Brunel University London
Department of Computer Science
Uxbridge
Middlesex
UB8 3PH
United Kingdom
T: +44 1895 203397
F: +44 (0) 1895 251686

Department of Computer Science

BSc (Hons) Computer Science (Artificial Intelligence)

Academic Year 2017 - 2018

The use of Facial Detection algorithms and Clustering techniques to estimate the total number of people in an image

Deepali Prakash Karsan Kerai

~1410622~

A report submitted in partial fulfilment of the requirements for the degree of Bachelor of Science

Contents

Word Count (Chapters Only): 17625

Page Count: 58

1. Abstract

There have always been causalities during events that have not been managed properly or due to a minor human-error. May it be a stadium full of football fans or school children on a school trip, there are many other forms of crowd events or situations that need to have effective precautions taken to ensure each person in the crowd or group is accounted for. This is definitely a situation to investigate and explore ways to help ensure better flow of people as the past has taken a toll for the worst, after all it's a person's life on the line.

As certain situations have their own way of dealing with the flow of people with cameras or acknowledging the visibility of people by wearing bright coloured vests, perhaps a single way of monitoring people is not reliable. Human error is inevitable therefore multiple ways of monitoring people is a need more than just an innovative idea. Image processing and face detection technology was explored as potential technical solution after being inspired by young AI developer who developed an app that takes a picture as input and outputs the degree of which the retina is diseased.

The implementation involved the exploration of facial detection techniques to capture the amount of people from a (uncopyrighted) digital image/ photographs. After software testing each aspect of the implementation, other machine learning and clustering methods were explored to increase the rate of correct detection of people in an image. The final solution involves face detection, facial features (nose, mouth and eyes) detection, and robust clustering after reduction of redundant detections (detections that qualified with no meaning). The evaluations inspect how concrete the results were against manual counts of the people within the test images. Overall the solution reflected results of detecting people from a photograph well, though was not to the quality of current detection methods.

2. Acknowledgements

I would like to thank my supervisor Dr Stephen Swift as I'm grateful for the guidance he's provided me with his expertise in Artificial Intelligence and Data Science throughout this project. The regular meetings and check-ups on the course of the project encouraged me to remain on track and to always apply my knowledge attentively to reach a high-quality software solution and dissertation towards the end.

I would also like to thank Dr Allan Tucker for reviewing my project idea and consulting in some interesting feedback regarding the technical concepts that I had not encountered. Due to this, I have taken his advice into account to overall deliver a better dissertation.

Finally to my family specifically my parents and all my friends, for their sincere support not only during this project but through-out my degree and time at University, I express my sincere gratitude which to whom this degree would deem impossible to do obtain.

I certify that the work presented in the dissertation is my own unless referenced.

Signature	[illegible]
Date	05/04/2018

3. Introduction

The project explores a series of artificial intelligence stemmed techniques that helped inspire the technical development of potentially solving the problems of crowd-based management situations. A common ground was discovered after investigating past events like the Hillsborough disaster (Kearns, 2011) and a school trip gone wrong (Williams, 2015), these situations needed a way to quantify accurately the amount of people there were. This eliminated human error and therefore became a solution essential to involve technology.

Photographs are a way to capture a moment of reality, like an image of a retina to check for eye disease or a group of people to check how many people there are (Bleicher, 2017). Image processing and segmentation methods were the first route taken as most images being with the filters to expose certain areas or colours of an image. The most extractions occurring from an image are the edges, regional grounds and colour differences according to light reflection. The Otsu, Hough and Watershed segmentation methods were implemented and tested to view how these methods were used to quantify how many people there are in a given image.

Further investigation into convolutional neural networks and machine learning techniques, was done to understand how their applications are used today. Face detection is one of the applications and became very relevant for this project since it associates with images. The detection of faces and possibly other facial features in the images being detected help extract data to then process and quantify how many people there are in the image just like Google discovered 'internet is made of cat's' through neural networks (Clark, 2012).

The clustering of the extracted data from detection algorithms enabled grouping of data containing similar attributes. Robust clustering was used when clustering and interpreting gene expression data (Swift et al, 2004). This was specifically useful when a face detection was not useful in detecting a face, but a feature detection was. However the outputs represented the same thing which means that the features in a face detection represented the same face, hence clustering.

After the implementation and testing stages, the evaluations took place to compare and identify how close the total is against a manual count of people in the same image. Several images were used to increase the validity of the system. Obviously the higher the accuracy the better. This project is also applicable for such problems like the crowd-based situations mentioned previously that inspired the purpose of this project.

This area of research was chosen due to the interest and unexplainable attachment to Artificial Intelligence. The project was an opportunity to learn more and discover fascinating technologies that may come together to create a technical solution for the proposed problem. At the very beginning of this journey, the first task was to specify the aims and objectives to lead a successful project.

3.1. Aims

The aim of this project is to develop a program that accurately counts the amount of people in an image. This program will include combinations of image processing and face detection applications to exhaust the overall systems efficiency to quantify the correct number of people in an image.

7 objectives were set to encourage this project to meet this aim. Each objective associates to the project directly in terms of how it will be carried out, the research, tools, delivery, data-collection and ethical concerns.

3.2. Objectives (7):

1. To research literature upon the similar problems of this sort and review their solutions to help elevate similar strategies for this project's technical solution. Also review publications of tools that may be used for image processing and face recognition to use them to their maximum potential.
2. To establish design(s) which will include a series of diagrams to describe the component process flows, theories and boundaries of the system. Unified Modelling Language will be used for representing the pre-determined solution, hand-drawn designs to view the assumptions of the output.
3. To successfully follow the Rapid Application Development methodology which will ensure iterative and experimental development of the project's technical solution to obtain accurate results in a short space of time.
4. To test the implementation of the program using grey-box testing, to confirm the system is bug-free and matches the designs to deliver expected results. The test data will include copyright-free images collected from the internet. Application to for ethical approval will be submitted for this data collection.
5. To evaluate the accuracy of the program by running the program to see how close the total of faces detected an image compared to a manual sum of the people in the image. Multiple compilations will occur to aid in statistical analysis to publish a universal degree of accuracy.
6. To present the project in its final stages including the evaluation of the results and implementation of the algorithms and receive guidance when completing the dissertation for submission.
7. To be able to record and write up the process of the overall project and to produce the dissertation document ready for assessment of the final year project.

