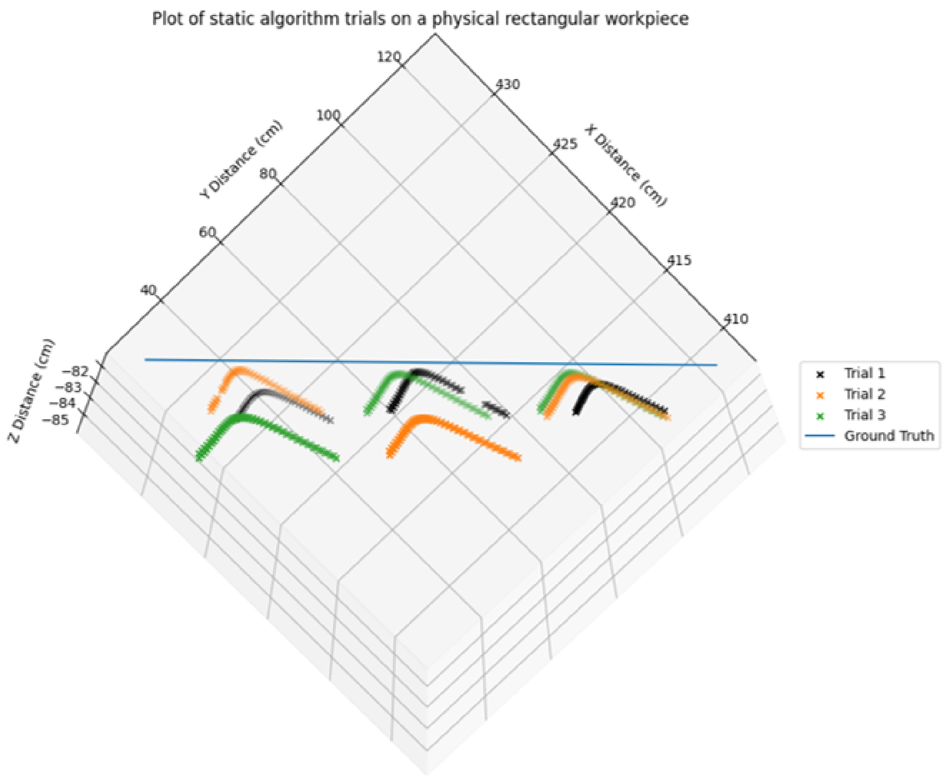


Figure 4.28: A graph showing the accuracy of the Static Base Algorithm when tracing a physical, rectangular workpiece

made by the arm correlate to the ground truth, although there is a larger offset in all Cartesian directions. All movements made by the arm when tracing the rectangular workpiece were within 3cm of the intended feature, which either matched or exceeded the expectations set by the simulated trials. Similar to the static trials, much of the offset between the traced path and the workpiece is in the Z axis, something which could be tuned and reduced through additional trials.

A nearest-neighbour analysis showed that when tracing a physical, rectangular workpiece (Figure 4.30) there is no dominant algorithm which is constantly closer to the ground truth between the two algorithms, however, the Moving Base Algorithm is closer to the ground truth the majority of the time. This is a departure from what was seen in the simulated experiments, where the opposite was true – the Static Base Algorithm performed better. This change in nearest neighbour is likely due to the 'bowing' effect seen in the physical trials, which

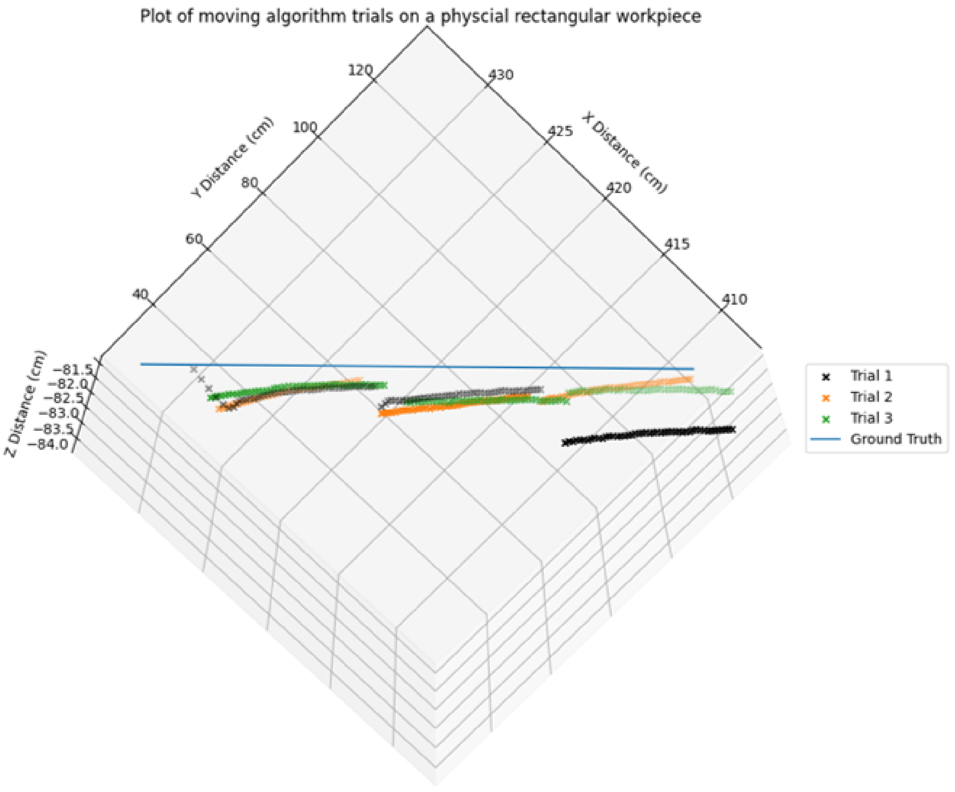


Figure 4.29: A graph showing the accuracy of the Moving Base Algorithm when tracing a physical, rectangular workpiece

means that many recorded points during a static trial would have a larger-than-expected Z value.

Time Results

In addition to the accuracy, the physical experiment completion times for each workpiece and algorithm were also recorded and compared to investigate the inherent differences, and, the differences compared to the simulated experiments (Figures 4.31). There was a notable difference in time when running the Moving Base Algorithm compared to the Static Base Algorithm. This echoed what was seen in the simulated results, with the Static Base Algorithm taking substantially longer to trace a workpiece, although both algorithms took roughly three times as long to complete a physical experiment as they did a simulated experiment. This was due to the differing acceleration times of the arm, as, in

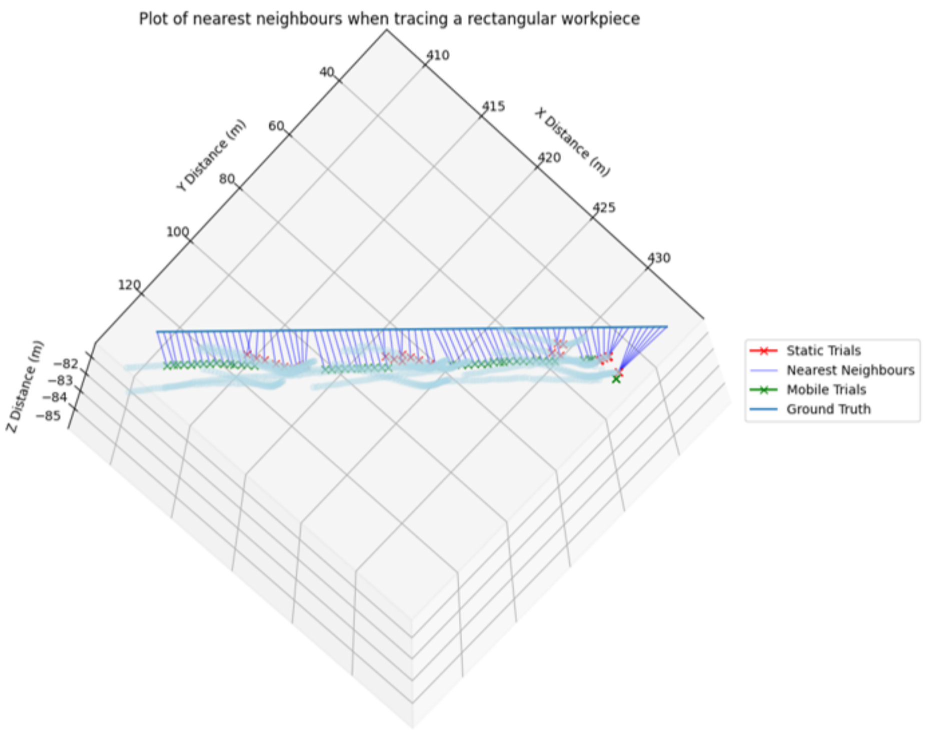


Figure 4.30: A graph showing the nearest neighbours of both algorithms when tracing the physical, rectangular workpiece

simulation, the arm could run at the maximum velocity while the physical Kuka Youbot had to be limited for safety reasons.

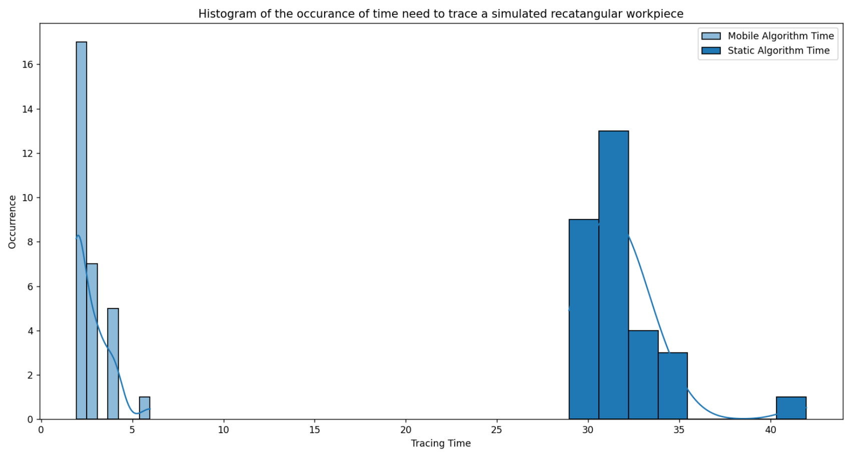


Figure 4.31: A histogram showing the times needed to trace the physical, rectangular workpiece with both algorithms

Accuracy over Time Results

The recorded times of the physical experiments could then be directly compared to a means square analysis of the algorithm's accuracy, to gain an understanding of how the physical accuracy of the arm was reflected in the extended completion times (compared to the results in section 4.4.1). An immediate look at Figure 4.32 show that both algorithms were slower and less accurate than the simulation experiments. As discussed earlier, the extended time it took to trace objects was a direct result of the arm not moving as quickly due to safety precautions, and the arm's inaccuracy was due to a variety of physical interactions that the simulation didn't account for. Outside of this inherent disassociation between the physical and simulated experiments, there is a clear corelation in the patterns they produced; the Moving Base Algorithm was quicker and less accurate, and the Static Base Algorithm was slower but more accurate.

