THE UNIVERSITY OF BUCKINGHAM

UNIVERSITY OF BUCKINGHAM
SCHOOL OF COMPUTING

# AUTOMATIC DETECTION OF EMPTY SUPERMARKET SHELVES

PROJECT REPORT

Submitted by
PATRICK CARRINGTON - 1909003

Supervisor(s):
DR NASEER AL-JAWAD

Autumn 2022

## Declaration

This declaration is to confirm that the following project is all my own work and that I have appropriately acknowledged the work of others that I have referenced.

# Table of Contents

# Table of Figures

## Abstract

Over the course of this project, I have researched and developed a system that takes an image of a shelf in a supermarket and analyses it. This analysis looks for the gaps on the shelf and the location of the gap in order to determine what product is missing.

I undertook this project because during my work in Tesco stores, I identified an issue where there is stock of products in the back warehouse, but since this hasn't been displayed on the empty shelves, we have needlessly missed out on sales. Checks for this scenario are conducted once a day, whereby a team of colleagues walk around the shop to manually look for gaps – this can be removed and done in real time by my automated system.

I researched and trialled a number of different methods for the gap detection functions, and fully implemented two methods – a 'traditional' image processing approach and a deep learning based solution. The traditional method used OpenCV and a number of operations to detect gaps. While this method uses less computational power and I was able to accurately detect with some images, changes to parameters needed to be manually made for each image, and the detection was badly affected by shadows etc. The deep learning method was based on the YOLOv5 algorithm and had much greater precision and recall compared to the traditional method, even with the relatively small dataset I used.

# 1. Introduction

In a retail store, one of the greatest customer irritants is lack of availability for their most essential items. According to research conducted by Corsten and Gruen, if a product is out of stock, 9% of consumers would not purchase the item and 31% would go to another store to purchase their desired item [1]. In Tesco, a further problem with lack of stock availability is that pickers for our tesco.com Grocery Home Shopping (GHS) service would be unable to pick products for online customers in an efficient way, which could lead to either the product being substituted for a customer (the number 1 complaint for the GHS service) or the picker unnecessarily searching in the warehouse for a product, wasting valuable time and reducing their pick rate.

## 1.1 Motivation & Background

Since 2017, during my A Levels, Gap Year and the COVID-19 pandemic, I have worked in a number of Customer Assistant and Team Support roles at my local Tesco stores. As the UK's largest retailer, Tesco uses technology to make their operations more efficient and help them to meet their purpose of serving their 'customers, communities and planet a little better every day'. I have always been interested in how Tesco uses technology and have decided to base my project on solving a big problem in our stores – on shelf availability.

In most larger supermarkets, fresh produce is displayed in trays to make it easier to transport and replenish. Grocery and General Merchandise products are delivered to stores on cages or pallets, consisting of cases (boxes) of products. Most cases of products can be opened and placed directly on the shelf. Often when products are out of stock on the shelf, the retailer has additional stock in the warehouse, but it is not put out for customers due to the lower capacity of the shelf (often only two trays/cases per product). Furthermore, as large supermarkets have a vast sales floor (Tesco Superstores and Extras have floor space ranging 15,000 to 35,000 sq. ft [2]), it takes a while for a store colleague to survey any out of stock products and walk to the warehouse to fetch additional stock to replenish the shelf.

At Tesco, in the average large store, 11 hours per day are allocated to the 'Gap Scan', where stock control colleagues manually walk the store to check which items are out of stock and scan the shelf label with their PDA. This checks the stock record and produces a list of products that are truly out of stock (and prints a temporary out of stock sign), and

items that have a positive bookstock (positive stock record, often meaning that the item is out of stock on the shelf but is somewhere in the warehouse). My solution could significantly reduce the time taken on the daily gap scan. Even if my solution only reduces the time to complete the routine by one hour, multiplied across the large store estate, represents a saving of 5,117 hours per week, or a cost saving of £2,687,448 annually, which could be reinvested into serving customers or completing other, more complex routines.

## 1.2 Aims/Objectives

My project will utilise a fixed camera pointing at a product shelf in a large supermarket. The system will then analyse the shelf for any empty trays or gaps on a shelf, and with the aid of either Tesco's existing Product Location API or a file created by myself, lookup what is supposed to be stocked on the shelf. This information is then relayed through a webpage to the store colleague's PDA in order to support replenishment activities.

Furthermore, a dashboard webpage can be created for store management. This will show live store performance and will enable the manager to view historical availability data for the week. This will enable the manager to highlight trends in unavailability on different days for colleagues to be scheduled in a more efficient way. The manager's dashboard will also show a leader board for the geographical region that the store is in, this will enable area managers to reward stores with good availability and coach stores with lower availability.

The overall objectives of the solution are to increase product availability for customers and provide data for managers to coach their teams and plan resource.

Over the course of the project, I hope to create a prototype system that correctly detects that gaps and their locations on the shelf. Should I have time, I will also develop a web interface for this prototype system that would enable store staff to easily view the data generated.


**Figure 1 – Example of Gap Finding**

The images will be captured at a predetermined interval, to be decided during testing. Were my solution to be rolled out for real, I would recommend the usage of either fixed dedicated cameras, mounted on the ceiling or shelving, pointed at the mods with a large number of 'star lines' (the 500 most popular products in a store [3]), or by using the existing fixed CCTV cameras.

For the purposes of my testing and research, I was unable, due to Tesco policies, to set up a fixed camera. Therefore, I will use images taken from the SKU-110K dataset [4] and from images that I personally captured, using a camera and tripod at the Tesco Warwick Superstore and Petrol Filling Station (PFS), while the stores were closed to prevent any customers from being disrupted.

In order to comply with GDPR, I will implement a method of person detection, that will detect when a person is in the captured image and automatically delete it – before it is saved or analysed. The image can then be retaken after a predetermined interval.

## 2. Related Works

There have been many applications of computer vision relating to retail stores and the management of on shelf gaps and planogram compliance.

'Applying Image Processing for Detecting On-Shelf Availability and Product Positioning in Retail Stores' [5] is such an application where Applied Image Processing is used to detect the presence or absence of products on the shelf. The solution is written in MATLAB and utilises Feature Extraction and SURF algorithms. Reference images of products were given to the system which the SURF algorithm searched for in the target image. When a void on the shelf is detected or if misplaced products are detected, a notification is sent to the store staff. The developers noticed a limitation that similar coloured products may register an incorrect number of products. This is due to the angle and light levels on the target image.

'Supervised Learning for Out-of-Stock Detection in Panoramas of Retail Shelves' [6] utilises a FAST corner detection algorithm to detect shelves and items. Out of stock regions on the shelves are found by the *lack* of keypoints in the target image and a binary mask is then produced. The solution also detects Shelf Edge Labels (SELs), which I may want to include in my project given the legal importance of having SELs and the amount of time spent in a Tesco store to check each shelf for them. The solution described has a high accuracy of 84.5% for out of stock detections and 86.6% for label detections. Another aspect that is particularly relevant to my project is that it does not require a large dataset of product images.

