Real-Time Image-based Localization and Tracking of a Continuum Flexible Robot

Blanca Fernandez Salamanca

Master of Informatics

School of Informatics
University of Edinburgh

2020

Abstract

The following report describes research undertaken to accurately track a concentric tube robot throughout images or video frames. Its intended use is for minimally invasive lung surgeries. The robot contains three pre-curved tubes that allow for a wide variety of positions and can traverse complex 3D structures, such as the human body. An existing mathematical model can predict the robot's full shape to an accuracy of 91% (of a 20cm robot, the shape is off by 1.8cm). The goal of this project is to provide vision-processing methods that can track the robot to a higher degree of accuracy. A main factor to consider is the processing speed required to process images in real-time, a necessity for the final use of the robot. Several methods are explored and compared for their usability in real-time applications as well as their accuracy. The vision-processing methods proposed are used to turn two sets of 2D coordinates from both cameras into one set of real-world 3D coordinates. A final method with a higher accuracy than the mathematical model is produced by the end of this report.

Acknowledgements

Mohsen Khadem for his guidance as supervisor throughout the project.
Emile MacKute for her help and support in the data gathering phase of the project.
Bob Fisher for his helpful notes from his class Advanced Vision.
Jane Hillston for her support as personal tutor.
Aristide and Vangelis for the sanity provided.

Table of Contents

Chapter 1

Introduction

Flexible robots are continuously curving manipulators that can imitate the movements of a snake, or other similar invertebrate animals, due to their high number of degrees of freedom [1]. The robot used in this project (see Fig. 1.1) consists of a series of pre-curved tubes that can rotate and move individually while inside each other. This ability provides a wide range of motion that allows the robot to access complex 3D shapes accurately. A major condition for this ability is the awareness of the robot's position. It is necessary to know the robot's present location in order to calculate the next movements needed to arrive to the target destination. In order to accurately track the robot, particularly the end tip of the robot as its end-effector, a vision-processing method is proposed. Using different segmentation approaches it is possible to track the robot throughout frames, and as demonstrated in this project, to do so in real-time. The applications for the robot are explored in Chapter 2, and many require this speed and accuracy.

This project outlines the steps taken to find an accurate way of tracking the robot throughout frames through vision-processing. Firstly, an image acquisition environment was constructed to gather data from the robot, as explored in Chapter 3. The experimental environment simulates images gathered using a 3D cone beam CT scan. Next, several segmentation methods are explored in order to provide an time efficient and accurate way to track the robot throughout its many states, and find the coordinates of its body and end effector. Chapter 4 outlines the design and implementation of the different methods. A comparison is offered in Chapter 5, in terms of speed and accuracy, the main factors affecting the reliability of the robot's use, to gauge the best approach. Finally, a conclusion is made about the completion of objectives and the final contribution to the area of research.

This section outlines the original project objectives, their adaptation due to the project circumstances, and a summary of the final findings. Throughout this report, common terms in the field of vision-processing will be used, a list of these and their meanings can be found in Appendix 7.A.

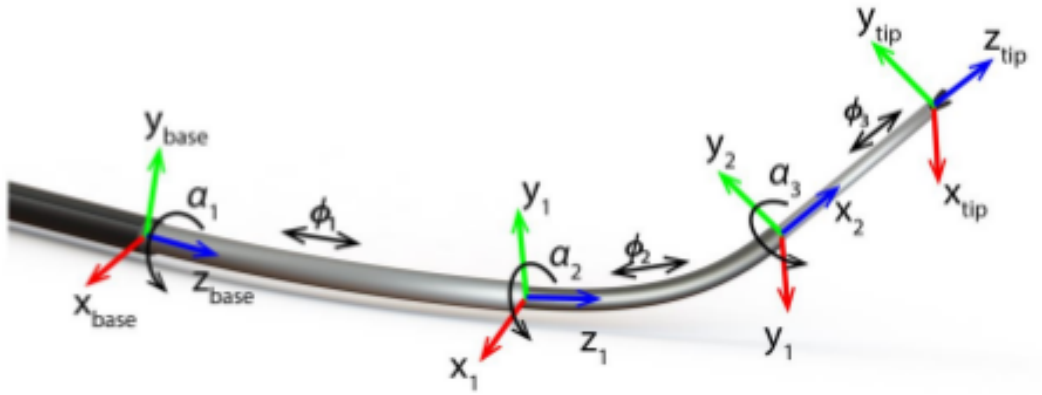


Figure 1.1: A concentric tube robot consisting of three tubes, with the ability to rotate or translate each tube.

1.1 Motivation

The robot analysed in this project is a type of a continuum robot designed for performing minimally invasive needle-based interventions. The specific details of the robot are particularly well-suited for these types of surgeries due to its small and flexible nature. Many physical interventions require the insertion of small instruments in small or hard-to-access places within the body. These instruments are made up of flexible needles for steering along curved paths, but their limitations in terms of steerability and flexibility severely affect their effectiveness.

The robot studied in this dissertation is a concentric tube robot. An example of these robots can be seen in Fig.1.1. The robot consists of three concentric tubes, one inside the other, with varying abilities to rotate and translate each individually. Different robots will have different levels of curvature and stiffness, different materials and number of tubes, but most can achieve complex 3D poses. The difference in these factors dictates the final use of each specific robot. For example, if a rigid tip is necessary to collect a sample, the maneuvering will be decreased. Overall, their ability to navigate complex passages, such as the human body, is a great improvement on the medical instruments used until now.

Lung cancer is one of the most deadly forms of cancer and has a low survival rate of 5% in the UK, a very low figure considering that survival depends on early diagnosis and treatment [2]. Needle-based biopsy through the chest cavity is the preferable diagnostic modality, as it is less invasive than open surgery and more specific than imaging modalities. However, high steerability and small size are requisites to perform this, and so it is currently impossible to access many areas of the lungs and bronchi with typical medical instruments. Concentric tube robots are a great alternative to this type of surgery, as it offers high accuracy to avoid sensitive areas, and could aid in early-stage definitive diagnosis of lung cancer. This concept also extends to neurosurgical procedures [3], and many other uses, as described in Chapter 2. The following section outlines some of these applications, as well as some more information about the

robot's motion.

1.2 The Concentric Tube Robot

There are two main aspects to using concentric tube robots: the kinematics needed to control and direct the robot's movement, and the method used to gain information about the robot's current location and pose.

The first aspect has been achieved by the beginning of this project, through the previous development of a kinematic and dynamic model of the robot for shape estimation. The model uses the information received from the tubes' micro-controllers in order to find its current shape. A similar model can be found in [4]. However, this method suffers from uncertainties about the external forces and mechanical parameters of the robot that can affect the robot's trajectory. The maximum error of the model is about 9% of the robot length, e.g, for a 20 cm robot the accuracy of the model in predicting the robot tip position is about 1.8 cm. Accurate real-time shape estimation is key for the robot's operation, and so a vision-based approach is explored to account for these unforeseeable events. Such methods can often become computationally inefficient, and since performing accurately in real time is a main requirement for medical applications, this project focuses on finding an efficient solution that finds a balance between speed and accuracy. Simple vision-processing techniques are often considered quick, over machine learning methods that require training with data for example, and these are explored in this project. A more complex approach using background subtraction is also explored and compared.

Machine learning methods are sometimes preferred for vision-processing purposes, as while training a model requires a long time, the technical debt is repaid at run-time. These models can perform well at identifying key features, and could eventually predict the full shape of the robot with only a partial image, a key element for the final expected use that could have the robot inside a human body and partially obscured. On the other hand, accurate training data is required for these methods which can be time-consuming and is currently inexistant for this project. However, labelled data, images where the robot is marked present or not, can be achieved through segmentation. Unsupervised methods often require the use of deep learning [5] and is better suited to image classification than localization (for the difference between these please consult the Glossary) and therefore not the most appropriate for this project.

The approaches proposed will use a set of 2D images from two cameras positioned orthogonally around the robot's image acquisition environment. The segmentation of the robot in each pair of corresponding images will provide two sets of 2D coordinates, which will be transformed using calibration data between the cameras, to provide one set of 3D coordinates of the robot's end-tip location. It must be noted that the main contribution to the project is based on the segmentation methods proposed and not the reconstruction of the coordinates acquired from these, which can undergo further optimization in the second installment of this project (MInf Part 2).

1.3 Original Objectives

The original objectives set out for this project are shown below.

1. To accurately track the robot throughout its environment, specifically to correctly localise the end tip of the robot, with the use of two cameras.
2. To maintain the tracking accuracy when using only one camera using machine learning on previous data.
3. To report the accurate coordinates of the robot using only a partial image of the robot from one camera.

However, the latter two were overambitious and unachievable due to delays in the building and set up of the robot. While the expected timeline was to have six months to work on the development of the segmentation of the robot and 3d reconstruction, the robot was only ready for data collection in the sixth month. A new image acquisition environment was also necessary in order to collect data. The design decisions and final results for data gathering are explored in Chapter 3. The objectives have therefore been adapted, and completed, as described in the next section.

1.4 Adapted Objectives

This research was heavily affected by the time available, which is why the project focused on providing an efficient segmentation approach to track the robot throughout frames, rather than predicting the robot shape based on a partial image. Past students working on the project provided accurate segmentation methods, but they were too slow for a real-time application. For comparisons on the speed achieved by the proposed methods and the previous approaches explored, refer to Chapter 5. Improving real-time processing of the robot became the focus of the project, to explore methods that were computationally viable. Since this is only the first installment of the project, there is further discussion on the steps (Chapter 6) that will continue in the second installment, with the aim of achieving the original goals. Below are the revised objectives the project tackled and accomplished. Chapter 6 outlines the final contribution made and how the objectives were achieved.

1. To create the image acquisition environment (Chapter 3) where a data set can be gathered from the robot for testing.
2. To provide accurate segmentation methods that can track the robot throughout frames in real-time (Chapter 4).

1.5 Summary of Findings

This project's focus has been to provide a segmentation method that would accurately track the coordinates of a concentric tube robot throughout different frames in real-time. In order to create this, a data gathering environment was designed and built. This includes design decisions for camera placement, to allow for the robot's motions, and

other factors. Several segmentation approaches were explored, discussed, and analysed in terms of speed and accuracy. A feature tracking method is suggested in order to provide deeper information about the robot's movements. The final segmentation method employs background subtraction techniques in order to detect the robot, and does so in real-time (can process up to 167 frames per second) and to a highly accurate degree (maximum error of 0.003cm for the tip coordinates).

While the objectives have been adapted due to time complications, as can often occur with physical instruments such as the concentric tube robot, the new steps towards accomplishing the original goals are clear and attainable. The following report provides sufficient evidence to deem the objective of providing a quick, accurate method, accomplished. Future research will use the segmentation results outlined in this report to achieve the final goal of predicting the full shape of the concentric tube robot from only a partial image. Future steps are outlined in Chapter 6.

1.6 Software Directory and Structure

In order to recreate or use the code provided, the following information is required.

Since the system created uses a Strategy Design Pattern [6], minimal changes are required to change the approach being used. The file `main.py` holds the high-level methods used, `getImages, processImages` that are used to order the data set and process the images. These steps occur using an object of the Data Engine class. When creating this object, a single parameter is given of `strategy` which can be set as **edge**, when the edge detection method is desired, **back** for the use of the background subtraction method or **final**, for the use of background subtraction with feature tracking. The few lines in the main file are shown in Appendix C.

All segmentation methods are in the appropriately called Segmentation folder. The data set is contained in the data directory. All functions used to calculate end points, apply filters, and for other purposes, are found in the directory `segmentation/office.py`.

Chapter 2

Previous Work

This chapter describes related work from separate research as well as the work previously done by other masters students working on this project. It is meant to demonstrate the variety of applications of the concentric tube robot, and the possible uses for vision-processing tracking algorithms, as well as the approaches that have already been tested.

2.1 Past Students Work

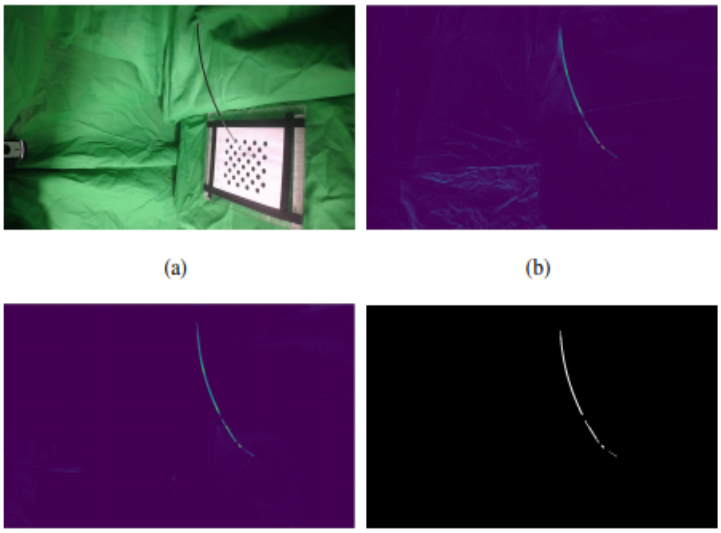
The results achieved by past students provided mostly accurate methods of calculating the robot's body within the image. However, no method was able to run in real-time, thereby making this the primary objective for the project. The segmentation methods presented include pixel-classification of background and foreground pixels, as seen in Fig.2.2, fitting polynomials to the curve found, and even deep learning based on manually-selected training data as seen in Fig.2.1. The images demonstrate the original frame with the robot and the results of segmentation. In the case of Fig.2.1 it can be noticed that the robot is barely visible in the original frame. This is a reason for why a new data-acquisition environment was created, as the old data set was noisy and not well-suited for segmentation, as is explained in Chapter 3. It can also be seen that the prediction contained noisy data and was not necessarily better than simpler edge detection approaches. This serves as an example of a case where more complicated, and therefore more computationally expensive, algorithms are not necessarily more accurate. The choice of methods in this project compares a simpler option to another that uses machine learning to explore this idea further.