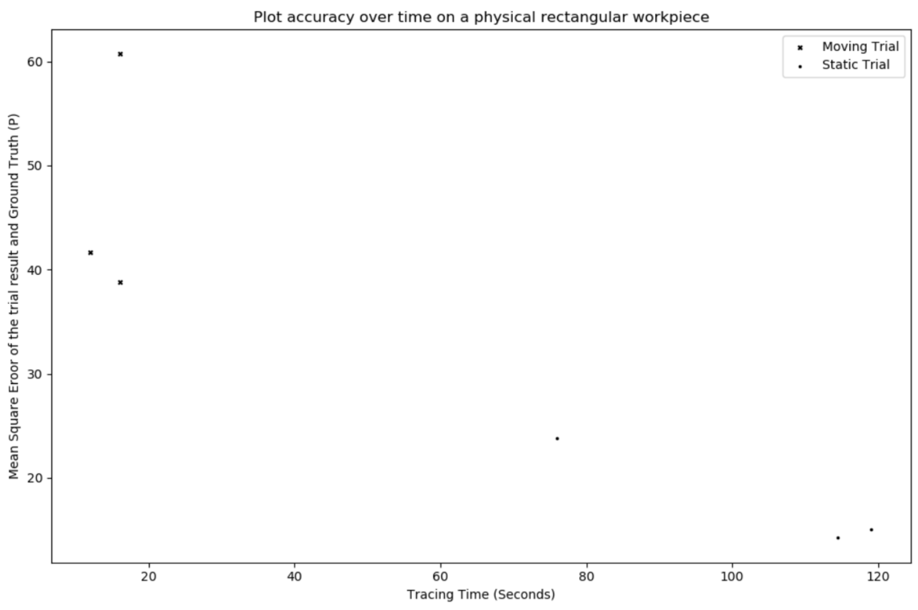


Figure 4.32: A scatter graph showing accuracy over time for both algorithms when tracing a physical, rectangular workpiece

By visually comparing the results from Figure 4.25 to Figure 4.32, it can be concluded that the simulated results can be verified by the physical experimentations, but not quantitatively. The data reported by each set of experiments follow a similar pattern, although the data itself isn't identical, and the conclusions drawn from the simulated experiments can also be drawn independently from the physical experiments.

4.5 Chapter Summary

In conclusion, when tracing a manufacturing workpiece, the use of distribution of labour algorithms can have a large effect on both the accuracy of the robot when tracing the workpiece, but also the completion time. The major factor in determining the results were the features of the workpiece, specifically the incline. When tracing a workpiece with little to no inclined features, such as the rectangular workpiece, the moving base algorithm worked best, having a similar accuracy to the static base algorithm, but a much increased time. The opposite can be said for the angled workpiece, where the static base algorithm was optimal due to have a substantially higher accuracy compared to moving base algorithm, even though it was slower. The aerostructure inspired workpiece showed similar results, with this static base algorithm outperforming the moving base algorithm accuracy wise, but not time wise. There was a clear corelation between the amount of inclined features of a workpiece, and the better performing algorithm, with the moving base algorithms performing better when tracing a feature with little to no incline, and the static base algorithm performing better when tracing a feature with an incline.

5

Development of a dynamic distribution-of-work algorithm to effectively and optimally complete the tracing of industrial workpieces

5.1 Introduction

After experimenting with both the static- and moving-base algorithms, an alternative, middle algorithm was developed. This algorithm, the dynamic-base algorithm, was created to help bridge the gap between the two previous algorithms, and apply both of their strengths accordingly when tracing a workpiece. It was able to detect different features of a workpiece using its camera, and then by analysing what it saw, dynamically change the way it would trace the workpiece according to what it saw. The result would be an algorithm that could trace a workpiece optimally, remaining accurate while reducing the tracing time of previous algorithms, by always switching to the most appropriate movement style.

The dynamic algorithm design was targeted specifically at the aerospace industry, and therefore, the aerostructure-inspired workpiece was the primary testing piece used. This was due to its versatility and variety of workpiece features. By rigorously testing the algorithm against the variety of features seen on such a complex workpiece, it was proven that such an algorithm could be more optimal than either of the extremes tested previously in Section 4.4.

5.1.1 Algorithm Design

After comparing the two previous algorithms against various workpieces, the results showed that each algorithm was suited to different detected features. The moving-base algorithm performed consistently quicker with a minimal drop in accuracy when tracing a straight-line feature, while the static-base algorithm was most consistent on all types of features but was substantially slower in tracing the feature. Therefore, a new algorithm was created which would use both systems automatically, and would dynamically adapt to the type of feature it was detecting.

The dynamic-base algorithm workflow started the same as all previous al-

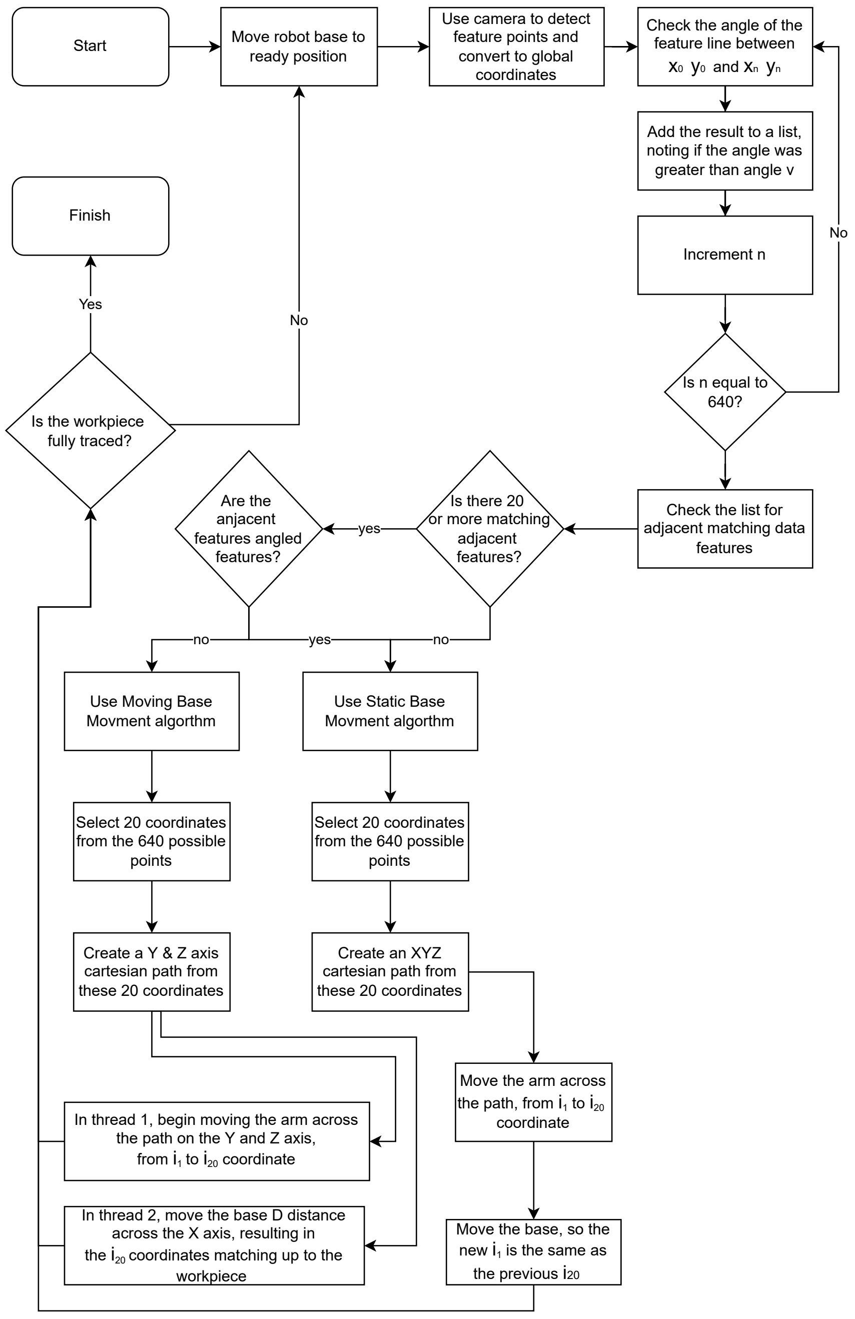

Start
Move robot base to ready position
Use camera to detect feature points and convert to global coordinates
Check the angle of the feature line between x0 y0 and xn yn
Add the result to a list, noting if the angle was greater than angle v
Increment n
Is n equal to 640?
No
Check the list for adjacent matching data features
Is there 20 or more matching adjacent features?
yes
Are the anjacent features angled features?
no
yes
no
Use Moving Base Movment algorthm
Use Static Base Movment algorthm
Select 20 coordinates from the 640 possible points
Select 20 coordinates from the 640 possible points
Create a Y & Z axis cartesian path from these 20 coordinates
Create an XYZ cartesian path from these 20 coordinates
In thread 1, begin moving the arm across the path on the Y and Z axis, from i1 to i20 coordinate
In thread 2, move the base D distance across the X axis, resulting in the i20 coordinates matching up to the workpiece
Move the arm across the path, from i1 to i20 coordinate
Move the base, so the new i1 is the same as the previous i20
Is the workpiece fully traced?
Yes
No
Finish

Figure 5.1: A flowchart of the dynamic-base algorithm workflow

gorithms, by moving the robot into its default position (Figure 5.1).

a) Firstly, the robot moves into a position parallel to the workpiece, and its arm moves into a starting position so that the camera can clearly see the feature which needs to be detected.

b) The camera detects all points of the desired feature and converts all the pixel coordinates to global coordinates, much in the same way as the previous algorithms. Unlike previous algorithms, this is done to allow the algorithm to analyse the feature and determine the best way to trace it.

c) The algorithm now checks the angle of a line between two points, X_0 Y_0 and X_n Y_n. The angle is checked to see if it exceeds a predetermined angle V, and a corresponding Boolean result is added to a list. Once completed, n was incremented until it reached 640, the maximum width of the camera. The outcome was a list of 640 values analysing the angle and steepness of the feature.

d) The list is then analysed for adjacent pairs and matching data, starting from the beginning. This is done to look for consistent features across the workpiece, such as a consistent straight line or a constant angle.

e) The algorithm will specifically look for a set of 20 or more matching adjacent Booleans on the list, and depending on what it finds, determines which algorithm it will use.

    a) If the algorithm cannot find at least 20 pairs of adjoining Booleans, the algorithm will use the standard static-base algorithm. This is due to consistency, since previous experiments have shown that the static-base algorithm is constantly accurate, if not optimal, due to the extended time it takes to trace a workpiece.

    b) If the algorithm finds at least 20 pairs of adjoining Booleans, it then does another check to read the Boolean values.

        a) If the adjoining Booleans all show that the feature is angled, then the algorithm will choose for the robot to use a static-base movement style, as experiments have shown it is best at dealing with angled workpieces.

b) If the adjoining Booleans all show that the feature is angled, then the algorithm will use a moving-base style

f) If the algorithm chose to use the static-movement style, then it will choose 20 points from the detected feature of 640 points and create and execute an XYZ Cartesian path. It will then move a predetermined distance equal to X.

g) If the algorithm chose to use the moving-base algorithm, then it will choose 20 points from the detected feature of 640 points and create an XZ Cartesian path. The algorithm then creates two threads, thread one and thread two, and executes them simultaneously. In thread one, the Cartesian path is executed, resulting in the arm moving vertically and towards the workpiece. In thread two, the robot base moved a modified distance D (Equation 5.1). The movement distance D is determined by getting the percentage of the detected feature F inside the current image frame C, and multiplying it by the default movement value of X. In essence, this gave a percentage of the predetermined movement value, which correlates to the percentage of the feature shown in frame.

h) Once the algorithm has finished, there is a final check to see if there is more workpiece to trace or not. If there is, then the workflow repeats, otherwise the algorithm terminates.

$$D = (X/100) * ((F/C) * 100) \tag{5.1}$$

While this algorithm shares similarities with the previously-tested static- and moving-base algorithms, various improvements should aid the dynamic-base algorithms' performance. The possibility of dynamically switching between the two movement styles while in process, as well as the dynamic movement offered when using a moving-base style, should elevate its performance against what was previously tested. It was hypothesized that the resulting algorithm would fall in between the previous two algorithms regarding both accuracy and time

5.2 Experimental Setup

5.2.1 Algorithm Tuning Experimentation

The dynamic-base algorithm was able to dynamically change its tracing style depending on what style of the feature was detected by the camera. For it to optimally decide which algorithm to use and to use them effectively, two predetermined variables had to be assigned, tuned and set: Predetermined Distance x, and Predetermined Angle v.

