Individual Engineering Project Final Report

Can Machine Vision be used to Improve Efficiency in Manufacturing?

Student Name: Luca Ricagni

Student Number: 16010178

Supervised by:

Dr. Gary Atkinson

Module code: UFMFX8-30-3

Submission Date: 23/04/20

Main Text Word Count: 11039

Degree Title: BEng (Hons) Robotics

Department of Engineering, Design, and Mathematics

University of the West of England

Preface

Abstract

Machine vision can be described as the technology and methods used to extract information from an image on an automated basis.

This paper will endeavour to determine whether machine vision has a viable application in tracking cargo moving through a factory. The Fischertechnik model factory will be used for the purposes of demonstrating this concept, while the product passing through the factory will be represented by coloured counters and referred to as 'cargo'. Many different techniques were tested to show that the three different coloured cargoes (red, blue and white) could be reliably found. Some of the methods used included: the use of thresholding, edge detection and HSV. In addition, several original techniques were used: comparing a given frame to a control frame and labelling and removing small regions of an image. The project was successfully concluded with the addition of large amounts of lighting, enabling all the differently coloured cargoes to be found with a success rate of $\geq$80%.

Disclaimer

I hereby declare that this dissertation is my own original work, except where attributed to another, and has not been previously submitted to any other institutions for assessment purposes.

Furthermore, I have acknowledged all sources used and have cited them in the references section.

Acknowledgments

I would like to thank Dr Gary Atkinson for his expertise, guidance and his consistent encouragement over the last year to enable me to write this dissertation.

I would like to thank my parents and siblings for continuing to push me to do the best I can.

I would like to thank my close friends Will, Liam, Izaak, Matt and Matthew for providing some much-needed laughs during late night lab sessions over the years.

Lastly, I would like to thank Emily, for being my rock and making sure I kept on track.

Contents Page

Table of Contents

1 Introduction

Machine vision is an increasingly important and expanding element of robotics. It is the term used to encapsulate the techniques and processes required to enable machines to see and as such has a multitude of applications including automatic inspection, process control and robot guidance (Owen-Hill, 2016). Examples of these include: inspecting electronic components as they come off a production line, controlling the input of reactants in industrial chemical reactions and providing environmental data to a robot so that it can traverse the local area. Machine vision incorporates many disciplines such as signal processing, image processing and pattern recognition in order to achieve a desired goal.

As industry and manufacturing becomes increasingly automated, the use of machine vision is becoming more widespread in an attempt to eliminate the potential for human error. Machine vision systems can replace humans carrying out repetitive tasks, by quickly examining things in greater detail.

Machine vision has applications in multiple areas of a production line, from part inspection and quality control within factories, to tracking and routing cargo in distribution warehouses.

In addition, machine vision can be extremely effective when employed in tasks to mitigate risk where failure could be catastrophic, such as inspecting aeroplane parts or load-bearing structures in buildings for damage.

Another application of machine vision is to track objects moving through an area, some examples include tracking people moving through a crowd and tracking vehicles or other obstacles in self driving cars. In order to do this the software often takes steps to predict where obstacles will be in the next time interval, resulting in faster response times. This is particularly important in the case of self-driving cars as they can be moving at speeds in excess of 70mph.

After consultation with the supervisor of the project a decision was made to focus this project on tracking cargo as it moved through a model factory. This is because a Fischertechnik model factory was available and would provide a good medium for tracking

cargoes. The Fischertechnik model factory provides a variety of locations where cargo tracking can take place, such as cargo racks, a pneumatic crane and several conveyor belts. Figures 1 & 2 show the Fischertechnik model factory mounted on its trolley.

The reader should note that the code to control the factory was not written by the author and was instead written by another student.

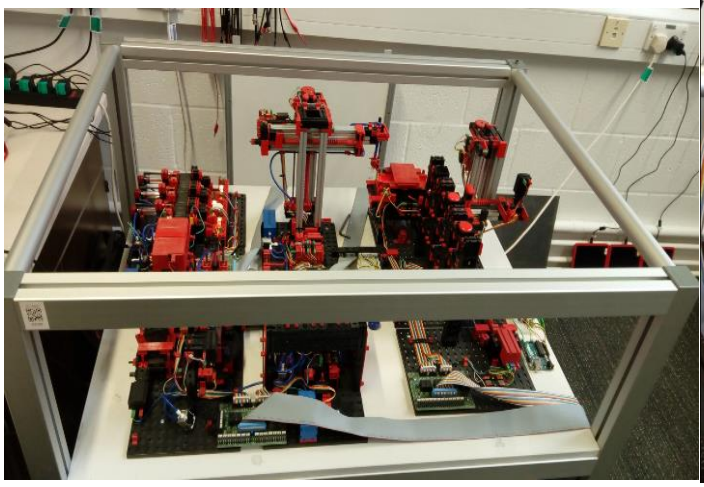


Figure 1

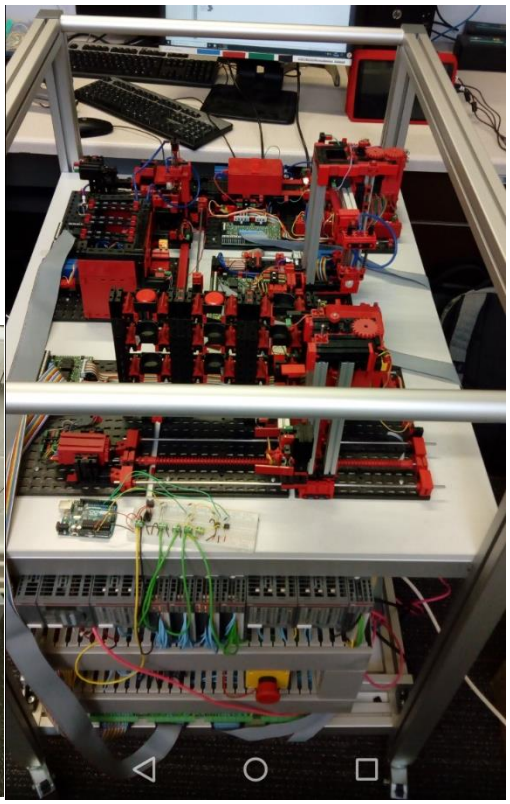


Figure 2

2 Scope and Objectives

The overarching aim of this project is to determine whether employing the use of machine vision is a viable and useful tool for tracking an object (cargo) moving through a factory. In order to attain this goal, the project was broken up into several clear objectives which were as follows:

1. Identify the medium which would provide the data used for testing
2. Identify the programming language to be used
3. Successfully locate circles in a still image
4. Successfully locate three differently coloured cargoes in a video
5. To track the cargoes moving through a video

There has not been any real deviation from the original plan outlined in the Interim Report. However, objectives 4 and 5 took slightly longer to complete than was originally planned, which was largely due to issues with lighting. The critical path for the completion of this project has been defined in the objectives, this is because these are the essential steps necessary to complete the aim of the project.

As discussed in section 1 the Fischertechnik model factory provides an appropriate medium to gather data to be used for testing and meet objective 1. This is because it provides a massively scaled down example of what a real factory could look like, and as such serves as a useful tool for demonstrating the concept outlined in the aim. The bulk of the data collection focused on the main belt in the Fischertechnik model factory, as it provided the longest uninterrupted, and more importantly unobstructed area in which the cargoes moved.

In order to fulfil objective 2, the programming language and IDE (Integrated Development Environment) MATLAB was chosen for the following reasons:

- It has several library add-ons which provide many useful functions for performing image processing and machine vision. The two libraries used in this project are the Image Processing Toolbox and the Computer Vision Toolbox
- It is a language which deals with many issues such as variable data type declaration, thus removing any added complexity

- MATLAB (Matrix Laboratory) was designed specifically with matrix manipulation in mind, and as such is very efficient at performing tasks involving matrices. This would prove extremely important as images are stored as matrices, with each cell representing a single pixel

3 Background Research

While many factories around the world employ people to perform tasks such as maintenance, quality control and even assembly, there is a definite push to remove humans from the manufacturing process. The concept of 'lights-out manufacturing', having factories with no human presence and thus no need for lighting or air conditioning, sounds futuristic but is in fact not new at all. The Japanese company FANUC has had a 'lights-out' factory in operation since 2001, where robots build other robots in complete darkness and no air conditioning (Anon, 2019). This saves a great deal of money in operating costs as the only human employees are several technicians who repair and maintain the machines, and there is no lighting or heating on. In addition, as a result of having no human workers, the factory can operate 24 hours a day without stopping, thus improving efficiency.

The 'lights-out' factory style can be used in other areas of the manufacturing process. For example: JD.com, a Chinese E-commerce giant, opened a large storage and shipping warehouse in Shanghai. This facility, which would normally need almost 500 workers to operate at capacity, now only has 5 technicians to repair and maintain the machinery (Anon, 2019). This facility acts as the distribution centre for the company and yet is completely automated.

Machine vision can be used in many areas of the manufacturing process, from material extraction and sourcing to the distribution and delivery stages. It is revolutionising many processes and enabling them to be more efficient financially and environmentally. The ability for machine vision to remove human labour and error from the process may prove to be invaluable in the future, allowing tasks to be carried out with a higher degree of accuracy. In addition, being able to remove lighting and air conditioning will drastically lower the carbon footprint of the factory, as less energy is required to run it. Not having workers commute in daily will reduce the carbon footprint too, as they won't need to use public transport or private vehicles.

Machine vision for tracking purposes is used to a lesser extent in manufacturing. Other applications include tracking people in crowds (Freitas, et al., 2016), autonomous rescue vehicles (allowing injured or sick people to be spotted and cared for), and for surveillance during large events such as music concerts. In the case of surveillance, it can be used to

ensure that the authorities can be alerted to any suspicious or dangerous behaviour. Another application of tracking using machine vision is to track cargo as it moves through an airport cargo terminal (Le, 2018). However, this is often done by scanning RFID tags attached to the cargo rather than analysing frames extracted from a video. Systems such as this enable cargo to be routed and tracked through a terminal to ensure it arrives at the correct destination.

4 Literature Review

The literature review performed in this section is divided into two sections: research conducted before the project commenced, and research conducted during the project (sections 4.1 & 4.2 respectively). The research conducted before the project started was performed primarily to determine whether the proposed project objectives and scope were viable by examining current literature. Research conducted during the project was done to ascertain whether certain techniques were reasonable approaches to solve issues faced during the project. Section 4.3 will be a summary of the literature review conducted throughout section 4.

4.1 Initial Research

The Circle Hough Transform (CHT), or simply Hough Transform, is a commonly used and well documented approach employed in a vast number of image processing applications. This is highlighted by the fact that there are over 2500 research papers detailing its use, variants, properties and applications (Mukhopadhyay & Chaudhuri, 2015). There are also a plethora of modifications to the CHT which have been proposed and implemented including: using edge orientation, concurrent consideration of a range of different radii, where the phase is proportional to the log of the radius, and the application of CHT as a filter (Atherton & Kerbyson, 1998). These varying methods come with a host of advantages and disadvantages: most help to improve the accuracy of the CHT at the expense of computational power requirements although more efficient computational implementations have been designed. A notable exception to the previous is the use of edge orientation, which operates by noting the direction an edge faces, towards or away from the centre, allows for a simple arc of radius R to be used thus massively improving the performance of the CHT. This technique is highly relevant to this project as it forms the basis of the MATLAB function ‘imfindcircles’ (Mathworks, 2012)

Lighting is an extremely important aspect of machine vision. How well illuminated an image is can have drastic effects on how easy it is to find objects within an image and is critical to whether or not objects can be found at all. Shadows (moving or static), blur and reflection are all obstacles to overcome as they can disrupt the use if machine vison. As such a logical way of dealing with lighting issues is to design a system where the lighting is controlled

throughout. One such way is to setup a double lighting system (Jahari, et al., 2015), in which a front-light and a backlight are used. Using only a front-light will result in the background being dark which would therefore allow an easy distinction to be made between target objects and the background. Using only the backlight will mean that the image will be brighter in general, with features more easily visible, at the expense of the background being less easily distinguishable from target objects. The front and back lights can then be turned on/off alternately which means there are now two images for any given time which can be compared and used in tandem to find target objects. Another method is to use multiplexing. Multiplexing is a method used to combine multiple analogue or digital signals into one signal to be sent over the same medium. With regard to machine vision this normally takes the form of using a 'constellation' of light sources (Schechner, et al., 2007), which could be many hundreds of lights. The use of many light sources allows for objects to be fully illuminated, thus mitigating shadows entirely as the lighting can be fully controlled, sources can be switched off or dimmed down depending on what best suits the need. As this project is primarily about machine vision lighting is likely to be very important.