2.2 Related work on CTR

The use of continuum robots is an emerging field in robotics and has found many applications in areas such as robotic surgery, for security and defence protocols such as bomb disposal or rescue missions, and in industrial scenarios such as aerospace and nuclear [7]. The robot's highly dexterous body allows for high maneuverability [8] in unstructured or confined environments, explaining the variety of its applications.

(a) Image (b) Label (c) Prediction

Figure 2.1: The results of the deep learning approach demonstrate the predicted results of the shape of the robot based on labelled training data.



(a) (b) (c) (d)

Figure 2.2: The results of pixel-level background subtraction and curve fitting.

Robotics research in this field, such as [9], show huge potential for many applications, a field that is only expanding with the growth of robotic surgery. Many continuum robots, a concentric tube robot is a type of continuum robot, can be enhanced with other sensors that could be used to gather more data, such as tissue collectors (for lung sample retrieval) or electromagnetic sensors that can reveal partial coordinates of the robot. However, vision-based approaches are preferred in terms of generalizability, as less set up is required, and these algorithms can often adapt to obstacles, and other types of unexpected events, better than any kinematic model, as can be seen in [10] and [11]. This is why this project will focus on mostly sensor-less data to make sure the segmentation process is able to generalise well across different situations.

2.3 Related work on Segmentation

Computer vision differs from kinematic shape estimation in that it is not affected by several hardware parameters or unknown factors that lower the accuracy of kinematic models. For example, in [11] a system was developed that could estimate the shape of a robot by extracting features from X-ray fluoroscopic images. Tracking objects and

vision-processing are well paired because vision-processing does not rely on accurate sensor data, for example, to be aware of the robot's body location. The applications of vision based-tracking are varied, such as [12] and [13], as are the methods for doing so. From continuous detection, explored in this project with the edge detection method, to region-of-interest (ROI) tracking, as is explored in [14], the method of choice often depends on the sensors available and the specific context of each project. From electromagnetic to fluoroscopic sensors, to X-Ray images, there are a wide variety of options when designing methods to track and locate robots in images. The biggest challenge is often the process of making these methods computationally efficient.

A solution for this problem can often be found in Convolutional Neural Networks (CNNs), and these are often considered the future of vision-processing [15] as they have have demonstrated highly accurate, and sometimes time-efficient results. Machine learning methods such as this can have lower computational costs at run-time, when compared to traditional techniques, but can require a lot of training time in order to define models and patterns, and needs a lot of accurate labelled data that is often hard to find. This project outlines two types of segmentation methods, one using traditional techniques, and another that uses optimized learning algorithms in order to find the robot in the image, so as to shed more light on the accuracy and speed compromises that are made with the different approaches.

Chapter 3

Data Gathering

Although an existing data set with images containing the robot in different positions already existed, there were several reasons for why a new data set was necessary. The main reason is that the old environment was not repeatable and did not exist any more, meaning a future environment would be necessary to perform experiments with the robot even if the old data was used to create the segmentation methods. The old images also had several problems in terms of segmentation and a new one was deemed necessary for the purposes of faster segmentation. These problems are explored further in the next section. The following sections describe the different approaches considered for the design of the image acquisition environment, such as what materials to use, how to make it repeatable and how the cameras were set up. The method for gathering truth data is also demonstrated at the end of this chapter. For the purposes of demonstrating the data set, consider that pairs of images are of the same robot position but from the two different cameras.

3.1 Decisions for the Environment

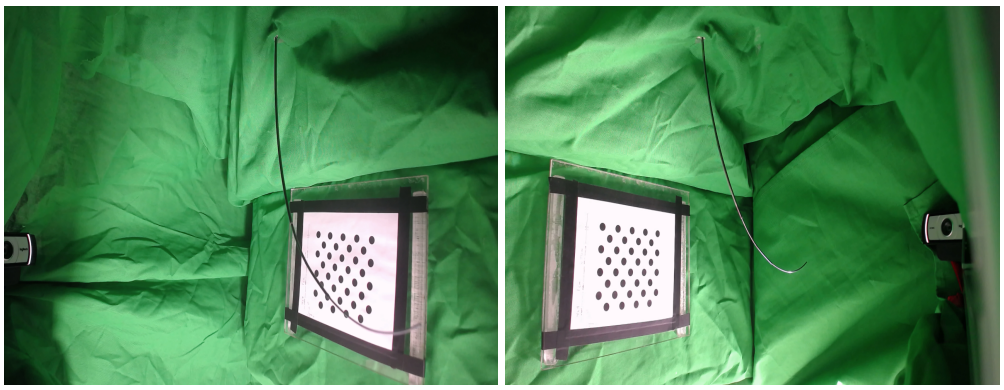
The old data set was deemed inadequate for the reasons described below. These problems can be seen in Fig.3.1.

1. The multiple folds found in the green material used as background for the robot. These create shadows and make identifying the robot, or removing the background, based on colour impossible as the robot and the shadows share a lot of properties.

2. The harsh lines of the robot set up, such as the horizontal square around the grid. These harsh lines are also harder to distinguish from the robot body than a smooth surface would have been.

3. The fact that both cameras have a view of the other camera is problematic as, once again, it makes the separation of the robot through texture or colour impossible.

4. The big black and white grid, used for calibration purposes as explained above, is



(a) Image from old data set of robot as seen from camera one.

(b) Image from old data set of robot as seen from camera two.

Figure 3.1: Old environment with several flaws.

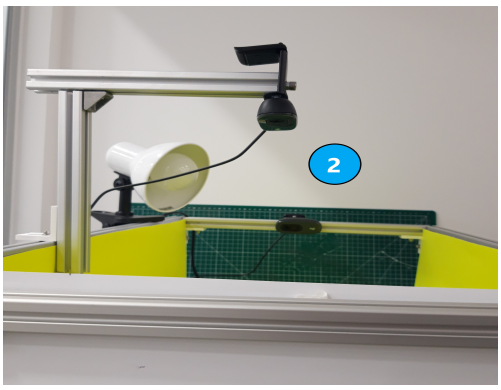
actually unnecessary. While determining the transformation matrix is important for 3D reconstruction, this can be pre-calculated and does not require it to be in every frame of both cameras.

To fix these issues, the final environment was created according to the following principles:

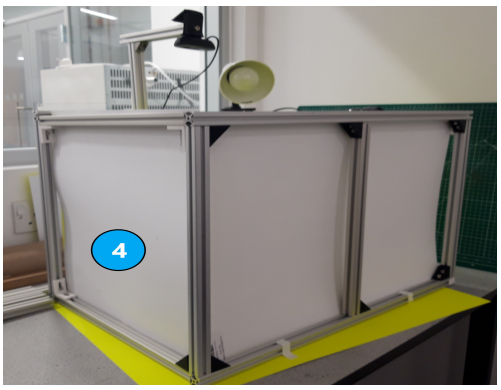
1. For both cameras to have a wide image of the robot's possible positions.
2. For both cameras to share image space for calibration purposes.
3. For both cameras to have an angled view of the robot to avoid extreme cases of robot facing camera directly.
4. For the background to be uniform to ease the segmentation process.
5. For lighting present to be uniform and sparse to avoid flashes and reflections on the robot's surface.

The final environment can be observed in Fig.3.2, with annotations added to aid in the visualisation of the objectives for the space. The image-acquisition environment was built using aluminium rails, two LG 720p cameras and neon yellow fluorescent cards for the background. The choice of aluminium and the two cameras used was due to available resources. This part of the project was designed and physically completed by myself and the other student working on the robot, Emile MacKute.

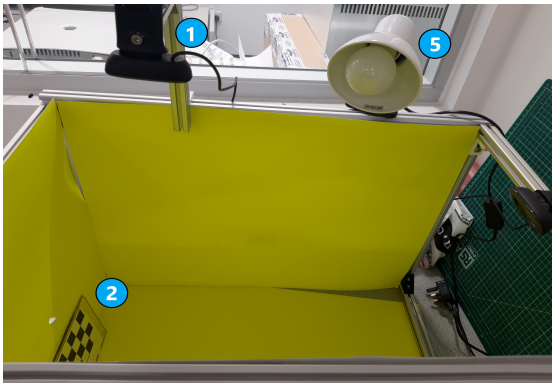
To ensure Objective 1, it was necessary to know the minimum distance between the robot and camera that would provide visibility for the range of the robot's possible positions. Although the robot shape and possible lengths are known, the natural zoom of the cameras also had to be taken into account. The final minimum distance necessary was deemed to be 60cm between the entering point of the robot (where the robot enters the environment) and the camera. To ensure Objectives 2 and 3, different camera angles were tested with different robot positions. The conclusion was that the most effective way to capture the full shape of the robot would be to position one camera observing the robot an an angle from the side of the environment (Objective 3.) The



(a) Lamp, top and angled camera placement in the image-acquisition environment.



(b) The environment as seen from the outside, where you can see the aluminium rails and the back of the neon board used as background.



(c) Looking inside the environment, both cameras, the lamp, and the grid for calibration are visible.

Figure 3.2: The new environment created labelled with what objectives each element helped accomplish.

second camera was positioned directly above the robot to ensure the robot shape was clearly visible even when the robot was pointing at the first camera (and therefore obstructing its own body.)

The shared space between the two cameras was necessary to calibrate both cameras in regards to each other. The method chosen to do this was to use a square grid of alternating black and white squares. By calculating the coordinates of the same object from both cameras, tools from Matlab's Stereo Toolbox [16] are able to calculate the transformation matrix required to transform coordinates from both cameras' 2D environment, to the real-world 3D coordinate space. The ability to calibrate the cameras in regards to each other is paramount for 3D reconstruction in order to account for camera parameters as well as the transformation between the two different coordinate spaces.

In order to achieve objective number four, a sufficiently distinct background was needed. Past students working on this project opted for a big green cloth, as pictured in Fig.3.1, but as mentioned, this option added shadows and other noise to the image, a disadvantage for the segmentation process. It is important to remember that although segmentation should not require a complete lack of noise to perform well, designing the

image-acquisition space with these factors in mind can facilitate the goal of real-time processing.

The final decision for the environment background was neon yellow hard card. The colour was expected to be contrasting enough to be easily separated from the robot and other reflections. Interestingly however, the colour of the card on camera reads more as white than the bright yellow/green neon of the card. This could be due to the cameras' colour perceptions, the visualising software, or other factors to do with the cameras' hardware. Fortunately this did not present a problem when segmenting the robot, aided by the lack of extra shadows in the card. Alternatives to this method would include using other types of fabric, but there is no solution which provided a background while hiding the aluminium rails, which would make segmentation of the robot harder, apart from the one used.

In order to complete the desired objective of uniform lighting, objective five, the goal was to purchase several LEDs that would be placed along the top of the environment 'box'. These would have been diffused with cloth in front of them if necessary. However, due to time and resource complications, an available lamp was used. While the lamp does an adequate job of lighting the space, it also creates several reflections on the robot body throughout frames, but mostly on the frames from the top camera and not the one placed on the side of the environment.

The final images obtained from these camera angles and within the environment created are very adequate for segmentation purposes, excluding the reflection the lamp sometimes creates on the robot body. However, it must be mentioned that if the robot changed to a larger length, the existing environment would likely not encompass enough space to allow for all of the robot's possible locations.

The next section contains examples of corresponding top and angled camera frames of the robot in the same position. It can be seen that the robot is in great contrast to its background, and that there are very few harsh lines, without folds or shadows, making the new data set of images much more suited for segmentation.