Predetermined Distance x was a variable which dictated what the default base movement would be – how far would the base move after or during a tracing cycle before modification. For example, it could be set to 40, which would result in an unmodified movement of 40cm. The initial distance value had to be manually set to ensure that the base always had a known distance it could move, else it would effectively never stop moving.

Predetermined Angle v was the value used to determine if a detected feature was straight or not. For example, if v were set to 1, then values between 180° and 179°, -180° and -179° and 1°, 0° and -1° would all be classed as a straight line. This variable was used to allow for inaccuracies when detecting the outlines of features and to minimize the number of false positives or negatives when tracing a feature.

Since both variables have to be set before runtime, a limited set of experiments had to be completed to tune the variables to ensure that they were set correctly. Three settings were chosen for each variable (Table 5.1), with a reasonable minimum, maximum and median setting. These settings were used to ensure that each trial could still provide a valid result.

Table 5.1: A table showing the variables to be tested when tuning the dynamic base algorithm

Angle v	Movement x
0.5°	40cm
1°	45cm
2°	50cm

The settings used for distance x were related to the dimensions and workspace of the Kuka Youbot, using both pieces of information to provide three possible initial distances. The minimum was set to ensure some level of overlap when tracing the workpiece from position to position, guaranteeing the whole workpiece gets traced, while the maximum was set to ensure the robot never retraces parts of the workpiece.

Each trial was run three times, with the robot attempting to trace the previously-used aerostructure-inspired workpiece (as seen in Section 4.4.1). This was done to allow a comparison between the results obtained to determine which set of variables provided the best results. The aerostructure-inspired workpiece was the one used as it was the most realistic of all workpieces, and provided a range of features for the robot to detect and trace.

While running the experiments, both the time and the robot's end-effector positions were recorded, so the most accurate and fastest version of the dynamic algorithm could be determined. Both of these variables were recorded in a similar way to the experiments conducted in section 4.4.1, using TF as a way to get the robot's end-effector local positions and time each second, and transforming the coordinates of the end-effector into global coordinates.

It was hypothesized that out of all the available options, a median approach to angle v would be best, as that would catch all straight lines while avoiding as many false positives as possible. It was important to minimise false positives as much as possible to ensure that the moving base portion of the algorithm could be used, which would directly reduce the workpiece-tracing competition time. Additionally, it was hypothesized that a maximum approach to distance x would be optimal for both accuracy and time, as it would minimise retracing of the workpiece and ensure that a 1-metre workpiece would be traced in 3 workflow cycles or fewer.

5.2.2 Algorithm Experimentation

Once optimal settings for both angle v and distance d were chosen, testing of the dynamic algorithm could be completed. These experiments were completed in simulation, allowing for a wider variety to be performed in a similar style to previous algorithm experiments, as seen in Section 4.4.1. Three separate workpieces were used:

- A 100cm by 40cm rectangular workpiece, which was used as a baseline for the algorithm
- A 100cm-long angled rectangle, measuring 386mm at its lowest end, and 421mm at its highest end
- An aerostructure-inspired workpiece, measuring 118cm long, and ranging in height from 384mm to 406mm.

The workpiece and robot were set up identically to the moving algorithm trials completed in section 4.4, to allow a direct comparison between the dynamic algorithm and the previously-tested static- and moving-base algorithms. The robot was positioned parallel to the workpiece, with a varying distance between the two depending on the workpiece being used. The algorithm was tested 30 times per workpiece to ensure that any anomalies were caught, and a thorough understanding of the algorithm's performance against different workpieces was recorded.

Alongside the setup being similar, the same data were also recorded: the robots end-effector position, and tracing time. Similar to both sections 4.4.1 and 4.4.2, TF was used to record local end-effector positions and times, which could then be used to compare the robot's accuracy against a known ground truth.

Due to the similarities between the experiments completed in section 4.4, a comparison can be drawn between the results of the dynamic algorithm experiments to previous experiments. This comparison is important because it allows a direct comparison to previous results, allowing a qualitative way to review the effectiveness of the dynamic algorithm compared to the previous algorithms.

It was hypothesized that the dynamic algorithm would perform better than the previous experiments conducted in section 4.4.1. Specifically, it was hypothesized that the dynamic algorithm would complete tracing tasks quicker than the static algorithm while being more accurate than the moving-base algorithm. The dynamic algorithm would also ideally be more consistent than the static or moving algorithm, providing an optimal and robust algorithm for use when tracing large-scale industrial workpieces.

5.3 Results and Discussion

5.3.1 Algorithm Tuning Experimentation

To tune the dynamic algorithm, experiments were performed while modifying the two major variables which dictate the algorithm's performance: Angle V and Distance D. These variables were altered in small increments, and trials were then run using the aerostructure-inspired workpiece (Figure 5.2). For the most part, the majority of trials had similar results – clearly mimicking the ground truth and at times having a sub 1cm discrepancy compared to the ground truth. Some variable sets, such as the 45cm D, 1° V and 50cm D, 0.5° V trials, showed that the trials lacked consistency, with some results being significantly different to the others.

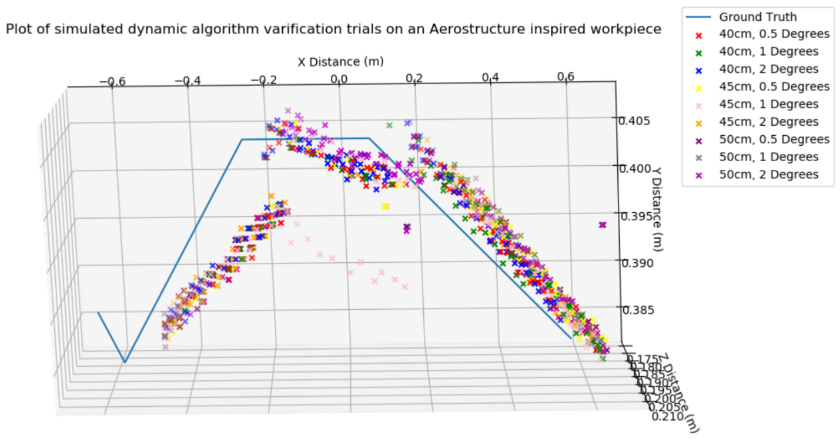


Figure 5.2: A scatter chart showing how each tuning experiment traced the aerostructure-inspired workpiece.

When analysing the majority of the experimental results shown in Figure 5.2, the most inaccurate, but still consistent, results have a discrepancy of 2cm at most, which is significantly lower than the moving-base algorithm results seen earlier in section 4.4.1 and on par with the static-base algorithm results seen in section 4.4.1, giving a preliminary look at the effectiveness of the dynamic algorithm when compared to previous trials.

Analysing the time results in Figure 5.3, a wide variety of times were recorded, ranging from 15 seconds to over 50 seconds. While, ideally, an optimal tuning would result in a quick completion time, consistency is also important. For example, when experimenting with a D of 50cm and V of 2°, it produced one of the quickest completion times but also some of the slowest. Therefore, it is important to find not only the quickest pair of variables but also the quickest and most consistent pair.

A good candidate for a fast, consistent completion time is 50cm 1°, which had the quickest result for one trial running for less than 15 seconds, and two other trials running for 30 seconds. While the variance between 15 and 30 seconds is larger than ideal, both recorded times are quicker than over 50% of all recorded times. Another reasonable candidate would be 50cm 0.5°, which had a similar maximum and minimum spread of 15 and 32 seconds, although the results were

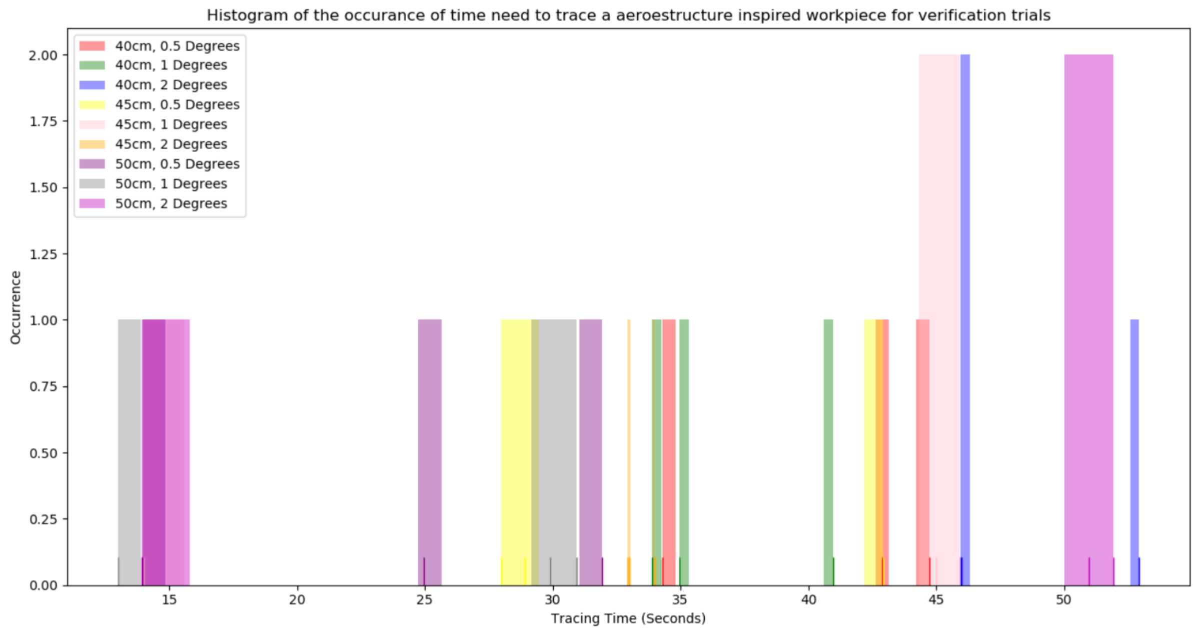


Figure 5.3: A histogram showing the times of each verification experiment.

less clustered and more spread across the range of times.

By combining both the accuracy and time results gathered, an analysis can be performed to investigate the accuracy of the variable combinations with regards to the completion time (Figure 5.4). Most tested variables had equivocal accuracy, having a very low P score, with the exception of 40cm, 1°. Due to the equivocal accuracy, the main variable which needs to be considered is completion time, and constancy of time. By looking at the scatter plot, it can be seen that when the variables are tuned to 50cm, 1°, the aerostructure-inspired workpiece gets traced the quickest while also being consistently accurate. While not all trials traced the workpiece with the same speed, the recorded times for 50cm, 1° were the most consistent.

Therefore, when doing further testing of the dynamic-movement algorithm, and comparing it to previous algorithms, Angle V will be set to 1° and Distance D to 50cm. This should give the best results regarding the algorithm, and allow for an accurate comparison to other algorithms.