'Shelf Auditing Based on Image Classification Using Semi-Supervised Deep Learning to Increase On-Shelf Availability in Grocery Stores' [7] uses a combination of the previous two papers and deep learning to classify images. It uses One-stage Deep Learning Object Detection, semi-supervised learning and Explainable Artificial Intelligence (XAI). The system builds a classification model to detect 'Product', 'Empty Shelf' and 'Almost Empty Shelf' but does not identify specific products, just the category. It requires a large dataset (something I want to avoid) and time to be spent manually labelling images. An aspect of this solution that I would like to have in my project is how images with people in the frame are automatically discarded and another image is taken after a predetermined period of

time. This would be useful to me since if there is a frame with people present, it would make it impossible to detect the shelves and may present a GDPR/privacy issue.

'Empty space detection in Retail Shelves where there is no product present' [8] is an article on LinkedIn and GitHub which explores various different methods to detect empty spaces on supermarket shelves. The first method used was to use the OpenCV library for python to analyse the image. In order to analyse the image, it is read as a numpy array and a histogram-based threshold is applied to separate the foreground and background. Canny edge detection is then used to find objects in the image which is dilated and eroded to merge edges together. The OpenCV rectangle function is then used to identify rectangles of empty space on the shelf. A problem faced with this method is that any products placed further back on the shelf may produce false positives and any other objects such as the floor, roof, shelf racking are not separated using Canny edge detection. Furthermore, any products with a dark logo/box may be incorrectly classified as the background in the threshold, therefore an adaptive method to threshold the image was needed. It was suggested that a Machine Learning model could be trained for adaptive thresholding, and Multinomial Logistic Regression could be used.

YOLOv5 is an object detection algorithm developed by Ultralytics and is available as an Open Source project on GitHub. The name YOLO (You Only Look Once) comes from how the algorithm predicts with a single network, during a single pass [9]. The structure of the algorithm is a Convolutional Neural Network that divides each image into a grid, to which bounding boxes are predicted for each class. The bounding boxes are then weighted with the expected probability [10].

The YOLOv5 model architecture contains three main parts – the model backbone, the model neck and the model head. Towards AI is a leading online AI publication that was very helpful in explaining the overall architecture of YOLO. I found that the model backbone of YOLO uses Cross Stage Partial Networks to extract features from an image. In the model neck, feature pyramids are created to extract features and generate multiple feature map layers. The YOLO model neck uses a Path Aggregation Network (PANet) for this. The model head processes the features and predicts the bounding boxes/classification. [11].

Since the YOLO architecture was significantly more complex than any AI model I had studied before, I did some research to try to better understand the architecture. A paper originating from the National Technical University of Athens [12] helped to explain the YOLO model and provided the diagram below that I used to help link back to the article I found on Towards AI. The research I conducted gave me a basic understanding of the YOLO architecture but I was still a little confused, so had to treat the YOLO aspect as more of a black-box than I had intended.

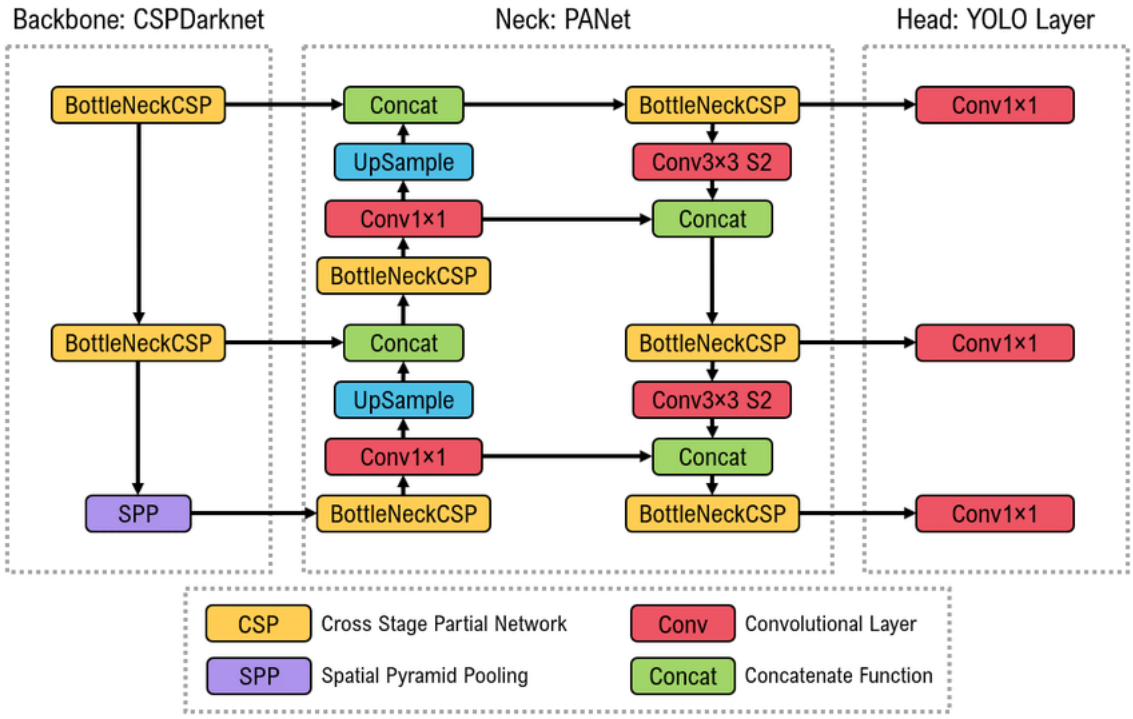


**Figure 2 – YOLO architecture diagram [12]**

# 3. User Requirement Analysis

## 3.1 Example Use Case

| **Gap Detection System: View List of Gaps** | |
|---|---|
| Actors | Store staff members wishing to replenish stock, Gap Detection System, Cameras |
| Description | When a store user wishes to replenish stock, they can use the system to generate a 'shopping list' of items out of stock on the shelf that they can then retrieve from the warehouse. This list contains the list of gaps and the 'bookstock level' denoting whether there is stock in the warehouse to be put out. |
| Data | Gap location data, Product Location Database, Stock Records |
| Stimulus | A user request to view gaps |
| Response | A list of gaps with stock records |
| Comments | The gap detection algorithm will run at set times, the request for the list of gaps will take the latest run of the detection algorithm. |

## 3.2 Requirements

The system will consist of two main parts, first is the driver code that detects the gaps, and the other is the web app/display interface for the data. I shall primarily focus on the driver code, and will move onto the web app should time constraints allow.