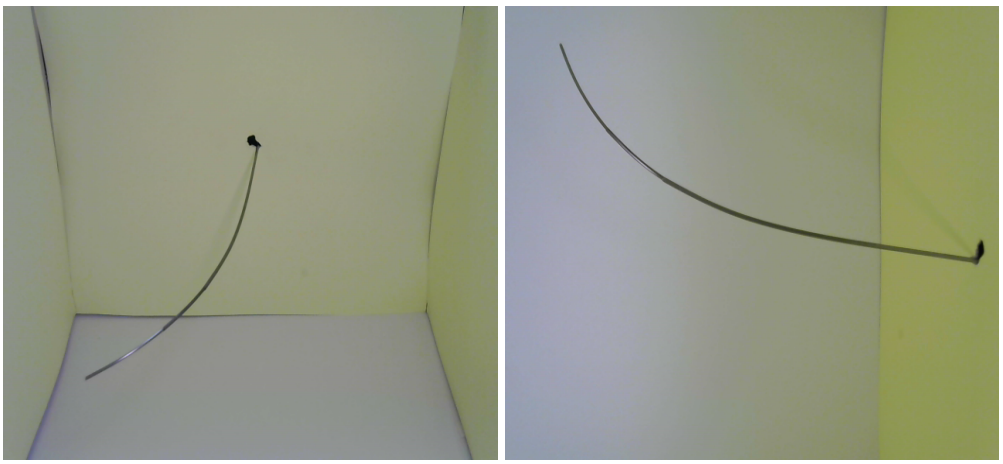
3.2 Final Data

3.2.1 Images

Images of the robot in its environment were taken, with different rotation and length settings in order to account for most possible positions of the robot in the space. This variety of positions is particularly important to ensure the tracking methods described apply to most, if not all, possible robot shapes. An example of corresponding images of the same robot configuration, as seen from the different cameras, is shown in Fig.3.3. More data can be found in the project's directory under 'data'.

3.2.2 Truth Data

An electromagnetic sensor was used in order to gain truth data for the coordinates of the robot. The sensor works in partnership with an electromagnetic board below it to



(a) Home position of robot as seen from camera 1,above and at an angle from the robot.

(b) Home position of robot as seen from camera 2, which is placed directly above the robot.

Figure 3.3: Corresponding images of robot in home position from both cameras.

report the 3D coordinates of its tip at any given point. When taking said images, with different rotation and length settings, the sensor was placed on the tip of the robot, as well as through its body. The objective for this data was to be used as the true coordinates of the robot at the given time, and compared to the results obtained from the 3D reconstruction of the resulting coordinates from the segmentation process.

Several difficulties were encountered with the sensor due to several factors. The sensor data returned coordinates of the tip of the sensor several times per second, and these readings were severely affected by shaking of hands. The resulting data is very noisy and includes many outliers that are very hard to avoid, as the sensor is placed by human hands and susceptible to shaking. Approaches where the sensor was held in place by another object, not humans, were unsuccessful as it was impossible to place the sensor as wanted while maintaining it still. Therefore, due to the large number of samples taken, and the inaccuracy of these, the EMT data needs further processing to be used as an error measure.

The alternative is the following. Using an algorithm provided by the project's supervisor, as can be seen in Appendix 7.C, which employs Matlab functions such as **triangulate** [17] in order to return a set of 3D coordinates from the provided 2D coordinates. The function uses camera parameters such as intrinsic, extrinsic and lens distortion, which can be obtained using other Matlab functions from the Camera Calibration toolbox [16].

This algorithm is in charge of taking corresponding points of the tip and the base of the robot from both cameras, and returning the calculated coordinates (using known information about the cameras' positioning in the real world). The process of gathering truth data consisted of using this algorithm with manually selected coordinates of the robot's tip and base. This option was somewhat error-prone. The process of manually selecting coordinates was repeated several times in order to see the variation in the coordinates chosen, which have an effect on the resulting 3D coordinates. A difference of 1% (for example if the x coordinate was 600, the range of the values for that

coordinate would be +/- 6) was noticed in the coordinates taken for the same point, a result of human error. However, this error was mostly cancelled by taking the average of said readings.

The resulting data is therefore a set of 3D coordinates for the tip of the robot in different images. The error measures in Chapter 5 are calculated by finding the difference between the calculated coordinates and the true coordinates. It must be taken into account that while this gathering method is liable to human error, the final intended method of gathering truth data is to use an EMT sensor.

Chapter 4

Methodology

The following chapter outlines the steps taken to segment the robot and reconstruct the 3D shape of the robot. The reconstruction will focus on the coordinates of the end tip of the robot, i.e, the robot end-effector. It is important to know the end tip coordinates to track the robot successfully. Two segmentation methods are investigated: one that uses edge detection and typical vision-processing techniques, and a second one that uses frame history to create a model of the background, in order to define the robot as the foreground of an image. Next, feature tracking is explored as a potential addition to segmentation that could reveal more pose information about the robot. Finally, the process of selecting corresponding 2D coordinates for 3D reconstruction is explored.

4.1 Segmentation

The results of the following sections were produced with the aim of accurately defining the exact coordinates in which the robot body is present within the image. Images will often be demonstrated with the original frame on the left, and the results of the segmentation approach being tested on the right. A good segmentation result would be to detect the whole body shape, including the beginning and the end of the robot. As mentioned previously, a recurring problem is the presence of reflection on the robot body, resulting in fragmented portions of the body when segmentation is performed. The next sections explain how this issue was averted within the segmentation approach, as well as the details of the implementation's logic.

4.1.1 Edge Detection

The goal of the first method was to provide a baseline for accuracy when using common vision-processing methods such as edge detection, noise reduction, and filtering. The perfect method is able to create a detailed mask (see glossary in Appendix 7.A for more information on these terms) of the robot's position in the image. Example images from the data set are demonstrated in Chapter 3. In order to design an algorithm to define the robot the following factors must be accounted for:

1. The dark lines in the background as they are immediately picked up by any edge-detection algorithm.
2. The wide gap in the opening of the environment, where the robot enters the image acquisition space, is particularly deformed and big. This is due to the brittle nature of the robot, meaning more area was given to avoid contact with surfaces. This circle will likely distort segmentation and may make the beginning of the robot a less defined shape.
3. Due to reflections from the lamp, the robot body is often fragmented. This means the program has to account for the robot being in different fragments and must join these when necessary, in order to attain the full robot shape.

In order to remove noise and detect the robot accurately, the following steps were taken. Firstly, the image is blurred using a Gaussian filter with a (5,5) kernel. The purpose of this step is to minimise the presence of small details in the image and leave only big, marked, shapes such as the robot. This will make it easier for the edge-detection algorithm to find the main bodies in the image and exclude smaller details. After converting the image to grayscale, reducing an image with three colour channels to one intensity channel, a personal function is applied. The transformation of the image to a single channel is often a requirement for edge detection algorithms as a single intensity value is necessary to calculate gradients in the image. The function created finds the optimal parameters to be used for the Canny filter provided by the OpenCV-Python library [18]. The Canny filter is a very useful tool for edge detection. It applies several processes on the image in order to detect all the edges present, such as applying different filters and thresholds to identify edges. However, steps can be performed outside of the Canny filter in order to optimize this, such as applying a smoothing filter before edge detection. The different results of using the Canny filter with pre-processing and without can be seen in Fig.4.1 and Fig.4.2. More about the Canny filter and the personalised function follows.

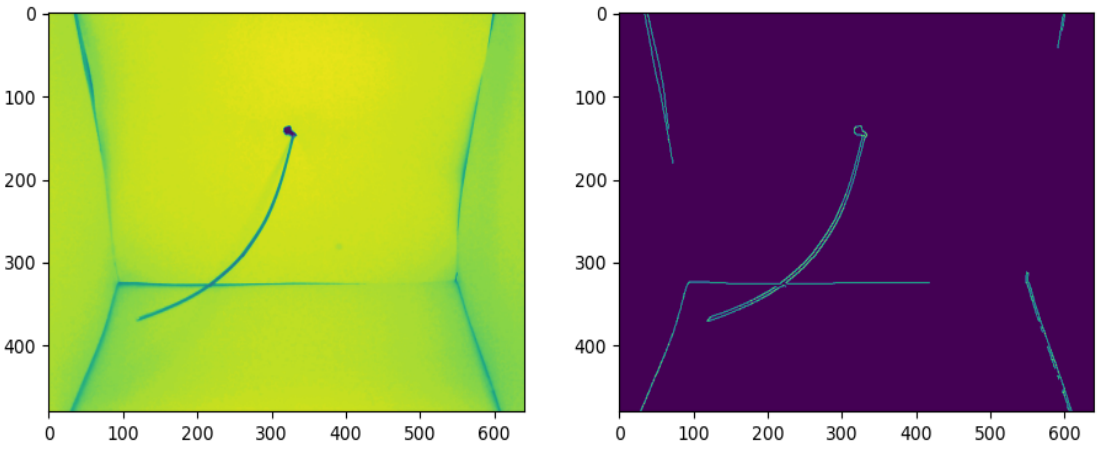


Figure 4.1: On the left is the original frame, and on the right the processed image. When the Gaussian filter is not applied before canny, there is more noise left in the image such as the edges of the environment.

For a pictorial representation of the Canny filter's process, see Fig.4.3. Firstly, it

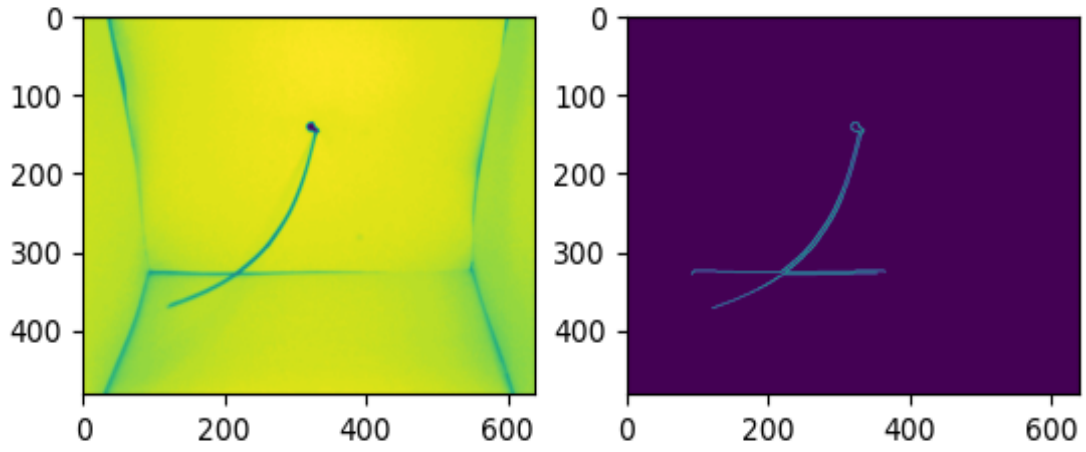


Figure 4.2: On the left is the original frame, and on the right the processed image. When the Gaussian filter is applied before the canny filter, only the robot and other unavoidable noise is left behind.

smoothes the image again, using another Gaussian filter to remove high-frequency noise. Next, it computes the gradient intensity for the image, using four filters to detect horizontal, vertical and diagonal edges in the blurred image (the reason using the grayscale version of the image). An example of these filters can be seen in Fig.4.3, in the step where the derivative is found. In order to detect horizontal lines, the matrix on the right, or G(y), is applied, the left matrix is used to detect vertical lines, and so on. The edge detection operator returns a value of the first derivative in the horizontal and vertical direction. From this edge, gradient and direction can be determined. Particular gradient and direction patterns are used to identify where the edges are in the image. Finally, the canny filter then ensures the most important edges are chosen by applying an edge thinning technique called 'non-maximum suppression. This step is applied to each pixel in the gradient image, where pixels are either suppressed or kept according to their 'edge strength' in comparison to nearby pixels.

Once these steps are completed, the remaining pixels provide a somewhat accurate representation of the strongest edges present in the original image. To account for noise and color variation, a binary threshold is applied in order to remove pixels with a weak gradient value. All pixels are therefore null or white at the end of this process. However, because this step can have a side effect of removing wanted pixels from edges, a final step of applying blob analysis to each pixel is performed. This step looks at the 8-pixel neighbourhood of any given pixel and checks whether a strong edge point is present - if so, the weak pixel is considered as part of another edge and kept.

A downside of the Canny filter, is the lack of optimal tuning of the parameters. Two thresholds are given to the Canny filter, describing the lower and upper thresholds to be applied in the last step. This parameter sets the degree of sensitivity to noisy edges. However, because each image is different, depending on light shifts that can be created by the lamp or by shadows, a function was created to find the optimal parameters for a given image. This function is responsible for finding the optimal parameters and apply the Canny filter.

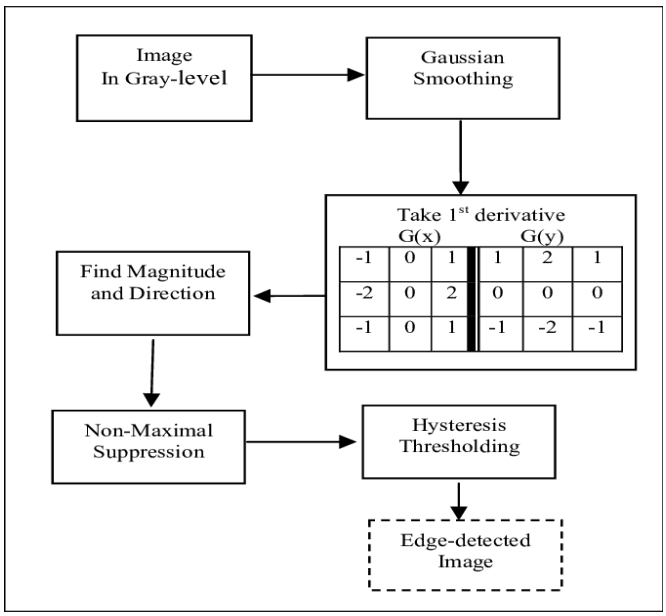


Figure 4.3: The processing of each image by the Canny filter [19].