4. Project Approach

Since MATLAB has a variety of statistical and machine learning packages, as well as the implementation may not require and object-oriented approach. It is most likely that MATLAB will be used to develop the system. However, other tools maybe used that may produce a better-quality implementation. Other tools that were used by case studies explored in the literature review will be noted and weighted against MATLAB to consider pursuing for this projects implementation.

The literature review will take place where the studies of previous software and technical solutions were developed relating to this project. Some of the topics may include image processing, machine learning concepts, neural networks and face detection. Applications and real-life problems will also be explored such as crowd-based situations to better understand the prototype that needs to be built.

The designs will include Unified Modelling Language diagrams to view the process of the techniques and algorithms used in the program. Abstract models will be shown (paper-based designs) to express the expected output visuals according to the techniques applied on an image. Pseudocodes will be presented to give an idea of how the algorithms will be programmed.

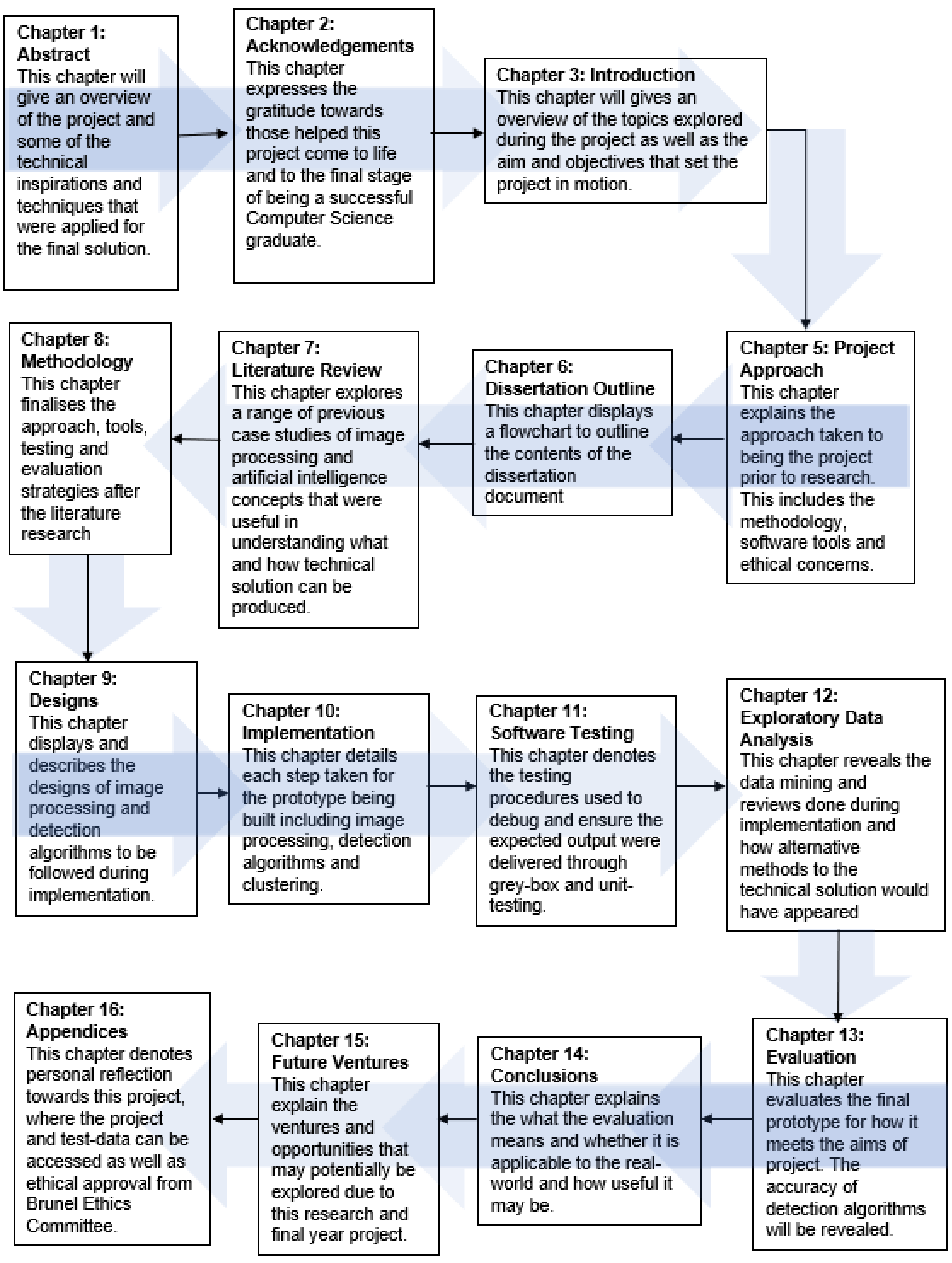
The Rapid Application Development methodology will be used to prioritise the intended functionalities of the system to work. These include image processing and face detection algorithms; therefore more time will be spent on ensuring the end results are as accurate as they can be.

This project will not involve any participants as the images/ photographs of people will be collected online. These images will be made sure to have no copyrights attached to ensure ethical approval. They will be used during the implementation, testing and evaluation stages of the project. Time is a limitation as the final year project and the dissertation has to be submitted in April of 2018 therefore the organisation and keeping up with priorities is key to achieve maximum potential and marks for this project.

After the literature review, more insightful knowledge will reveal some of the techniques, theories and the tools that were used. After these areas are assessed, a final decision will be made to which tools, methodology, techniques to apply during the course of this project.

5. Dissertation Outline

This chapter breaks down the brief notations of each chapter in this dissertation. This provides an insight of how the project took place and the effecting chapters that influenced the following chapters. There are 16 chapters in total starting from the abstract to the appendices. There is an additional chapter that reflects the creative motions of the implementations of the project – Exploratory Data Analysis.