It was imperative that a good Integrated Development Environment (IDE) was chosen, bearing in mind the projects requirements. MATLAB (Matrix Laboratory) is a well-known IDE with a great deal of high-quality documentation, as well as several add-on libraries with lots of additional functions specifically written to assist in machine vision (McAndrew, 2004). Some examples of the functions added include: 'imfindcircles' a function which finds circles in the image based off the Circle Hough Transform, 'imresize' which just simply scales an image to a given size and 'rgb2grey' which converts a given image from colour to greyscale, to name but a few.

In industrial applications machine vision is commonly used for implementing precise position verification systems which have to be accurate in the order of microns (Mahapatra, et al., 2015). There are many tools to obtain position measurement and verification, for instance linear encoders, coordinate measuring machines (CMM) and powerful microscopes. These various techniques work in different ways, with unique strengths and weaknesses (Mahapatra, et al., 2015). The resolution required for the application would dictate the appropriate method, with techniques such as laser interferometers, atomic force

microscopes and scanning electron microscopes being the most suitable for examining microscopic objects. These techniques are often employed in parallel, to account for the disadvantages of single methods. One such example is using atomic force microscopes (AFM) in tandem with scanning electron microscopes (SEM) as AFMs are much more efficient at providing highly detailed scans of smooth or flat surfaces, whereas with rough surfaces SEMs, with their significantly larger field of depth mean they produce better images (Russell & Batchelor, 2001).

In some machine vision applications, it is a viable, and indeed sometimes necessary, strategy to design a system where the camera is able to follow the object of interest. This allows for the object to be tracked even if it moves in such a way that would leave it obscured by another object or obstacle. One such method of dealing with these obstacles is to apply a Fast Fourier Transform (FFT) algorithm to a given image (Montironi, et al., 2014). By applying an FFT and using the phase information from it the movement necessary to clear the line of sight and reacquire the target object can be calculated. In addition, by considering both the images before and after a given movement, the fact that the perspective will have shifted can be accounted for. Subsequently by locating corresponding points in the two images and correcting for the distortion caused by the movement, object recognition is easier as a single template can be used.

Inspecting parts for damage and defects is also a common application of machine vision. From airplane components such as wings to electronics such as PCBs and even individual chips or integrated circuits (ICs) mounted on a board. Using machine vision offers a much quicker and often more accurate way of inspecting objects. In some of these electronic applications it becomes necessary to adjust the position of the camera in order to fully examine the object and assess the level of damage. In these cases, it if often useful to employ a neural network with the purpose of classifying the given IC and then inspecting it (Edinbarough, et al., 2005). This allows the system to decipher which object from a given set it is looking at, then decide how to best approach examining said object.

4.2 Research conducted during project

Thresholding is a common technique used in machine vision and is relevant to this project as its use was investigated for segmentation. It involves taking a greyscale image and, based

off the threshold value set, assigns pixels with a value below it to 0 (black) and pixels with a value above it to 255 (white). The easiest way to apply thresholding is to use manual thresholding. This is simply where the user choses a value T at which to threshold. This can be done either by coding in the value of T before runtime, or by asking a user for a value of T during runtime, and is often chosen by making an 'educated guess' based off a histogram of the image (Gonzales-Barron & Butler, 2006). However manual thresholding has limited applications given that systems often need to be designed in order to deal with things automatically. As such several techniques for applying automatic thresholding have been developed. One of the more prolific algorithms used is known as Otsu's method (Nobuyuki, 1979). This method involves iteratively changing the threshold based off the variance of each class, that is the group of pixels above and below the current threshold value T. This particular technique performs much better if the histogram for an image has a bimodal distribution but performs poorly if an image has a unimodal distribution (Hui-Fuang, 2006).

Edge detection is also a useful tool in machine vision and is utilized in order to help extract significant properties or features of an image. This technique often works by looking for sudden changes in brightness or discontinuities in the image (Umbaugh, 2010). This change is most obvious when it takes the form of a step function, in the case of image processing this would mean shifting from 0 to 255, or black to white. However this is a rare occurrence when not dealing with black and white images and as such tolerance must be built into the system to enable edges that perhaps are not so apparent to be found (Ziou & Tabbone, 1998). Just how much tolerance is applied depends on how robust the particular filter being used is, as well as what the application of the filter is. One of the first filters to be designed is the 'Canny' filter, this is a filter built into MATLAB (Mathworks, 2011). The Canny filter works by: applying a gaussian filter to remove noise, finding the intensity gradients, applying non-maximum suppression to remove false responses to edge detection, applying a double threshold to determine potential edges, and then removing edges which are weak (Canny, 1986). Edge detection is important to this project as it forms the basis for the Circle Hough Transform (CHT), and is also investigated as a standalone technique.

Images are commonly represented as a combination of red, green & blue (RGB) and while this is a useful way to convey images it is not the only way. Presenting images as a

combination of their hue, saturation and value (HSV) is another popular way as it aligns more closely with the way the human eye perceives colour (Joblove & Greenburg, 1978). The hue value of a given pixel represents the dominant colour, the saturation denotes the purity of the colour and value denotes the brightness (Poynton, 1996). One of the reasons HSV is a useful tool for image processing is because it is able to separate the image intensity from the colour information. This in turn enables HSV displays to more effectively ignore shadows, as the image intensity can simply be removed. Conversely, shadows can be ignored and the focus switched to the colour of a given pixel, depending on the task (Bhaskaran, et al., 2013). The use of HSV is relevant to this project as it is a useful way to represent images and was investigated in this project.

4.3 Summary of literature review

The literature review conducted in section 4.1 explores the importance of lighting, as well as the Circle Hough Transform and some if its modifications. Also discussed are some common applications of machine vision such as part inspection and position verification. In addition, using multiple automatically repositioning cameras is also considered. The literature review conducted in section 4.2 examines and discusses several techniques which will be used within the project such as thresholding, edge detection and HSV.

5 Methodology

This section will detail the methodology of the project, starting with a brief overview of the Fischertechnik model factory, followed by an overview of how the function 'imfindcircles' (Mathworks, 2012) was tested on still images. Gathering data by taking videos of the model factory is then discussed in section 5.3. Sections 5.4 – 5.10 detail the use of various techniques such as thresholding, edge detection, HSV and removing small regions. Section 5.11 details the final solution of adding more lighting and regathering the data.

5.1 Model factory details

As mentioned previously this project will make use of the Fischertechnik model factory. This model factory consists of five major areas: the 'oven' which serves as the start point and is positioned in the lower centre of Figure 3. The start of the main conveyor belt is located in the lower left corner of Figure 3. The bulk of the main conveyor belt, the colour detection block and the crane bays can all be found in the upper left section of Figure 3. The upper centre section of Figure 3 shows the crane itself and the motors which drive it. Lastly the right side of Figure 3 shows the cargo racks and the apparatus which moves the cargo to a specific shelf.

The factory comes with three different colours of cargo, white, red and blue, which will all be used as test cases to determine the success rates of the image processing techniques.

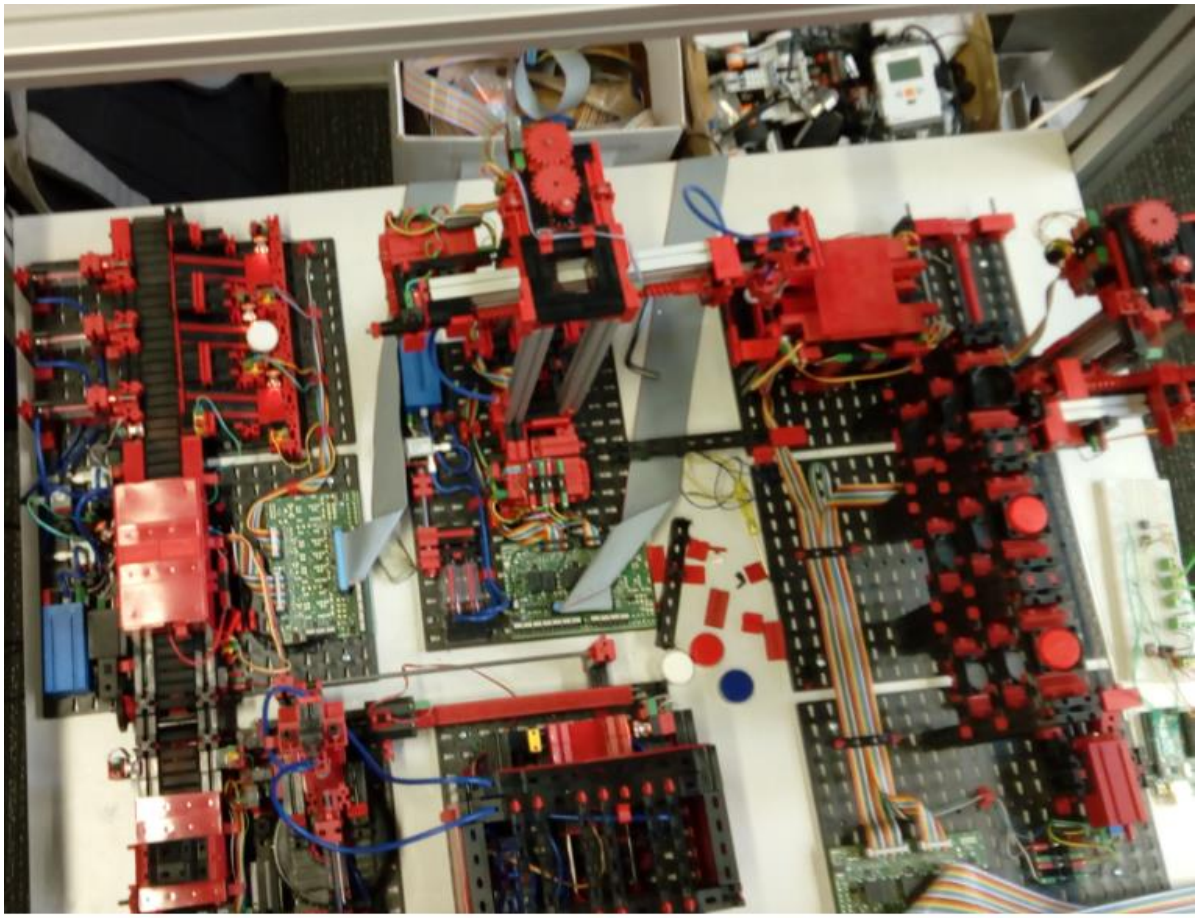


Figure 3

5.2 Find circles from a still image

5.2.1 Implementation

One of the key tasks is to implement a reliable circle detector and demonstrate its accuracy on some still images. As such he first stage of practical work was to find a suitable function in MATLAB which could be used to do this. The 'imfindcircles' (Mathworks, 2012) MATLAB function, which is part of the Computer Vision Toolbox was identified as a main suitable

candidate as it implements a Hough circle transform solution to analyse an image looking for circles.

It was imperative that an existing function could be found for this task as without it a significant time investment would have been required to create a bespoke function.

The function was initially tested on a colour top-down image of the cargo in the cargo racks (see Figure 4) but unfortunately the code was unable to find the cargo.

Figure 4

The 'coins.png' image from the mathworks page (Mathworks, 2012) on 'imfindcircles' (Figure 5) was then used as a test image. The function then successfully found the coins before drawing a blue outline on them (Figure 6).

Figure 5

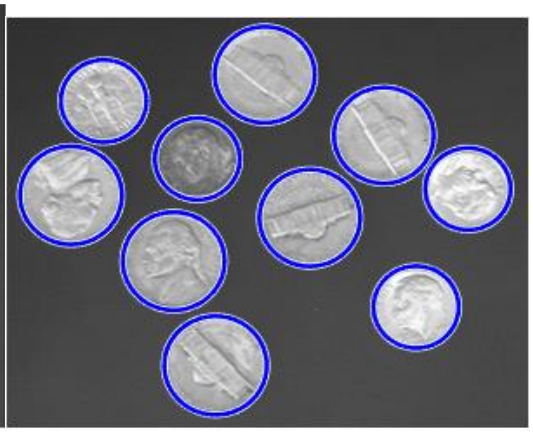


Figure 6

As the 'coins' image was a greyscale image it was thought that this could be the reason for being successfully able to find the circles in the image, so the next test was to convert the

image of the cargo in the racks (Figure 4) into greyscale (Figure 7) using the 'rgb2greyscale' function to try to replicate the previous success with the coins image. However, this did not solve the issue as the circle was still not found.

Figure 7

A simple white circle on a black background created using MS Paint (Figure 8) was then used in an attempt to create a well-defined and distinct edge for the algorithm to run on. However, the circle was still unable to be found.