The function takes an argument sigma that can be used to vary the percentage of thresholds the filter should use. In this case, the final sigma value was chosen through experimentation for the best edge detection results. This means finding a compromise between detecting the whole body, including the tip, while also including other objects in the image, and a more accurate segmentation that misses out some parts of the robot body. The function begins by finding the median value of the image, in grayscale (otherwise there would be three median values, for each colour channel). The median is used, with sigma as a percentage parameter, to construct the lower and upper thresholds for the canny filter. A low value of sigma would provide a tighter threshold, where the median has more weight, while a higher value gives a wider threshold. A balance between both options was found by fine tuning sigma.

The different results after applying the canny with different sigma values are shown in Figures 4.4 and 4.5.

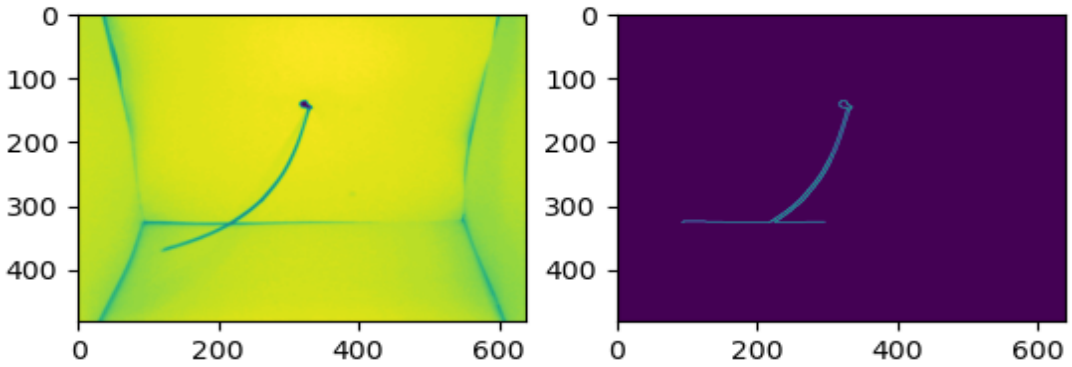


Figure 4.4: The robot tip is included in the segmentation when a higher sigma value is used.

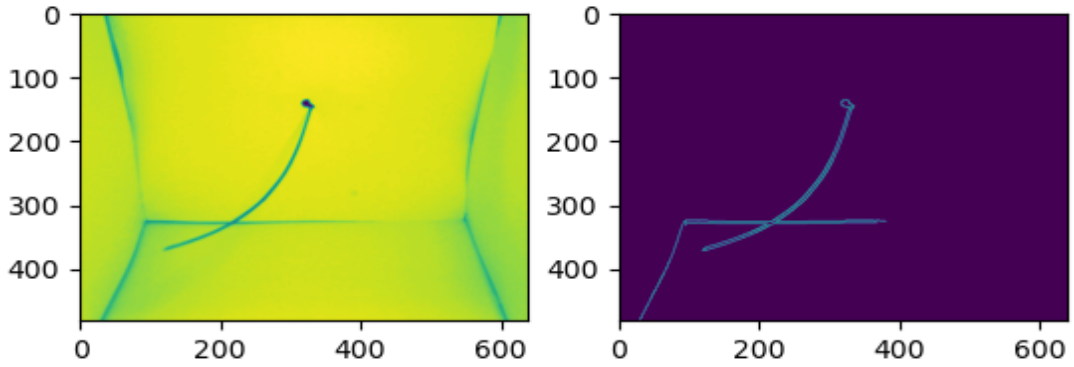


Figure 4.5: The same frame without the tip segmented as a tight threshold (low sigma) was used.

Once the canny filter is applied, more morphological operations are performed on the frame in order to widen the wanted robot shape and minimise the presence of other elements, such as lines or leftover pixels. Once the robot is segmented, it is time to check and account for reflections on the robot body. As can be seen in Fig.4.6, the robot was divided by the reflection on the robot. If this is the case, the length of the main body would not match the real length, and so the program checks for nearby shapes that are big enough to be another part of the robot body. The function created for this purpose considers the distance and size of other shapes found in the image, excluding the main body, to find the biggest shape and the closest valid shape (valid in terms of area). It also removes small shapes that are not part of the main robot body.

If this step is necessary, the two robot parts must then be joined. This step is performed by calculating the extreme points of each shape, meaning the top, left, right and bottom coordinates. Taking the extreme coordinates of both shapes allows us to create a line joining these points, and effectively the robot. The main robot body is then joined with the remaining body and the line connecting them, providing the final robot body that will be used as a mask in the next step. The steps for this process can be seen in Fig.4.7. The mask shown is the final product of this method. Discussion on its accuracy and speed can be found in Chapter 5.

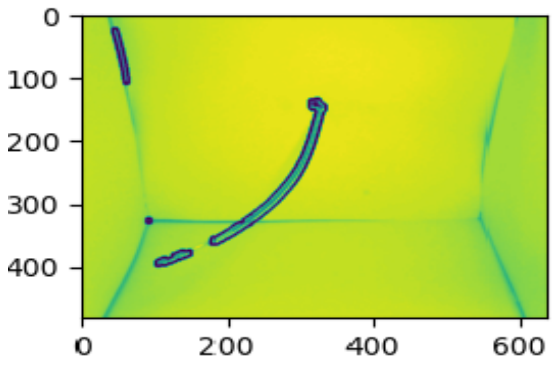


Figure 4.6: After applying the canny filter and finding the shapes, the robot is fragmented due to reflection.

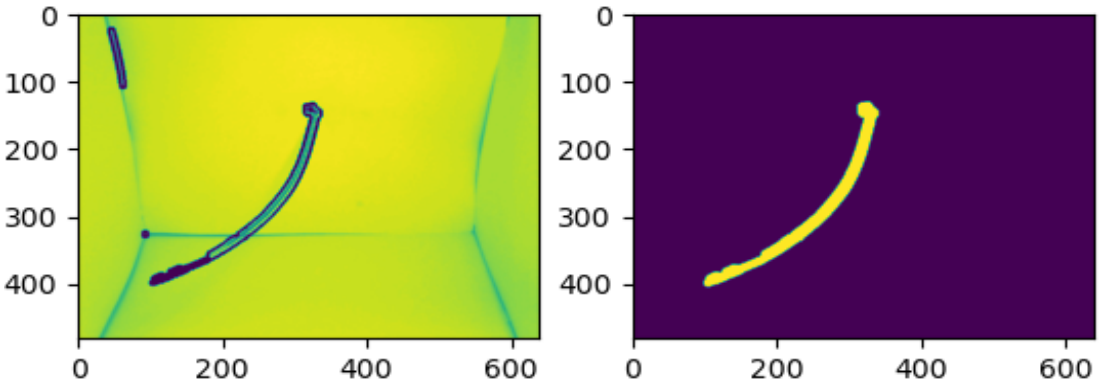


Figure 4.7: The left image shows the shapes including the line joining the wanted contours, and the right shows the final chosen contour of the robot body.

4.1.2 Background Subtraction

The second approach proposed has many benefits over the edge detection method. The main reason for its efficacy and accuracy is due to a recently created background subtraction method [20] called Background Subtraction CNT (sometimes shortened to CNT). Background subtraction is a basic operation used in computer vision in order to eliminate background pixels and leave the foreground behind, in this case the robot body. This process can often be computationally expensive as past students demonstrated (for processing-time comparisons, see Chapter 5, section 5.2). A past student created a machine learning classifier that was trained on background images and learnt the pixel value distribution. This classifier was later applied to every pixel in order to classify it as background or foreground. Although it was mostly accurate, there is evidence that the algorithm often did not find the end tip point accurately.

The increased efficiency of the subtraction method offered in this project is due to several reasons [21]. Firstly, the logic applied to each frame is sound. Secondly, it uses the optimization framework Valgrind [22], which optimizes memory management and threading. It is used to build dynamic analysis tools without the common downside of increased computational cost at run-time. Secondly, this background subtraction method is faster than other background subtraction approaches provided by the Python vision-processing library OpenCV [18]. The method documentation was recently added to an unofficial version of the OpenCV library and the results in this project demonstrate its extensive potential applications. A comparison between the accuracy of this method and other background subtraction methods (background subtractor MOG and MOG2 [23]) offered by OpenCV is demonstrated in Fig.4.8, with a detailed description of the accuracy of the final approach available in Chapter 5. As can be seen in the image, while the CNT approach captures the beginning and the end of the robot, the other two methods only show half of the robot body.

The run-time of this method is over twice as fast as other background subtraction methods, particularly when using cheaper hardware with a weak processor. This can be seen in Chapter 5, section 5.2. This ensures the real-time application of the algorithm is possible, a key factor in the success of the project. The accuracy of this method however, depends on tuning certain parameters such as definitions for pixel stability and how the

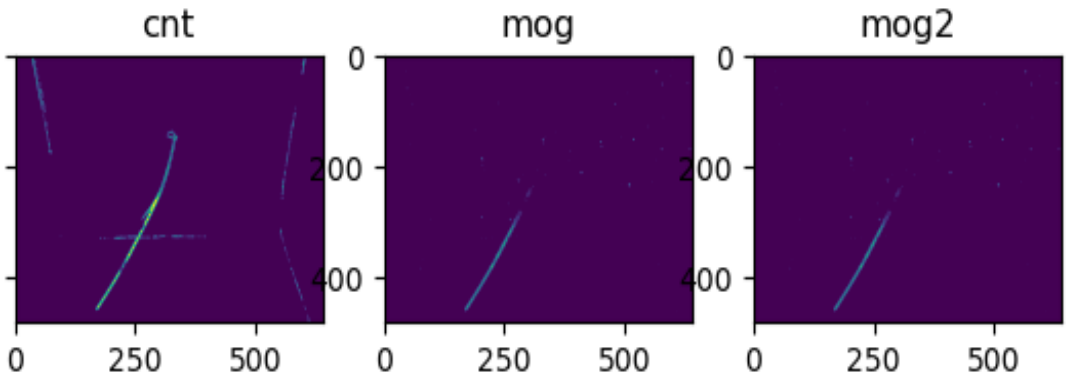


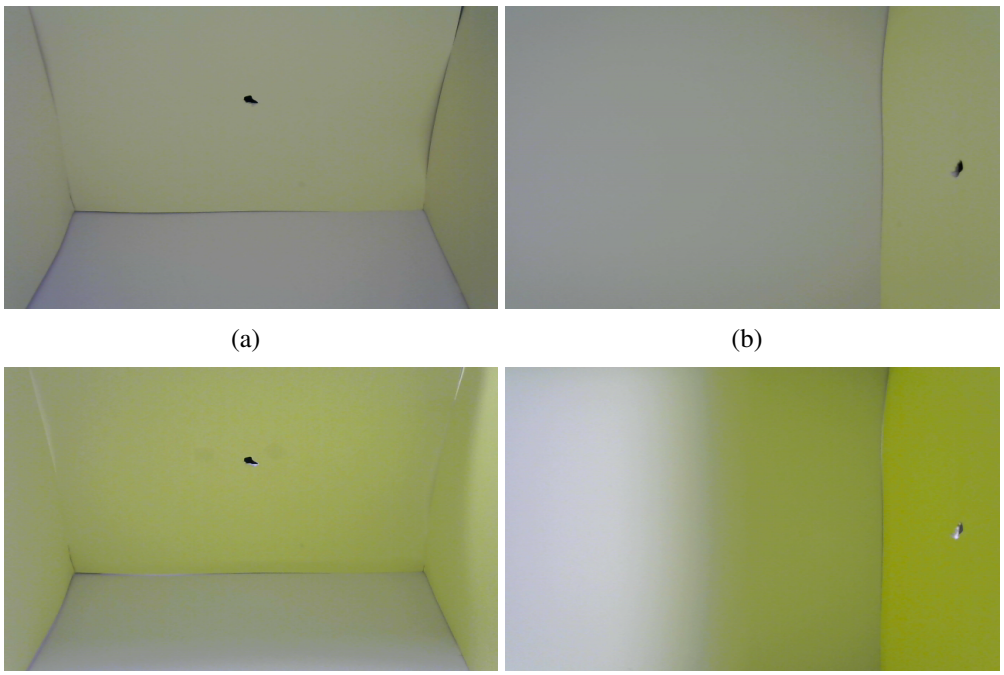
Figure 4.8: A demonstration of the CNT algorithm's higher accuracy. The original image is not shown for visibility, but the CNT method clearly defines the full robot shape while the others do not.

learning of each frame should progress. These parameters change the way the background is perceived. For example, the parameter of **minPixelStability** defines how long a pixel must be stationary before being marked as background. This is supposed to represent the way humans naturally differentiate background and foreground, as we wait some time, perhaps only one second, to consider an item as background. This value was set very low in this project due to several reasons.

Firstly, each pair of consecutive frames is not a smooth transition of robot states. This means the frames are not a video and there are no intermediate positions between them. This means the variance of where and what the background pixels are between images is large, so a rapidly adapting and assuming algorithm is necessary. Therefore the variable mentioned was set very low. Another variable to account for is the use of history or not. When this setting is on, previous frames applied are considered in order to determine background pixels. However, this history can be limited and shortened, which can also increase the adaptability of the algorithm to the consecutive frames being very different, as it does not need a lot of proof to mark a pixel as background. While this method would be faster if parallel threads were available, the available hardware is described in Appendix 7.B, but this must be considered for further optimization in the future.