Chapter 1: Abstract
This chapter will give an overview of the project and some of the technical inspirations and techniques that were applied for the final solution.
Chapter 2: Acknowledgements
This chapter expresses the gratitude towards those helped this project come to life and to the final stage of being a successful Computer Science graduate.
Chapter 3: Introduction
This chapter will gives an overview of the topics explored during the project as well as the aim and objectives that set the project in motion.
Chapter 5: Project Approach
This chapter explains the approach taken to being the project prior to research. This includes the methodology, software tools and ethical concerns.
Chapter 6: Dissertation Outline
This chapter displays a flowchart to outline the contents of the dissertation document
Chapter 7: Literature Review
This chapter explores a range of previous case studies of image processing and artificial intelligence concepts that were useful in understanding what and how technical solution can be produced.
Chapter 8: Methodology
This chapter finalises the approach, tools, testing and evaluation strategies after the literature research
Chapter 9: Designs
This chapter displays and describes the designs of image processing and detection algorithms to be followed during implementation.
Chapter 10: Implementation
This chapter details each step taken for the prototype being built including image processing, detection algorithms and clustering.
Chapter 11: Software Testing
This chapter denotes the testing procedures used to debug and ensure the expected output were delivered through grey-box and unit-testing.
Chapter 12: Exploratory Data Analysis
This chapter reveals the data mining and reviews done during implementation and how alternative methods to the technical solution would have appeared
Chapter 13: Evaluation
This chapter evaluates the final prototype for how it meets the aims of project. The accuracy of detection algorithms will be revealed.
Chapter 14: Conclusions
This chapter explains the what the evaluation means and whether it is applicable to the real-world and how useful it may be.
Chapter 15: Future Ventures
This chapter explain the ventures and opportunities that may potentially be explored due to this research and final year project.
Chapter 16: Appendices
This chapter denotes personal reflection towards this project, where the project and test-data can be accessed as well as ethical approval from Brunel Ethics Committee.

6. Literature Review

6.1.1 Crowd Management: Stampedes in Events

There has been many crowd control or management situations that have gone wrong. People have been uninformed of the events time or location information. In 1989, a disaster at Hillsborough stadium in the UK struck where people became victims of a Human stampedes and crushes (Kearns, 2011). 96 Liverpool FC fans were killed and 799 suffered injuries.

Situations like these can be controlled with the use of loud speakers which lead the crowd to safety or keeping the arms close to the chest to have more breathing space (Ripley, 2008). These are potential solutions when masses of people are in dangerously tight area, however there are ways that this situation can be avoided from the start. If people that were entering the compact area were accounted for, and properly monitored with real facts rather than face value, then the authorities would have made better decisions.

Technology in 1989 was obviously not as advanced as it is today, but the lesson can be learned to ensure that this event in history does not repeat. As grateful as it is to hear of the victim's families, survivors and supporters recently gaining justice, many precautions have been taken since by the authorities including stadium construction engineers to ensure safety of the fans in the future for any sporting event.

A total number of people in a crowd need to be accounted during a given time, here raises a question as to how to track the flow of people, and that computationally. The next case study answers this question.

6.1.2 Crowd Management: Lost children in School Trips

Handling a situation full of adults who don't like to be told what to do can be difficult. Children however is whole different situation. As children young as 5-10 years old tend to be mischievous, it can be difficult to manage their behaviour. They have to be looked after all of the time, otherwise losing track of their movements can cause serious panic.

A case of losing a child occurred when a 5-year-old boy went missing for 2 hours in 2015 in Scotland during a school trip at a zoo (Williams, 2015). There were 300 students accompanied with 70 adults, this proportional ratio indicates the children kept well in check. The child was missing since the busses left from the zoo and nobody knew where he was. In fact, he was in an area full of wild animals. Imagine if a child lost on the London tube during rush hour with oncoming trains and countless underground tunnels (Newland, 2012).

This must have been a classic form of mis-calculation error. Even if children nowadays have to wear 'hi vis' vests in an attempt for children to be visible to teachers on a trip (Hoodies4Schools, n.d).

Adults count children in pairs to make sure everybody is present. Unfortunately human error is inevitable and has to be forgiving. This raises a question of 'what if human-error is out of the equation?'. What if a system or technology allows an accurate acknowledgement of every person at given time.

An affordable device with sensors can be useful to potentially analyse the crowd somehow, for example a camera phone, can be used to take pictures to analyse further. The next case study is an inspirational story of how the developer used camera technology to solve a problem using Ai.

6.1.3 AI Eye Disease Diagnosis System

A (then) teenage computer scientist developed an AI system to help diagnose and reduce damage to her grandfather's eyes, which was a complication of diabetes that damages the blood vessels in the retina (Bleicher, 2017). Kopparapu, the developer of this system, was motivated to develop a system that be of use by doctors to diagnose diabetic retinopathy in a more affordable manner. Meaning that millions that didn't have access to insurance or the expenses to use a multi-dollar retina image scanner just for a diagnosis, can now have a better chance of being treated (Bhave, 2003).

This technology simply required a smartphone with a 3D lens camera to capture an image of an eye. Her team began by using an architecture known as convolutional networks that aided in parsing vast sets of data which was 34,000 retina scans. The patterns of similarity between these images came to surface which then successfully diagnosed diabetic retinopathy of the same stature of a human pathologist for the first time in 2016.

The inspiration projected by this story and the success of this system generated many different leads for this project. For example photographs of people can be used capture the moment where people are at a given time. Just like previous retina scans were used for medical diagnosis. Images in general can be processed in many ways to expose certain useful information like objects, shapes, and patterns just like image processing and segmentations systems.

6.2 Machine Learning: Neural Networks

Artificial intelligence is the larger concept that encompasses human intelligence to act such tasks as humans do (Marr, 2018). Machine learning is the application derived from artificial intelligence which learns to do specific tasks after being trained from learning using data. Majority of these application are automation system that are used for people in business, entertainment, engineering and much more. Some of the technology that people use in their daily lives include Smart cars, video games, online customer support (chatbots), and security systems (Albright, 2016). The applications are endless and are constantly being used to find solutions to recent problems everyday.

As machine learning and neural networks are constantly in the news of their rapid advances relating to data science, big data, and business intelligence (Oswal, 2018). As well as the previous case study also used (convolutional) neural networks in their venture, this project may well benefit from neural networks. Therefore an in-depth research into neural networks and a plan of how these would be used for this project was commenced.

Neural networks, with its name, depicts a biological connection specifically the brain (Artificial Neural Networks Technology, n.d). Since the enigma of the brain has been studied for thousands of years, its idea of construction came about in the early 20th century. Along with this time, statisticians and electronic engineers managed to harness this concept.