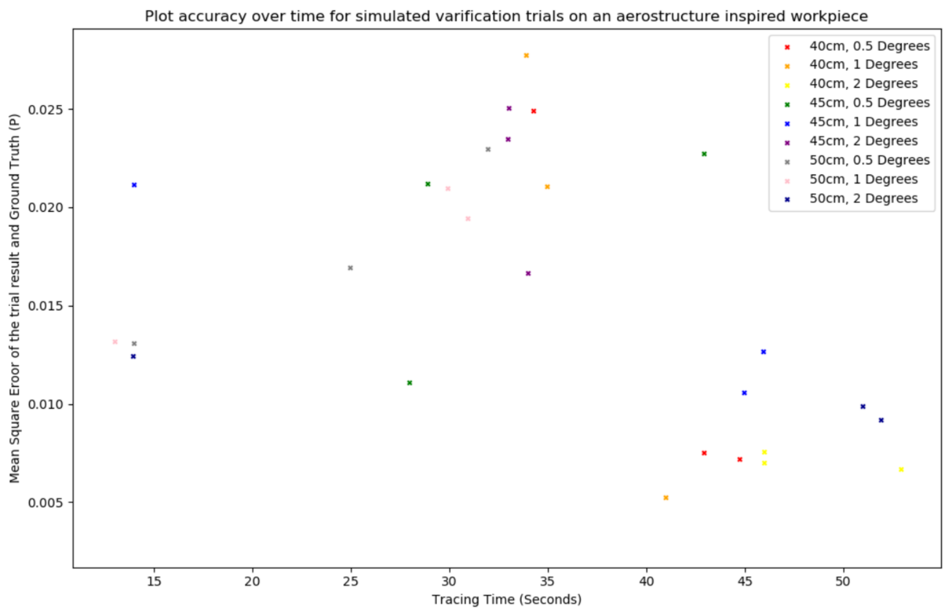


Figure 5.4: A scatter graph showing the accuracy over time for the dynamic algorithm's tuning experiments

5.3.2 Algorithm Experimentation

After tuning the dynamic-moving algorithm, it could then be used to trace a variety of workpieces and to gather an understanding of its consistency and accuracy. 30 trials were completed on each workpiece: the rectangular workpiece (Figure 5.5), the angled workpiece (Figure 5.6), and the aerostructure-inspired workpiece (Figure 5.7). When tracing all workpieces, the traced path completed by the robot was mostly accurate to 2cm or less, and, in many cases, was accurate to less than 1cm. Similar to experiments completed in section 4.4.1, the accuracy seen in the Z axis is less than the accuracy seen in the X and Y axes, due to the inaccuracy of the camera used when attempting to ascertain the depth of the edge of a workpiece.

In addition to the accuracy of the algorithm, by analysing Figures 5.5, 5.6 and 5.7, an understanding of how the dynamic algorithm traced each workpiece can be obtained. Looking at the tracing path of the rectangular workpiece in Figure 5.5 shows that the dynamic algorithm continually used a moving-base style of tracing, due to the lack of angle change, which, as previous experiments

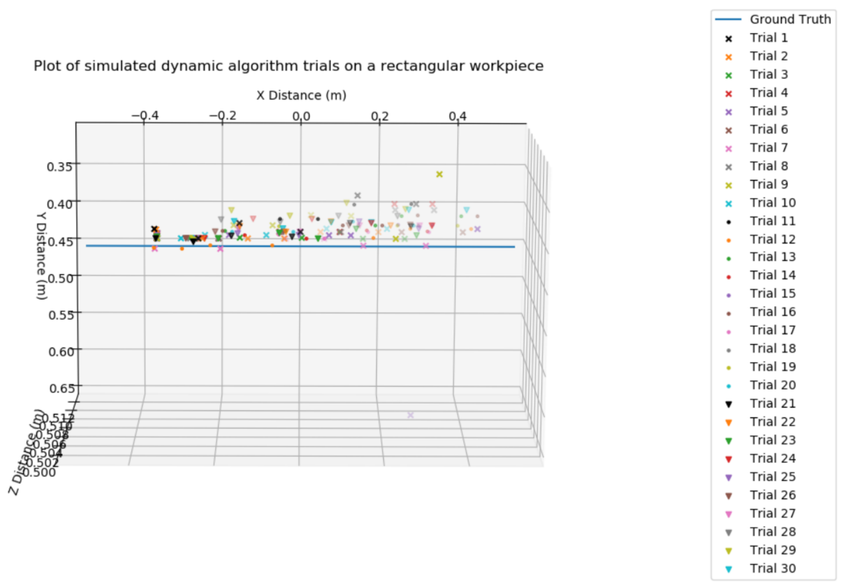


Figure 5.5: A scatter graph showing the traced path of the dynamic algorithm on a rectangular workpiece

in section 4.4.1 showed, is the optimal way to trace such a workpiece. On the contrary, Figure 5.6 shows that the dynamic-base algorithm used a static movement style, due to the incline of the workpiece, and showed a comparative result to the one seen in section 4.4.1. The final workpiece, seen in Figure 5.7, used a combination of both movement styles – using a static-base style when the workpiece had an incline (such as at the beginning and end), and using a moving-base algorithm in the centre, where there was no active incline.

A direct comparison to the previous experiments conducted in Section 4.4.1 can be done by completing a nearest-neighbour analysis of the dynamic algorithm compared to the static- and moving-base algorithms. Figure 5.8 shows a completed nearest-neighbour analysis of all three algorithms when completed on a rectangular workpiece. It demonstrates that there is a clear mix between the static and dynamic algorithm, with a singular occurrence of the moving-base algorithm. Therefore, it can be concluded that when tracing a rectangular workpiece, the dynamic algorithm is equally as accurate as the previously-tested static-base algorithm. This pattern continues when completing an analysis on the angled workpiece, as seen in Figure 5.9. When tracing the angled

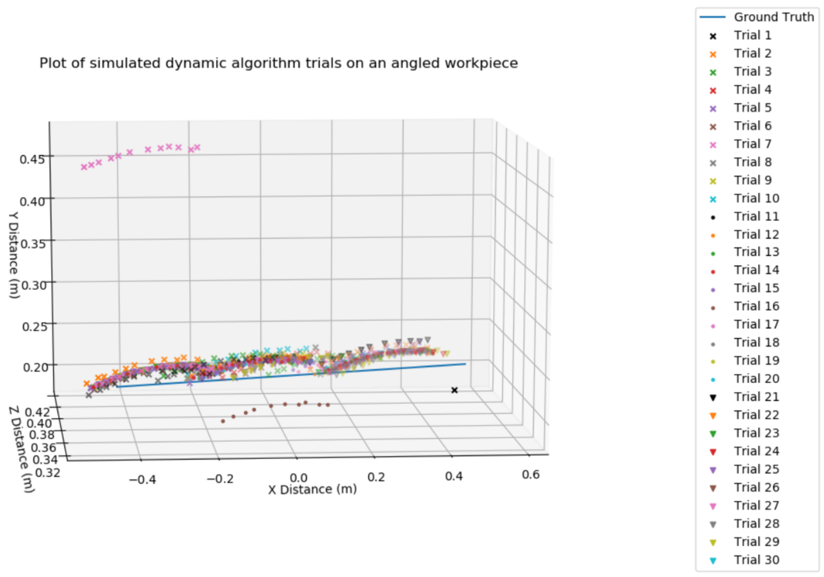


Figure 5.6: A scatter graph showing the traced path of the dynamic algorithm on an angled workpiece

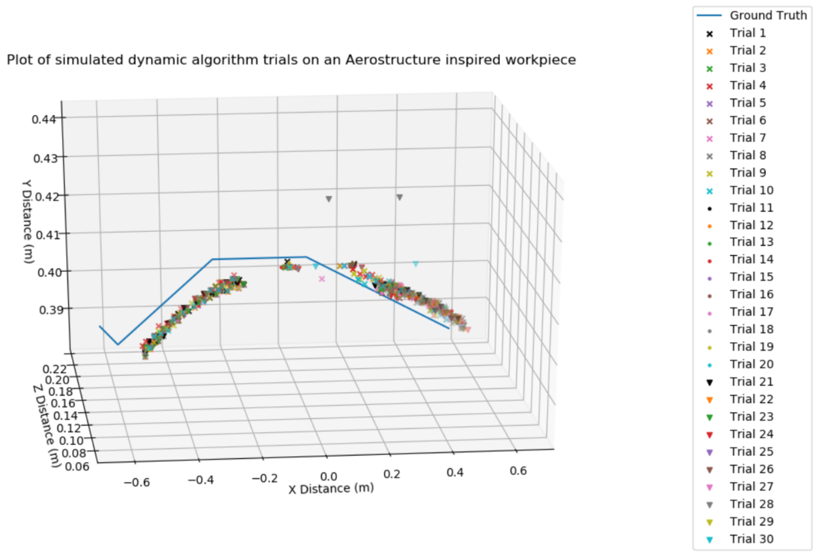


Figure 5.7: A scatter graph showing the traced path of the dynamic algorithm on an aerostructure-inspired workpiece

workpiece, there was an even split between the dynamic- and static-movement algorithms. This was due to how, when the algorithm traced the angled workpiece, it prioritised the static-movement style due to the incline of the workpiece. When tracing the aerostructure-inspired workpiece (Figure 5.10), a different pattern emerged. Unlike the previous workpieces, the nearest-neighbour analysis showed that the static-base movement style was dominant, and the dynamic style was only nearest on 3 occasions. This is due to the complex nature of the workpiece, and while, when running the dynamic-movement style, both static and moving styles were used, the algorithm is less accurate overall compared to using the default static-base algorithm.

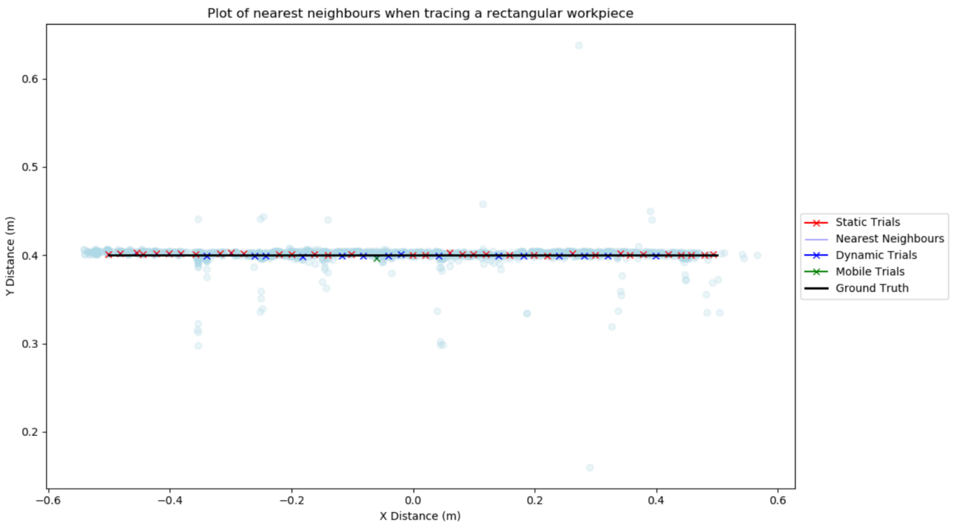


Figure 5.8: A scatter graph showing a nearest-neighbour analysis of the dynamic algorithm, when compared to previous algorithms on a rectangular workpiece