The specifics of the order of steps performed on each frame is not publicly available. However, it can be inferred that the algorithm carries out a version of other background subtraction methods [24] that have been optimized. Fig.4.10 demonstrates the common processing pipeline.

The CNT algorithm is applied is as follows. First, it is applied to pictures taken of the background (of the vision environment) without the robot, so as to provide a good representation of the different states of the background . These background images are all different in terms of what lighting was present and from what camera the image was taken, in order to make the algorithm as resilient as possible to variance in the background that naturally occurs throughout frames. Some of the background images can be seen in Fig.4.9. Once the CNT object is created and applied to the existing background images, it is systematically applied to all the frames present. Other opera-



(a) (b)

(c) (d)

Figure 4.9: Different lighting options present when taking background images to account for possible variance and increase the generalizability of the algorithm.

tions are then used, such as finding shapes in an image with OpenCV's **findContours**, in order to remove leftover shapes, as well as to see the state of the algorithm.

The CNT algorithm requires the application of several frames before it is able to provide the accurate location of the robot. An example of the first result when applying CNT can be seen in Fig.4.11. The reason for the leftover robot shapes seen in the image is the use of frame-differencing to separate the background and the foreground, which has not stabilised yet. This means it has not been able to define the background model yet. This is also present in the second image, which directly corresponds to the value passed for the parameter of **minPixelStability.** Since this variable is set to two, due to the mentioned need of quickly adapting the background model, two frames must be passed before the algorithm can correctly segment the robot. This is not considered a hindrance as two frames of a video correspond to less than a second passed. The applications of this algorithm in real-time are explored in chapter 5.

To account for this occurrence, a limit is placed on the number and size of the shapes that can be present in the image. To eliminate the lines leftover, as seen in Fig.4.11, only shapes with an area bigger than the set threshold are accepted. This often leaves the robot only, but in the case of the first few frames when it is possible to have multiple shapes, a further check is performed on how many valid shapes are found. If more than 3 are found, the result of applying the CNT is deemed unusable due to the lack of a unique robot body. The reason for 3 shapes being acceptable is due to the fact that the overall robot body is often made up of several shapes as defined by the **findContours** function. Later on these shapes are joined in order to find out the necessary coordinates,

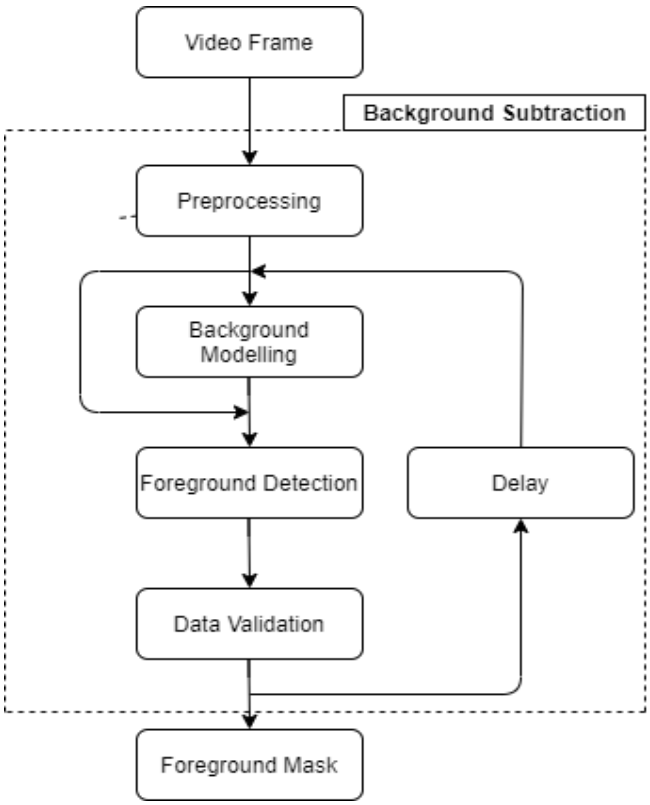


Figure 4.10: The common steps of Background Subtraction algorithms as described in [24] (image re-created for visibility).

as described in section 4.3.2. While this only affects the first two frames, the check is placed in order to minimise error possibilities.

The process of applying the CNT subtractor, cleaning up the leftover shapes, and filtering for the early erroneous results, is repeated for all frames in the data set, to provide a complete outline of the robot in every frame. The accuracy of measuring the tip is discussed in Chapter 5, including the results in edge cases. Results of applying the CNT algorithm can be seen in Fig.4.12.

4.2 Key Feature Tracking with ORB

Feature tracking is a method often used in 3D reconstructions to identify gradient extrema points in images (effectively edge points), and then compare them across frames using local point descriptors, in order to identify the same point in different images. There are different algorithms that perform this, such as SURF, SIFT, KAZE, and ORB. More on the differences between these and their intended applications can be found in [25]. Each method differs in the following aspects:

1. The criteria for finding key features.
2. The description of the neighbourhood of each feature point for future comparison.

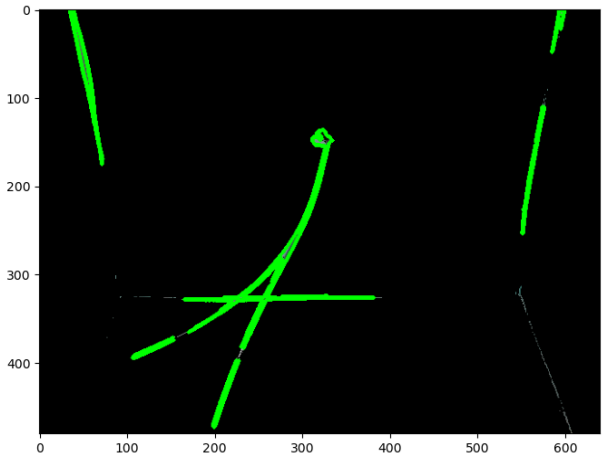


Figure 4.11: The filtered image, after applying CNT, contains many robot shapes and leftover noise. This is the case for the first two frames until the background model is set.

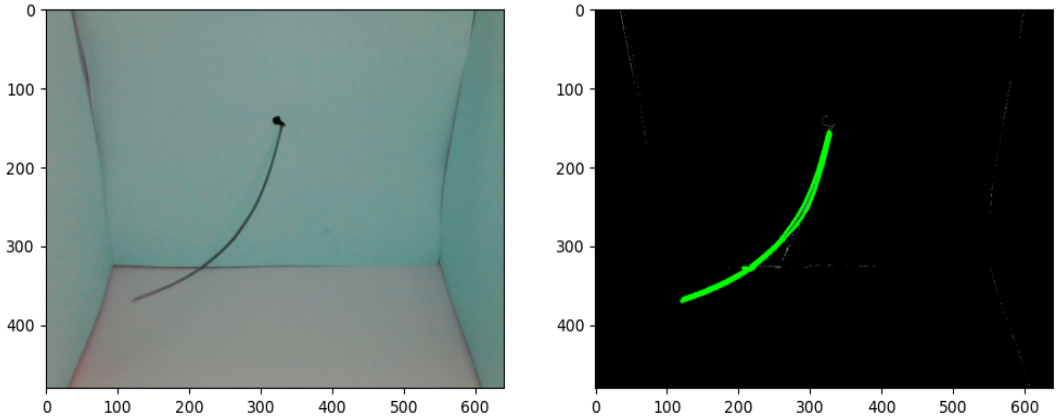


Figure 4.12: The filtered image, after applying CNT and removing invalid shape, contains the full robot shape including the tip.

3. The measure used to validate and choose a matched pair.

4. Whether the approach is resilient to orientation and scale changes.

In this case ORB (Oriented FAST and Rotated BRIEF) is used due to its speed advantage over other feature-finding methods, as can be seen in [26]. ORB uses FAST [27] as a key point detector, and uses BRIEF [28] with some modifications, outlined in the linked pages, to produce the descriptors (since the details of these methods are quite complex and verbose, these will not be described in depth as it is not original work). It then uses a Harris corner measure to find the better-matched points. A great benefit of this method is that it is rotation invariant, meaning it can identify corresponding points even if the image has rotated, an event that is expected to occur from frame to frame in the case of this robot. Finally, the features found are matched and validated using multi-probe LSH [29].

With the robot segmented, we can apply the feature tracking algorithm to find main features and track them across frames. However, another problem remains. Due to the exaggerated opening of the robot entry, many matched feature points were found around this opening, as can be seen in Fig.4.13. This is expected as consecutive frames in the data set are not part of a video sequence and quite different, in terms of length and rotation of the individual tubes, as explained below. This means that the most reliable points to match are around the robot opening due to their high similarity, and therefore those are selected. To avoid this, a filtering process was carried out on the features found, by removing key points from consideration if they were around the top of the robot. Since the opening coordinates do not change, this is an easy step to include. The leftover points are matched if valid, and the top 20 matches, those with the highest score on the Harris metric used, are tracked and displayed.

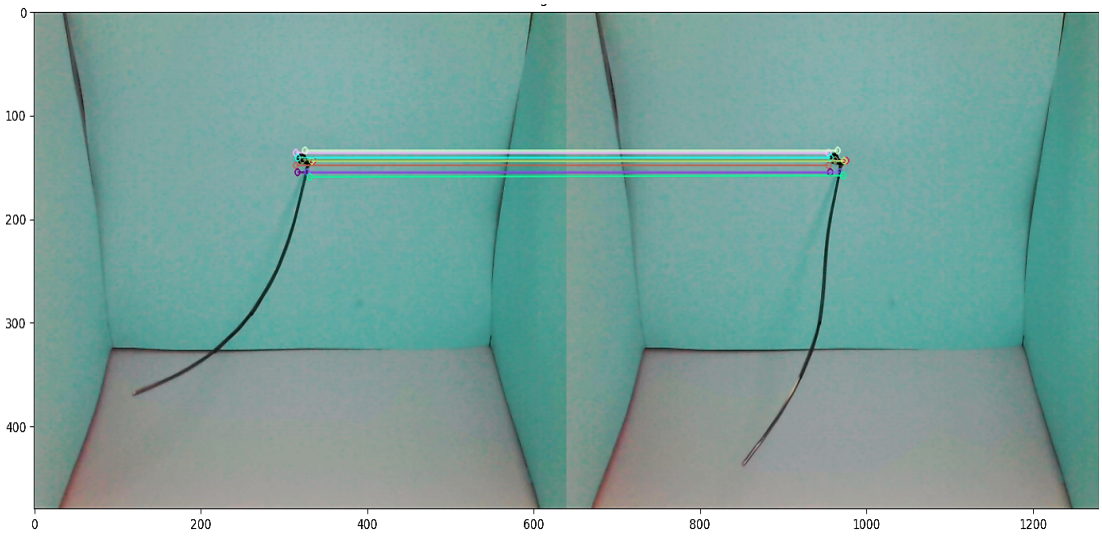


Figure 4.13: The image shows consecutive frames and the points matched between them. The features matched were all around the opening before filtering took place.

Something to take into account when observing these results is that consecutive frames in this data set do not correspond to similar configurations of rotation and lengths for the different robot segments. This is because the images were taken to collect data for several different robot configurations, rather than a video demonstrating gradual change between positions. Due to availability problems, I was unable to record a video of the robot moving between configurations. This information is relevant because it means features are being matched between very different frames, rather than consecutive frames that would allow for much more accurate matching as it is more likely to receive a good match when the feature points have not changed very dramatically, as is the case now. On the other hand, the results demonstrate feature points being matched consistently and quite accurately between some frames. Using a video in the future would provide a better way of estimating the information gain from feature tracking. If matched points could be found to be corresponding with the robot's tubes for example, this would be very useful information, particularly when training machine learning classifiers to estimate 3D shape based on less information than currently used, such as only one camera.

4.3 3D Reconstruction

Once the robot body is segmented and its coordinates are known, it is time to put both sets of coordinates, from the two different cameras, on the same plane. This step is necessary in order to achieve the 3D coordinates of the robot. As mentioned in Chapter 3, the data used as truth for this project is manually selected coordinates. Matlab functions then apply triangulation and camera parameters for both cameras to produce one set of 3D coordinates. The goal is to do same process but using coordinates achieved from segmentation rather than manually selected locations. For the triangulation to work, both sets of coordinates attained from both cameras must refer to the same point even if from different perspectives. The process of selecting this point is described below.

4.3.1 Point Correspondence

As mentioned above, finding corresponding points is a critical step to accomplish accurate 3D coordinates. Fig.4.14 demonstrates the different dimensions of the coordinates, with segmentation results being two-dimensional and the desired result being three-dimensional.