A major requirement for the driver code is accuracy – if stores are going to be measured on their performance, a high level of accuracy is required to ensure that the measure is fair. Therefore I have decided that the accuracy of the detection algorithm must be no less than 80%, with a preferred value closer to 90%.

The system will allow store colleagues and management to better manage the availability of products in the store. This is achieved through two web apps.
The first web app is the Gap List for the store colleague. The app simply shows a list of the detected gaps in the store along with other information such as the location of the gap, the product number and the current bookstock. This can be printed off or displayed on a store PDA and used as a 'shopping list' for the store colleagues to collect stock from the warehouse and take to the shopfloor for replenishment activity.

The managers dashboard will show times and locations with the most out of stocks in order to highlight trends and issues. A weekly performance screen will give the store managers a ranking of blue, red, amber or green (BRAG). This data is also fed into a regional leaderboard for the regional director.

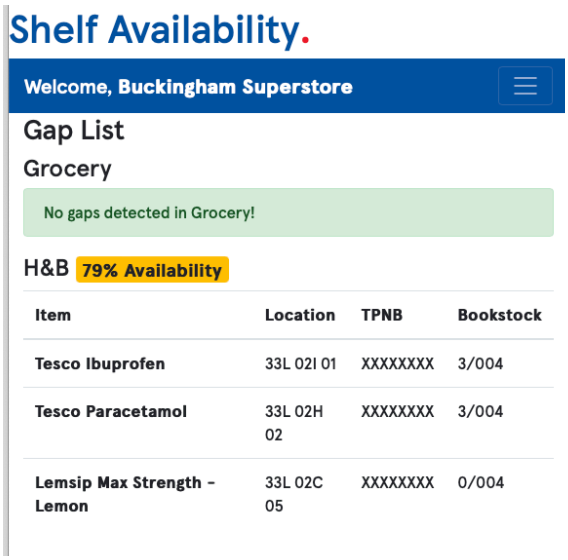


**Figure 3 – Mockup of Colleague PDA screen**

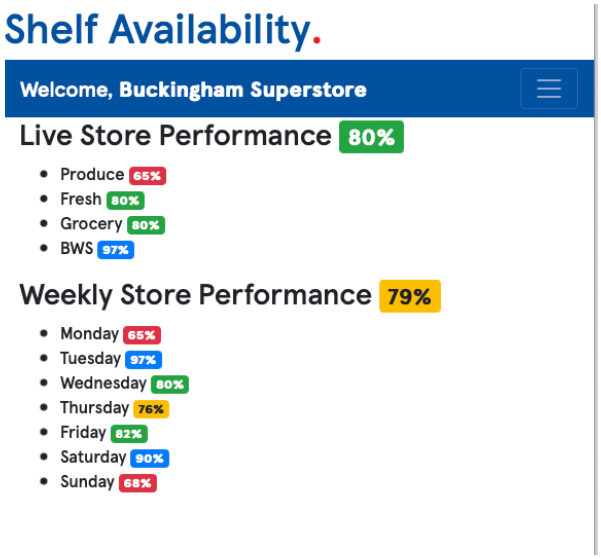


**Figure 4 – Mockup of Managers' Dashboard**

## 4. Methodology

At Tesco, each product is mapped to a specific aisle, on a specific mod (section of aisle), specific shelf on mod and then to a position on the shelf. The diagram below explains the mapping. For example, the yellow shaded item in the figure below is at position 01L02C02, meaning it is on Aisle 1, on the left hand side, on the second mod, on shelf C in position 2.


**Figure 5 – Tesco Product Location Explanation Diagram**

I plan to use Image Processing techniques to detect the location of gaps on the supermarket shelves. This location can then be matched to a location in the product database. A list of out of stock products can then be generated and sent to the PDAs of store colleagues, and an availability figure can be calculated and sent to stores and regional management teams to understand where and why gaps are forming, and to better plan resource or deliveries to help minimise gaps.

## 5. Implementation, Prototypes & Testing

The implementation of my solution has been split into different functions. I implemented the system in Python, making extensive use of the OpenCV and NumPy libraries.

There are six main steps to the system, which are outlined in the figure below.

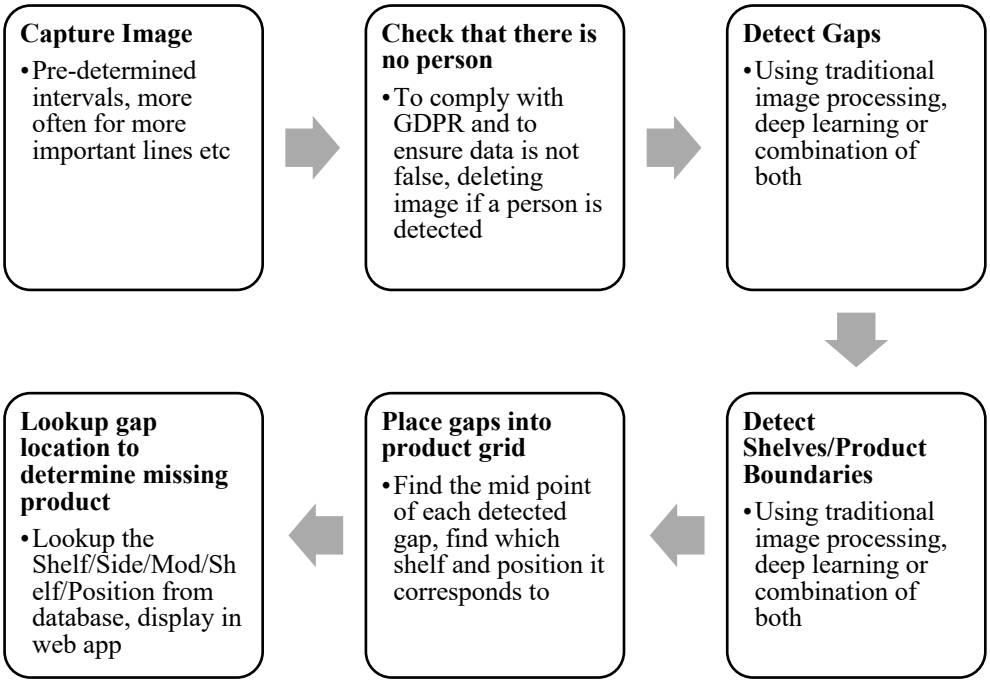


**Figure 6 – System Flowchart**

First an image from the pool of dataset and my own images is read into the system using the cv2.imread() function. The image is then resized and converted from an RGB image into a greyscale image using the cv2.cvtColor() function. Next, using the cv2.equalizeHist() function, the histogram is equalised to enhance the contrast of the image.

### 5.1 Implementation of Gap Detection

#### 5.1.1 Traditional Implementation of Gap Detection

Perhaps the most important part of the project is the gap detection algorithm. I first wanted to use 'traditional' image processing techniques to detect the gaps on a shelf. The figure below summarises the method that I decided to use.