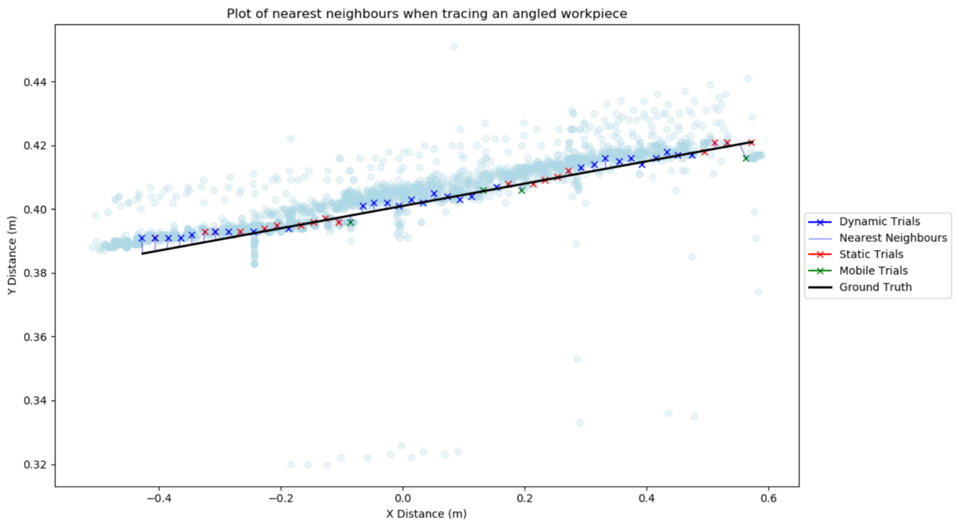


Figure 5.9: A scatter graph showing a nearest-neighbour analysis of the dynamic algorithm, when compared to previous algorithms on an angled workpiece

In addition to analysing the accuracy of the dynamic base algorithm, the completion times were also analysed. When tracing the rectangular workpiece

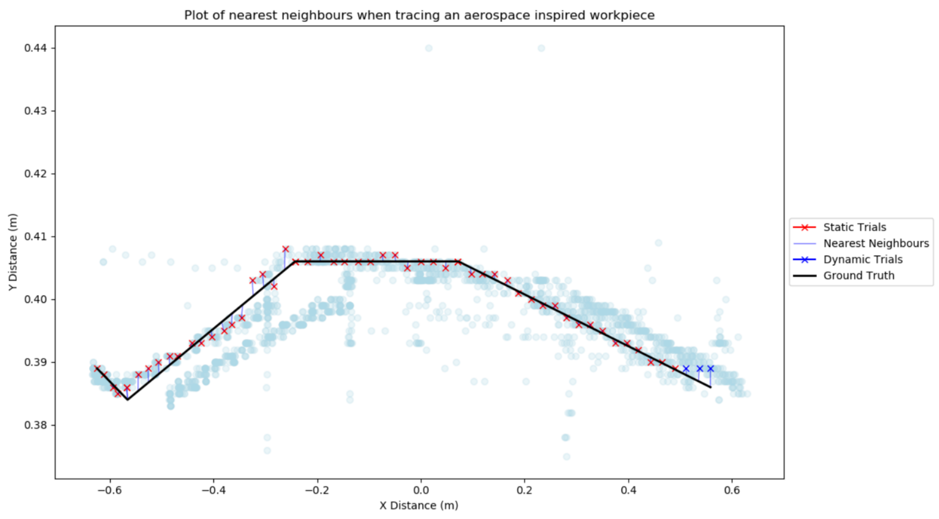


Figure 5.10: A scatter graph showing a nearest-neighbour analysis of the dynamic algorithm, when compared to previous algorithms on an aerostructure-inspired workpiece

(Figure 5.11), the dynamic algorithm performed similarly to the previously-tested moving-base algorithm, completing each trial in 5 seconds or less. Unlike the previously tested static-base algorithm, the dynamic-base algorithm is substantially more consistent time-wise, with its completion times clustering much more than what was seen in previous algorithms. This is likely due to the dynamic algorithm preferring the use of the moving-base algorithm, but the tuning allowing for a more consistent time output.

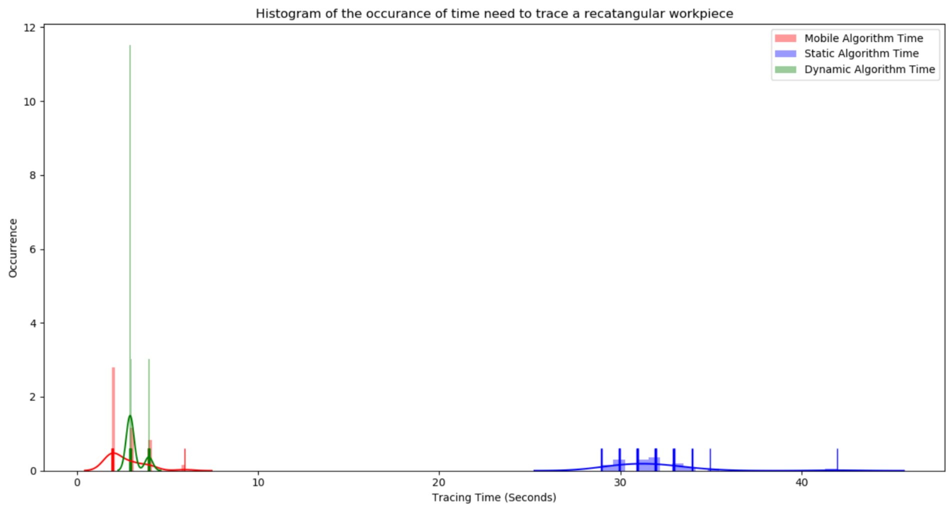


Figure 5.11: A histogram showing the completion time of the dynamic algorithm on a rectangular workpiece

The opposite can be seen in Figure 5.12, where due to the dynamic-base algorithm favouring a more static approach, the completion times for the angled workpiece were more comparable to the results seen when using the static-base algorithm. This is expected, since time-wise, the dynamic algorithm performed almost identically to the static-base algorithm, resulting in a similar histogram distribution.

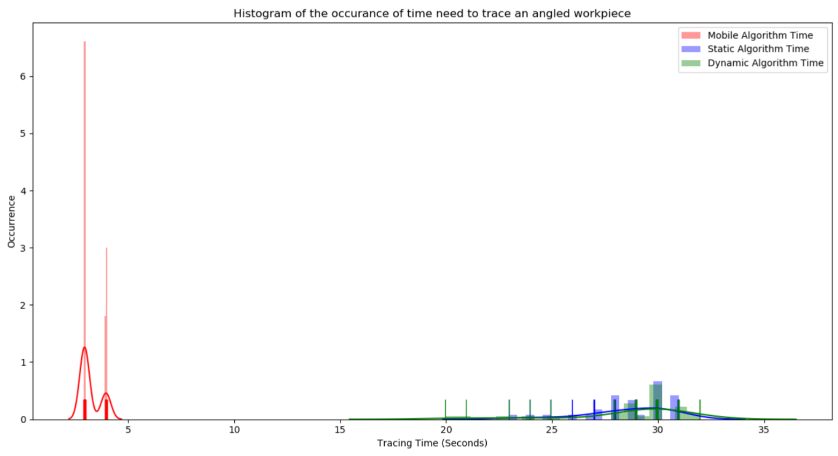


Figure 5.12: A histogram showing the completion time of the dynamic algorithm on an angled workpiece

When tracing the aeostructure-inspired workpiece (Figure 5.13), the resulting completion times reflected how the dynamic algorithm traced the workpiece. Unlike having a distribution on either end of the time axis (like the static- or moving-base algorithms), the dynamic algorithm's distribution was placed in between the two, averaging a completion time of 22 seconds, compared to the 30-40 seconds of the static algorithm or the 4 seconds of the moving algorithm. This was due to how the dynamic algorithm used a combination of both static- and moving-base styles but favoured a static style for ¾ of the workpiece due to its incline.

Using both accuracy and time data, a time-over-accuracy analysis was performed. This allowed a more in-depth analysis, and determined how optimal the dynamic algorithm was when compared to the previously-tested algorithms. The completed time-over-accuracy analysis for the rectangular workpiece (Figure 5.14) showed that in general, the dynamic algorithm was as quick as the

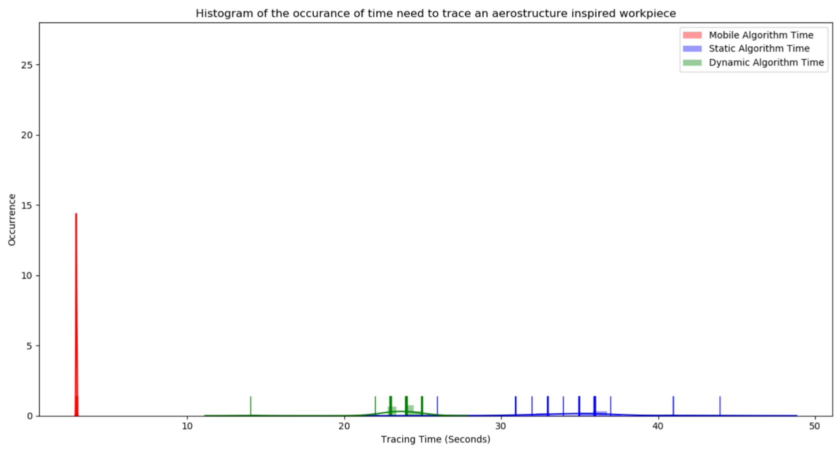


Figure 5.13: A histogram showing the completion time of the dynamic algorithm on an aerostructure-inspired workpiece

moving-base algorithm, while being more accurate (but still slightly less accurate than the static-base algorithm). Given the huge completion time decrease, with only a slight decrease in accuracy (0.008 P compared to the static bases 0.002 P), it can be concluded that the dynamic-base algorithm is a more optimal option when tracing a workpiece with little to no inclined features.

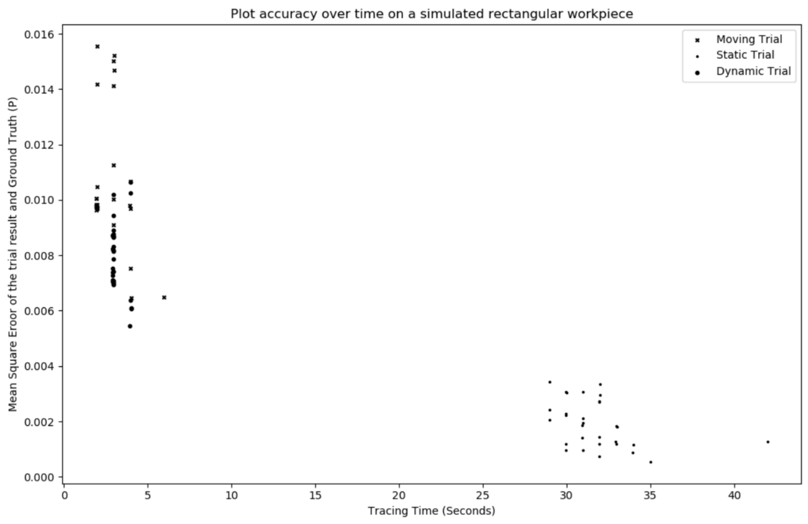


Figure 5.14: A scatter graph showing time over accuracy of the dynamic algorithm on a rectangular workpiece

The analysis of the angled workpiece (Figure 5.15) showed that, unlike the rectangular workpiece, the speed of the dynamic algorithm was equivalent to the times recorded for the static-base algorithm, but the accuracy was closer to the previously-seen moving-base algorithm. Some trials showed that the speed was sometimes quicker than the static-base algorithm, but this wasn't consistent. This was likely due to the tuning of the algorithm, and how much the base moved. This could be remedied in the future by adding a more intelligent moving style, which would consistently and dynamically change how much it moved depending on the workpiece's features.