Warren McCulloch and Walter Pitts with their combined expertise, being a mathematician and neurobiologist modelled a simple neural network with electrical circuits. These concepts were advanced in the 1950's by Frank Rosenblatt a respectable neurobiologist, who realised a fascinating approach. Most of the processing that tells a fly to flee is done in its eye (O'Riordan, 2005-6). Meaning that this perception occurs from stimuli eventually relating to a decision the fly makes in the end.

The concept of a Perceptron was a result from his research, which is an approach that is still being taught and industrially used today. The concept introduced a single-layer perceptron that involves two sets of input classes into a neuron that results in one of two possible outcomes depending on the weighting of those inputs. Figure 1 is diagram that illustrates it (Tucker, 2017):

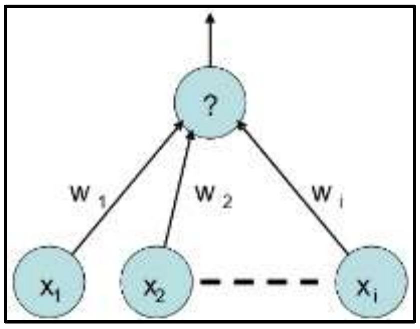


Figure 1: *Perceptron Diagram*

Neural network ideology continues with visual perceptions with Hubel and Wiesel's research in 1962 (Hubel et al, 1962). Their findings introduced a remarkable understanding of how the retina is transformed into the orientation sensitivity in the cortex. For example knowing the difference between a dog and a cat according to its body curvature or certain features. Most dogs have a longer snout or a cat's ears may be small and pointy.

The documentary perfectly demonstrates this (Christian G, 2013). The crackling sounds as the light beam moves across the retinal image is the respondent of stimulation of neurons. This operation is a form of analogous edge detection (SteveLehar, n.d) which consists of adjacent positive and negative values. Imagining this on a larger scale sheds a light on how living organisms tend to perceive and recognise animals or any other objects for that matter.

This study directed this project to look into edge detection and other segmentation and image processing methods that are commonly used to today. Especially when processing digital images inevitably involves image processing. So a few studies were explored to understand their applications and technicality to feasibly apply this if it became a suitable option to implement.

6.3 Image processing

During image analysis the first stage is usually segmentation. Each pixel of an image (RGB) is assumed to have different regions or intensity values for each pixel (Automatic Thresholding, n.d). Thresholding images is a massive part of image processing as it's bias (T) classifies each pixel according to an intensity value given to it. Two of the most common image processing methods were researched, and an incredible study that explored these methods was reviewed as inspiration to this project.

6.3.1 Image processing: Hough Transformation

As brilliant as it is to see the earliest simulation of edge detection developed by Hubel and Wiesel. A popular edge detection image processing technique available to work with today is called the Hough Transformation. This image processing tool has been expanded upon since the 19th century until P.V.C Hough developed in 1962 (Hart, 2009). It is used to detect geometric features like straight lines and curves in digital images.

The edge pixels are collected using a Canny edge detector or a Sobel edge detector. This will map out the highlighted parts of the image that are detected as edges or curves. The Hough transformation will provide the geometrical representation of the edge, including the slope and intercept using the equation of a straight line (y = mx + c). When re-arranged to favour the intercept (c = y – mx), the Hough algorithm places lines from the intercept according to the gradient. The intersection of the lines are represented as parameters of the line. Figure 2 demonstrates this (Sinha, 2010).

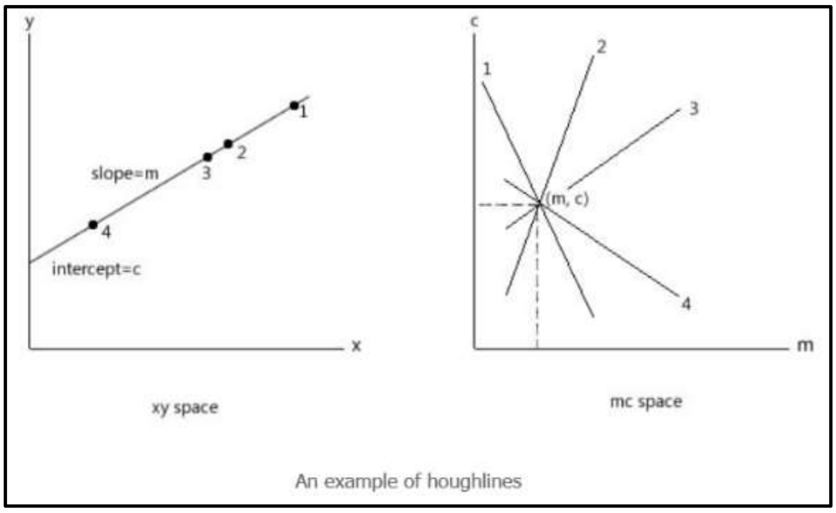


Figure 2: *Hough Transform Diagrams*

6.3.2 Image processing: Otsu Segmentation

In the case of Otsu segmentation developed in 1979 by Nobuyuki Otsu, which uses global thresholding (Baxi et al, 2013) by default where the classifications fall into variances (measure of spread) of foreground and background. A histogram of greyscale is produced to show the range of grey pixels (black to white). The background and foreground histogram bars are then separated by calculating the overall variances (measure of spread).

The level of thresholding can vary according to the figure 3 and the experimenter’s evaluations (The Lab Book Pages, 2010). For example, the image here is processed with a threshold to the value of 3. The result of the Otsu segmentation implicates that all pixels with a level less than this threshold are the background (black pixels) and the pixels that are greater or equal to the threshold

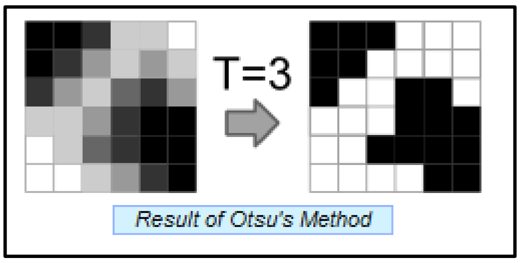


Figure 3: *Otsu Segmentation Diagram*

are the foreground (white). This is a good example of segmentation between the original image to the processed image.

6.3.3 Combinations of Segmentation

Baygin et al (Baygin et al, n.d) published research based on a machine vision application for object counting, with the use of image processing. Their goal was to be able to accurately count items that require the counting of objects during production, typically on a conveyor belt. They used Otsu and Hough transformation techniques to uncover key objects (grouped pixels that count as one object) on a real-time digital image to later be used to count the objects.