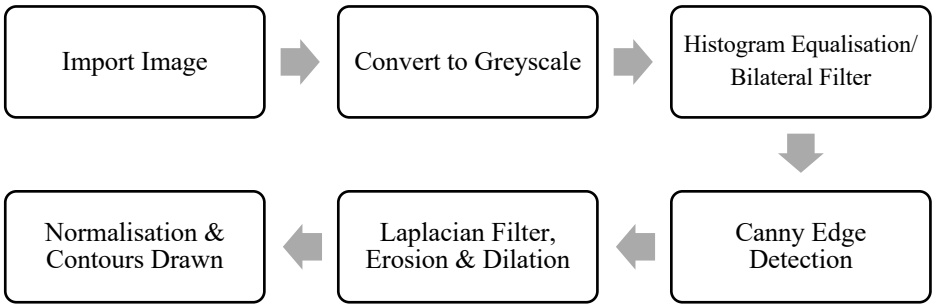


**Figure 7 – Flowchart of Traditional Gap Detection**

After the image has been imported and equalised, it is passed to the gap detection function. The gap detection function first applies a bilateral filter onto the image to preserve the edges, then Canny edge detection is used to find the edges of the products.

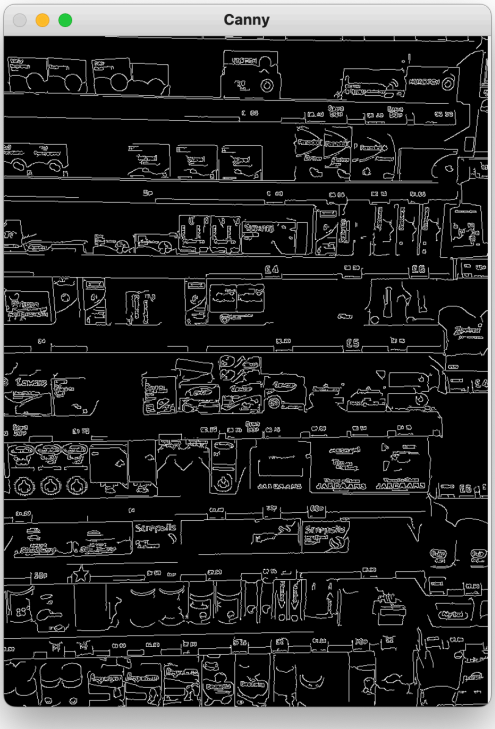


**Figure 8 - Canny Edge Detection**

I chose to use Canny edge detection because it gave the clearest, most defined edges. I found that Sobel edge detection would be faster and use less computational power, but as shown below, it doesn't produce as clear edges as Canny, and since the images aren't being analysed in realtime, I did not think that it would be an issue.

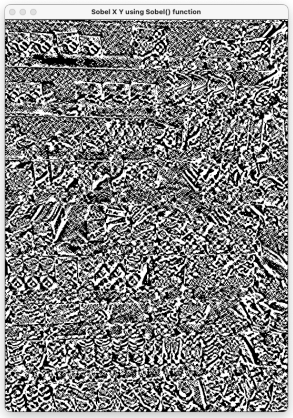


**Figure 9 – Sobel XY detection using Sobel() function**

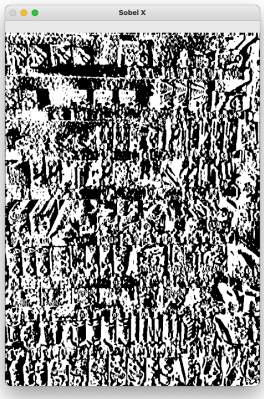


**Figure 10 – Sobel X**

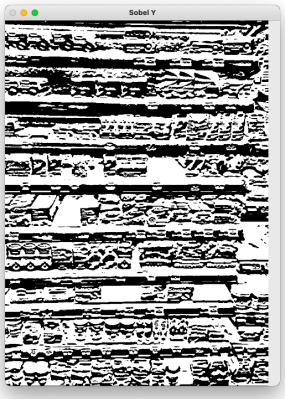


**Figure 11 – Sobel Y**

After the edge detection, I applied a laplacian filter to make the Canny edges stronger, then applied erosion and dilation to close up any holes. Next, I normalised the image using the cv2.normalize() function, to give the image below. The white sections are where a gap is detected.

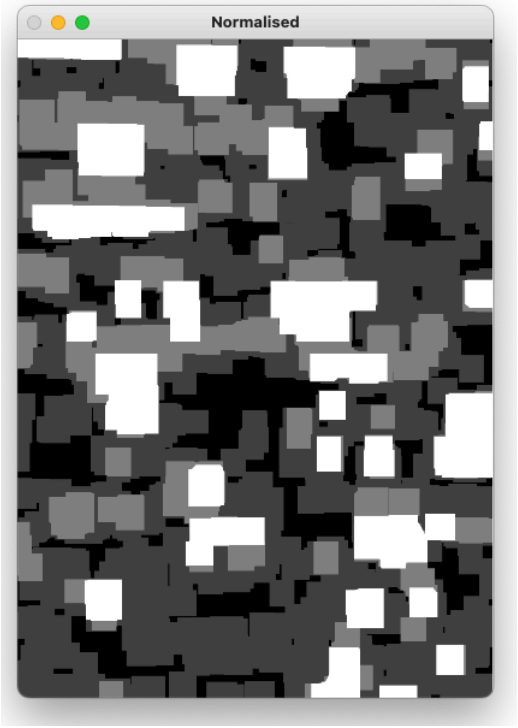


**Figure 12 - Normalised image**

After the image is normalised, I use the cv2.contours() function to draw contours around the gaps.


**Figure 13 - Example of contours drawn by cv2.contours()**

From my testing, there was a number of false positives caused by smaller areas on the normalised image. From testing a number of images, the false positives were mainly made up of the smallest quartile of gaps, so I sorted the gaps by area and filtered out the smallest quartile – I found that this drastically reduced the number of false positives and didn't cause many false negatives.

In order to generate definitive values of precision and recall for my implementation, I tested my implementation on a set of 18 images, taken from a mixture of the SKU-110K dataset and my own images that I have taken in both Tesco stores and other supermarkets. I manually analysed each image to find the location of the gaps – finding 102 gaps in total.

For the traditional approach to gap finding, the values I calculated were as follows:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} = \frac{24}{12 + 24} = 0.667$$

$$Recall = \frac{(True\ Positives)}{True\ Positives + False\ Negatives} = \frac{24}{24 + 66} = 0.267$$

These values are well below the target I set for the gap detection to be 80% accurate, and therefore I decided that I needed to explore further methods of gap detection.