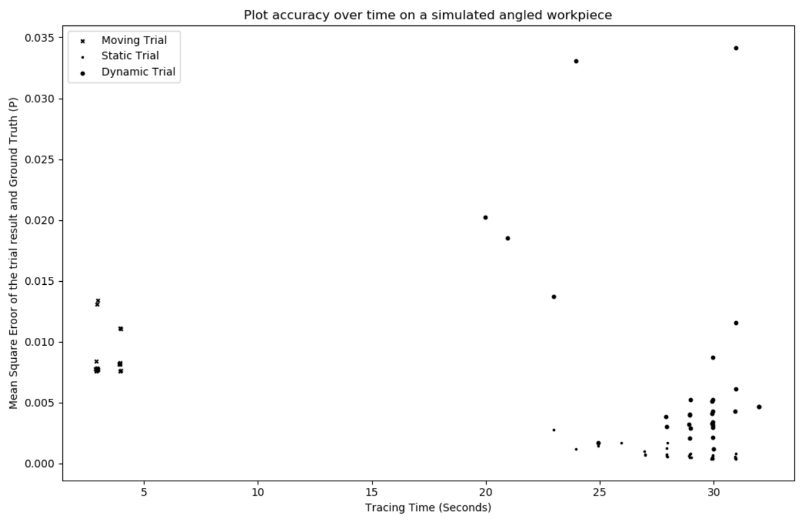


Figure 5.15: A scatter graph showing time over accuracy of the dynamic algorithm on an angled workpiece

An analysis of the aerostrucure-inspired workpiece (Figure 5.16) showed that the dynamic-base algorithm performed the median in both time and accuracy compared to the static- and moving-base algorithms. The dynamic-base algorithm was quicker than the static algorithm by almost 10 seconds, while being very slightly less accurate (0.0075 P compared to 0.005 P). It was also substantially more accurate than the moving-base algorithm (0.0075 P compared to 0.016 P), while being 20 seconds slower. While the speed of the dynamic algorithm compared to the moving algorithm feels substantial, a consistently-moving algorithm isn't practical for tracing a workpiece with such complex features – as can been seen via the moving-base algorithm's poor accuracy score. Given the dynamic algorithm's speed increase over the static algorithm, with only a miniscule decrease in accuracy, it can be concluded that using a dynamic-base algorithm is an optimal choice over either purely static or moving algorithms, as it allowed the robot to adapt to different workpiece features and use the strengths of each algorithm's movement styles when appropriate.

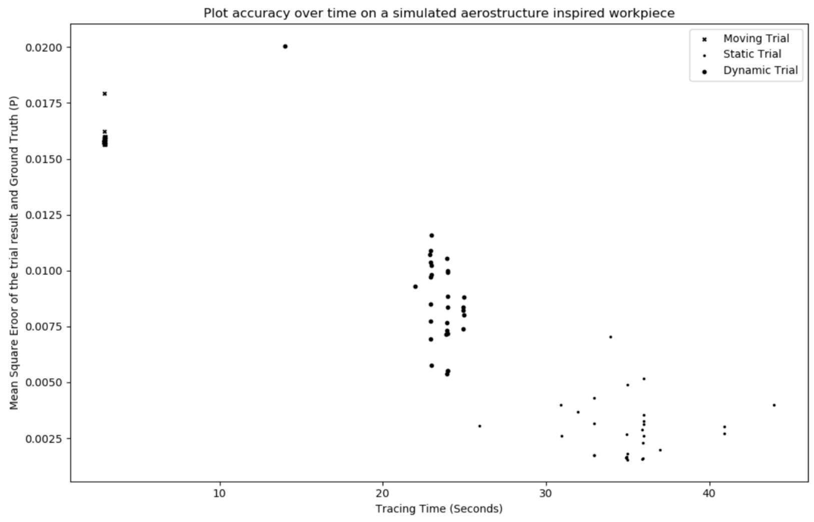


Figure 5.16: A scatter graph showing time over accuracy of the dynamic algorithm on an angled workpiece

5.4 Chapter Summary

In conclusion, the dynamic-base algorithm performed optimally when tracing the more complex aerostructure-inspired workpiece, as well as the rectangular workpiece. For the aerostructure-inspired workpiece, this was due to the ability of the dynamic algorithm to switch between moving styles, allowing the use of the moving algorithm during straight features, and the static algorithm during more spatially-complex sections. The dynamic algorithm performed optimally while tracing a rectangular workpiece due to the additional tuning compared to previous algorithms, ensuring that the robot could cover more of the workpiece while committing to less movement. Unfortunately, the algorithm didn't improve the efficiency of tracing the angled workpiece, mostly due to the consistent, inclined feature of the workpiece (although it also didn't perform any worse than any previously-provided algorithms).

The tuning used for the dynamic algorithm, while limited, proved to be an effective way to explore the variable options the algorithm has, and was able to optimise the performance of the algorithm, ensuring it could perform to its full potential while tracing the workpieces provided.

6

Discussion and Conclusion

6.1 Research Findings and Contributions

The findings of the research are summarised in the following sections, separated by project objective and chapter.

6.1.1 Creation and Implementation of a Visual Servoing System

Chapter 3 *'Implementation of a flexible solution for feature detection and tracing of manufacturing workpieces'* focused on achieving the first objective:

`Develop and implement a vision system for the feature detection and tracing of spatially-varying, partially-known manufacturing workpieces'

The aim of the first objective was to create a lightweight and dynamic visual servoing system which could detect and trace the edge features of partially-seen, spatially-varying manufacturing workpieces. This was done in two parts: firstly, by creating a vision system which could detect the edges of workpieces; and then turning that system in a visual servoing system.

When creating the system, experiments were completed to understand how environmental factors and camera variables affected the clarity of a detected workpiece edge. The results showed that when detecting a workpiece's edge, lighting and aperture are the two most important variables – ensuring that the camera is not flooded with light, or else exclusively detecting workpiece features becomes difficult without additional post-processing. By gathering this data, an optimal environmental setup was created: the most optimal setup for detecting the edge of a manufacturing workpiece.

This setup was then used to classify different styles of common RGB cameras, to gather an understanding of their effectiveness when tracing manufacturing workpiece features. These experiments concluded that cameras which can provide an image quality of 720p or higher can easily trace a realistic workpiece, and extra features such as a higher resolution or depth-sensing can be used to aid in the deprojection of image pixels to 3D coordinates.

Using the camera, a variety of workpieces were traced, each with a different spatial profile. These ranged from multiple, straight-edged workpieces, to workpieces with a curved, cylindrical edge. The end result showed that all workpiece edges were detected within a reasonable level of accuracy – less than 3cm worth of discrepancy when compared to the ground truth.

Regarding the literature, most of the research completed for the first objective was novel and unique, especially within the context of manufacturing workpieces. Aspects such as how environmental factors affect camera work were already well known within the computer imaging field, but the direct effects on workpiece edge detection is novel. Additionally, the detection and tracing of curved edge workpieces was completely novel, having no previous published work.

6.1.2 Creation of distribution-of-labour algorithms for tracing manufacturing workpieces

Chapter 4 *'Evaluation of various distribution-of-labour algorithms for tracing a partially-known workpiece'* focused on achieving the second and fourth objectives:

'Create and evaluate various distribution-of-labour algorithms for tracing a spatially-varying, partially-known manufacturing workpiece'

'Validate and test the developed system and algorithms on test samples inspired by real-life engineering parts.'

The aim of these objectives were to investigate the different possible ways a mobile kinematic arm robot could interact and trace a manufacturing workpiece, specifically by investigating the possible distribution of labour between the main two parts of the system. By doing this, an investigation into how a robot may trace a workpiece was completed, and how different styles of tracing would affect the accuracy and time of the tracing operation. This was completed both in simulation and on real manufacturing workpiece samples, to ensure the validity of the results.

For these experiments, two fundamentally-opposed algorithms were designed. One algorithm would prioritise base movement, while the other would prioritise arm movement. The goal was to run identical experiments with each algorithm, to analyse the results, and to see what benefits and negatives each algorithm had when tracing different workpiece features. This data could then be used in the future for creating a dynamic algorithm.

The results concluded that the moving-base algorithm was optimal, time-wise, within all the experiments, while the static-base algorithm was optimal, accuracy-wise, with all workpieces – although the difference in accuracy between the two algorithms depended heavily on the workpiece feature. Specifically, when the workpiece feature was not inclined, it was concluded that the accuracy difference between the two algorithms was so miniscule that the time saved via the moving-base algorithm made it the optimal choice.

When regarding the literature, there is little specific work focusing on the distribution of labour between the two tested aspects of the mobile manipulator. Some parts of the literature analyse the use of mobile manipulators generally, and a subsection of this is dedicated to industries such as manufacturing. In all cases, they are generally trying to solve a specific problem, and never consider the distribution of labour as a particular focus. Some of the findings from the experiments have been known for some time, such as that a moving manipulator is less accurate than a static manipulator, but the content of the experiments, as well as the experiments themselves, are unique. Therefore, the investigation into the distribution of labour between the two robotic systems is novel within the literature, as are the algorithms being created and compared. Additionally, the experiments conducted on the manufacturing sample workpieces help strengthen the novelty of the research conducted in chapter 4, by validating both the initial hypothesis and the results seen from all experimentation.

6.1.3 Creation of an optimal distribution-of-labour algorithm, to optimally trace manufacturing workpieces

Chapter 5 '*Development of a dynamic distribution-of-labour algorithm to effectively and optimally complete the tracing of industrial workpieces*' focused on achieving the third objective:

'Create a dynamic distribution-of-labour algorithm to optimally trace a spatially-varying, partially-known manufacturing workpiece'

After gathering data from the previous objective, the goal of the final objective was to create a dynamic algorithm which could dynamically change how it traced a workpiece according to the results. The algorithm used the results gathered from Objective 2's experiments to inform what tracing style to use dependent on the feature it had seen. If the detected feature was inclined, it would use the static-base algorithm, whereas if the feature wasn't inclined, it would use the moving-base algorithm – therefore attempting to create an algorithm which would optimally trace a workpiece.

Two sets of experiments were completed: an initial set of experiments to tune the algorithm and some of its variables, and a set of in-depth, simulated experiments on a variety of workpieces. The tuning experiments had to be completed to ensure that the dynamic algorithm would perform as expected, and to ensure the best possible results were obtained. The simulated experiments used a variety of workpieces in a similar setup to the previous objective, to ensure that the results from the dynamic algorithm could be compared to previous experiments, allowing for a detailed comparison and analysis.

The results of the experiments showed that, in most situations, the dynamic algorithm performed equally or better than the previous algorithms, depending on the features of the workpiece.

- When tracing a rectangular workpiece, it was concluded that the dynamic algorithm performed better than the previously-tested static- and moving-base algorithms, being as quick as the moving-base algorithm, and almost as accurate as the static-base algorithm.
- When tracing the angled workpiece, the dynamic algorithm performed on par with the previously-tested static-base algorithm.
- When tracing the aerostructure-inspired workpiece, the dynamic algorithm performed optimally, having an accuracy equal to that of the static-base algorithm, while having a tracing time in between the static- and moving-base algorithms.

Overall, it was concluded that the dynamic algorithm was generally more efficient than using either of the previously-tested algorithms, mainly due to how it either matches or outperforms the optimal results seen in previous experiments, while actively being able to adapt to a workpiece with a variety of spatial features.

Regarding the literature, the creation of a dynamic algorithm which can trace a spatially-varying workpiece dynamically, is novel. There has been no previous investigation into such a system, as much of the current literature focuses on using additional data to help guide the robot when tracing a workpiece, such as CAD files.

6.2 Limitations of Research

The conducted research was developed and experimented at a TRL level of 4 [89] due to the majority of laboratory-based experiments, and lack of any real world, industrial grade experimentation. Additionally, the general scope of this research was quite strict, and many aspects of mobile robotics, such as safety, were not a major consideration.

During the environmental, variable experiments in Section 3.4, the variables which were analysed and experimented on were very major, and only a factorial design of experiments was created for it. As a result, not many conclusive results were derived from this, and the optimal environmental setup was very simplistic. Additionally, the cameras which were used after that were limited in scope – only having three major types of RGB camera to compare against. This was a very limited set of possible cameras, and, having more categories covered, such as a time-of-flight camera, may have proven useful and more informative.