An example is shown in figure 4, of their successful results was when the digital image they used was of egg packets where, along with other techniques, the Otsu and Hough Transformed images detected each and every egg which enabled the machine vision application to accurately count the eggs (Baygin et al, n.d).

What is most intriguing about this study is that, it showed how the techniques worked well when together. This involves the testing of different orders of these methods (Otsu then Hough or, Hough then Otsu) to review the output images that can be acceptable for accurate counting. This means that there may be many other techniques that would be useful with each other to reveal interesting visuals from an image.

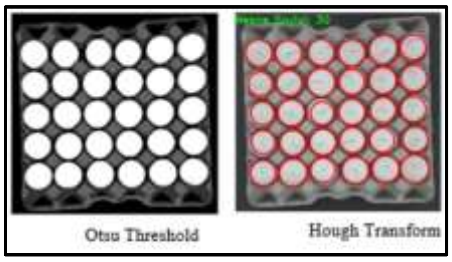


Figure 4: *Image Processing of and Egg Packet*

6.3.4 Image processing: Watershed Segmentation

The Watershed image segmentation method is closely related to the Otsu and Hough transform methods. This is because it is region (background vs foreground) and contour-based (edge). A Watershed basin analogy relates the ridge between two basins or regional draining. This segmentation begins with the simple separation of the objects of from the background, generating many morphological formations. Some examples are the segmentation of DNA dotted samples or grainy objects (MathWorks, 2002).

The adaptive nature of Watershed segmentation was put to the test by Hodneland et al (Hodneland et al, n.d). Their research was based on testing 'level sets' of the watershed segmentation. This means to construct a binary image from a grey-scaled image of RBG, to morph into watershed segmented image. Their image clearly stated a boundary of the shape and the inner regions. They

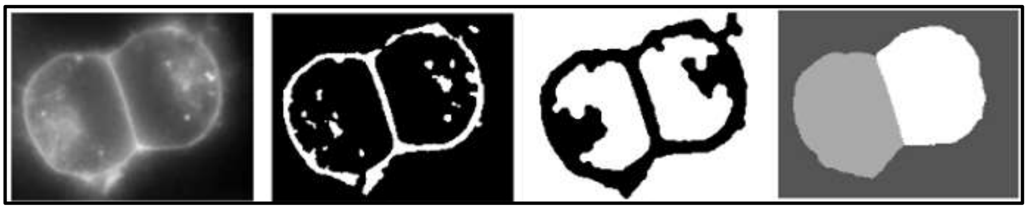


Figure 5: *Watershed Segmentation Examples*

further introduced the result of the Euclidean influence (the last image) from the previously processed images, which depicted two contrasting regions (identification of coloured locale).

As the resulting images shown in figure 5, of being processing through these 3 methods, they clearly reveal information that separates key aspects of the image. By detecting the edges, regional areas and coloured locales, the segmentation methods combined can help identify key objects in an image. For the purpose of this project, these methods (Otsu, Hough, Watershed segmentation) can help identify the bodies of the people in terms of shape, outline or even shade colour. This will then help reveal the amount of people there are quantitatively.

6.4 Machine Learning: Convolutional Networks

Convolutional networks were first introduced by Yann LeCun in 1994 after many iterations starting from 1988 (Culurciello, (2017). This architecture was called LeNet5 which expresses in their study that a smaller/ learnable number of parameters are an effective way to learn and extract features at multiple locations on an image (RGB). Yann LeCun et al also state that the convolutional nature of their program achieved their smallest test error rate compared to other various models they were testing (LeCun et al, 1998).

Understanding the technology available during that time - no GPU's and slow CPU's, this architecture was inevitable and extremely computationally cost effective. There has many advances after this discovery, that pooling also controls overfitting which affects generalizability since the training is done so precise that the test data may not be recognizable. Hinton et al (Hinton et al, 2014) has also specified multiple ways to program better optimizing algorithms by accepting to let go of established binary-output neurons as well as method like drop-out layers and back-propagation.

Ai and machine learning, specifically neural networks is being used and developed now more than ever. Since the very first simulation of Perceptron, to different branches of machine learning with biological and life-like ideologies including Genetic Algorithms, Hill-Climbing, Ant Colony Optimization and the Travelling Salesman Problem (Le, nd). This is increasingly being absorbed in the world of Data Science which uses mathematical approaches like Regression, Decision trees, Bayes probability and Logic. Together forming a new disciplinary art form.

Big corporations like Facebook, Pinterest, Amazon and Google use Ai to optimize their respective systems. Like Amazon for their product recommendations and Google for an effective photo/image search. (Deshpande, 2016) There are many uses of NN including image processing. Interestingly Google corroborated that 'the internet is made of cats' July 2012 (Clark, 2012) by running a series of unlabelled images through a neural network, eventually learning a cat-detecting neuron.

Understanding neural networks and its applications, a key extension of image processing includes recognition. This form of recognition is the result of vast machine learning operations in the background that help identify the final image. Just like Google claim the internet is made of cats, recognition software can be used to detect anything that its been trained for. Facial recognition in particular is very popular and is very relevant to this project which led to focussing more on this subject.

6.4.1 Machine Learning: Facial Recognition

It was only inevitable for a facial recognition system to be developed using convolutional networks. Mansouri et al explained how they managed to extract facial features with different lighting and facial expressions from digital images (Mansouri et al, 2012). Their network architecture was inspired by LetNet5 where the network comprised of 7 layers involving convolutional, sub-layers, fully connected layers and an output layer. Their results showed the benefits of training the convolutional network as a feature extractor and applying a logistic regression, allowing them to achieve their highest accuracy being 86.06% within their dataset.

There are many applications of facial recognition systems that are developed for their own purpose. For example Facebook uses their facial recognition software to automatically tag a user on a photo according to the users friends-list (Geitgey, 2016). The very first part of these systems extract information like the distance between the eyes, width of the nose, length of jawline and much more (Ex-Sight, n.d).

In MATLAB's Computer Vision System Toolbox, there are a series of available object detectors that are useful for recognition purposes (MathWorks, n.d). The 'visionCascadeObjectDetector' in particular is useful as it is set to detect faces by default unless the classification model is altered to detect other specific objects. Viola-Jones algorithm is a straightforward algorithm that returns an m-by-4 matrix which includes the coordinates and dimensions of the bounding box objects used to surround the faces detected.