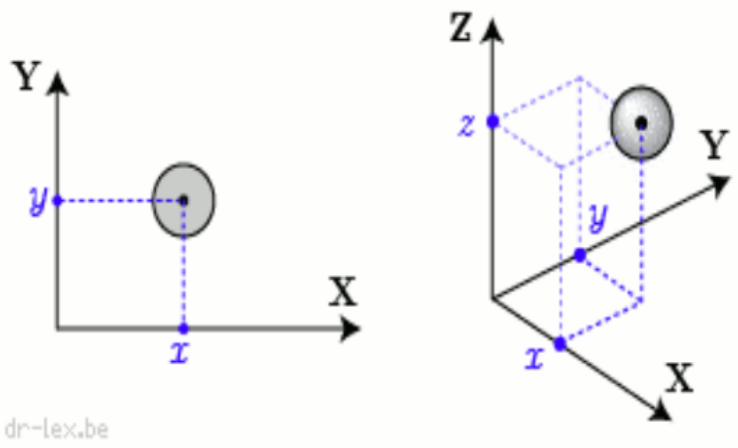


Figure 4.14: The left shows a two-dimensional coordinate system while the left shows a 3D coordinate system. Since we want the real-world coordinates, we will use the 2D coordinates captured in the images to reconstruct the 3D equivalent. Image as seen in [30].

Because the images are from different perspectives, the coordinates obtained from one camera are not going to correspond to the same point on the second camera. This is shown pictorially in Fig.4.15. That is why a transformation matrix is necessary, to map the coordinates as obtained from one camera, onto the same plane, or perspective, as the second camera. However, to find the transformation between two coordinates, it must be ensured that both coordinates refer to the exact same point, or the transformation will be erroneous.

Since the goal of the project is to know exactly where the robot is, the best representation of the robot's location are the coordinates of its end tip. As discussed in the last

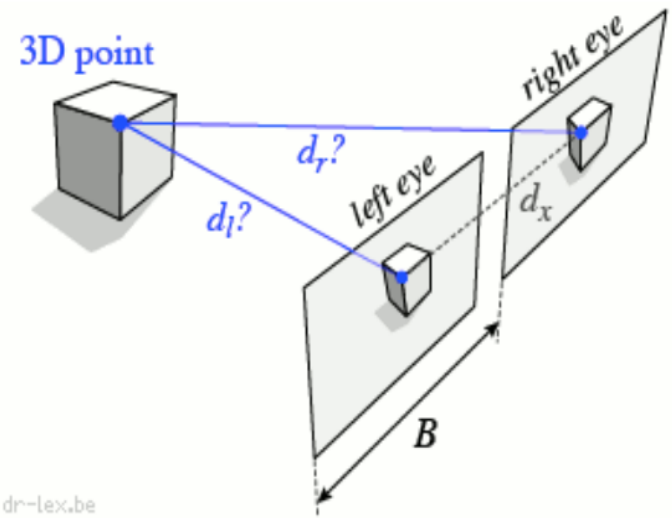


Figure 4.15: An example of how the perspective of the beholder, the camera or the human eye, changes the coordinates of the same 2D point, as seen in [30].

section, the segmentation of the robot is successful at finding the tip. However, it is harder to find the exact end as the robot shape will be made up of other shapes, where the most north and south coordinates are not necessarily the end and beginning of the robot. An example of a robot position for which this logic would fail can be seen in Fig.4.16. This is why a more complex solution was created in order to find the most accurate end-tip coordinates. For credit purposes, let it be known that all the functions described in this section were created and designed by the author unless specified.

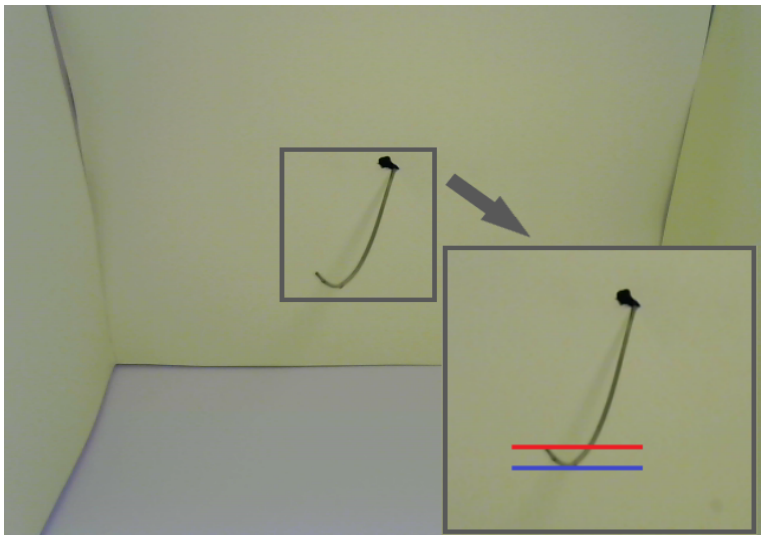


Figure 4.16: An example of a robot position where the most south coordinates do not correspond to the robot end. The red line represents the y value of the end coordinates, while the blue demonstrates what coordinates would have been picked if following the erroneous logic.

Firstly, I created a function that calculates the most north, south, west and east coordinates for all the detected shapes that make up the robot body. These coordinates will be

referred to as **extrema** for abbreviation purposes. These coordinates are then given to another function that calculates the extrema coordinates of all the shapes combined in order to account for the whole shape. These coordinates are then joined if close enough to each other. When running experiments, it was often the case that some coordinates were very close to each other. It therefore made sense to merge these coordinate pairs in order to describe a region. This is important because when finding the tip coordinates, probabilities are provided for each point. If two points are too close together, they will have competing probabilities. If the two coordinates are the most likely to be the tip of the robot, we gain more information by merging them than by choosing one or the other. An example of a situation where the west and southern points are on the tip of the robot can be seen in Fig.4.17, demonstrating the reasoning for merging these points in order to acquire the most accurate end coordinates possible.

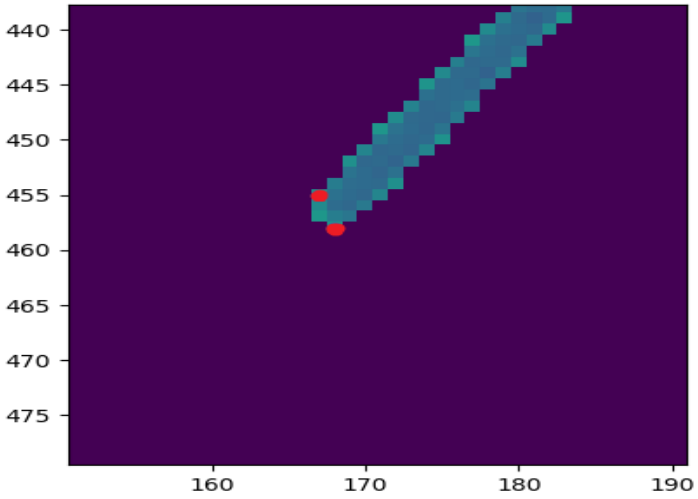


Figure 4.17: Example of situation where the most west and south coordinates of robot shape are on the end of the robot. The extrema coordinates, marked in red, should be merged rather than picking one or the other.

Once the extrema coordinates have been calculated and filtered, it is time to calculate the probability that these points are the robot end that we are looking for. In order to do so, the following logic was assumed. Firstly, it must be taken into account that the image at this point is a simple binary mask, meaning each pixel value is either 0 or 1. This means that the robot body has values of 1 and the background of 0. So if the point being examined is surrounded by many positive pixels, it means it is likely part of the main robot body. However, if the ratio of positive pixels to null pixels is low, this would mean the point is near one of the robot edges. Since the coordinates of the beginning of the robot are fixed, at its entering point, it is not necessary to account for these coordinates. Therefore, if a low ratio is found, knowing it cannot be the beginning of the robot, it must be the end.

For this experiment, a neighbourhood of 40x40 pixels is considered when calculating the ratio of positive to null pixels around a point. The average distance of all the positive pixels' coordinates to the extrema point being analysed is calculated, providing a vector. The magnitude of this vector is expected to be large if the points are mostly on

one side of the point, and small if there are points on both sides of the point. Therefore, a large magnitude denotes points on the ends of the robot body. A pre-made **softmax** function is then applied to these magnitude, which computes the exponential of each magnitude divided by the sum of the exponentials of all the magnitudes [31]. The softmax function provides a list of probabilities for the likeliness of each point being the end tip of the robot. The maximum element is chosen to be the end tip coordinates. The accuracy of this logic is 100% as the end of the robot is always identified, despite the different shapes present in the data set. This means we are now able to find the transformation matrix mapping one set of coordinates to another. This process is described in the following section.

Chapter 5

Analysis and Evaluation

The following chapter aims to answer the following questions:

1. What is the most accurate segmentation method?
2. What is the fastest method?
3. Which ones are appropriate for accurate real-time processing?

5.1 Accuracy

When measuring the accuracy of the segmentation methods, there are several factors to consider, including the measuring metric and what is tested. In the case of this project, the goal of segmentation is to provide accurate sets of 2D coordinates (of the position of the robot) that will be used to find the real-world 3D coordinates. Therefore, the error calculated in 2D is not particularly relevant, instead, the error of the 3D reconstructed coordinates provides a better representation of the accuracy of each method. As mentioned previously in this report, the final accuracy of the mathematical model's predictions of the robot's shape was of 91% in its worst case. The robot being tested is 20cm in length, meaning any higher accuracy than 1.8cm will be considered a success in terms of accuracy. The speed evaluation of the approaches can be found in section 5.3, considering that the ideal solution finds the most accurate solution that is able to run in real-time. For the purposes of showing the error data of each method, box-plots are chosen to demonstrate the distribution of the error: the average, the outliers, the lower and higher percentiles, etc.

5.1.1 Edge Detection Accuracy

The edge detection method was expected to perform worse than the approach using Background Subtraction for several reasons. Firstly, it requires many more steps and calculations in order to gain similar results. Secondly, it does very little machine learning and so is a solution that is somewhat specific to the current problem. This means that while this specific solution works well for the given problem, due to appropriate

tuning of important parameters throughout the algorithm (decided on through experiments and observations), it would not be expected to work as well in a different setting. This could be relevant if the robot environment is changed dramatically. It must also be taken into account that this method sometimes under-calculated the coordinates due to missing part of the end of the robot (this challenge was discussed in chapter 4) due to reflections present in the environment. This is something to consider in the future, as is explained in Chapter 6.

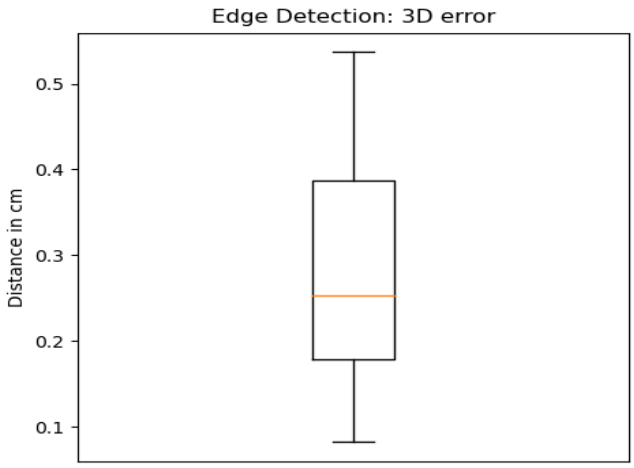


Figure 5.1: Boxplot demonstrating the distribution of the error found between the calculated 3D coordinates and the real-world coordinates.

Despite the shortcomings of this method, it provides more accurate results than the mathematical model, as can be seen in 5.1. With the maximum error being 0.5 centimetres and the majority of errors being between 0.2 and 0.4 cm, we can deem this method to be more accurate than the mathematical model. Whether it is sufficient for the level of accuracy required for human surgery is unknown yet, as accuracy requirements have not been set yet.

5.1.2 Background Subtraction Accuracy

The approach that uses the efficient CNT Background Subtraction algorithm, described in Chapter 4, has several advantages over the edge detection method, in terms of accuracy, for the following reasons. Firstly, it uses machine learning in order to create a model of the background, while providing many options through parameter-tuning that could make the algorithm adaptable to changes in the robot environment. Secondly, it has been optimized to be resilient to light changes, which can often occur throughout a set of frames due to several external factors such as small light changes in the room due to shadows. This means that problems encountered in the edge detection method, such as reflections on the robot body, are minimised and often not a problem. It is therefore no surprise that this method's accuracy, seen in Fig.5.2, is a great improvement on the edge detection method. With a maximum possible error of 0.08 centimetres, it is a great improvement on the mathematical model's 1.8 and the other method's 0.5.

As the boxplot demonstrates, the main percentage of the error is within 0.025 and

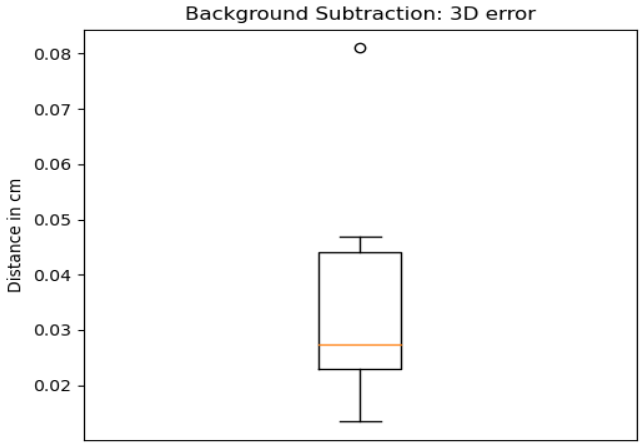


Figure 5.2: Boxplot demonstrating the distribution of the error found between the calculated 3D coordinates and the real-world coordinates.