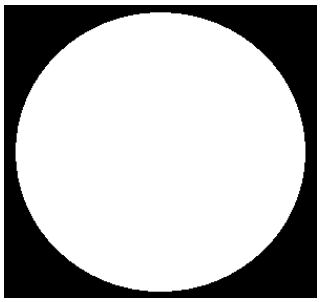


Figure 8

Following on from this it was noticed that the circle in Figure 5 actually comprised a larger total area than the background, leading on to the possibility that the function 'imfindcircles' was operating under the assumption that the white colour was the background, which was not what was required. Hence it was decided to use the same white circle but with a much

larger black background and retest the same function, shown in Figure 9. As shown in Figure 10 this worked perfectly; the circle was found and a blue outline drawn onto it.

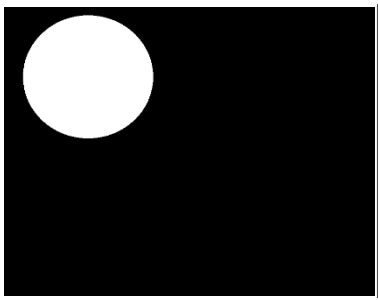


Figure 9

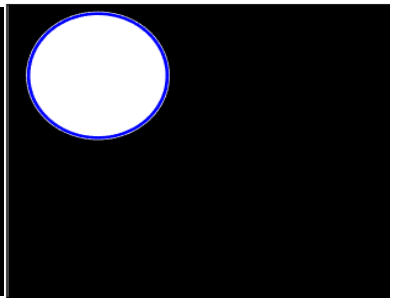


Figure 10

The logical assumption for the next test was therefore to ensure that the cargo had an ample background of a different colour in order to ensure that it could be distinguished. However, this was evidently not the issue, as the background made up a larger percentage of the image than the cargo.

The reader may wish to note that the image had to be scaled down in order to display it correctly. This was done simply by using the function 'imresize' (Mathworks, 2006), which simply scales down a given image by a specified scale factor. A scale factor of 0.1 (scaling down to 10% of the original size) was chosen.

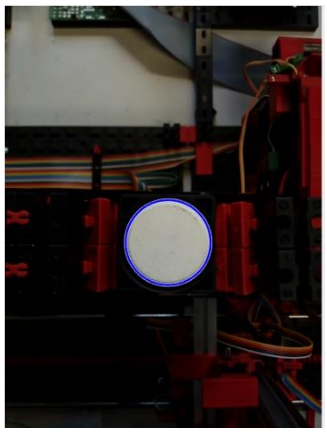


Figure 11

Upon scaling down the original image, the cargo was able to be found successfully with the blue outline drawn on it (see Figure 11).

5.2.2 Discussion of results

The function 'imfindcircles' was successfully implemented on a variety of images containing circles, as shown in Figure 11.

It was observed that the function 'imfindcircles' works much more efficiently on smaller images with smaller circles using a radius range which is as small as possible. Smaller images also mean that there are fewer pixels to be analysed which leads to a reduction in computation time. Making the range of radii as small as possible is covered in (Mathworks, 2012) where the following formula is stated:

$$r_{max} < 3 * r_{min} \quad \& \quad r_{max} - r_{min} < 100$$

This states that the maximum specified radius should be less than three times the minimum specified radius, and that the difference between the two should be less than one hundred. However, it should also be noted that the difference between r_{max} and r_{min} should be greater than 5.

While testing the use of converting the image to greyscale it was also found that converting to greyscale increased runtime and did not improve the speed of 'imfindcircles'.

The following results were found:

- Changing from RGB to greyscale gives an elapsed time of 0.616292 seconds
- Keeping the image in colour gave an elapsed time is 0.544687 seconds

Therefore, a preliminary look would indicate that not changing the image to greyscale improves performance by approximately 13% when compared to the image to greyscale, assuming no improvement in accuracy is seen. The fact that circles had been able to be successfully found in still images means that objective 3 has been completed.

5.3 Data gathering

The next stage was to gather some data, in the case of this project the data will take the form of videos taken of various parts of the factory in operation.

The videos taken included the 3 available colours of cargo included with the factory (blue, red and white) in different parts of the model factory. Figure 12 shows an example image of the first section of the model factory, it shows the ‘oven’ in the top right with the red track that the suction crane travels along. Figure 13 shows an example image of the second section of the model factory, from the spinning cleaner in the bottom left to the colour sensor box in the top right. Figure 14 shows an example image of the third stage of the model factory, from the output of the colour box to the three bays for the sorted cargo to be picked up by the crane.

These videos make up the bulk of the data for the project, with the ones belonging to the third stage being particularly pertinent. This is because the cargo is visible for the greatest amount of time during this stage.

The reader should note that in order to calculate the percentage of frames when the cargo could be found the videos taken were edited to remove footage where the cargo was not visible within the frame. All percentage values calculated are based off extracting a frame every 0.5 seconds.

Figure 12

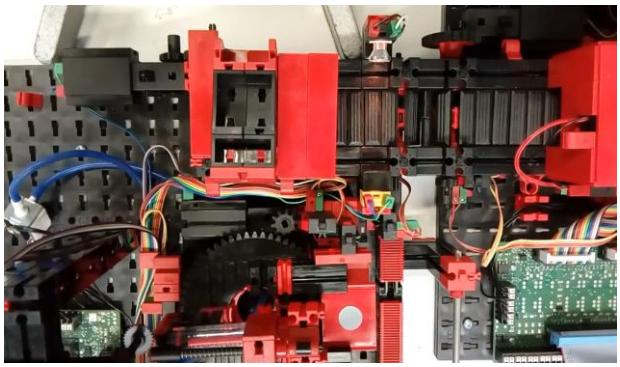


Figure 13

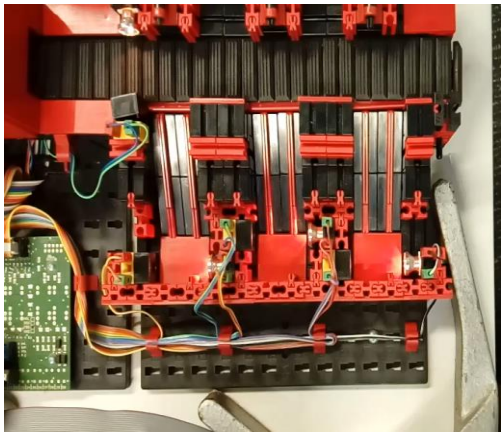


Figure 14

5.4 Find the cargo from a colour video

5.4.1 Implementation

Upon collection of the video data, tests could be conducted into how well the cargo could be found from a video. This was an especially important test as the technique of analysing a video to find a cargo formed the basis of the entire project, as the eventual aim was to track it in real time.

The initial pseudocode took the following form:

- Import video
- For loop to iterate through the frames
    - Extract frame
    - Display frame
    - Find circles within frame
    - Draw blue outline on circles
- Plot points

It was decided, based off the tests in section 5.2, that this was a good basic layout for the code to take in this task. The idea of extracting a frame every specified time interval (0.5 seconds was used) was especially important as it drastically reduces the computational load if only 2 frames need to be analysed per second instead of 30 or even 60 (depending on the camera used to acquire the data).

This pseudocode was then written up into MATLAB and can be seen in appendix 1.

This code, when implemented, was found to work almost perfectly for the white cargo; there were a few frames where the cargo was obstructed by the crane where the image tracking was expected to fail. However, this implementation did not work for the blue or red cargo colours. An attempt was then made to test it after firstly converting the images to greyscale, but as before this changed nothing, the white cargo was still the only one that could be found.

An attempt was then made to test the system by using the red, green and blue images individually using the code:

```
frame = frame(:,:,1);
```

It was believed that using the blue/red frame would help to 'bring out' the corresponding colour of cargo and make it easier to detect; unfortunately this was not the case and none of cargoes could be found.

5.4.2 Discussion of results

As mentioned above the code only worked for the white cargo, why this occurs is not clear. It is thought to be because the red cargo is a very similar colour to the background of the model factory and thus doesn't stand out very well (Figure 15). For the blue cargo it was more difficult to understand the reasoning, however it was believed to be because the blue colour is too similar to the black of the conveyor belt for it to be effectively distinguished (Figure 16).

Figure 15

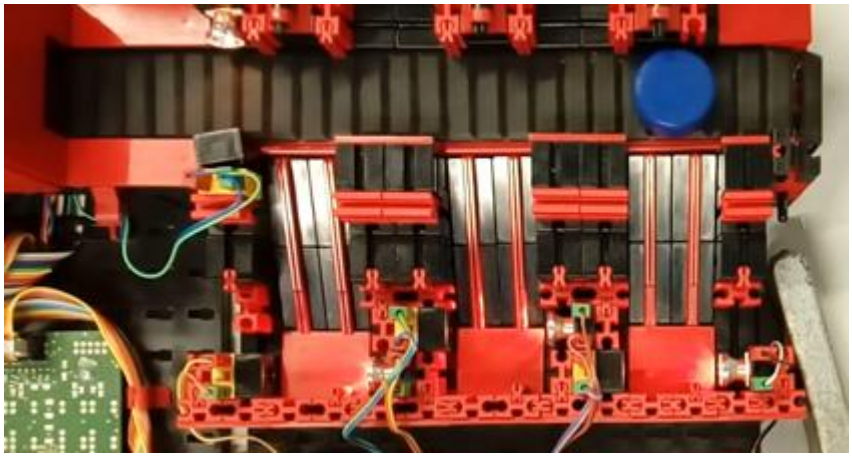


Figure 16

The calculations showed that this method was successful approximately 83% of the time for the white cargo, 5% for the red cargo and 0% for the blue cargo.

Overall, this section was successful as the basic functionality of extracting a frame every given time interval worked and the white cargo was able to be found in most cases, even if the other colour cargo was not able to be found.

5.5 Thresholding

5.5.1 Implementation

The next step was to attempt find a system in which all three of the cargo colours could be found with a reasonable degree of reliability. As such it was decided that a thresholding system would be employed in an effort to remove the colour of the cargo out of the equation entirely.

This was first tested by simply selecting an arbitrary value from 0-255, the range of possible values for any given pixel in an image, this is known as manual thresholding. However, this is generally not very effective, especially when trying to build a system which is capable of applying thresholding to many different images. As a result, an automatic thresholding system needed to be implemented.

Automatic thresholding is simply a system whereby the thresholding value T is chosen automatically. In this case it works by iteratively altering T based off the mean of the pixels above and below it until the mean of these is equal to the value of T selected in this iteration.

The pseudocode for the automatic thresholding used can be found in appendix 2.

5.5.2 Discussion of results

Figure 17

The result of this technique appeared to be quite good (see Figure 17), as the cargo (red in reality) can be fairly easily distinguished from the background by the naked eye. However, it did not allow for the cargo to be found at all, for any of the coloured cargoes, giving a percentage success rate of 0% for all three colours. This is because it is no longer circular. As can be seen in Figure 18, there are several spike-like shapes along the bottom as well as a larger shape at the top of the image. These abnormalities create difficulty for the function 'imfindcircles' and inhibit, or in this case remove entirely, its ability to find circles in a given image.

Figure 18

5.6 Edge detection

5.6.1 Implementation

Given that the automatic thresholding seemed to produce reasonably clear results it seemed sensible to build off this and 'clear up' the circle in order to make it more circular. A common method used in machine vision for this is edge detection, whereby algorithms are run on an image to try and locate objects by finding their edge. There are 6 edge detection filters built into MATLAB: Sobel, Prewitt, Log, Zerocross, Canny & ApproxCanny which all work slightly differently and are used in the 'edge' function (Mathworks, 2011).

5.6.2 Discussion of results

This endeavour did not meet with much success as none of the filters built into MATLAB gave good outlines for the cargoes, as can be seen clearly in Figure 19. This result did not help solve the previous issue of detecting the circles as the algorithm still does not function on these. This is due to the fact that the outlines left by the filters are fragmented and thus do not leave a full circle.

However, for the sake of completeness edge detection was tested on the video data. It was found that the white cargo could be found about 70% of the time, with the red and blue cargos being found approximately 5% of the time. A similar performance to the original colour tests.