As this project simply requires a face detection system (detecting faces), not specifically a recognition system (recognises faces for authentication or security). The Viola-Jones algorithm (open source) does exactly that. Wang (2014) published a study which analyses the Viola-Jones Face detection algorithm by breaking it down and explaining using the pseudocodes (Wang, 2014). An effective part of this algorithm works by the use of boosting technique called AdaBoost. This is when there is a combination of weak classifiers into a single strong classifier. This is so the performance is better than random learning.

When defining classifications, depending on the problem they can usually give you accuracy results above 50%. However there can be many mis-qualifications with regular approaches, which the AdaBoost technique can help improve. An example would be to establish weak classifiers just as much as strong classifiers if not more. This trains the system to look for what is intended as well as knowing what not to look for when testing. In the case of the Viola-Jones algorithm, the weak classifier is the selection of non-face images and the stronger classifier images are of actual faces.

As it's shown that Viola-Jones algorithm is useful and used for further research, this project may also benefit from its implementation. As MATLAB also supports the language especially as it was initially implemented on OpenCV application, MATLAB may also be where the initial implementation will take place. An understanding of how the detected faces are stored and how else to manipulate this data will also be acknowledged.

6.4.2 Machine Learning: Training & Testing

A study by Majid et al on face detection using Adaptive Boosting (AdaBoost) technique explains, with analysis of Viola-Jones algorithm, that face detection can be difficult to get right due to a large amount of variation and complexity brought by the appearance (Majid et al, n.d). They concluded that (Support Vector Machine) SVM-based supervised learning along with AdaBoost achieves great results relating to the problem they were solving. This makes one realise that there may be many other reasons why a face, after a powerful learning session, is still not detected. When understanding why a face may not be detected due to lighting or a certain angle the photo is taken, this opens an opportunity of creating supervised approach to develop a new technique for feature selection to hopefully achieve even higher results.

MATLAB's object detector has the option to train classifiers according to the users' project (MathWorks, n.d). As a tutorial, the detector is used capture stop signs with the use of the AdaBoost technique. This is where a series of 'negative' images and 'positive' images can be trained against each other using the 'trainCascaseObjectDetector' objector. Then tested on an image using this newly trained objector which is stored as an XML file. In the context of this project, this can be used to train faces that aren't detected using default face detection.

For training images according to their set classifiers, there needs to be a sufficient amount of data to work with. Many algorithms require a different amount of data in order to get a high accuracy. The example given above has 42 positive samples and 84 negative samples. Which implicates that any amount less that these may result in the algorithm not being effective. Perhaps a smaller positive set of images can be used if they are higher quality? No, as the training set is used to predict the future results when testing begins, the results will either be inaccurate or misleading. In fact the quality of images are irrelevant as a study in Image Quality assessment in Neural Networks by Chen et al explaining that their proposed scheme achieved high prediction across different distortion types and levels (Chen et al, 2017).

When training an algorithm for face detection of which its purpose is already being face detection seems somewhat absurd. However there may be many reasons why a face is not detected, which will only be discovered during testing phases. It may be necessary to train the dataset after noticing a potential classifier missing. Though it can be very time-consuming and difficult to find a large dataset to work with therefore options need to be weighed carefully before taking this step.

6.4.3 Machine Learning: Facial feature Selection

Just as a face has its own properties to detect that it's a face (position of eyes, nose and mouth etc). For a facial feature like a mouth or a nose, there are deeper properties to acknowledge. For example a width of a nose, the fact that there are two nostrils per nose etc. The Viola-Jones algorithm applies this approach to detect the nose, mouth, and upper body of a person in an image. Maghraby et al have examined these properties as feature extraction and have also built upon this to develop a better algorithm that has a detection rate of 37.2% above the Viola-Jones algorithm (Maghraby et al, 2014). Nevertheless the inspiration has come from Viola-Jones' approaches itself. The point here is that feature extraction, individually, can identify a person that may not be detected by face detection due to environmental distortions (E.g - lighting). Especially when only one aspect of a face is visible.

When carrying out feature selection methods as well as initial face detection, the results are usually vast depending on the problem. Feature selection is mostly useful to bring to light what is not risen from the first method. Like eyes, nose and mouth detection as feature detection and facial detection as the initial method. With visual representation, data can emerge to a new perspective and can lead to further analysis. Like the clustering of data closely related face detections. However the redundancy of data needs to be controlled, this is when multiple data represents the same thing. A method of elimination needs to be set to refine the dataset.

With regards to this project, the face detection may detect all faces as well as their facial features. This means that an account of these detections in total will mean that there are more people than there actually are in the image. However face detection alone may have less error than features detection as features detection itself, which encompasses many variables supporting face detection while features detection relies solely on the shape and position of a potential feature. As the benefits of feature selection were explored in detection algorithms, an elimination process needs to take place to refine the results before quantifying the amount of people in the image.

6.5 Artificial Intelligence: Optimization Algorithms

Heuristic algorithms are an iterative process which uses mathematical comparison to collect better solutions until the best is found (termination). This means that the proper solution to a problem may not be found, however the best possible solution is found within reasonable time. Hill Climbing (HC) is a classic algorithm (GeeksforGeeks, n.d) that uses the greedy/ exhaustive approach, moving in one direction (descending or ascending) within the search space, hence the hill climbing analogy.

There have been algorithms that have advanced using this algorithm like the Stochastic HC, Random Restart HC (RRHC) and more. These algorithms are fit for their own purpose, like stochastic involved random uphill moves and the RRHC involves further iteration practises and is usually very fast. Hill Climbing is good to begin with to understand the behaviour of the algorithm and the results that appear (StackExchange, n.d). Then further improvements can be made to get better solutions than before.

As discussed thresholding during image processing, they are a great way to segment data in their own respect to reveal information. The concept of data being separated during image processing via a threshold value is helpful in many respects. For example a neural network may produce units or values at the end of its training and is separated by a threshold value (ExplainTahtStuff, 2018). Those values achieve a simple classification of 0 (nothing) or 1 (trigger/fire). This is a similar ideology to Entropy, where thresholding is a measure of fuzziness (WaveMetrics n.d). A major application of entropy as a solution would be to use 'Shannon entropy' to help classify unpredictable results (Preiswerk, n.d). This threshold value is a result of multiple testing and evaluation sessions to ensure proper classification.