During the experiments in Section 4.4.2, limited, real-world experiments were conducted to ensure the validity of the large volume of simulated experiments. When completing the real-world verification, only a small amount of trials were completed with a limited number of workpieces, due to time constraints and the difficulty of replicating simulated experiments with industrial robotics. Whilst the experiments completed were able to verify the simulated experiments, completing the physical verification with additional workpieces (matching each one used in simulation) and additional trials to make them ISO 9283 compliant, would have been beneficial.

During the creation of the dynamic-base algorithm in Section 6, the algorithm was tuned by tweaking two particular variables in 9 total configurations, by doing a rudimentary, factorial experiment design. This was done to aid the algorithm's accuracy and to ensure that the algorithm performed at its best when completing experiments. This verification was very limited, since both variables being tuned were continuous. While the results showed an effective and tuned algorithm, the tuning process could have been improved by running many more simulations with a much wider range of variables.

6.3 Proposed Future Work

The developed dynamic algorithm works effectively within simulation when tracing the aerostructure-inspired workpiece, but the tuning done to it and specific algorithm features were very limited in scope. A proposed addition to the algorithm, to allow it to alter its movement more freely, would greatly aid the accurately when tracing larger workpieces, especially when using a moving-base tracing style. One of the biggest issues with the dynamic algorithm is that it overestimates how much it can move, since all distance calculations are done before the robot starts moving. Such a feature would allow the algorithm to dynamically start and stop in-process, aiding with accuracy while tracing.

All the algorithms developed have only focused on tracing a workpiece once it has manually been put into position. This is a limiting factor, as it means that the amount of data the algorithms can leverage for information on how to trace a workpiece is very limited. Currently, it can only use partially-observed vision data, meaning the robot can only motion plan for the distance it can see – generally less than 1 metre. It is proposed that, by adding an intelligent SLAM movement system alongside a high-level planning system, a robot would be able to actively take `breaks' from tracing a workpiece to observe the workpiece as a whole, and thus gather more information. This could come in many forms, from a robot observing the whole workpiece by travelling around a factory floor before tracing it all, or just using an intelligent navigation system to observe larger portions of the workpiece.

6.4 Conclusions

In conclusion, the use of dynamic robots for the manufacturing of low-volume, high-value workpieces is increasing, and is expected to increase in the future. As robotic equipment becomes more accurate, specifically regarding kinematic arms, the ease in which the manufacture of high-precision workpieces can be completed will be greatly enhanced. Research was completed to show the viability of kinematic arms completing a high-value manufacturing task, by interacting with particular features of a large, partially-observed manufacturing workpiece. The results showed that it was possible to do so to a given level of accuracy and precision and presented a comparison of multiple techniques to achieve the task. Multiple algorithms were proposed which provided different techniques for tracing a manufacturing workpiece, by splitting up the distribution of labour between the kinematic arm and mobile base. These algorithms were then compared, the results of which were used to inform the creation of a dynamic distribution-of-labour algorithm, which could dynamically change how the robot would interact with the manufacturing workpiece, depending upon what sort of features were detected. The results showed a capable algorithm which could easily trace a variety of spatially-varying workpieces with no prior knowledge or data.

Unlike much of the previous literature, which focused on custom-built robotic cells, this research showed that the use of a generalised robotic algorithm is not only a viable option but comparable to some currently used aerospace manufacturing techniques. This was done through the experimental use of distribution of labour algorithms, something not seen in the literature previously. This explored the relationship between mobile bases and manipulators and the roles they share when tracing manufacturing workpieces. While the outcomes matched some previously formed hypotheses, both the algorithms showed various strengths and weaknesses when tracing workpieces with different spatial features of workpieces. This data was then used to create a new, novel algorithm which was able to dynamically control its division of labour, something not previously seen in literature. This research has also provided much-needed details regarding the edge detection of spatially varying manufacturing workpieces and the effect environmental factors have on edge detection quality. Specifically, the research added details regarding a technique on how to trace obtusely angled manufacturing workpieces, something which was not previously available in the literature.

This work will enable a faster and more generic approach to robotics with manufacturing, allowing the automation of operations which were previously thought to be too costly to be automated. The creation of the division of labour algorithms, and the analysis of them when tracing various workpieces, will allow a larger and more detailed investigation into the role mobile manipulators will play in the future of high value manufacturing. Future work could involve higher TRL examinations of division of labour algorithms, when interacting with larger, more realistic manufacturing workpieces could be investigated, with the aim of making them more accurate, and increasing the overall speed of the operations.

References

[1] A. Liaqat, W. Hutabarat, D. Tiwari et al. `Autonomous mobile robots in manufacturing: Highway Code development, simulation, and testing'. In: *International Journal of Advanced Manufacturing Technology* 104.9-12 (2019), pp. 4617–4628 (cit. on pp. xiii, 13).

[2] Federico Boniardi, Tim Caselitz, K Rainer and Wolfram Burgard. `Robust LiDAR-based Localization in Architectural Floor Plans'. In: *International Conference on Intelligent Robots and Systems* (*IROS*) (2017), pp. 3318–3324 (cit. on pp. xiii, 17, 18).

[3] Steve Grehl, Mark Sastuba, Marc Donner et al. `Towards virtualization of underground mines using mobile robots – from 3D scans to virtual mines'. In: *Mine Planning Equipment Selection 2015* (June 2015) (cit. on pp. xiii, 23).

[4] Ling Zheng, Bijun li, Bo Yang, Huashan Song and Zhi Lu. `Lane-Level Road Network Generation Techniques for Lane-Level Maps of Autonomous Vehicles: A Survey'. In: *Sustainability* 11 (Aug. 2019), p. 4511 (cit. on pp. xiii, 23).

[5] Vinayak Ashok Prabhu, Boyang Song, John Thrower, Ashutosh Tiwari and Phil Webb. `Digitisation of a moving assembly operation using multiple depth imaging sensors'. In: *International Journal of Advanced Manufacturing Technology* 85.1-4 (2016), pp. 163–184 (cit. on pp. xiii, 28).

[6] Panagiotis Koustoumpardis, Paraskevi Zacharia and Nikos Aspragathos. `ntelligent Robotic Handling of Fabrics Towards Sewing'. In: *Industrial Robotics: Programming, Simulation and Applications*. Ed. by Low Kin Huat.

Pro Literatur Verlag, Germany / ARS, Austria, 2006. Chap. 28, pp. 559–582 (cit. on pp. xiii, 32, 33).

[7] ABB. *ABB to provide paint solutions to SAIC Volkswagen's first electric vehicle factory in China*. 2019 (cit. on pp. xiii, 35, 36).

[8] Shashank Sharma, Gerhard Kraetzschmar, Christian Scheurer and Rainer Bischoff. 'Unified Closed Form Inverse Kinematics for the KUKA youBot'. In: 2012, pp. 1–6 (cit. on pp. xv, 78).

[9] Monika Florek-Jasinska. *YouBot Detailed Specifications*. 2015 (cit. on pp. xv, 78, 82).

[10] Jon Martin, Tobias Fink, Stefan May and Robert Bosch Gmbh. 'An Autonomous Transport Vehicle in an existing manufacturing facility with focus on the docking maneuver task'. In: *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)* (2017), pp. 365–370 (cit. on pp. xix, 14, 15).

[11] Luis Pérez, Íñigo Rodríguez, Nuria Rodríguez, Rubén Usamentiaga and Daniel F. García. 'Robot guidance using machine vision techniques in industrial environments: A comparative review'. In: *Sensors (Switzerland)* 16.3 (2016) (cit. on pp. xix, 22).

[12] Mark Williams. *JOBS02: Workforce jobs by industry*. Tech. rep. Office for National Statistics, 2019 (cit. on p. 2).

[13] Michael Rüßmann, Markus Lorenz, Philipp Gerbert et al. 'Industry 4.0: The future of productivity and growth in manufacturing industries'. In: *Boston Consulting Group* 9.1 (2015), pp. 54–89 (cit. on p. 2).

[14] Kuka. *KUKA omniMove moves heavy parts in confined spaces during construction of the A380 at Airbus*. 2016 (cit. on p. 3).

[15] Airbus. *Wing of the Future*. 2017 (cit. on p. 3).

[16] Ulrike Götz. *KUKA omniMove transports wing for A350 XWB | KUKA AG*. 2016 (cit. on p. 12).

[17] H. H. Götting. ‘Automation and Steering of Vehicles in Ports’. In: *Port Technology International 10* (2000), pp. 101–111 (cit. on p. 13).

[18] Iris F.A. Vis. ‘Survey of research in the design and control of automated guided vehicle systems’. In: *European Journal of Operational Research* 170.3 (2006), pp. 677–709 (cit. on p. 13).

[19] Tuan Le-Anh and M. B.M. De Koster. ‘A review of design and control of automated guided vehicle systems’. In: *European Journal of Operational Research* 171.1 (2006), pp. 1–23 (cit. on p. 14).

[20] Reinhardt Jünemann and Thorsten Schmidt. *Materialflußsysteme*. Berlin: Springer, 2000 (cit. on p. 14).

[21] James A. Tompkins, John A. White, Yavuz A. Bozer and J. M. A. Tanchoco. *Facilities Planning*. Fourth Edi. Wiley, 2010 (cit. on p. 14).

[22] Pericle Salvini, Diego Paez-Granados and Aude Billard. ‘Safety Concerns Emerging from Robots Navigating in Crowded Pedestrian Areas’. In: *International Journal of Social Robotics* 14.2 (2022), pp. 441–462 (cit. on p. 16).

[23] Christoph Sprunk, Boris Lau, Patrick Pfaff and Wolfram Burgard. ‘An accurate and efficient navigation system for omnidirectional robots in industrial environments’. In: *Autonomous Robots* 41.2 (2017), pp. 473–493 (cit. on p. 16).

[24] Alexander Schaefer, B Daniel, Lukas Luft and Wolfram Burgard. ‘A Maximum Likelihood Approach to Extract Polylines from 2-D Laser Range Scans’. In: *International Conference on Intelligent Robots and Systems* (2018), pp. 4766–4773 (cit. on p. 17).

[25] Federico Boniardi, Tim Caselitz, Rainer Kümmerle and Wolfram Burgard. ‘A pose graph-based localization system for long-term navigation in CAD floor plans’. In: *Robotics and Autonomous Systems* 112 (2018), pp. 84–97 (cit. on p. 18).

[26] Richard P. Paul. *Robot Manipulators: Mathematics, Programming, and Control : the Computer Control of Robot Manipualtor*. illustrated. Richard Paul, 1981, p. 279 (cit. on p. 19).

[27] Per Bergman, Steven J. Blacker, Nicholas Kottenstette, Omid Saber and Saeed Sokhanvar. 'Robotic-Assisted Percutaneous Coronary Intervention'. In: *Handbook of Robotic and Image-Guided Surgery*. Elsevier, Jan. 2020, pp. 341–362 (cit. on p. 20).

[28] Kathrin Cresswell, Sandeep Ramalingam and Aziz Sheikh. 'Can Robots Improve Testing Capacity for SARS-CoV-2?' eng. In: *Journal of medical Internet research* 22.8 (Aug. 2020), e20169–e20169 (cit. on p. 20).

[29] Wai Mar Myint and Theingi. 'Kinematic Control of Pick and Place Robot Arm'. In: *International Journal of Engineering and Techniques* 1.4 (2015), pp. 63–70 (cit. on p. 20).

[30] Carlos Martinez, Remus Boca, Biao Zhang, Heping Chen and Srinivas Nidamarthi. 'Automated bin picking system for randomly located industrial parts'. In: *2015 IEEE International Conference on Technologies for Practical Robot Applications* (*TePRA*). 2015, pp. 1–6 (cit. on p. 20).