0.045, demonstrating the method's suitability to this project, as well as the potential of efficient machine learning algorithms. The outlier shown in the boxplot shows a worst-case scenario, of the tip of the robot being wrongly calculated by 0.08 cm, or 8mm, which was caused by faulty segmentation of the tip. However, as can be seen by the distribution of error, this was an exception in the data set, which is deemed to represent most of the robot's possible positions. Since the outlier error is still lower than previous errors, it is not considered a problem, and this method is considered appropriate for accurately tracking the robot throughout frames.

5.2 Speed

In order to find the fastest segmentation approach, the time taken to process a single frame is calculated. This time is assumed to be the difference between the beginning of the processing of each frame. However, because different instances of the same algorithm will result in slightly different measurements due to uncontrollable hardware factors, 30 measurements are taken for each method in order to present an accurate description of the distribution of the time taken. Each measurement contains processing times for all 20 images in the data set. The number 30 is chosen according to [32] and other sources that use the Central Limit Theorem as a guide for what represents an accurate distribution of a group, and this number is often stated to have a minimum value of 30. The mean and standard deviation of each sample is calculated and demonstrated in graphical form in this section. Because what is shown is the difference between readings, there are half as many measurements as frames in the data set. For recreation purposes, the specifics of the hardware used to run these experiments can be found in Appendix 7.B.

5.2.1 Speed Measure

While the graphs in the following sections describe the processing time per frame for each of the methods explored, it is not simply a matter of how quick the algorithm is. All times will be directly compared to the estimated requirement. The estimated requirement is that the algorithm will be used for a video, which often contain up to 30 frames per second [33]. For the context of this project, that measure is used to deem whether an algorithm can run in real-time. Any value lesser than 30 but above 20 is still considered plausible as the final camera parameters are still unknown. It is also possible that 20 frames are enough to grasp the robot's movements as the robot's moving speed is not high and sudden moves are unexpected. It must be taken into account that since there are two frames to be processed in order to gain one set of 3D coordinates, the time demonstrated in the graphs (per single frame) must be doubled in order to account for one iteration of finding 3D coordinates, as shown in the calculations in the following sections.

5.2.2 Edge Detection

The edge detection approach is the slower segmentation method, demonstrating the possibilities of optimization tools such as Valgrind [22] that are employed in the background subtraction method. As the graph in Fig. 5.3 demonstrates, the processing time required per frame is mostly under 0.008, approximately 300% of the time required in the background subtraction method. Despite being the slower method, the calculations below demonstrate it still provides a processing rate apt for video processing. However, the method is considered sub-optimal as it has the higher error rates in terms of accuracy, as described in section 5.1. More on the method's overall evaluation can be seen in section 5.3.

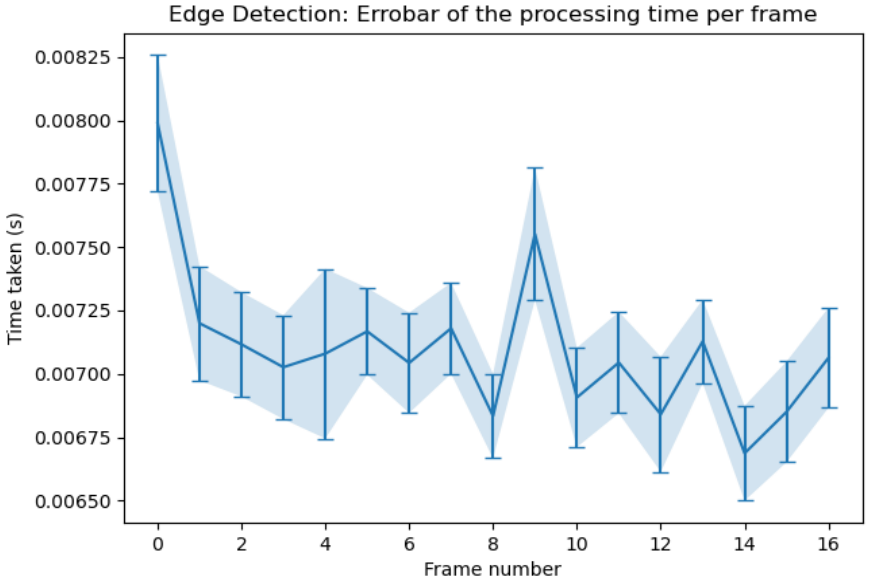


Figure 5.3: The mean and standard deviation of the time taken, in seconds, to process each frame using the edge detection method.

In order to estimate whether the method meets the desired video frame rate (30 frames per second), the calculation 5.1, using the worst-case time demonstrated in the graph. This method is able to process 62.5 sets of frames per second as demonstrated in the calculations below, double the speed that a video would require. If the method was refined and more accurate, it could become a valid option for tracking the robot as time is not a bottleneck.

$$1s/0.008s * 2frames \approx 62.5FPS \tag{5.1}$$

5.2.3 Background Subtraction

Fig.5.4 demonstrates the processing time required by the background subtraction approach, for each frame in the data set. The main graph line represents the mean time taken while the error bars demonstrate the variance from the mean between samples. As can be seen in the left axis, the maximum time taken in seconds is of 0.0030, or 3 milliseconds. Considering the accuracy, as demonstrated in the last section, of this method, this speed is remarkable. The background subtractor CNT algorithm has proved that it one of the best options for this type of vision-processing. The time range is of less than 0.0005, or 0.5 milliseconds, demonstrating the reliability of the algorithm for the whole data set.

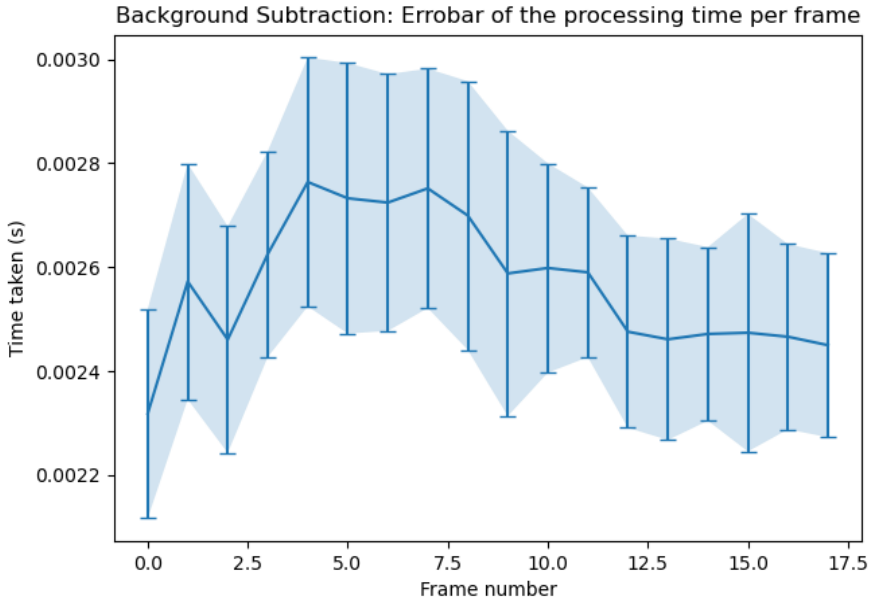


Figure 5.4: The mean and standard deviation of the time taken, in seconds, to process each frame using the background subtraction method.

This method is able to process 166.7 sets of frames per second (using the worst case scenario) as shown in calculation 5.2, which is more than is necessary for video-processing, therefore this method is deemed efficient enough to be applied in real-time.

$$1s/0.003s * 2frames \approx 166.7FPS \tag{5.2}$$

5.2.4 Pose Estimation

The following tests were performed on a method that uses the previously described background subtraction method in partnership with the pose estimation that key feature tracking can provide. While the accuracy of this method is arguably useful, as described in the previous section, it is a promising addition to segmentation that could provide more useful information about the robot's location, such as which tube has moved and in what direction, rather than just the end tip coordinates. As can be seen in the graph in Fig.5.5, there is a distinct difference in the processing time of the first few frames. As mentioned before, the CNT algorithm requires two frames before it can work accurately, which can be seen in the spike of the first frames. The reason for the height of the spike however, in comparison with the similar method described in the last section, is the addition of the feature matcher. The first time the object is created, there is a retardation in processing time. However, as explained previously, this is not necessarily a drawback of the approach seeing as this time only lasts 2 or 3 frames, which are still processed in less than a second and could occur in a calibrating stage of the algorithm.

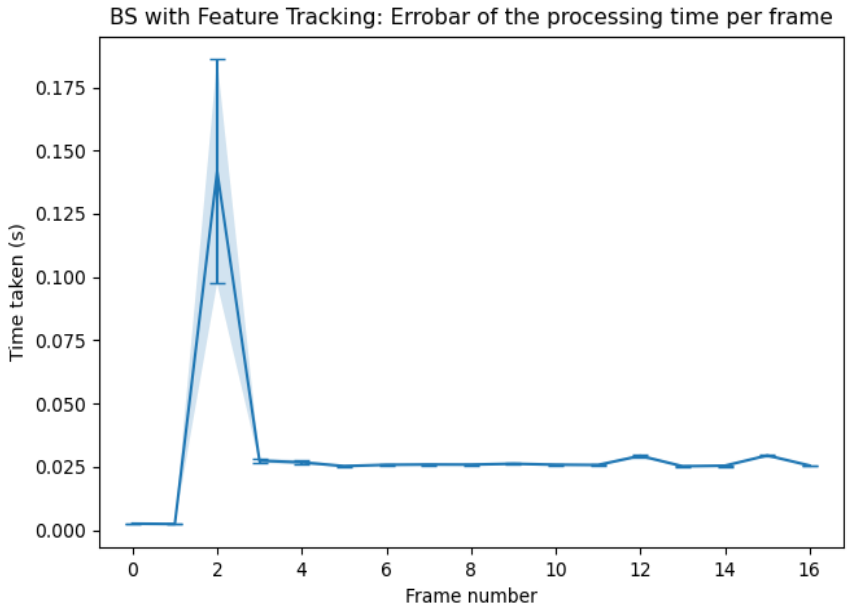


Figure 5.5: The mean and standard deviation of the time taken, in seconds, to process each frame using the background subtraction method with pose estimation.

This approach is able to handle 12.5 sets of frames per second (when using the median) as shown in calculation 5.3. This rate could still provide accurate coordinates of the robot throughout frames, and considering the speed of the robot's movements, it is possible that a 12.5 frame rate could still account for all the relevant robot positions occurring in a video. However, as established in the beginning of this chapter, success is determined by the algorithm's ability to run in real-time, therefore this method is deemed invalid for the task.

$$1s/(0.04s * 2frames) \approx 12.5FPS \tag{5.3}$$

Chapter 6

Conclusion

6.1 Final Evaluation

This section aims to establish the final solution proposed by the research performed in this project. For context, previous results from students are compared to the results achieved in this research, as a big objective was to improve on those results. Both students' work can be found by their titles but are not yet published, so no online record of them can be referenced. The titles of their projects are demonstrated in [34] and [35].

	Previous Work	**Edge Detection**	**BS**	**BS with Feature Tracking**
Speed (FPS)	A: 8 B: NA	62	167	12.5
Accuracy	A: 1.6cm B: 2.6% of length	0.25 cm	0.03 cm	NA

Table 6.1: Table of final speed and accuracy results compared to previous results, where BS stands for Background Subtraction (adapted for visibility). NA stands for Not Applicable, either the data was not found or the experiment not performed.

As can be seen in Table 6.1, both approaches proposed have achieved lower processing times than previous students, highly impacting the viability of this project for real-time tracking. The results compared are of the average accuracy for each method, while for the speed comparisons the worst-case is used. The background subtraction approach clearly demonstrates a much higher possible processing rate of 167 frames per second as a worst-case, an improvement on the previously achieved 8 frames per second (as a worst case scenario, and up to 14 frames per second) and the edge detection approach.

Together with its higher accuracy, this method is decidedly the best segmentation method found in this research. On the other hand, when using Key Feature tracking (as described in section 4.2) the algorithm is too slow for video-processing. The accuracy of this method is hard to gauge, and further research is necessary to find all the advantages and disadvantages of using this method. So while feature tracking could provide a deeper knowledge of the robot's coordinates and pose, for the purpose

of speed, this approach is not considered valid. If later tests reveal that 12.5 frames per second is an appropriate rate for the movement of the robot, it could be revisited. The final conclusion is that a valid, accurate, and efficient method has been found using background subtraction to accurately track and localise the robot throughout frames.

6.2 Possible Improvements

Several aspects of the methods created throughout this project can be modified for improvement, and there are several future steps to be carried out in order to achieve the final algorithm that meets all the original objectives (section 1.3). The following are several improvements that could be expected to increase the two methods proposed in this project.