### 5.1.2 Gap Detection Using YOLOv5

After having conducted some research of the different types of object detection algorithms, I chose to explore the YOLOv5 algorithm further. This is because I found lots of documentation and tutorials on training a model using a custom dataset, and because there had already been work conducted using the SKU-110K dataset with YOLO (albeit for a different purpose).
On the official YOLOv5 GitHub page, Ultralytics provides a guide on training the model with custom data [13]. I used this in conjunction with an in depth tutorial by YouTuber Nicolai Nielsen [14], to train a model using images.

First, I annotated a dataset of 116 images using Heartex Labs' open source annotation tool *LabelImg*. Once I had annotated all the gaps in the dataset, I uploaded them to *Roboflow.* and once the images were uploaded, I resized all the images to 416 x 416 pixels.

Once this pre-processing was completed, I turned my attention to mitigating the risk of overfitting. Overfitting is where the input training data is too closely aligned to the model. I overcame this by augmenting the dataset by rotations and flips. Once the augmentation was completed, I had a set of 300 images to train the model on.

In order to train the model, I utilised code adapted from the YOLO wiki and Nielsen's tutorial in a Google Colab environment to run the train.py function. Originally I trained with the default parameters for 50 epochs. Once I had an idea of how the algorithm trained, I upped this to 100 epochs. The YOLO documentation recommends that you start with 300 epochs, but due to the additional time required verses 100 epochs and the impressive levels of precision and recall at 100 epochs, I decided not to increase.

YOLOv5 has a number of hyperparameters (around 25) that can be used to produce better results from training [15]. Since the scope of my project was primarily a proof of concept and seemed to work quire well without changing the hyperparameters, I left the hyperparameters to their default values. If the project had a greater scope and I had more

time, I would have experimented with the hyperparameters – especially the ones concerning the augmentation of the training dataset.

The results of the training are detailed in the below graphs.

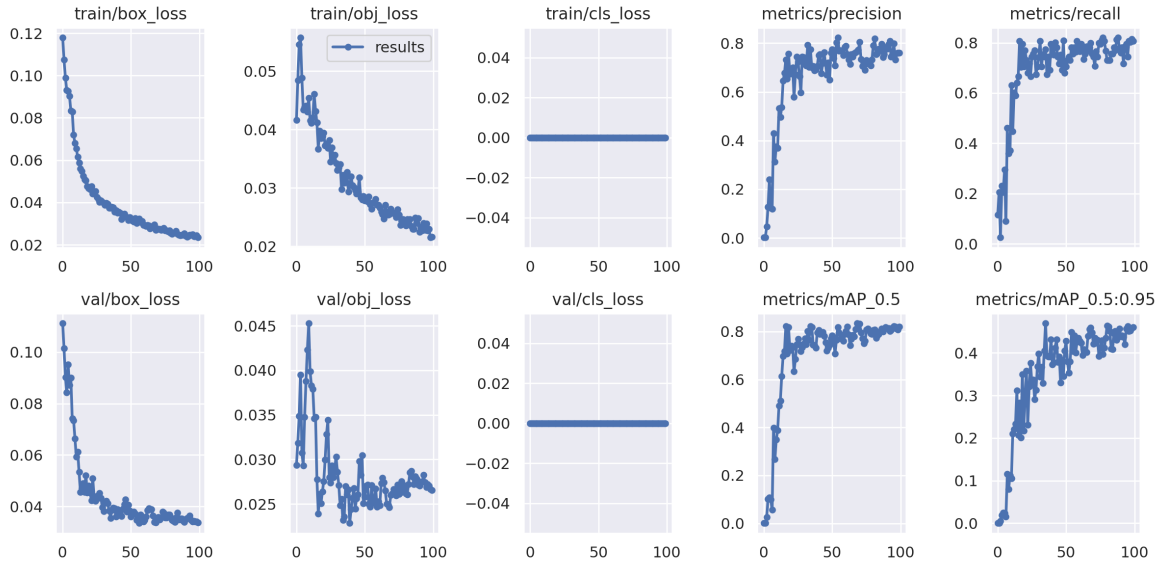


**Figure 14 – Automated Graphs from YOLO training**

The following set of images is generated by YOLO and depicts the predictions of the model for part of the validation set.

**Figure 15 – Validation set with predictions**

Once the model was correctly trained, I used the detect.py function to test the YOLO model, using the same dataset as the traditional method above. The results were as follows:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} = \frac{95}{95 + 4} = 0.960$$

$$Recall = \frac{(True\ Positives)}{True\ Positives + False\ Negatives} = \frac{95}{95 + 3} = 0.969$$

These values represent a large improvement in the values for recall and precision. Therefore, I took the decision to not use the more traditional methods for gap detection and use a purely AI based approach to the gap detection.

## 5.2 Shelf/Product Detection and Separation

The next function that I implemented was the vertical shelf detection. To do this, I took the equalised image, then used cv2.reduce() to take the average values of the image going down the image vertically.

**Figure 16 – Visualisation of the Shelf Detection**

Then using the SciPy peak finding algorithm, I found the peaks of the average values, which correspond to where there is a shelf on the image.

Once the shelves are detected, I then split the image up into the individual shelves. Then I run the product detection algorithm, which is similar to the shelf detection, but uses the troughs on the graph to act as a marker between products/gaps.

**Figure 17 – Visualisation of the Product Detection**

## 5.3 Product Location function

Once the gaps, shelves and products have been detected, the next task is to place each gap into a location on the product 'grid'. To do this, I used the cv2.rectangle() function to draw a rectangle around each of the contour areas. From the coordinates of these rectangles, I was able to calculate the mid point of each gap. From this, using the coordinates of the shelves and the products, it is possible to deduce the shelf and product location of the gap.

```
GAP: Shelf C Position 11
GAP: Shelf C Position 5
GAP: Shelf B Position 7
GAP: Shelf D Position 9
GAP: Shelf B Position 4
GAP: Shelf D Position 1
GAP: Shelf A Position 4
GAP: Shelf D Position 12
GAP: Shelf E Position 12
GAP: Shelf C Position 13
GAP: Shelf C Position 6
GAP: Shelf B Position 9
GAP: Shelf A Position 3
GAP: Shelf C Position 1
GAP: Shelf C Position 9
GAP: Shelf B Position 12
GAP: Shelf B Position 1
GAP: Shelf B Position 11
```

**Figure 18 - Screenshot from output from Product Location function**

## 6. Ethical Issues

A main ethical consideration for this project is how people may stray into the view of the camera when in the store, thereby getting captured. It will be possible to overcome this by adding a person detection function, which will be run before any further analysis of the frame takes place. This function will detect if there is a person in the frame, if there is, the image will be automatically deleted. This is to properly comply with GDPR.