Optimization algorithms help find the best possible solution in their own respect, but they do not always work. There are countless alternatives and ideologies that can be applied. The common denominator here is there is always switch or a thresholding concept that causes data to flow in different directions. Whether it being grouped or a simple elimination process. Regarding the refinement of multiple detection algorithms, many tests and practical ideas may stem from the concepts mentioned above to reach the intended objective.

6.6 Artificial Intelligence: Robust Clustering

Clustering is the act of organising data into subsets of characteristic(s). These characteristics can be many depending on the problem (like age, eye colour, etc.). A reason for generating clusters may be for data visualization for predictive analysis, that reveals certain observations, or instances simply represented on a table (Dummies, n.d).

Clustering algorithms in Data mining specifically have several models to implement the partitioning of data (Big Data Made Simple, 2015). This includes centralized, distributed, density and many more approaches. These techniques are adopted with respect to the developers understanding of a potential relationship within the data obtained. These models have different cluster states which involve soft and hard partitioning. Hard partitioning allows an object to strictly belong or not belong to a cluster while soft partitioning has specific divisions which allow objects belonging to multiple clusters. The soft partitioning approach involve hierarchical trees (decision trees) to which reflect the anticipation of object clusters.

There are many cluster algorithms, like K-means being very popular. The partitioning methods (mentioned above) grant object passes into clusters deterministic and forceful way. An algorithm relating to this is the robust clustering algorithm. This algorithm engages in carrying each object or group of objects to a cluster that already has a similar object containing it. This also means that clusters that contain the same objects merge in agreement without question.

A study by Swift et al developed an application of robust clustering for functional interpretation of gene expression data to generate consensus clusters (Swift et al, 2004). They express that their data had been subdivided into smaller clusters for a better definition (higher agreement) of the characteristics like ‘B-cell lymphoma’ and ‘ASC’. This means that this algorithm allowed and accelerated fine tuning of the data set, in this case the gene-expression dataset.

As this project includes a dataset of faces and facial features detected. The robust clustering method can be used to further refine the dataset, especially the facial features. An example of this would be to cluster the facial features close to each other and count as one person.

6.7. Literature Review Summary

The literature review will be useful in appropriately using the techniques and concepts in certain areas of the implementation. The case studies and their successful results has inspired the project to take forward similar applications to fulfil the aim of this project. The image processing and detection algorithms will be used for development in particular. The face detection literature in particular flowed in place to the point that it may be successful in detecting face when used together. There were also other paths that are open if something does not work, like training a classifier using AdaBoost technology.

The literature review brought out a view of the project that may involve creative processes and trial-and-error situations to help develop the prototype at its best of its ability and be applicable to the crowd-based scenarios.

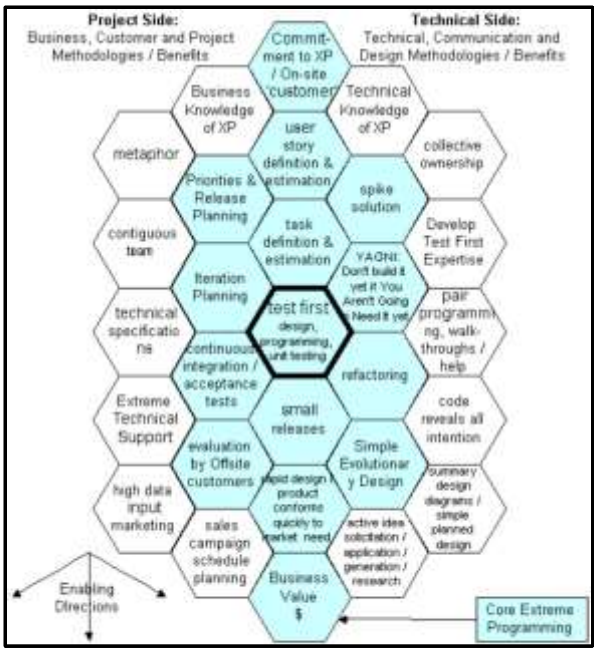
7. Methodology

This project consists of image processing and detection mechanisms that require iterative software testing stages, due to better decision-making after reviewing the results and knowing which way to take the project further. Agile is actively being used in software engineering industry, 94% of firms practise agile methodologies. Obviously due to a constant change in requirements and cost management in refactoring software (QA Symphony, n.d). The popular methodologies are explored to make a decision on which methodology will fit the nature of this project.

The classic methodology that has been practised since 1970 is the waterfall method (Powell-Morse, 2016). This imitates the 6 stages of falling water, these includes the forward motion of software development stages like requirements, analysis, design, implementation, testing and maintenance. This method is advantageous in terms of costs as the next stage is perfectly executed before it is dispatched to the next. However a major issue with this method is that it does not allow room for refactoring or changes, which in turn becomes costly to re-do.

Extreme programming is a popular section of Agile methodology. It's primary principle is to continuously develop a software program according to dynamic requirements in rapid forms. It is closely related to test-driven development as it follows a test-first approach shown in figure 6. Meaning that a program will be developed highly revolving the requirements and errors would be closely guarded (Ghahrai, A., 2016). Though extreme programming requires intense training in a specific environment to develop the software which is cost and time consuming (Partogi, 2016). Not to mention extreme programming is usually a collaborative undertaking and will most probably be useful this way than a single developer.