[31] E. R. Davies. 'Computer and Machine Vision'. In: *Computer and Machine Vision* (2012) (cit. on p. 21).

[32] Larry Li. *Time-of-Flight Camera – An Introduction*. 2014 (cit. on p. 22).

[33] Péter Fankhauser, Michael Bloesch, Diego Rodriguez et al. 'Kinect v2 for mobile robot navigation: Evaluation and modeling'. In: *Proceedings of the 17th International Conference on Advanced Robotics, ICAR 2015* (Oct. 2015), pp. 388–394 (cit. on p. 24).

[34] Gabriel Danciu, S.M. Banu and A. Caliman. 'Shadow removal in depth images morphology-based for Kinect cameras'. In: Jan. 2012, pp. 1–6 (cit. on p. 24).

[35] G J Agin. *Real time control of a robot with a mobile camera*. Technical note. SRI International, 1979 (cit. on p. 25).

[36] Seth Hutchinson, Gregory D. Hager and Peter I. Corke. `A tutorial on visual servo control'. In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), pp. 651–670 (cit. on pp. 25, 26).

[37] François Chaumette and Seth Hutchinson. `Visual servo control. II. Advanced approaches [Tutorial]'. In: *Robotics Automation Magazine, IEEE* 14 (Apr. 2007), pp. 109–118 (cit. on p. 25).

[38] Jenelle Armstrong Piepmeier and Harvey Lipkin. `Uncalibrated eye-in-hand visual servoing'. In: *The International Journal of Robotics Research* 22.10-11 (2003), pp. 805–819 (cit. on p. 25).

[39] Arthur C. Sanderson and Lee E. Weiss. `Adaptive Visual Servo Control of Robots'. In: *Robot Vision* (1983), pp. 107–116 (cit. on p. 26).

[40] G. Palmieri, M. Palpacelli, M. Battistelli and M. Callegari. `A comparison between position-based and image-based dynamic visual servoings in the control of a translating parallel manipulator'. In: *Journal of Robotics* 2012 (2012) (cit. on p. 26).

[41] Andrew Kusiak. `Smart Manufacturing'. In: *International Journal of Production Research* 56 (2017), pp. 508–517 (cit. on p. 26).

[42] A. Tiwari, C. J. Turner and B. Majeed. *A review of business process mining: State-of-the-art and future trends*. 2008 (cit. on p. 27).

[43] M. Edwards. `Robots in industry: An overview'. In: *Applied Ergonomics* 15.1 (1984), pp. 45–53 (cit. on p. 27).

[44] Sakae Takahashi, Mamoru Maezima and Yuji Takahashi. *ASSEMBLY LINE CONSTRUCTION AND METHOD FOR ASSEMBLNG AUTOMOTIVE VEHICLE BODES*. 1993 (cit. on p. 27).

[45] Panagiota Tsarouchi, Sotiris Makris, George Michalos et al. `Robotized assembly process using Dual arm robot'. In: *Conference on Assembly Technologies and Systems*. Vol. 23. Elsevier B.V., 2014, pp. 47–52 (cit. on p. 27).

[46] L. Zhou, T. Lin and S. B. Chen. 'Autonomous acquisition of seam coordinates for arc welding robot based on visual servoing'. In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 47.3 (2006), pp. 239–255 (cit. on p. 29).

[47] De Xu, Linkun Wang and Min Tan. 'Image processing and visual control method for arc welding robot'. In: *Proceedings - 2004 IEEE International Conference on Robotics and Biomimetics, IEEE ROBIO 2004* (2004), pp. 727–732 (cit. on p. 29).

[48] Boeing. *Robot Painters*. 2013 (cit. on pp. 30, 36).

[49] Rui Chen, Guolei Wang, Jianguo Zhao, Jing Xu and Ken Chen. 'Fringe Pattern Based Plane-to-Plane Visual Servoing for Robotic Spray Path Planning'. In: *IEEE/ASME Transactions on Mechatronics* 23.3 (2018), pp. 1083–1091 (cit. on p. 30).

[50] F. Prieto, T. Redarce, R. Lepage and P. Boulanger. 'An automated inspection system'. In: *International Journal of Advanced Manufacturing Technology* (2002) (cit. on p. 31).

[51] N J Shipway, T J Barden, P Huthwaite and M J S Lowe. 'Automated defect detection for Fluorescent Penetrant Inspection using random forest'. In: *NDT and E International* 101.October 2018 (2018), pp. 113–123 (cit. on p. 31).

[52] Jacques A. Gangloff, Michel de Mathelin and Gabriel Abba. 'Visual servoing of a 6 DOF manipulator for unknown 3D profile following'. In: *Proceedings - IEEE International Conference on Robotics and Automation* (1999) (cit. on p. 32).

[53] Jacques A. Gangloff and Michel F. De Mathelin. 'Visual servoing of a 6-DOF manipulator for unknown 3-D profile following'. In: *IEEE Transactions on Robotics and Automation* (2002) (cit. on p. 32).

[54] Haiyong Chen, Kun Liu, Guansheng Xing et al. 'A robust visual servo control system for narrow seam double head welding robot'. In: *International Journal of Advanced Manufacturing Technology* 71.9-12 (2014), pp. 1849–1860 (cit. on p. 33).

[55] Airbus. *Production*. 2019 (cit. on p. 34).

[56] Anon. 'Robot-based machine tool for the aerospace industry(Article)'. In: *Industrial Robot: An International Journal* 20.3 (1993), pp. 11–13 (cit. on p. 34).

[57] Nigel R Roche. 'Automatic Riveting Cell for Commercial Airplane Floor Grid Assembly'. In: *SAE Transactions* 103 (1994), pp. 1884–1896 (cit. on p. 34).

[58] Brian Rooks. *Automatic wing box assembly developments*. Bradford, England : 2001 (cit. on p. 34).

[59] Karl-Erik Neumann. 'True Mobile/Portable Drilling and Machining, a Paradigm Shift in Manufacturing'. In: *SAE International* (2017) (cit. on p. 35).

[60] N. Jayaweera and Phil Webb. 'Automated assembly of fuselage skin panels'. In: *Assembly Automation* 27.4 (2007), pp. 343–355 (cit. on p. 35).

[61] Robert Bogue. 'The growing use of robots by the aerospace industry'. In: *Industrial Robot: An International Journal* 45.6 (2018), pp. 705–709 (cit. on p. 36).

[62] Rio Tinto. *Mine of the Future™*. 2019 (cit. on p. 36).

[63] EVELYN M. RUSLI. *Amazon.com to Acquire Manufacturer of Robotics*. 2012 (cit. on p. 36).

[64] Ángel González. *Amazon's robot army grows by 50 percent*. 2016 (cit. on p. 36).

[65] Stanford Artificial Intelligence Laboratory. *Robotic Operating System*. 2018 (cit. on p. 45).

[66] Gary Bradski. 'The OpenCV Library'. In: *Dr. Dobb's Journal of Software Tools* (2000) (cit. on p. 46).

[67] Intel. *ROS Wrapper for Intel® RealSense™ Devices*. 2018 (cit. on pp. 46, 76).

[68] Ayssam Elkady and Tarek Sobh. 'Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography'. In: *Journal of Robotics* (2012), pp. 1–15 (cit. on p. 46).

[69] N. Koenig and A. Howard. 'Design and use paradigms for Gazebo, an open-source multi-robot simulator'. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. 2004 (cit. on p. 46).

[70] Lucas Nogueira. 'Comparative Analysis Between Gazebo and V-REP Robotic Simulators'. In: *ICMREE2011 - Proceedings 2011 International Conference on Materials for Renewable Energy and Environment* 2.December 2014 (2014), pp. 1678–1683 (cit. on p. 46).

[71] John Canny. 'A Computational Approach to Edge Detection'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1986) (cit. on p. 47).

[72] Adrian Rosebrock. *Zero-parameter, automatic Canny edge detection with Python and OpenCV*. 2015 (cit. on p. 47).

[73] Jakob Engel, Thomas Schöps and Daniel Cremers. 'LSD-SLAM: Large-Scale Direct Monocular SLAM'. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele and Tinne Tuytelaars. Cham: Springer International Publishing, 2014, pp. 834–849 (cit. on p. 47).

[74] Clément Godard, Oisin Aodha, Michael Firman and Jourdan Gabriel. 'Digging Into Self-Supervised Monocular Depth Estimation'. In: 2019 (cit. on p. 47).

[75] Vrushali Patil, Indraneel Patil, V. Kalaichelvi and R. Karthikeyan. 'Extraction of Weld Seam in 3D Point Clouds for Real Time Welding Using 5 DOF Robotic Arm'. In: *2019 5th International Conference on Control, Automation and Robotics, ICCAR 2019* (2019), pp. 727–733 (cit. on p. 49).

[76] Mark Anderson and Patrick Whitcomb. 'Chapter 3: Two-Level Factorial Design'. In: *DOE Simplified: Practical Tools for Effective Experimentation, Second Edition*. 2007, pp. 1–30 (cit. on p. 49).

[77] R Manish and S Denis Ashok. 'Study on Effect of Lighting Variations in Edge Detection of Objects using Machine Vision System'. In: *International Journal of Engineering Research Technology* (*IJERT*) 4.26 (2016), pp. 1–5 (cit. on p. 58).

[78] Commonplace Robotics. *Robotic Arm Mover6 User Guide*. Commonplace Robotics GmbH, 2014, pp. 1–32 (cit. on p. 69).

[79] International Organization for Standardization. *Manipulating industrial robots — Performance criteria and related test methods*. Standard. Geneva, CH: International Organization for Standardization, Apr. 1998 (cit. on p. 70).

[80] Kleber Roberto da Silva Santos, Emília Villani, Wesley Rodrigues de Oliveira and Augusto Dttman. 'Comparison of visual servoing technologies for robotized aerospace structural assembly and inspection'. In: *Robotics and Computer-Integrated Manufacturing* 73.July 2021 (2022), p. 102237 (cit. on p. 72).

[81] Saeid Mokaram, Jonathan M. Aitken, Uriel Martinez-Hernandez et al. 'A ROS-integrated API for the KUKA LBR iiwa collaborative robot'. In: *IFAC-PapersOnLine* 50.1 (2017), pp. 15859–15864 (cit. on p. 78).

[82] Rainer Bischoff, Ulrich Huggenberger and Erwin Prassler. 'KUKA youBot - A mobile manipulator for research and education'. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2011), pp. 3–6 (cit. on p. 78).

[83] MAS Group. 'youbot$_{s}imulation$'. In: (2021) (cit. on p. 79).

[84] Pal Robotics. *Intel RealSense Gazebo ROS plugin*. 2021 (cit. on p. 79).

[85] David Coleman, Ioan Sucan, Sachin Chitta and Nikolaus Correll. 'Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study'. In: (2014), pp. 1–14. arXiv: 1404.3785 (cit. on p. 79).

[86] Tully Foote. `tf: The transform library'. In: *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*. Open-Source Software workshop. Apr. 2013, pp. 1–6 (cit. on p. 87).

[87] Pauli Virtanen, Ralf Gommers, Travis E Oliphant et al. `SciPy 1.0: fundamental algorithms for scientific computing in Python'. In: *Nature Methods* 17.3 (2020), pp. 261–272 (cit. on p. 99).

[88] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort et al. `Scikit-learn: Machine learning in Python'. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830 (cit. on p. 99).

[89] W Nolte, Brian C Kennedy and R J Dziegiel. `Technology readiness calculator'. In: *6th Annual System Engineering Conference* (2004), pp. 1–16 (cit. on p. 136).