Tesco stores already have posted signage (similar to the figure below), that warns customers and visitors about the use of CCTV and the scheme controller information. If my solution were to be implemented in a real store that utilises fixed cameras for gap detection, I would suggest an edit to the signage to include the purpose of 'efficient running of stores'. This would ensure that the organisation complies with GDPR should the automated detection algorithm fail.

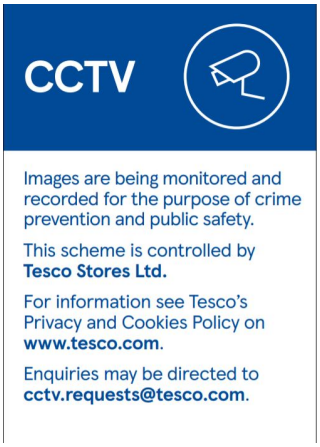


**Figure 19 – Tesco CCTV Signage**

## 7. Evaluation

Overall, I am pleased with the results from the AI training – a precision of 0.960 and a recall of 0.969 is well above the 80% threshold that I had set at the beginning of the project and is also above my target of 90%.

In order to gain further confidence with my solution, I would conduct further tests with a larger set of validation images – including images that are harder to detect with, such as images with shadows/poor lighting.

A further problem that I came across is the problem with messier shelves. If the items on the shelf have fallen over, or there is debris at the back of the shelf, the algorithm may not classify the location as a gap. For example, in the figure below, the area indicated in yellow has a few stray products and debris at the back of the shelf.


**Figure 20 – Example of messier shelf**

This debris prevents the algorithm correctly classifying the location as a gap, despite the intended products clearly not being present. It may be possible to overcome this by adding a few messier images to the training set and retraining the model. The figure below shows a further example of where equipment at the back of the shelf can cause false negatives.


**Figure 21 – Further example of messier shelf**

A further problem with the messier shelves is that it can corrupt the data from the shelf/product detection function, as this purely works off of traditional image processing. Since the problem is how the function itself is designed, I think the only way to properly overcome this would be to rewrite the shelf/product detection algorithm, potentially using some kind of AI model instead.

## 8. Plan

During the early stages of the project I was relatively on track with my plan. Something that I had been struggling with is actually finding a large dataset of images to analyse. The SKU110K dataset, while very large, mostly contains images of full or nearly full shelves due to its main purpose of research being the detection of objects in densely packed areas. Furthermore, since it often only includes single images of shelves, I couldn't verify that my algorithm would behave in the correct way over time. Therefore, I attempted to contact the Technology team at Tesco for assistance, however due to Tesco policies and those of third party partners, I was unable to obtain images. Therefore, my initial testing and implementation was somewhat slow.

My Gantt chart below contains all the major deliverables and milestones in the project. I do not have any project activity scheduled for the June exam diet as I would prefer to focus on my exams. The plan is a rather optimistic timetable, something that can be mitigated by extending some deadlines, since I do not have too much activity planned for the Summer Term. Furthermore, I have not included any activity to take place in the three week Summer holiday – if necessary, I could include some working time, while ensuring that I do have some time off.

I have based some of my estimations for the time to complete some aspects of the implementation on how long I spent working on the initial implementation during the first term/holiday. Therefore, the implementation times could be somewhat inaccurate.

Since the first progress report I have found that some of my estimations for completion were optimistic . At the time of writing, I have only implemented the Gap Detection, Shelf/Product detection and the function to locate the products in the product 'grid'. I still have person detection, the AI implementation and the webpages to design and implement. This should be achievable by the due date for the project.

Since I was behind my original schedule, I had intended to work throughout the summer holiday. However, I found the June exam diet more stressful and challenging than expected so I felt that it was necessary to have a good period of time away from all university work. Therefore, the only project work I completed over the holiday was minor bug fixes, some elements of the second progress report and some background reading.

After submitting the second progress report and the summer holiday, I worked on implementing the AI based approach, after my research during the holiday period. This was a more straightforward process than I had imagined because of the very well documented nature of the YOLO project. The poster on the other hand took many attempts to get to a high standard.

Over the course of the summer term, I decided that I needed to be more realistic with the work that I could actually complete before the project due date. Therefore, I had to make the decision to not implement a web app for the project, focusing mainly on the underlying detection code.

A further aspect of the project that I did not have time to implement was the person detection function. This is because I recognised the complexity of the task early in the project, so decided to focus on the more important aspects (i.e. the gap and shelf detection). During the course of my research into YOLO/AI approaches, I realised that it may be simple to implement if I had more time (using a pretrained YOLO model) but I would not have time to test or properly implement the function with the rest of my project.

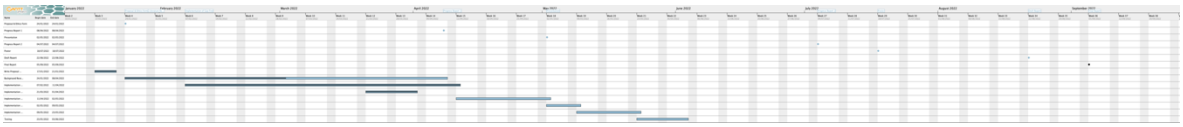

**Figure 22 - Initial Project Gantt Chart**

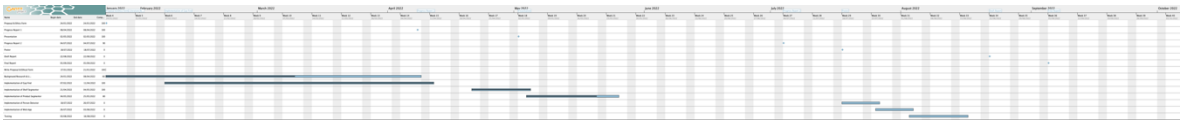

**Figure 23 - Updated Gantt Chart**

| Name | Begin date | End date | Comp... |
|---|---|---|---|
| Proposal & Ethics Form | 24/01/2022 | 24/01/2022 | 100 |
| Progress Report 1 | 08/04/2022 | 08/04/2022 | 100 |
| Presentation | 02/05/2022 | 02/05/2022 | 100 |
| Progress Report 2 | 04/07/2022 | 04/07/2022 | 100 |
| Poster | 08/08/2022 | 08/08/2022 | 100 |
| Draft Report | 12/09/2022 | 12/09/2022 | 100 |
| Final Report | 26/09/2022 | 26/09/2022 | 90 |
| Write Proposal & Ethical Form | 17/01/2022 | 21/01/2022 | 100 |
| Background Research & Literat... | 24/01/2022 | 08/04/2022 | 100 |
| Implementation of Traditional ... | 07/02/2022 | 11/04/2022 | 100 |
| Implementation of AI Gap Find | 01/06/2022 | 15/08/2022 | 100 |
| Implementation of Shelf Segm... | 21/04/2022 | 04/05/2022 | 100 |
| Implementation of Product Seg... | 04/05/2022 | 02/06/2022 | 100 |
| Testing | 03/08/2022 | 16/08/2022 | 100 |
| Write Final Report | 01/08/2022 | 26/09/2022 | 90 |