Firstly, it is assumed that a lack of harsh lighting in the environment, such as the one present in most frames of the data set, would greatly increase the accuracy of both methods, particularly in the case of edge detection, as its main challenge is discarding a part of the end of the robot due to reflections. In order to correct this, different lighting options could be used, such as many small LED lights with a diffuser in front to reduce harsh lights and shadows. However, this would also entail repeating the process of data gathering for a new data set. Since the final method is very accurate nonetheless, the small increase in accuracy would probably not make up for the time lost in data gathering, but this might be revisited once the specifications for the necessary accuracy for viable use is defined. Another improvement in the area of data gathering would be to collect more accurate EMT data that could be used as accurate truth data for the coordinates of the robot.

Secondly, it is probable that further parameter tuning could result in more accurate results. However, because processing time was a concern, a search for optimal parameters is often not viable. Better parameter tuning could particularly improve the background subtraction method, as its learning approach relies heavily on its parameters. This can affect the way past frames are considered and how much they are weighted in order to adapt the background model. However, as mentioned previously, increasing this method's accuracy is not deemed necessary at this stage as the accuracy is deemed acceptable.

Thirdly, the edge detection method could be improved in several ways. For example, the logic of how to find the whole robot when it is fragmented due to reflection does not work for every frame. A check for the current length of the robot would improve this logic, as it could be taken into account to gauge how much of the robot is missing from the segmentation results. However, a method of providing live length readings would be necessary, and this is not yet available.

6.3 Future Steps

The following are some of the steps the project will require in order to approach the final objectives. The main step that can be carried out in continuation of this research,

is to tackle the second original objective (section 1.3) of predicting the full shape of the robot using only one camera. In order to do this, several options are available. For example, a possible solution would be to pre-calculate the transformation between two camera's perspectives and apply this to one set of coordinates from one camera. However, this still technically requires two cameras, and so the next option is more appropriate. A method that uses machine learning with labelled training data, supervised learning, would likely accomplish this goal. The labelled training data would consist of the segmentation results produced by the final approach proposed.

Another future step would be to perform more in-depth research about the applications and benefits of using Feature Tracking, the approach discussed in section 4.2. Information about how corresponding points have changed between corresponding frames could shed a lot of light on the specific changes of the individual tubes, and the curvature of the robot. This extra information about the robot's pose could greatly increase the accuracy of machine learning algorithms that are hoping to recreate the full robot shape based on a single image.

Finally, one of the next steps in this project will be to gather much more data. While the current data set is small, it contains many different possible robot positions that can account for the majority of the robot's range of movement. However, the data set lacks videos to provide real-time processing, and more data would be necessary to train a machine learning model.

6.4 MInf 2 Plan

This section aims to demonstrate the current plan for the continuation of this project towards the final goals, as described in Chapter 1. Due to the unforeseen circumstances Covid-19 has brought, the exact start and end of this timetable is unknown. However, the steps to be taken are known. These and their expected completion time can be seen in Fig.6.2.

Action/Process	Expected Completion Time
Gathering more data: images and EMT	1 month
Process EMT data for use	1 month
Use Part 1 segmentation for labelling new data to train a Machine Learning classifier that could predict shape with only one camera.	2 months
Research more about feature tracking with the robot and potentially include this information to increase classifier accuracy.	1-2 months
To improve classifier accuracy with or without the use of feature tracking.	1 month
Writing of the report.	1 month

Table 6.2: The steps planned for the second installment of the project, with an estimate of the time these will require.

Chapter 7

Appendix

7.A Glossary of frequently used terms

The following are frequently used words and their explanation. They are the author's explanation of the terms to increase understanding and should not be considered as formal definitions.

- **Segmentation:** The process of splitting an image into different segments. For example, separating the robot from the rest of the image.
- **Background Subtraction:** The process of identifying background and foreground pixels, in order to eliminate the background of an image. What constitutes as back or foreground depends on the situation.
- **Edge Detection:** The process of identifying edges in an image by calculating the gradient and direction of the image.
- **Real-time Processing:** In this project, refers to the ability to produce sets of 3D coordinates as quickly as frames are captured. Currently, this rate is 30 frames per second for videos. Therefore any algorithm that processes frames faster than this rate is appropriate for real-time processing.
- **Feature Tracking:** The process of identifying key features in images, edges for example, and given two sets of key features, comparing neighbourhood information from these pixels in order to validate their similarity, to gauge whether these pixels refer to the same point across different frames.
- **Semantic Segmentation/Dense Prediction:** The process of classifying every single pixel in an image. This approach is useful for tracking objects as it is specific about the location of the object in the image.
- **Image Classification:** Assuming a data set with labelled images is given, this is the process of providing a label for an image depending on the main object within the image. This is of course much more vague in terms of localisation than semantic segmentation.

- **Object Detection:** The process of detecting the number of objects within an image.
- **Unsupervised vs Supervised learning:** Supervised learning replies on labelled training data the classifier can use to classify incoming data points. Unsupervised however, is given no labelled data and must find underlying structures or patterns in order to learn more about the data. Supervised learning is appropriate for this project, while unsupervised is ill-suited as it is unlikely that the whole robot would be identified considering the colour variations in the images.
- **Continuous Detection:** An approach to vision-processing where each image is processed fully, rather than a specific area as is the case in ROI tracking.
- **ROI-tracking:**An approach to vision-processing that focuses on a specific area of given images rather than the whole image. This is helpful in cases where coordinates are known.
- **Noise Reduction:** Noise refers to added error to data, for example blurry pixels in an image. Noise can sometimes refer to unwanted elements within an image, such as random lines or circles.
- **Mask:** Often refers to a black and white image that can be joined with another in order to select areas and discard others. For example, a mask for a face would imply the resulting image only has a face and the rest of the pixels have a value of 0.
- **Gray-scale:**An image format that implies the image has been reduced to a single channel, from three colour channels, that represents the amount of light/intensity in an image. Often a required step to apply edge detection algorithms.
- **Colour Channels:** Typically images contain a Red, Green and Blue channel. These channels contain the values for their specific colours of each pixel. This means a normal image will contain 3 colour values per individual pixel. Other channels such as saturation and hue can also be found.
- **Contours:** Contours can be defined as the outline of a shape within the image. They are a useful tool for shape analysis and object detection. The functions provided by Python to find contours are often accurate and have a high degree of adaptability with their variety of parameters.

7.B Hardware Used for Speed Measurements

```
Processor: AMD Ryzen 5 2600 Six-Core, 3.85 GHz
RAM: 16.0 GB
OS: Windows 10
```

7.C Matlab 3D Reconstruction code

```
clear all
```

```
clc

% Loads the pre-calculated camera parameters
load('stereparam.mat')

img1 = imread('images_02_13\cam2_rot_1.png');
img1 = undistortImage(img1,stereoParams.CameraParameters1);
imshow(img1)

% The code below gets the coordinates of the points selected
% in the image. However, in order to use the calculated
% coordinates from the segmentation results rather than
% the selected coordinates.

[x1, y1] = getpts;
point1 = [x1, y1];
img2 = imread('images_02_13\cam1_rot_1.png');
img2 = undistortImage(img2,stereoParams.CameraParameters2);
imshow(img2)
[x2, y2] = getpts;
point2 = [x2, y2];
close all

% Uses camera parameters to find the transformation
% between the points and applies it to get the final
% coordinates.
point3d_tip = triangulate(point1, point2, stereoParams);

img1 = imread('images_02_13\cam2_rot_1.png');
img1 = undistortImage(img1,stereoParams.CameraParameters1);
imshow(img1)
[x1, y1] = getpts;
point1 = [x1, y1];
img2 = imread('images_02_13\cam1_rot_1.png');
img2 = undistortImage(img2,stereoParams.CameraParameters2);
imshow(img2)
[x2, y2] = getpts;
point2 = [x2, y2];
close all

point3d_base = triangulate(point1, point2, stereoParams);

% The final coordinates are of the tip of the robot
% in respect to the base of the robot.
point3d_tip - point3d_base
```

7.D Main.py file required to recreate experiments

```
% In order to change the strategy used, simply change
% the word to the appropriate method (described in
% section 1.5) in the constructor of the Engine object.
engine = Engine(strategy='back')
imgs1 = engine.getImages(1)
imgs2 = engine.getImages(2)

coors1 = engine.processImgs(imgs1)
coors2 = engine.processImgs(imgs2)
```

Bibliography

[1] "Snake-arm robot." [Online]. Available: https://en.wikipedia.org/wiki/Snake-arm_robot

[2] "Statistics by cancer type: Lung cancer." [Online]. Available: https://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/lung-cancer

[3] Y. Zhang, H. Sun, Y. Jia, D. Huang, R. Li, Z. Mao, Y. Hu, J. Chen, S. Kuang, J. Tang, X. Xiao, and B. Su, "A continuum robot with contractible and extensible length for neurosurgery †," pp. 1150–1155, 06 2018.

[4] P. Dupont, J. Lock, and E. Butler, "Torsional kinematic model for concentric tube robots," *IEEE International Conference on Robotics and Automation : ICRA : [proceedings] IEEE International Conference on Robotics and Automation*, vol. 2009, pp. 2964–2971, 05 2009.

[5] F. SHAIKH, "Essentials of deep learning: Exploring unsupervised deep learning algorithms for computer vision." [Online]. Available: https://www.analyticsvidhya.com/blog/2018/06/unsupervised-deep-learning-computer-vision/

[6] "Strategy." [Online]. Available: https://refactoring.guru/design-patterns/strategy

[7] D. Harvey, "Robot snake created by bristol engineers." [Online]. Available: https://www.bbc.co.uk/news/uk-england-bristol-15073514

[8] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, 2015.

[9] "Tendons, concentric tubes, and a bevel tip: Three steerable robots in one transoral lung access system." [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4492535/

[10] S. S. R. Reilink and S. Misra, "3d position estimation of flexible instruments: marker-less and marker-based methods," international Journal of Computer Assisted Radiology and Surgery, vol. 8, pp. 407–417, 2012.

[11] R. Xu, A. Asadian, S. F. Atashzar, and R. V. Patel, "Real-time trajectory tracking for externally loaded concentric-tube robots," pp. 4374–4379, 2014.

[12] D. Focken and R. Stiefelhagen, "Towards vision-based 3-d people tracking in a smart room," pp. 400–405, 2002.

[13] V. N. Dobrokhodov, I. I. Kaminer, K. D. Jones, and R. Ghabcheloo, "Vision-based tracking and motion estimation for moving targets using small uavs," pp. 6 pp.–, 2006.

[14] "Computer vision based tracking approaches," accessed: March,2020. [Online]. Available: https://medium.com/teleidoscope/computer-vision-based-tracking-approaches-88d49819c9e6

[15] A. M. Neha Sharma, Vibhor Jain, "An analysis of convolutional neural networks for image classification," pp. 377–384, 2018.

[16] "Camera calibration and 3-d vision," accessed: March,2020. [Online]. Available: https://uk.mathworks.com/help/vision/camera-calibration-and-3-d-vision.html?s_tid=CRUX_lftnav

[17] "3-d locations of undistorted matching points in stereo images," accessed: March,2020. [Online]. Available: https://uk.mathworks.com/help/vision/ref/triangulate.html

[18] "Opencv," accessed: March,2020. [Online]. Available: https://opencv.org/

[19] A. Alshumrani and A. Papagiannakis, "An improved canny edge detection application for asphalt concrete," pp. 3683–3687, 10 2009.

[20] "The backgroundsubtractorcnt project," accessed: March,2020. [Online]. Available: https://github.com/sagi-z/BackgroundSubtractorCNT

[21] "Fastest background subtraction is backgroundsubtractorcnt." [Online]. Available: https://www.theimpossiblecode.com/blog/fastest-background-subtraction-opencv/

[22] "Valgrind," accessed: March,2020. [Online]. Available: http://valgrind.org/

[23] "Opencv background subtraction." [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html

[24] N. El-Gamal, H. E.-D. Moustafa, and F. abou chadi, "A new combination method for background subtraction in video sequences," *2012 8th International Conference on Informatics and Systems, INFOS 2012*, pp. MM–21, 01 2012.

[25] S. A. K. Tareen and Z. Saleem, "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk," 03 2018.

[26] E. R. V. R. K. K. G. Bradski, "Orb: an efficient alternative to sift or surf." [Online]. Available: http://www.willowgarage.com/sites/default/files/orb_final.pdf

[27] "Fast algorithm for corner detection." [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_fast/py_fast.html

[28] "Brief (binary robust independent elementary features)." [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_brief/py_brief.html

[29] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: Efficient indexing for high-dimensional similarity search ." 01 2007, pp. 950–961.

[30] "Why 3d will fail," created:2010, Accessed: March,2020. [Online]. Available: https://www.dr-lex.be/info-stuff/3dfail.html

[31] "Scipy documentation," accessed: March,2020. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.softmax.html

[32] "Central limit theorem," accessed: March,2020. [Online]. Available: http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_Probability/BS704_Probability12.html

[33] "Video resolution vs. frames per second," accessed: March,2020. [Online]. Available: https://thinpigmedia.com/blog/decisions-decisions-video-resolution-vs-frames-per-second

[34] S. Xenides, "Image-based localization and tracking of a contiuum flexible robot."

[35] X. Wang, "Image-based localization and tracking of a contiuum flexible robot."