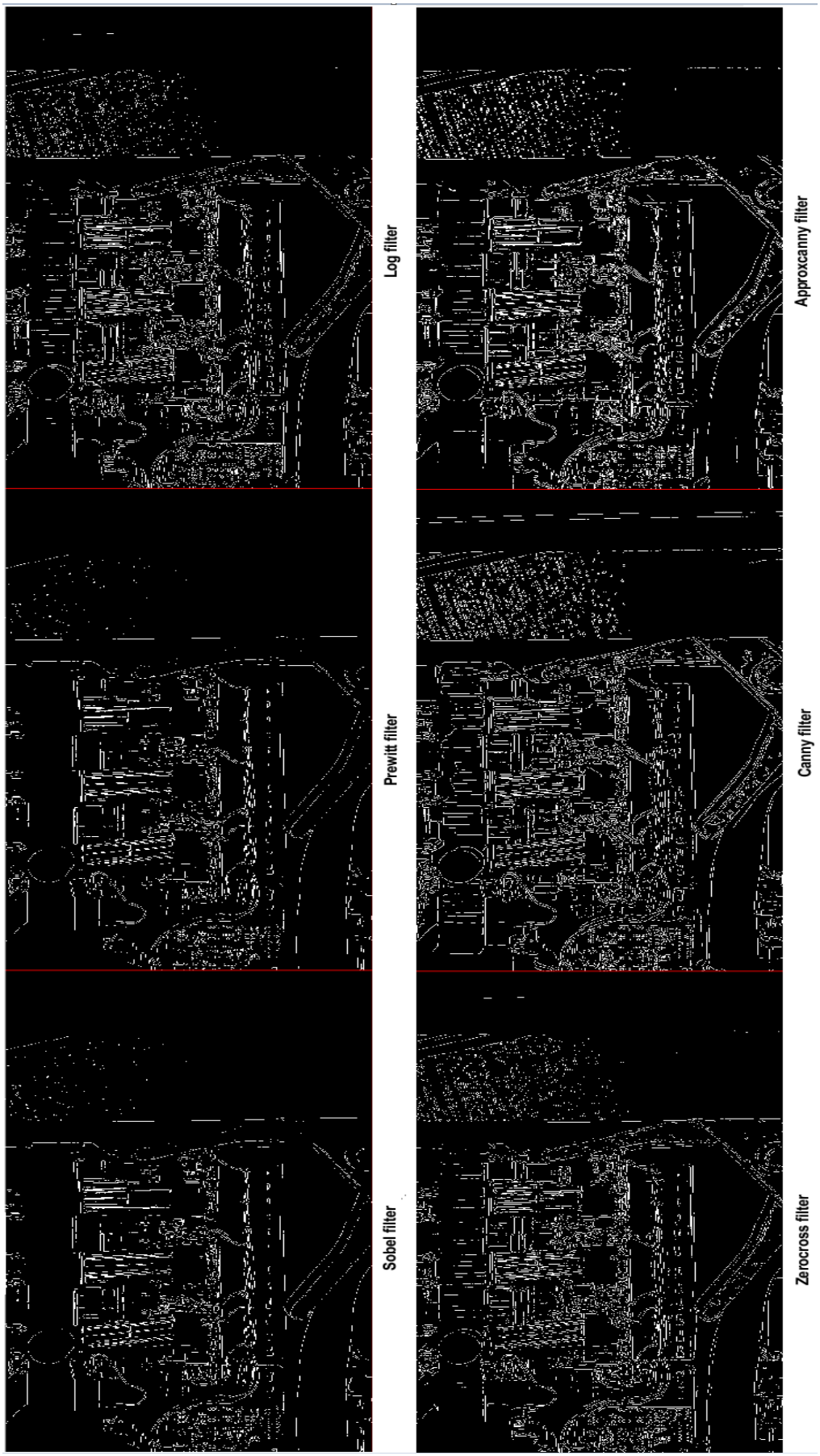

Log filter
Approxcanny filter
Prewitt filter
Canny filter
Sobel filter
Zerocross filter

Figure 19

5.7 HSV

5.7.1 Implementation

An attempt was then made to use the HSV (Hue, Saturation and Value) method as an alternative to the more standard RGB system. This method stores each pixel as a combination of these variables and is thought to be closer to how the human eye perceives colour. In MATLAB a given image can be switched from RGB to HSV by the simple use of the line:

```
frameHSV = rgb2hsv(frameRGB);
```

Where the function ‘rgb2hsv’ takes as input an RGB image and converts it to an HSV image.

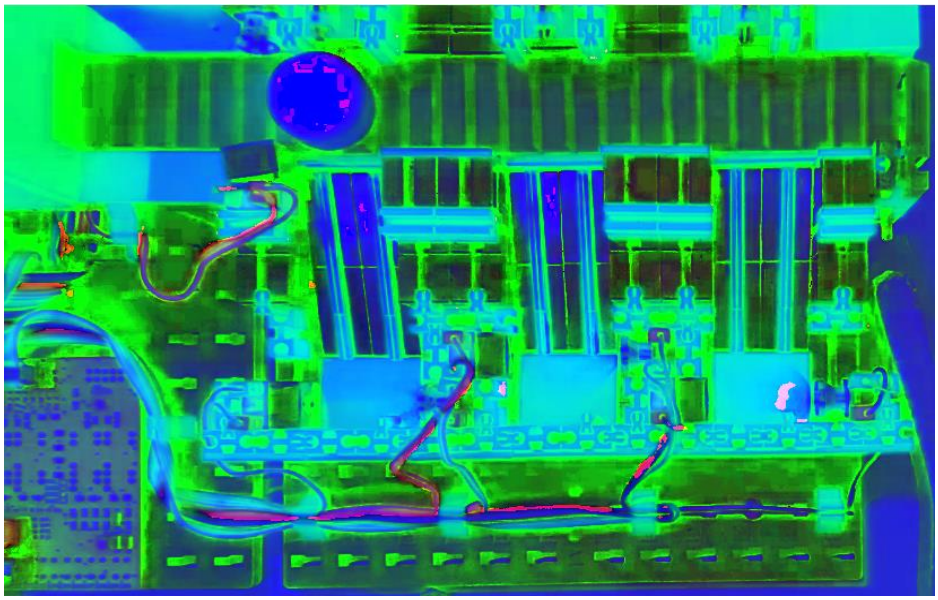


Figure 20

Figure 20 shows an example of a standard RGB image Converted to HSV. As can be seen the cargo (in this example the white cargo) looks fairly distinguishable from the background with reasonably defined edges. However, the cargo was still unable to be found using this method.

As in section 5.4, this was also tested on the hue, saturation and value frames individually, but to no improved success. Figures 21, 22 & 23 show the Hue, Saturation and Value frames extracted, however as can be seen none could be found.

Figure 21

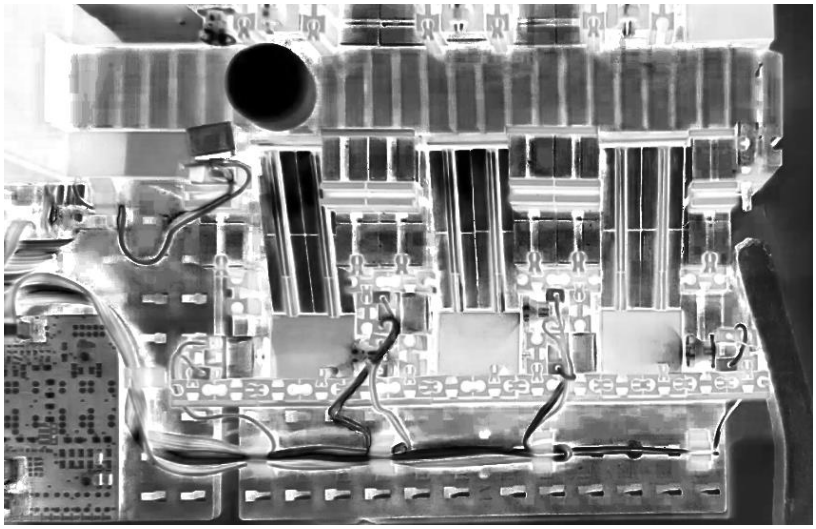


Figure 22

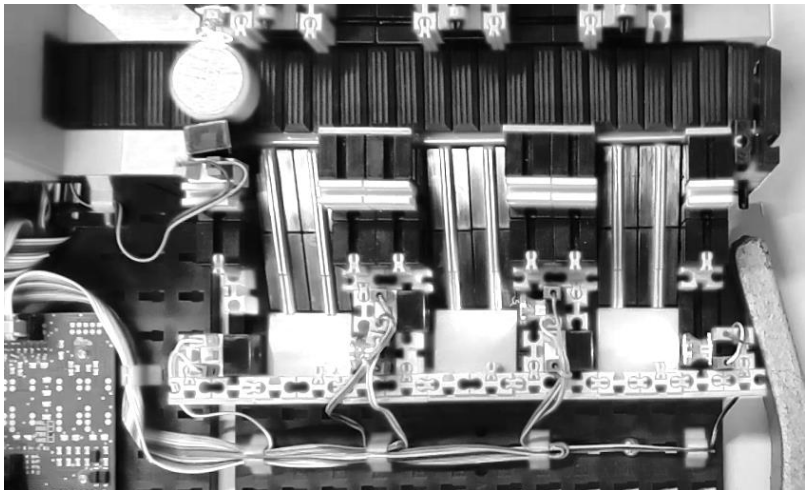


Figure 23

5.7.2 Discussion of results

The attempt to use HSV did not meet with any success as the cargo was not able to be found. This is due to the fact that, while the image is stored and therefore represented in a different way, there is still not enough contrast between the cargo and the background. This means that the function 'imfindcircles' (Mathworks, 2012) is unable to find the cargo, as there is an insufficient difference between the cargo and the conveyor belt.

Further tests included exploring different combinations of the three channels (Hue, Saturation and Value), by extracting and multiplying them. An example of this was testing the product of the hue and the saturation channels. However, these attempts were also unsuccessful as the cargo could not be sufficiently distinguished from the background.

The use of HSV was then tested on the video data. When tested on full HSV image, the hue frame and the saturation frame none of the three colours of cargo were ever able to be found, giving a success rate of 0%. However, when the code to find circles was run on just the value frame the white cargo was able to be found approximately 60% of the time, the red cargo 65% and the blue cargo 10%. While this was not as good at finding the white cargo as the method outlined in section 5.4, it was better at finding the red and blue cargos, particularly the red cargo.

5.8 Comparing to a control frame

5.8.1 Implementation

Following the previous failures, a different approach was needed. It was considered that one could compare each frame to be analysed against a control frame to examine the change. In this case the change sought would be the cargo within the frame, as the control frame would need to be a frame that could be guaranteed to not have the cargo within it. Finding the difference between two frames can be done with a simple subtraction:

```
frameDiff = frame-controlFrame;
```

This needs to be followed by finding the absolute values, as the actual difference is needed, not whether the difference is positive or negative. This can be done via the simple execution of this line:

```
frameDiff = abs(frameDiff);
```

It is important to note that either immediately preceding or following these operations the matrix containing the difference values should be consolidated into a 2-dimensional matrix. Or in other words, converted to greyscale.
Subsequently the matrix containing absolute difference values should have a threshold applied to it, such that the majority of noise such as minor lighting changes is removed, and only large differences remain.

Figure 24

Figure 24 shows an example of the frame after the aforementioned actions have been applied to it, the threshold value used in this case was 130. The reader may note that this has not removed all of the noise and there are lines within the cargo. Lowering the threshold value would help to remove the lines from the cargo but would increase the amount of noise present in the image. Conversely raising the threshold value would help to remove noise but would remove even more of the cargo.

Looking at Figure 16 it is clear that something must be done to try and clean up the image. The use of erosion and dilation was needed for this purpose. Erosion is when a pixel's value is changed to 0 if any of the neighbouring pixels have the value of 0. This serves to trim away at the shapes in an image. Dilation is the opposite, if enough neighbours have the value 255 then the pixel in questions value is also turned to 255. A combination of these two procedures was thought to help smooth out the image and allow the cargo to be found.

5.8.2 Discussion of results

Unfortunately, this method was unsuccessful is enabling the cargo to be found as the percentage success rate was 0% for all cargo colours. A wide variety of combinations of dilation and erosion were tested, from the order in which they were applied to the number

of times each was used. If too much erosion is used, then objects can become too small and 'growing' them back to their original size becomes very challenging. If too much dilation is used then any noise present in the image, unless removed previously via erosion or otherwise, will become much larger and cause much more of an issue.

During extensive testing it was realised that the image should first be dilated, eroded and then dilated again. This is done so that the overall shape and size of the object trying to be found is preserved. If too much dilation is applied areas of interest, in this case circles, in an image tend to become more quadrilateral in shape (see Figure 25). This is obviously not desirable for this application as the object being searched for is circular in shape.

Figure 25

5.9 Labelling regions and removing small ones

5.9.1 Implementation

Rather than trying to erode then dilate an image to eliminate the noise why not simply remove any area in the image with a small enough area to be considered noise? Built into MATLAB's Image Processing Toolbox is a function called 'bwlabel', which takes an image, (must be a black and white image), finds all the areas in an image and assigns a label (or index) to them (Mathworks, 2006).

Figure 26

Figure 26 shows a test image which has had the 'bwlabel' function applied to it. The gradient used helps to distinguish between the different regions. In order to apply this gradient simply execute the line:

```
imshow(t,[])
```

Following this it is a simple task to find and remove any areas with a small enough area, a value of 1000 pixels was arbitrarily chosen for this.