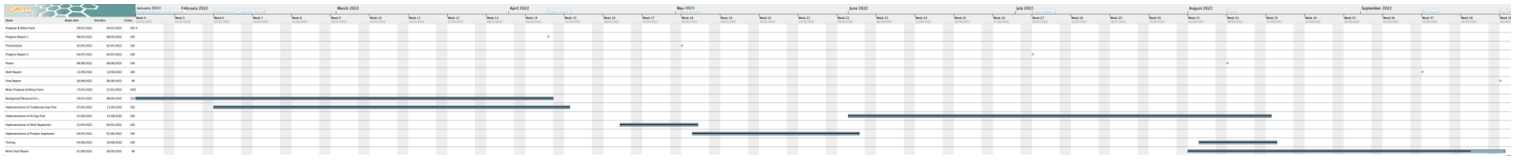

**Figure 24 - Final Gantt Chart with Start/End dates**

## 9. Conclusion

During this project I have researched and developed a system to detect empty supermarket shelves. I had two main methods of doing this – traditional image processing and the use of the YOLOv5 object detection algorithm.

The traditional image processing method was promising at first, but after testing on a number of images, the values for precision and recall were far too low – 0.667 and 0.267 respectively. Therefore, I had to research a new method in order to improve my prototype. This new method, utilising the YOLOv5 algorithm had vales of 0.960 and 0.969 for precision and recall, which well exceed the targets set at the beginning of the project.

I think that this project can be considered a success, as I have been able to achieve my main aim of producing a prototype system for the detection of gaps on supermarket shelves. I was very happy that eventually I achieved such high values for precision and recall for the detection algorithm.

During the project, I have developed many skills that I feel will be very beneficial in my future career as a software engineer. For example, the research of different methods of implementation and time management.

## 9.1 Further Work

If I were to continue the development of the project, I would add in some of the missing features – namely the person detection and web app. It was disappointing to not have the time to develop these aspects but leaving them was essential to ensuring that the bulk of the prototype was in a fit state to be submitted with the report.

Furthermore, I would seek to continue to improve the model by continually training with a larger dataset and by experimenting more with the hyperparameters. I left most parameters of YOLO to the default ones, which while they produced a good result, I think the result could be improved by fine tuning them.

A further piece of work that I would like to do is the improvement of the shelf/product detection functions. These were written alongside the traditional gap detection function, so at the time I did not have a good enough knowledge of AI to produce a more accurate solution.

## Bibliography


[1] D. Corsten and T. Gruen, "Desperately Seeking Shelf Availability: An Examination of the Extent, the Causes, and the Efforts to Address Retail Out-of-Stocks," *International Journal of Retail & Distribution Management,* vol. 31, no. 12, 2003.

[2] Tesco PLC, "Property - Tesco PLC," 2022. [Online]. Available: https://www.tescoplc.com/contacts/tesco-property/. [Accessed March 2022].

[3] Tesco, "Star Lines To Deliver Great Availability (Large Format)," 2020.

[4] E. Goldman, R. Herzig, A. Eisenschtat, O. Ratzon, I. Levi, J. Goldberger and T. Hassner, "Precise Detection in Densely Packed Scenes," 2019.

[5] R. Moorthy, S. Behera, S. Verma, S. Bhargave and P. Ramanathan, "Applying Image Processing for Detecting On-Shelf Availability and Product Positioning in Retail Stores," 2015.

[6] L. Rosado, J. Gonçalves, J. Costa, D. Ribeiro and F. Soares, "Supervised learning for Out-of-Stock detection in panoramas of retail shelves," 2016.

[7] R. Yilmazer and D. Birant, "Shelf Auditing Based on Image Classification Using Semi-Supervised Deep Learning to Increase On-Shelf Availability in Grocery Stores," *Sensors,* vol. 21, no. 2, p. 327, 2021.

[8] D. Sharma, "Empty space detection in Retail Shelves where there is no product present," 2020. [Online]. Available: https://www.linkedin.com/pulse/empty-retail-shelves-lets-find-em-geeky-way-opencv-yolo-deepika-/. [Accessed February 2022].

[9] D. A. Ali, *Artificial Intelligence Lectures, Chapter 6 - Artificial Neural Networks,* Buckingham: The University of Buckingham, School of Computing, 2022.

[10] A. Garg, "How to Use Yolo v5 Object Detection Algorithm for Custom Object Detection," 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/12/how-to-use-yolo-v5-object-detection-algorithm-for-custom-object-detection-an-example-use-case/. [Accessed 2022].

[11] Towards AI, "YOLO V5—Explained and Demystified," 1 July 2020. [Online]. Available: https://towardsai.net/p/computer-vision/yolo-v5%E2%80%8A-%E2%80%8Aexplained-and-demystified. [Accessed September 2022].

[12] I. Katsamenis, E. Karolou, A. Davradou, E. Protopapadakis, A. Doulamis, N. Doulamis and D. Kalogeras, "TraCon: A novel dataset for real-time traffic cones detection using deep learning," National Technical University of Athens, Athens, 2022.

[13] G. Jocher, "Train Custom Data," 2021. [Online]. Available: https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data. [Accessed 2022].

[14] N. Nielsen, "YOLOv5 Custom Object Detection with Code and Dataset - Neural Networks and Deep Learning," 2021. [Online]. Available: https://youtu.be/rZyY2pNzypQ. [Accessed 2022].

[15] Ultralytics, "Hyperparameter Evolution," [Online]. Available: https://docs.ultralytics.com/tutorials/hyperparameter-evolution/#1-initialize-hyperparameters. [Accessed September 2022].

## Appendix 1 – Meeting Minutes

**Week 1 – 24/1/22**

**Matters Discussed:**

- Now that proposal/ethical forms submitted need to start:
    - Collect literature
    - Start to think about questions:
        - What is the process?
        - How can the data be used?
- Decided on three states to look for
    - Shelf Stocked
    - Semi Empty
    - Gap
- Fixed camera, angles and location
    - How often should the system conduct the check
        - Process:
            - Takes image
            - Analyses trays
            - Triggers X

- Start
    - Images
        - Analyse
            - Row/Column
            - Grid? (shelf is laid out in a grid pattern, we can specify the size and camera is fixed so easy to make a grid to go over image
        - How fast/pattern recognition
            - Time since replenishment
                - Prediction/Patterns for analysis/display
        - Provide statistics
            - Time to replenish since gap etc

        - Predict where gaps will occur to pre-empt
        - Date/Time Stamps, can provide reminders to date check

- Not just fresh produce, think about applications in Grocery/General Merchandise/Beers, Wines & Spirits
    - Makes the solution more desirable

- Consider adding reference point to shelf in case the camera is knocked as would ruin the analysis

**Work for next week:**

- Look at literature - ie published/looked at before
- Background - ie what I need to learn
    - Understanding of CV/Image Analysis
    - Feature Extraction
        - How do we know if empty/not
- Collect some starter images to analyse
    - Taken with phone
    - Tesco Technology email - chase!