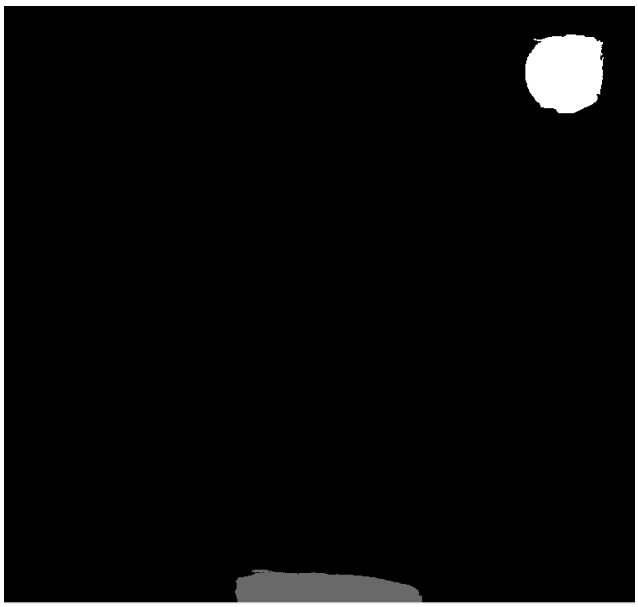


Figure 27

Figure 27 shows the image after the removal of all objects with an area smaller than 1000 pixels, as can be seen much of the noise has been removed, with only the cargo in the top

right and another object along the bottom remaining visible. This illustrates that this is an effective method of removing noise from an image. The code for this can be found in Appendix 3.

After this some cleaning up of the cargo was still required, as the edges were uneven which would prevent the cargo from being found.

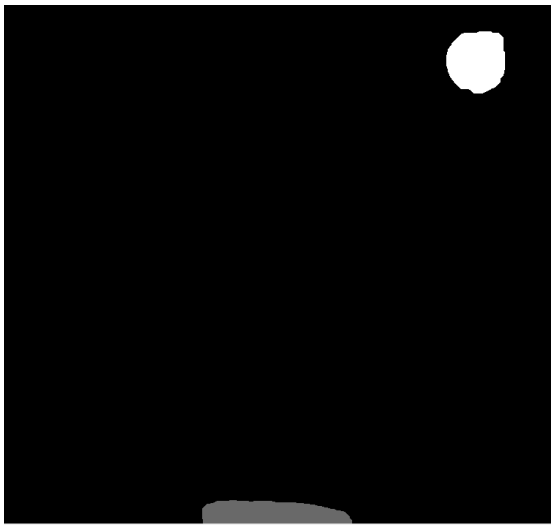


Figure 28

Figure 28 shows the image after having the functions 'imopen' (Mathworks, 2006) and 'imerode' (Mathworks, 2006) applied to it. As shown the edges are less jagged but are still not completely smooth.

5.9.2 Discussion of results

The reasoning behind using this technique was to remove noise from the image without using erosion and dilation, because as mentioned previously in section 5.8, the use of these techniques can cause circular objects to become more quadrilateral in shape. The use of the function 'bwlabel' (Mathworks, 2006) as well as a simple loop proved very effective in removing the vast majority of the noise in a given image (Figure 27). Unfortunately, while it does demonstrate a useful technique, this method was ultimately unsuccessful in finding the cargo as the percentage success rate was 0% for all cargo colours. This was because, as mentioned previously, the cargo was not circular enough to be found.

5.10 Combining finding the difference with removing small regions

5.10.1 Implementation

Building on the previous section it was decided to attempt to combine finding the difference between a given frame and a control frame, with labelling regions in an image and removing ones below a certain size. After much experimentation (while testing using the white cargo), the Hue frame was chosen as the difference gave the clearest resultant image. Figure 29 shows an example of this. In addition, the image was also cropped such that only areas where the cargo could, the conveyor belt and the crane bays, be would be visible in the image, thus removing a great deal of extra noise and speeding up the program.

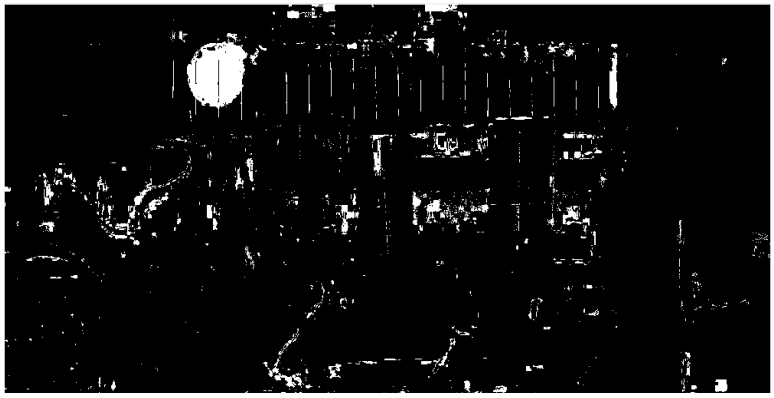


Figure 29

The next step was to remove the noise. As mentioned previously this was to be done via the use of the 'bwlabel' function (Mathworks, 2006), to label all the regions in an image, followed by again making use of the code found in Appendix 3 to remove any region smaller than a certain size, in this case 1000 pixels.

Figure 30

Figure 30 shows the result of this implementation. As the reader may note the application of this technique has removed almost all the noise in the image, leaving just the cargo with some extra lines and some gaps. However, the majority of this can be cleaned up with some erosion and dilation. Figure 31 shows the image after the functions 'imopen' (Mathworks, 2006) and 'imerode' (Mathworks, 2006) have been applied. As can be seen while the extra lines have been cleaned up the cargo now has several blob-like shapes along the edges and as such the cargo could not be found.

Figure 31

After cleaning up the image slightly the cargo should have then been found. However, the function 'imfindcircles' (Mathworks, 2012) was unable to find the cargo. A different technique was needed. It was thought to create a system which would find the average x & y coordinates of the white pixels present in the image. Then find the average radius of white pixels from the epicentre in the four compass point directions. The code for this can be found in Appendix 4.

5.10.2 Discussion of results

Figure 32 shows the code in Appendix 4 working on the image shown in Figure 30, as the steps taken in Figure 31 do not really change anything they were not applied. The circle is successfully found, even if the location isn't 100% accurate. However, there are some issues. If there is any noise left in the image the centre and radius of the cargo will be distorted, leading the system to incorrectly believe it had found the cargo. Additionally, if there is no difference in the image then nothing will show up causing the cargo to again not be found. This system had a percentage success rate of 50% for the white cargo, 10% for the red cargo

and 30% for the blue cargo. The frame used as well as the set threshold value were changed around but gave no higher success rate.

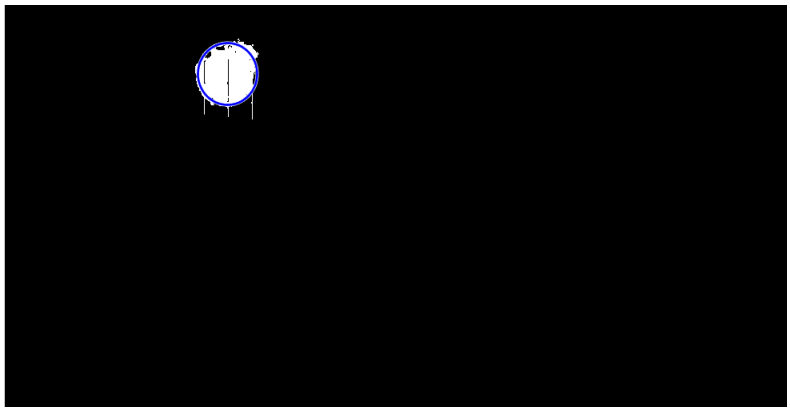


Figure 32

5.11 Data gathering with controlled lighting

5.11.1 Implementation

Given that all previous attempts to solve the issue of finding the blue and red cargo with any degree of reliability were met with failure, it was decided to go back to the data gathered in section 5.3 and examine it more closely. Upon performing some frame by frame visual inspections based on where the code was able to find the cargo and where it was not, it became clear that lighting was the main issue. As such it was necessary to perform further tests.

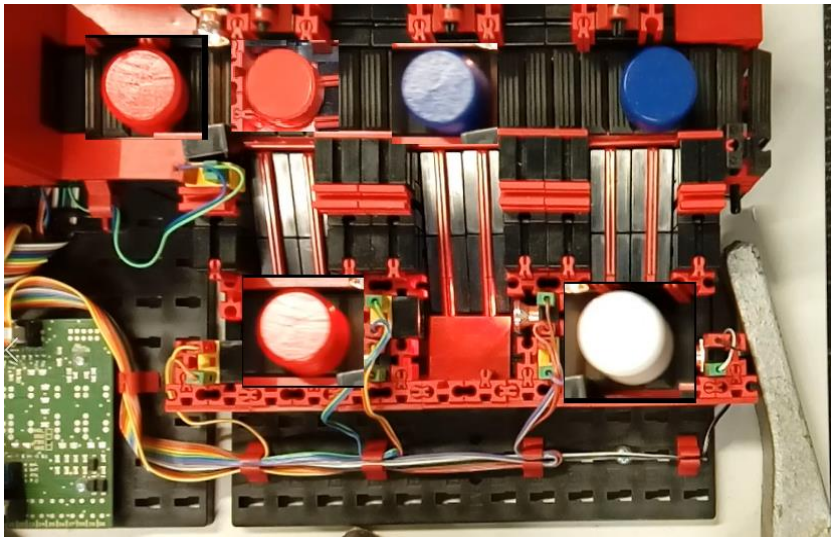


Figure 33

Figure 33 shows the red, white and blue cargoes at various levels of brightness (note that they have been pasted over from different images and thus the location in the image is not indicative of anything). As the reader will note, especially in the case of the blue cargo, the lighting can have a drastic effect on the perceived colour of an object.

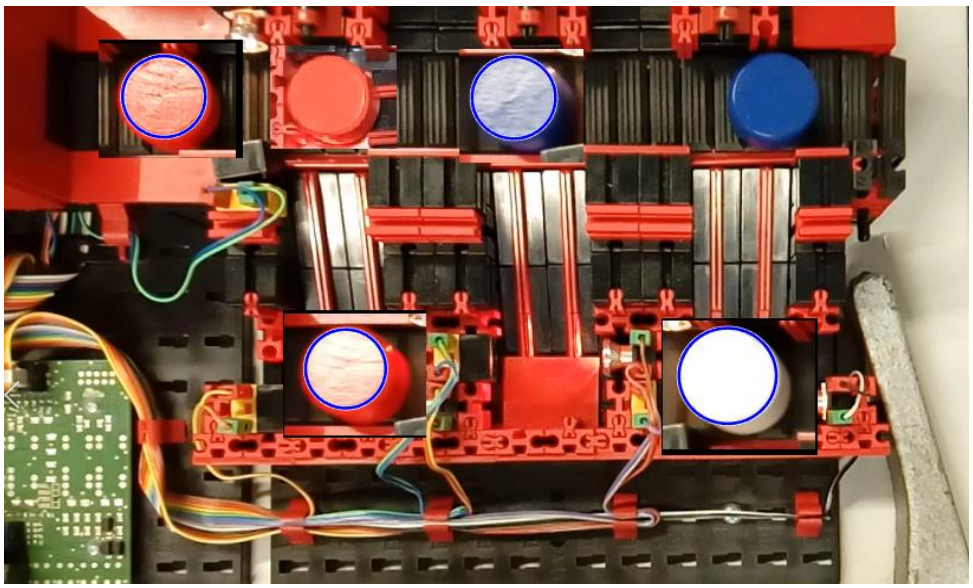


Figure 34

Figure 34 shows the previous image after the function 'imfindcircles' (Mathworks, 2012) has been run. As expected, the white cargo was found with no issue, but an interesting observation was that some of the red and blue cargoes were also successfully found. Upon closer inspection it became apparent that the red and blue cargoes were found when the lighting on them created a sheen. The more light that was shining onto the cargo, the more it reflected, the whiter it appeared and thus the easier it was to find.

Following this discovery, the next step was to regather the data, but this time making sure the lighting was more controlled in order to ensure that all the different coloured cargoes could always be found. Due to time constraints it was decided to focus the efforts of this endeavour onto 'Stage 3" of the model factory as it has the longest uninterrupted belt to track the cargo along. This was done primarily to make sure that shadows were limited and thus would not interfere with the image. Originally, this was to be done by turning off the ceiling lights and drawing the blinds, and then using two lamps, one to illuminate the belt and another to illuminate the bays where the cargo waits for the crane. However, upon

testing it quickly became apparent that this was not enough as at several places along the belt and in the crane bays the blue and red cargo could still not be found.

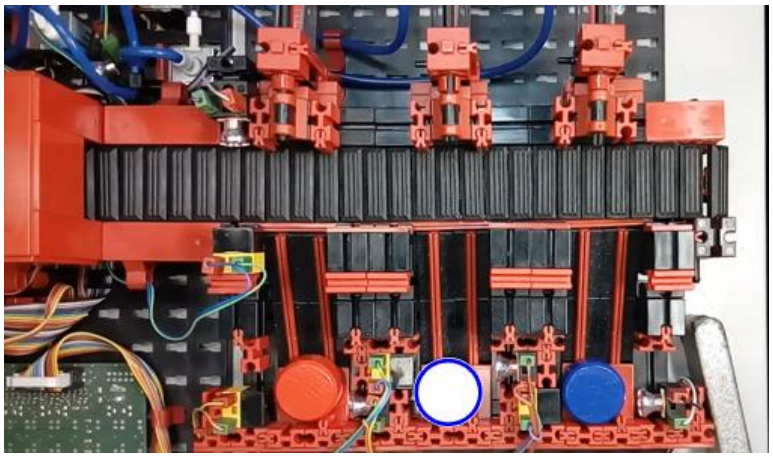


Figure 35

Figure 35 shows the three different cargoes in their respective bays, as can be seen only the white one could be found, due to insufficient lighting. In order to resolve this, four lamps were used to ensure that both the belt and the bays were sufficiently illuminated for the cargo to be found. Whilst solving this it became apparent that in order to fully light up the bays a lamp had to be positioned very close to it. Figure 36 shows an example of this with the blue cargo having been found, as the reader may note the blue cargo looks completely white due to the reflected light.

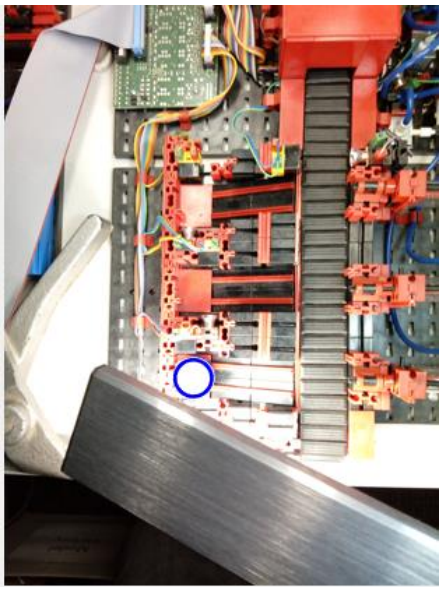


Figure 36

5.11.2 Discussion of results

With the lighting system shown below in Figures 37 and 38 being used to ensure the cargo was always reflecting as much light as possible, additional videos of the red and blue cargoes were taken and the code in Appendix 1 run on them. All the coloured cargoes could then be found at all locations along the main belt, as well as in the crane bays. The success rate for this method was 85% for the white cargo, 86% for the red cargo and 80% for the blue cargo. As all cargoes can now be found in the majority of locations with fairly equal success rates, objectives 4 and 5 have been completed, thus successfully concluding the technical aspect of this project.

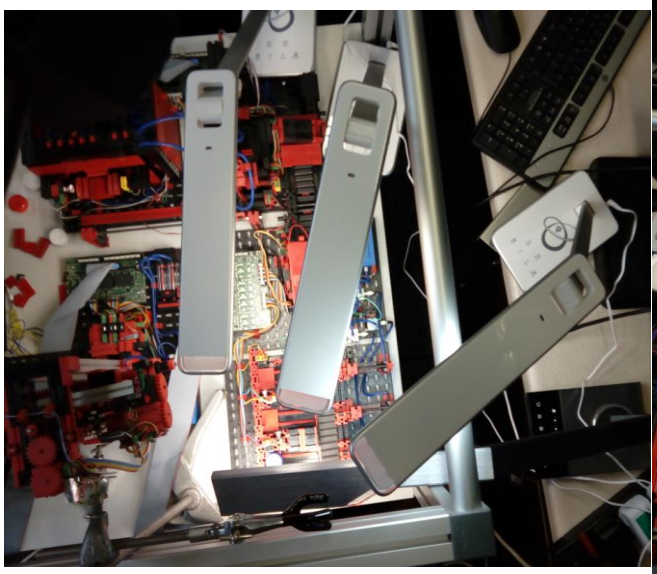


Figure 37

Figure 38

6 Analysis and Evaluation

6.1 Discussion of results

Section	Technique	Success Rate for White (%)	Success Rate for Red (%)	Success Rate for Blue (%)
5.4	From colour video	83	5	0
5.5	Thresholding	0	0	0
5.6	Edge Detection	70	5	5
5.7	HSV (all frames)	0	0	0
5.7	Hue frame	0	0	0
5.7	Saturation frame	0	0	0
5.7	Value frame	60	65	10
5.8	Comparing to a control frame with erosion/dilation	0	0	0
5.9	Removing small regions	0	0	0
5.10	Combining comparing to a control frame with removing small regions	50	10	30
5.11	Additional lighting	85	86	80

Table 1

As described throughout section 5 of this report, many different techniques were used in an attempt to find the red, white and blue cargoes, as they travel along the main belt of the factory. Table 1 shows of a summary of the different techniques used and their success rate.

The work conducted in section 5.2 was critical in proving the project's viability by demonstrating the successful use of the function 'imfindcirlces' (Mathworks, 2012) to find the cargo in still images.

The next integral part of the project was the data collection outlined in section 5.3. This data would serve as the means of testing the techniques outlined in sections 5.4-5.10 without which the project would not have progressed.

The technique discussed in section 5.4 and shown in appendix 1 was a highly important one that would form the basis of sections 5.5-5.10 by creating the skeleton structure for the code to extract a frame every given time interval and analysing it to search for the cargo.

Applying the thresholding algorithm described in section 5.5 produced a good image with a reasonably low amount of noise. However, this technique proved ineffective at allowing the

cargo to be found, due to it not being circular and meant that none of the cargo colours could be found.

Section 5.6 details the use of the 'edge' function and the various filters in MATLAB (Mathworks, 2011). While this technique proved useful in helping to give more understanding on how the function 'imfindcircles' works, as it essentially takes an image and breaks it down into edges in order to find circles (Mathworks, 2012), it neither improved nor decreased the performance of the system in finding the cargo.

The use of HSV (Hue, Saturation and Value) presented in section 5.7, and the research carried out during it, granted a great deal of insight into the way colour can be captured and displayed in order to best highlight certain attributes or features. Unfortunately, the use of this technique meant that none of the cargo colours could be found, regardless of what channel was used.

Comparing a given frame to a control frame by performing a subtraction, finding the absolute values and then thresholding as demonstrated in section 5.8, produced images with little noise, but the cargo becomes very jagged and the cargo cannot be found. This was then followed by applying erosion and dilation to the image, however this tends to have the effect of causing the cargo to become more quadrilateral in shape, as well as causing any noise left to grow.

Section 5.9 details the attempt to get around the problem of removing noise via erosion, and the issues this cause the desired object, by using the 'bwlabel' function (Mathworks, 2006) to label all the regions in an image and remove those with a small enough size. This method was very successful in removing most of the noise in an image; however, the cargo was still too jagged to be found.

Combining the previous two sections (5.8 and 5.9) together in section 5.10, as well as cropping the image, proved more successful. The difference was found by performing a subtraction and then calculating the absolute values and applying a threshold, the small regions were then removed as outlined previously using 'bwlabel' (Mathworks, 2006). This technique proved very good at removing all of the noise in an image, however the function 'imfindcircles' (Mathworks, 2012) was still unable to find the cargo.

In order to find the cargo a different method was needed. It was decided to calculate the average x & y coordinate of the white pixels in the image, then finding the distance in each of the four compass points and calculating the average to find the radius. Using this method, the cargo could be found with relative success, however this method does not work well if there is any noise left in the image, or for instance the crane moves into the image.

In section 5.11, after more closely examining the original data collected in section 5.3, it was noticed that the locations where the lighting was brighter made the cargo appear shinier which then enabled the red and blue cargoes to be found.

Using this information new data was collected using multiple lamps to ensure that the whole belt as well as the crane bays were as illuminated as possible. This causes the cargo to be very reflective which in turn enables the function 'imfindcircles' (Mathworks, 2012) to find the cargo at all points.

This shows that while it is possible to find the cargo using image processing techniques, simply controlling the lighting is a far simpler solution.

These findings have demonstrated just how important lighting is when it comes to machine vision. Setting up the environment for data collection by controlling lighting, the colour of the background and the colour of the object, are all very important factors. Using reflective objects and a non-reflective background can also further ensure the contrast between the object and the background is more pronounced.

6.2 How could the results be used in a real setting

Based on the findings of this project it is completely possible for a machine vision-based position verification and object tracking system to be implemented inside a real factory. This is shown in section 5.11 where all the colours of cargo were able to be found at all points along the belt and in the crane bays in that section of the model factory. However, there would be some issues to be tackled in order for it to be successfully implemented.

The system was only successful when the lighting was heavily controlled therefore, using this solution, if the desired outcome is to track cargo moving through a factory then the lighting must be controlled throughout. This presents its own issues, as some factories can have a floor area greater than $10000m^2$, and controlling the lighting for all that area would

be a very difficult and expensive task. This could be made even more challenging depending on what type of lighting is necessary, as some are more expensive than others. It may also be necessary, as part of controlling the lighting, to remove or block out other sources of lighting such as the sun as they may cast unwanted shadows. This could add further expenses as additional opaque structures may need to be built inside the factory which could also limit access for maintenance crews. In addition, if there are workers in the factory considerations would have to be taken for their health and safety. For instance, if there was very bright lighting throughout then the factory owners may have to provide workers with protective eyewear to minimise eye strain.

A potential alternative to using a bright white light, which is visible and potentially dangerous to humans without protective eyewear, could be to use infrared or near infrared light to illuminate the factory. This method would have the additional benefits of being more easily controlled than visible light, as well as being far less intrusive to workers due to it not being visible. However, it would still have safety concerns because infrared light, or radiation, is essentially heat and as such could present danger to workers. In addition, the use of infrared illumination would lack the colour discriminating properties of RGB or HSV images which, depending on the task, may make its use infeasible.

As discussed throughout section 5 when the cargo appeared whiter in colour it was able to be found much more easily as the background was mainly dark in colour. As such one potential alternative to a highly controlled lighting system would be to ensure that the cargo being tracked contrasted with the background as much as possible. However, depending on exactly what the factory is producing this may not be practical or even possible.

Another solution could be to give each object moving through the factory a barcode or label which could be easily identified by the image processing software. While this is commonly done in places such as distribution warehouses on finished products, applying this technique to certain parts of a production line can be significantly less feasible as barcodes are often added to packaging but placing barcodes on items before they are packaged is not usually viable.

The most pragmatic use of these findings in industry would be to focus on a small part of the production line, as demonstrated during the project where the primary focus was on a

long straight piece of conveyor belt. This would mean that the lighting would only need to be controlled in a proportionally small part of the factory, reducing setup and operational costs.

7 Critical Thinking

In terms of the initial plan laid out in the original Gantt chart not much has changed. Due to the Covid-19 outbreak in early 2020 university buildings, including the laboratory where the model factory is located, were closed. This could have potentially caused some issues regarding finishing the technical work, because as discussed in section 5.11 the data needed to be regathered which obviously would require the model factory. As a result, finishing the technical work became a priority and was finished as promptly as possible. In addition, the submission date of the final report was moved from 02/04/20 to 23/04/20. While this afforded an additional three weeks, due to other deadlines not being changed it resulted in no real additional time available to be allocated. This however allowed more time for proof-reading and formatting.

The viva, which took place in early March 2020, was an opportunity to discuss work done to date and what to do moving forward with the supervisor and a second marker. From this discussion it was clear that several methods had been tried and tested and the realisation that in order to solve the issues faced the data needed to be regathered with more lighting was correct. This solution was carried out in section 5.11 and proved to be a successful final solution.

The final Gantt chart has been updated to more fully represent when certain tasks were carried out, as well as to show new dependencies. The original Gantt chart can be found in Appendix 5, while the updated version can be found in Appendix 6. Most tasks went according to the original plan, however some tasks such as writing the final report ended up taking longer than initially planned.

The Risk Assessment Form and the Project Resources Form, found in Appendix 7 and Appendix 8 respectively, have not be changed from the original versions submitted with the interim report. This was because no additional risks became apparent during the course of the project and no additional resources were required in order to complete the project.

This project has met the criteria for several components of the UK-SPEC for the Incorporated Engineer (IEng). While there is evidence gathered during this project for most of the components, the two main ones for this project are components A and B. Component

A is about showing general and specialist engineering knowledge by maintaining and extending theoretical knowledge, developing solutions to new problems and solving issues with a high level of risk. This has been demonstrated within this project by performing background research, reading and analysing relevant papers, on the techniques outlined in section 5 such as HSV, edge detection and thresholding. Thresholding had been studied and used previously, this knowledge was built upon and employed to remove the issue of the colour of the cargo. In addition, some thought was given to the fact that any tracking system should be able to run in real time, and as such any code written should be as efficient as possible. This is shown by the fact that the code in Appendix 1 has a runtime only slightly longer than the length of the video. Some of this will be due to the code for plotting the graphs and other accessories, but this shows that it is possible to create a real-time system, as this project was a proof of concept.