**Week 2 – 24/1/22**

**Matters Discussed:**

- Literature gathering/Background research
    - Looking at SURF algorithms
    - K-D trees
    - Higa et al article
    - Background subtraction

**Work for next week:**

- Continue literature gathering

**Week 3 – 31/1/22**

**Matters Discussed:**

- Consider the use of a dataset
    - Can we use template matching
        - Could work for bases of produce trays
- Touched on SURF descriptors/feature detection
- Need to be careful with working with Tesco to ensure that we keep within university rules
    - Have a more formal relationship, potentially speak to Harin?
- Problem with people
    - Analyse to take people out of the frame
- Can we use a reference points on the shelf to ensure that camera hasn't moved

**Work for next week:**

- Continue literature gathering and research
- Produce powerpoint with research
- Speak to Harin about Tesco relationship
- Review ethics form

**Week 4 – 7/2/22**

**Matters Discussed:**

- Presented literature gathering powerpoint
    - I have a few more sources
- Since the shelf is arranged in a grid, could we analyse that once, then using a reference to ensure camera hasn't moved, analyse for changes instead of gaps?
- I expressed concern at using Deep Learning, NAJ said don't be afraid of using ML/DL.

**Work for next week:**

- Start implementing using Image Processing techniques (MATLAB)
- Add to literature review as and when

**Week 5 – 14/2/22**

**Matters Discussed:**

- Showed MATLAB, explained processes

    - Edge detection – used canny
        - Have a look at the others (Harris, Sobel etc)
- Tesco no reply to latest email – chase again

**Work for next week:**

- Work more on edge detection – research Harris/Sobel etc
- Have a look at using OpenCV library on python/C++

**Week 6 – 14/2/22**

**Matters Discussed:**

- Started to implement in Python
    - More success than MATLAB
- Found more articles using OpenCV/Python than MATLAB – easier to transfer to be more appropriate.
- Best results were with Canny Edge Detection and Laplacian Filter
    - Work on normalising the image
    - Use contour detection algorithm.

**Work for next week:**

- Using the GitHub repositories found, keep improving
- Keep improving literature review

**Week 7 – 21/2/22**

**Matters Discussed:**

- Lots of false positives using edge detection
    - Discussed using YOLO to detect objects, then removing any intersections with detected gaps.
    - Discussed using blob analysis
- Discussed merging the techniques found in various papers
    - Document this
- Massive differences in accuracy if erosion/dilation used, and if the structuring element is changed (works well for one image, less well for others)

**Work for next week:**

- Start progress report
- More research on YOLO/ML

**Week 8 – 28/2/22**

**Matters Discussed:**

- Not very productive this week as had to focus on assignments
- Only looked at YOLO
    - Very interesting YouTube tutorial
    - Fast but not very accurate
    - Difficulty in segmenting smaller objects in dense situations
        - Since shelves are very densely packed, may cause issues
    - Can we train to look for gaps?
- NAJ suggested to focus on 'traditional' image processing first and use ML as a last resort
    - Show evidence of it not working
- Don't worry about how long analysis takes (since we'll only be taking an image a few times an hour)
- Speak to Niall (previous student) about his project

**Objectives same as last week**

**Week 2 – 15/4/22**

**Matters Discussed**

- Progress report was handed in on time with no concerns
- Over the holiday I took extra photos at Tesco to complement the SKU dataset images
- Over the holiday worked gap detection algorithm (at a good stage now) and started exploring the gap finding algorithm
    - Discussed how I found OpenCV forum posts exploring similar shelf detection and how to implement
- Started to think about user requirements and use cases

**Objectives**

- Continue to explore shelf segmentation
- Start presentation slides

**Week 4 – 29/4/22**

**Matters Discussed**

- Showed the progress made on the shelf detection and product detection
- Spoke briefly about the requirements for the presentation
- Discussed the accuracy of using traditional image processing and the potential of moving to using AI/Deep Learning

**Objectives**

- Continue to explore shelf segmentation
- Continue presentation slides
- Background research on AI

**Week 5 – 4/5/22**

**Matters Discussed**

- Struggling with how to segment the products/gaps horizontally
    - Discussed how we could use the vertical shelf detection, then split the image into the individual shelves, then use the same method but horizontally, taking the troughs of the graph as the boundaries between products/gaps
- Presentation slides are good
    - Will add extra slides on the different edge detection algorithms and why Canny was used
    - Discuss the challenges faced

**Objectives**

- Finish and practice presentation
- Continue with implementation

**No meetings week 6/7 due to presentations**

**Week 8 – 27/5/22**

**Matters Discussed**

- Presentation was good
- The shelf detection and the horizontal detection are working well – just need to refine the distance parameter of the peak finding algorithm to meet fine line between false positives and negatives
- Discussed the deliverables for next term

- Discussed person detection and how it isn't working brilliantly
    - Potentially use yolov5/AI to complete person detection

**Objectives**

- Don't work too much on project over exams
- Research AI/Deep Learning and try to implement
- Start person detection implementation

**Week 2 – 21/7/22**

**Matters Discussed**

- Over the holiday, I researched different AI algorithms
    - Decided to explore YOLOv5 in more detail
- Discuss challenges for each image (ie messy images etc)
- Need to start poster
    - Produce draft outline soon

**Objectives**

- Start poster
- Start implementation of YOLO
- Bring draft poster next week

**Week 4 – 4/8/22**

**Matters Discussed**

- YOLO
    - Much, much more accurate
    - A few false results, experiment with confidence thresholds etc
- Poster
    - Good layout so far
    - Too much text
    - Make colours more vibrant

**Objectives**

- Continue poster

**Week 5 – 11/8/22**

- Feedback on poster

**Week 6 – 18/8.22**

**Matters Discussed**

- Final poster looks good
- Research into YOLO hyper parameters
- Discuss overfitting
- Discuss dataset balancing

**Objectives**

- Continue implementation
- Work on report

**Week 8 – 1/9/22**

**Matters Discussed**

- Need to focus on written report
- Work on literature review and add the recent YOLO research
- Talk briefly in the background section
- Calculate values for precision and recall

**Objectives**

- Continue with report

**Week 9 – 8/9/22**

**Matters Discussed**

- Need to talk more about hyperparameters in the report, discuss YOLO results more
- Discuss the challenges of messy shelves etc
- Look at MATLAB deep learning example

**Objectives**

- Continue report and hand in draft on 12/9/22