Component B focuses on identifying and defining project requirements, recognizing appropriate research that needs to be conducted and implementing solutions and evaluating their effectiveness. In the case of this project this was demonstrated by choosing an appropriate development language (MATLAB), testing different use cases (different coloured cargoes), carrying out risk assessments, collecting data and then changing the way the data was collected in order to improve the reliability of the system.

The project also met some of the specifications in component E of the UK-SPEC. Component E is about displaying commitment to professional standards such as codes of conduct, being knowledgeable on relevant safety standards and ethical issues, ensuring all work is done to sustainable standards as well as recording professional development evidence. This was shown by the fact that a logbook was maintained throughout the course of the project, ensuring that the fool-proof stop button was in close proximity at all times in case of emergency and by considering the ethical implications of the work conducted and the result of the project.

While this project did not have any major ethical concerns as it did not involve testing on human or animal tissue, there are some ethical implications to be considered. Perhaps the most obvious ethical issue would be that this technology could cause a loss of jobs as human operators are replaced with a machine vision system. This could severely damage

families' livelihoods, possibly removing their only source of income. However, many of these more repetitive jobs could be replaced by other roles, such as technicians to repair and maintain any such machine vision system. Furthermore, the use of this technology could also improve people's working conditions and remove people from dangerous or unpleasant work environments. For instance, a system could be designed to operate within a freezer or an oven, places where workers cannot go or where conditions are not ideal for humans.

In terms of environmental consequences there are several points to consider:

A cargo tracking system implemented in a factory could result in a desirable reduction of defective products which could then be recycled reducing waste.

However, the mining or harvesting of raw material and its associated transportation, the running of the factory itself, as well as the transport medium for distribution of the final product, all have a carbon footprint associated with them. If these were to increase due to a higher rate of production, then the carbon footprint of the whole process would grow. While this could potentially be circumnavigated by ethically sourcing raw materials and ensuring that electricity used was generated using renewable sources, this may not be possible or economically viable.

As described previously a system to track cargo throughout a factory would be expensive to implement, and as such would require wealthy companies or governments willing to invest. This could contribute to a greater gap between the rich and poor, because those able to afford it could reap the benefits of reduced waste products and a higher output from factories.

8 Conclusions

For the most part the project stayed true to the original overall plan, as can be seen by comparing the initial and final Gantt charts shown in Appendices 5 and 6 respectively. The number of individual techniques was not in itself planned but evolved due to the challenges faced as the project progressed. However, the time allocated for the technical work to be completed was reasonably accurate.

The bulk of the tested solutions outlined in section 5 proved unnecessary, particularly things such as the use of HSV (Hue, Saturation and Value) or thresholding, as the final solution was to simply increase the lighting and regather the data. While these techniques in themselves did not contribute to the final solution, they were worthwhile investments of time as they served as important investigations into commonly performed tasks in the field of machine vision.

There are several recommendations for further work which could not be accommodated within the scope and time scales of this project, but which could be appropriate future additions.

Multiple cameras could be incorporated into a machine vision system to track an object moving through a factory. This could potentially allow for a greater degree of accuracy as the coordinates of the object could be verified from multiple points and angles. In addition, this would add a level of redundancy to the system, allowing for the malfunction of one or more cameras. Multiple camera points could also allow for the tracking of three-dimensional complex objects regardless of which way up they were as they moved along a conveyor belt.

Implementing a template matching system which finds template images within the larger image could also be used to detect more complicated objects. A deep learning algorithm could also be incorporated to make the matching more efficient by using multiple layers of matching to extract different features. Lower layers would identify features such as edges while higher layers would identify more abstract concepts such as digits or shapes.

Although there were some issues tracking the cargoes, these were eventually overcome by additional lighting and so overall the project proceeded as planned. As the cargo was then

able to be found in the majority of locations the aims of this project were successfully attained and the project was a deemed a success.

9 References

Anon., 1978. *Color Gamut Transform Pairs.* 2nd ed. New York: ACM.

Anon, 2019. *CBINSIGHTS.* [Online]
Available at: https://www.cbinsights.com/research/future-factory-manufacturing-tech-trends/
[Accessed 3 April 2020].

Atherton, T. J. & Kerbyson, D. J., 1998. Size invariant circle detection. *Image and Vision Computing,* 17(11), p. 795–803.

Bhaskaran, S. et al., 2013. Mapping shadows in very high-resolution satellite data using HSV and edge detection techniques. *Applied Geomatics,* 5(4), pp. 299-310.

Canny, J., 1986. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 8(6), pp. 679-714.

Edinbarough, I., Balderas, R. & Bose, S., 2005. A vision and robot based on-line inspection monitoring system for electronic manufacturing. *Computers in Industry,* 56(8-9), pp. 986-996.

Freitas, A. D. et al., 2016. Autonomous crowds tracking with box particle filtering and convolution particle filtering. *Automatica,* July, Volume 69, pp. 380-394.

Gonzales-Barron, U. & Butler, F., 2006. A comparison of seven thresholding techniques with the k-means clustering algorithm for measurement of bread-crumb features by digital image analysis. *Journal of Food Engineering,* 74(2), pp. 268-278.

Gopalan, R. & Jacobs, D., 2010. Comparing and combining lighting insensitive approaches for face recognition. *Computer Vision and Image Understanding,* 114(1), pp. 135-145.

Harrison, R., Vera, D. & Ahmad, B., 2016. Engineering the smart factory. *Chinese Journal of Mechanical Engineering,* 29(6), pp. 1046-1051.

Hui-Fuang, N., 2006. Automatic thresholding for defect detection. *Pattern Recognition Letters,* 27(14), pp. 1644-1649.

Jahari, M. et al., 2015. Double Lighting Machine Vision System to Monitor Harvested Paddy Grain Quality during Head-Feeding Combine Harvester Operation. *Machines,* 3(4), pp. 352-363.

Joblove, G. H. & Greenburg, D., 1978. Color spaces for computer graphics. *Computer Graphics,* 12(3), pp. 20-25.

Kita, Y., Ishikawa, H. & Masuda, T., 2017. Guest Editorial: Machine Vision Applications. *International Journal of Computer Vision,* 122(2), pp. 191-192.

Le, W., 2018. Application of Wireless Sensor Network and RFID Monitoring System in Airport Logistics. *International Journal of Online Engineering,* 12(1), pp. 89-103.

Mahapatra, P. K., Thareja, R., Kaur, M. & Kumar, A., 2015. A Machine Vision System for Tool Positioning and Its Verification. *Measurement and Control,* 48(8), pp. 249-260.

Mathworks, 2006. *bwlabel.* [Online]
Available at: https://uk.mathworks.com/help/images/ref/bwlabel.html
[Accessed January 2020].

Mathworks, 2006. *imerode.* [Online]
Available at: https://uk.mathworks.com/help/images/ref/imerode.html
[Accessed Febuary 2020].

Mathworks, 2006. *imopen.* [Online]
Available at: https://uk.mathworks.com/help/images/ref/imopen.html
[Accessed Febuary 2020].

Mathworks, 2006. *imresize.* [Online]
Available at: https://uk.mathworks.com/help/matlab/ref/imresize.html
[Accessed December 2019].

Mathworks, 2011. *edge.* [Online]
Available at: https://uk.mathworks.com/help/images/ref/edge.html
[Accessed January 2020].

Mathworks, 2012. *imfindcircles.* [Online]
Available at: https://uk.mathworks.com/help/images/ref/imfindcircles.html#d118e148136
[Accessed December 2019].

McAndrew, A., 2004. *An Introduction to Digital Image.* Melbourne: Brooks/Cole.

Montironi, M., Castellini, P., Stroppa, L. & Paone, N., 2014. Adaptive autonomous positioning of a robot vision system: Application to quality control on production lines. *Robotics and Computer-Integrated Manufacturing,* 30(5), pp. 489-498.

Mukhopadhyay, P. & Chaudhuri, B. B., 2015. A survey of Hough Transform. *Pattern Recognition,* 48(3), pp. 993-1010.

Nobuyuki, O., 1979. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics,* 9(1), pp. 62-66.

Owen-Hill, A., 2016. *Robot Vision vs Computer Vision: What's the Difference?.* [Online]
Available at: https://www.roboticstomorrow.com/article/2016/07/robot-vision-vs-computer-vision-whats-the-difference/8484/
[Accessed 30 March 2020].

Poynton, C., 1996. *A Technical Introduction to.* 1st ed. New York: John Wiley & Sons.

Russell, P. & Batchelor, D., 2001. *SEM and AFM: Complementary Techniques for Surface Investigations,* Santa Barbara: Digital Instruments/Veeco Metrology.

Schechner, Y. Y., Nayar, S. K. & Belhumeur, P. N., 2007. Multiplexing for Optimal Lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 29(8), pp. 1339-1354.

Sun, T.-H., Tseng, C.-C. & Chen, M.-S., 2010. Electric contacts inspection using machine vision. *Image and Vision Computing,* 28(6), pp. 890-901.

Umbaugh, S. E., 2010. *Digital Image Processing and Analysis: Human and Computer Vision Applications with CVIPtools.* 2nd ed. Boca Raton: CRC Press.

Ziou, D. & Tabbone, S., 1998. Edge detection techniques: An overview. *International Journal of Pattern Recognition and Image Analysis,* 8(4), pp. 537-559.

10 Appendices

Appendix 1

//

```
%% Machine Vision code for final year project "Can machine vision improve efficiency in manufacturing?"
% This code finds the cargo at regular points in a video
% By Luca Ricagni
%% Clear
close all
clear
clc
%% Import video
% Stage one videos
% V = VideoReader('StageOneWhite.mp4');
% Stage two videos
% V = VideoReader('StageTwoRed.mp4');
% V = VideoReader('StageTwoWhite.mp4');
% V = VideoReader('StageTwoWhite2.mp4');
% Stage three videos
% V = VideoReader('StageThreeBlue.mp4');
% V = VideoReader('StageThreeRed.mp4');
% V = VideoReader('StageThreeRed2.mp4');
V = VideoReader('StageThreeWhite.mp4');
% V = VideoReader('StageThreeWhite2.mp4');
% V = VideoReader('StageThreeWhite3.mp4');

%% Declare variables (these can potentially be edited)
percentageDiff = 1.1; % Difference you look for to decide what data to use for line of best fit
timeInterval = 0.5; % Interval for frames in seconds
```

```
%% Declare other variables
frameIncrementSize = round((V.NumberOfFrames/V.Duration)*timeInterval); % Calculate
frames per 0.5seconds
a = round(V.NumberOfFrames/frameIncrementSize); % Calculates how many values to store
centreCoordinates = zeros(a,2); % Initialise matrix to store centre coordinates
frameNumbers = zeros(a,1); % Initialise matrix to store the frame numbers
counter = 1; % Used in for loop
%% Loop to go through image in specified increments
for i=1:frameIncrementSize:V.NumberOfFrames
   frame = read(V,i);

   imshow(frame);

   %% Find the cargo (if it is there) using Hough Circle Transform

   [centers, radii, metric] = imfindcircles(frame, [36 60], 'ObjectPolarity', 'bright');

   %% Draw blue outlines on the circles

   % Find number of circle centers
   b = numel(centers);
   numOfCircles = b / 2;

   viscircles(centers, radii,'EdgeColor','b'); % Draws the blue outlines
   frameNumbers(counter,:) = i;
   if(numOfCircles > 0)
      centreCoordinates(counter,:) = centers; % Stores centre coordinates
   end
   counter = counter + 1;
end
%% Plot in 3d
```

```
scatter3(frameNumbers,centreCoordinates(:,1), centreCoordinates(:,2), 'x')
```

//

Appendix 2

- Read in image
- Convert to greyscale
- for(each pixel in array)
    - Add up greyscale values to calculate mean
    - Add up number of each greyscale value in image
- Calculate middle pixel (to calculate median)
- Find the pixel value which contains the median greyscale value
- Calculate mean greyscale value
- Threshold = mean
- Display mean and median greyscale values
- while(unsolved)
    - for(each pixel in array)
        - if(pixel value < threshold)
            - add to lower mean
        - else
        - add to upper mean
    - mean = mean of lower and upper mean
    - if(mean = thresh)
        - break out of loop
    - else
        - reset counts
        - thresh = mean
- Display iteratively derived threshold value

//

Appendix 3

```
% Remove objects with an area less than minSize
```

```
t = bwlabel(frameInUse);
minSize = 1000;
for k = 1:max(t(:))
    a = t == k;
    if sum(a(:)) < minSize
        t(t==k) = 0;
    end
end
```

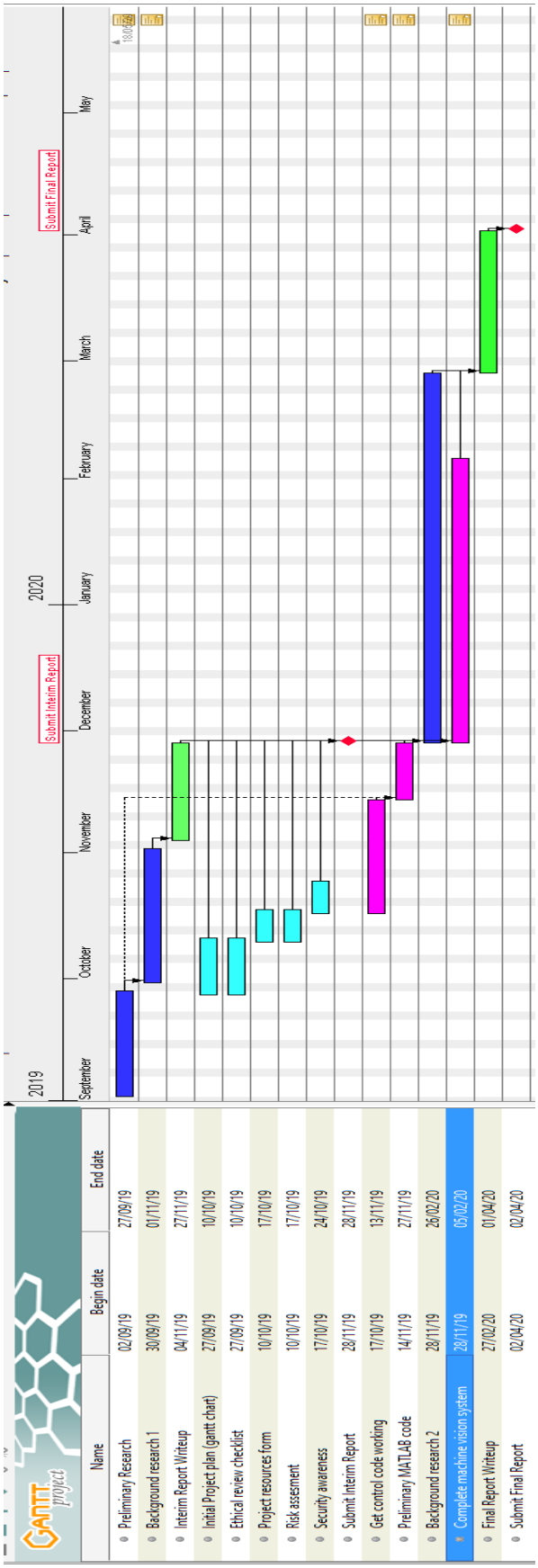
//

Appendix 4

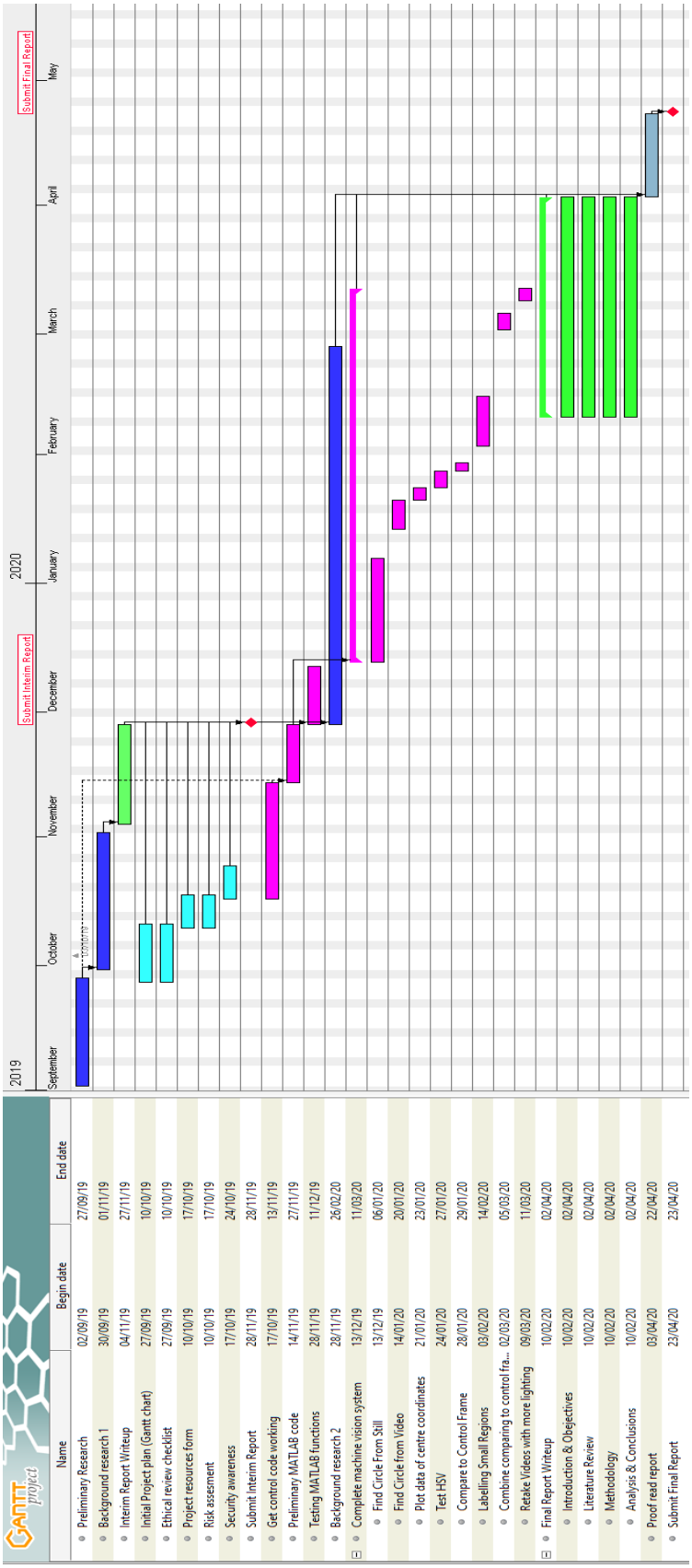
```
totalX = 0;
totalY = 0;
count = 0;

for n=1:500
    for m=1:1000
        if t(n,m) ~= 0
            totalX = totalX + m;
            totalY = totalY + n;
            count = count + 1;
        end
    end
end

centers(1,1) = round(totalX/count);
centers(1,2) = round(totalY/count);

centreCoordinates(counter,1) = centers(1,1);
centreCoordinates(counter,2) = centers(1,2);
counter = counter + 1;
```

```
if isnan(centers(1,1)) || isnan(centers(1,2))
    display('No Circles')
else
    distance1 = 0;
    for n=centers(1,1):-1:1
        if t(centers(1,2), n) ~= 0
            distance1 = distance1 + 1;
        end
    end
    distance2 = 0;
    for n=centers(1,1):1000
        if t(centers(1,2), n) ~= 0
            distance2 = distance2 + 1;
        end
    end
    distance3 = 0;
    for n=centers(1,2):-1:1
        if t(n, centers(1,1)) ~= 0
            distance3 = distance3 + 1;
        end
    end
    distance4 = 0;
    for n=centers(1,2):500
        if t(n, centers(1,1)) ~= 0
            distance4 = distance4 + 1;
        end
    end
    radii = round((distance1 + distance2 + distance3 + distance4)/4);
    viscircles(centers, radii,'EdgeColor','b');

end
```

Appendix 5


GANTT project
Name
Begin date
End date
Preliminary Research 02/09/19 27/09/19
Background research 1 30/09/19 01/11/19
Interim Report Writeup 04/11/19 27/11/19
Initial Project plan (gantt chart) 27/09/19 10/10/19
Ethical review checklist 27/09/19 10/10/19
Project resources form 10/10/19 17/10/19
Risk assesment 10/10/19 17/10/19
Security awareness 17/10/19 24/10/19
Submit Interim Report 28/11/19 28/11/19
Get control code working 17/10/19 13/11/19
Preliminary MATLAB code 14/11/19 27/11/19
Background research 2 28/11/19 26/02/20
Complete machine vision system 28/11/19 05/02/20
Final Report Writeup 27/02/20 01/04/20
Submit Final Report 02/04/20 02/04/20
2019
September
October
November
December
Submit Interim Report
2020
January
February
March
April
Submit Final Report
May

Appendix 6


GANTT project
Name
Begin date
End date
Preliminary Research 02/09/19 27/09/19
Background research 1 30/09/19 01/11/19
Interim Report Writeup 04/11/19 27/11/19
Initial Project plan (Gantt chart) 27/09/19 10/10/19
Ethical review checklist 27/09/19 10/10/19
Project resources form 10/10/19 17/10/19
Risk assesment 10/10/19 17/10/19
Security awareness 17/10/19 24/10/19
Submit Interim Report 28/11/19 28/11/19
Get control code working 17/10/19 13/11/19
Preliminary MATLAB code 14/11/19 27/11/19
Testing MATLAB functions 28/11/19 11/12/19
Background research 2 28/11/19 26/02/20
Complete machine vision system 13/12/19 11/03/20
Find Circle From Still 13/12/19 06/01/20
Find Circle from Video 14/01/20 20/01/20
Plot data of centre coordinates 21/01/20 23/01/20
Test HSV 24/01/20 27/01/20
Compare to Control Frame 28/01/20 29/01/20
Labelling Small Regions 03/02/20 14/02/20
Combine comparing to control fra... 02/03/20 05/03/20
Retake Videos with more lighting 09/03/20 11/03/20
Final Report Writeup 10/02/20 02/04/20
Introduction & Obejctives 10/02/20 02/04/20
Literature Review 10/02/20 02/04/20
Methodology 10/02/20 02/04/20
Analysis & Conclusions 10/02/20 02/04/20
Proof read report 03/04/20 22/04/20
Submit Final Report 23/04/20 23/04/20
2019
2020
September
October
November
December
January
February
March
April
May
Submit Interim Report
Submit Final Report
Today

Appendix 7

University of the West of England BRISTOL

GENERAL RISK ASSESSMENT FORM

Describe the activity being assessed: Programming the model factory	Assessed by: Luca Ricagni	Endorsed by: Gary Atkinson
Who might be harmed: Student undertaking project How many exposed to risk: 1	Date of Assessment: 15/10/19	Review date(s): 15/10/19

Hazards Identified *(state the potential harm)*	Existing Control Measures	S	L	Risk Level	Additional Control Measures	S	L	Risk Level	By whom and by when	Date completed
Electric shock	Ensure wires are insulated Voltage kept to 24V	1	1	1	None required	1	1	1	Luca Ricagni	20/04/20
Fingers getting trapped	Remain a respectable distance away whilst movement is occurring	2	1	2	None required	1	1	1	Luca Ricagni	20/04/20

RISK MATRIX: (To generate the risk level).

Very likely 5					
Likely 4					
Possible 3					
Unlikely 2					
Extremely unlikely 1					
Likelihood (L) ↑ / Severity (S) →	**Minor injury – No first aid treatment required**	**Minor injury – Requires First Aid Treatment 2**	**Injury - requires GP treatment or Hospital attendance**	**Major Injury 4**	**Fatality 5**

	1		**3**		

ACTION LEVEL: (To identify what action needs to be taken).

POINTS:	RISK LEVEL:	ACTION:
1 – 2	NEGLIGIBLE	No further action is necessary.
3 – 5	TOLERABLE	Where possible, reduce the risk further
6 - 12	MODERATE	Additional control measures are required
15 – 16	HIGH	Immediate action is necessary
20 - 25	INTOLERABLE	Stop the activity/ do not start the activity

Appendix 8

Dissertation Project Resources Form

Name: Luca Ricagni Student ID: 16010178

Supervisor: Gary Atkinson

[If you are carrying out a desk-based study and/or are only using the library or ITS facilities available at UWE, the faculty's risk assessment procedures will sufficiently cover your activities. In these circumstances, you should identify this on your risk assessment form, which you should attach - along with this form - to your Interim Research Proposal.]

Please sign below to confirm your request for resources.

Student: ....

Supervisor: .....Gary Atkinson.....

Further Resources – *what else will you need?*

R-Block workshop	☐	Laser cutting	☐
N-Block workshop	☐	Large format printing	☐
3D printing	☐	Miscellaneous	☐

Labs:

Structures	☐	Car/flight simulator	☐
Materials	☐	Engine test bays	☐
Composites	☐	Robotics	x
Hydraulics	☐	Dynamics	x
Wind tunnel/aero	☐		

If you have ticked any of the above, please fill out a risk assessment of your activities.

Have you spoken to a technician? Yes ☐ Not yet x