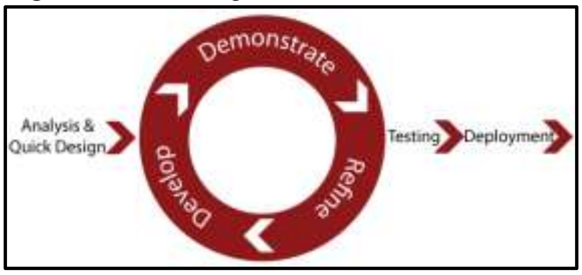
Figure 6: *Test-Driven Development*


Project Side:
Business, Customer and Project Methodologies / Benefits
Technical Side:
Technical, Communication and Design Methodologies / Benefits
Commit-ment to XP / On-site customer
Business Knowledge of XP
Technical Knowledge of XP
metaphor
user story definition & estimation
collective ownership
Priorities & Release Planning
spike solution
contiguous team
task definition & estimation
Develop Test First Expertise
Iteration Planning
YAGNI: Don't build it yet if You Aren't Going to Need it yet
technical specifications
test first design, programming, unit testing
pair programming, walk-throughs / help
continuous integration / acceptance tests
refactoring
Extreme Technical Support
small releases
code reveals all intention
evaluation by Offsite customers
Simple Evolutionary Design
high data input marketing
rapid design / product conforms quickly to market need
summary design diagrams / simple planned design
sales campaign schedule planning
active idea solicitation / application / generation / research
Business Value $
Enabling Directions
Core Extreme Programming

Rapid Application Development (RAD) is another sub-methodology of Agile (Rouse, 2016). It embraces the idea of iterative processes and stages during the development of a prototype. Figure 7 demonstrates this process, the circular motion is the iterative processes that depicts the prototype development (I.T.Weld, n.d). Due to the limited time of this project and the flexibility of changing requirements this methodology seems very relevant for this project (Activo Hub, 2016). Some of the cons include cheap modelling and tools for code generation and a high level of user involvement. Since this project has no user involvement, and the system is about processing an image using machine learning and image processing, that con is not worth considering. The nature of this project is experimental and consists the developmental strategies that involve trial and error. So the final decision is to use RAD as a methodology.

7.1 RAD Process

Figure 7: *RAD Diagram*



The analysis and requirements analysis will occur through the period of literature reviews and research. This will give a clearer understanding of what to expect with the studies and their findings on image processing. Unified Modelling Language (UML) design and abstract designs will be created to help visualize the processes of the system and what the outputs should display at relevant stages. Pseudocodes will also be presented to express some of the algorithms used for data analysis

The prototype will begin by implementing the image processing algorithms including Otsu, Hough, and Watershed segmentation. Then the testing and inspections will be carried out to ensure the outputs relate to the designs. The detection algorithms will also be implemented and tested. Further analysis will include evolutionary stages of the prototype to bring the total number of people in an image. Such evolutionary stages may include clustering exercises.

7.2 Test Data Input & Ethical Consideration

The input of the system (test images) will be found online, Google images can be used to select them with the tools section set to labelled for non-commercial reuse which means that the images are copyright free. Ethical approval from the Brunel Research Ethics Committee was received to confirm that this project does not require further ethical approval regarding participants (Brunel University London, n.d).

These test images (13) will be a series of photographs that include a range of multiple people in it, which hope to challenge the system for its betterment. The images are a range of real-life photographs. All will be used during the evaluation, first 5 will be used as input during the implementation and software testing stages.

7.3 Computer Tools

The literature review included simulations and image processing implemented on MATLAB. Not only does MATLAB have considerable data and table handling mechanisms for simple manipulations, they have an Image Processing, and Computer Vision toolbox that supports visualization, analysis, image and geometric segmentations (MathWorks, n.d). All useful when developing an application of graphical importance.

After research on other potential programming languages for this project, OpenCV appeared several times. This is because OpenCV and their Computer Vision Library is compatible with 4 programming languages and is open source (OpenCV, n.d). However this is seen as a disadvantage because the online support is limited, especially being specific to each language. Therefore the project will proceed with MATLAB.

7.4 Exploratory Data Analysis

The RAD methodology will require data analysis in order to move forward with the project. This includes visual reviews of the output images and noticing patterns that will help form the solution for the next stage of the prototype.

This means that an attempt to let the data and output reveal underlying structures and ideals as opposed to applying a strict implementation model to it (Engineering Statistics Handbook, n.d). This will be useful as the segmentation and detections methods do not always bring out concrete revelations. Also, the output images are not only the data which will be processed further, it is most likely to be a visualisation technique to discover patterns. But the data that is produced in accordance to segmentation and detections may allow manipulations to some kind of series of values that represent the output images. For example, the Hough Transformation produces Theta, and rho values.

A limitation of this approach is that the goals or final product may be altered into a product that serves a different requirement. This happens due an inconsistent reminder of what the ultimate goal of the system is (Universal Teacher, n.d). However this project does not require mining or handling of complex data structures, and this process may also provide solutions that fail which creates a learning curve to reach closer to the solution. Therefore this project may benefit from this stage to produce a system that applies artificial intelligence concepts to reach the end-goal.

7.5 Software Testing Strategy

A potential method of testing could be White-box testing. This is the act of testing the internal structure of a software system (Guru 99, n.d). This means that the components within a system that pass the input through is tested to ensure the expected output is delivered. White-box testing means a quick testing phase as the code associated with the input is tested so the code that does not related to the input is remained untested and may be the cause of bugs that will be difficult to find after progressing forward.

Opposite to white-box testing is black-box testing (Software Testing Fundamentals, n.d). This is an exhaustive approach that tests every single branch of the system regardless of its relevance to passing inputs, which makes sure every part of the program works and is bug-free. Though this method may be time-consuming to test every branch of the code.

Therefore grey-box testing will be used to test the system of this project. White-box testing will ensure the expected output is delivered while black-box testing will ensure every aspect of the code is bug-free. A practical approach to this is unit testing. Unit testing is when each line or aspect of the code is tested with test data or input of a previous operation (Rouse, n.d). As this project may not involve a wide range of branches since it is expected to be an evolving prototype, these strategies will work well together. The testing will occur with data that is output from a previous part of the code unless a particular condition is not being triggered, then newly formed test-data will be created to test it.

7.6 Evaluation

The evaluation stage will be conducted to review whether the aim of this project and technical solution was met. Which is to accurately count the total number of people in an image/photograph. And if so, how accurate is it to detect the amount of people in the image. Since this application is not a mobile application, there will be no evaluations relating to the usability of the system. The correct amount of people in an image will be manually counted to use to compare with detection or image processed outputs that derive the total number of people in an image.

The aim will be directly challenged by gathering 13 images as test data which will be inputted in to the system. The correct amount of people and the detected number of people will be compared to measure the accuracy. Depending on the prototype, and what algorithms consist of it, further analysis may take place to understand why the prototype did or did not meet aim, for example using face detection as the current means to detect the number of people in an image.

The data set will also be split into a smaller and larger data sets to compare which set allows the prototype to generate the number of people in the image or have a lower error-rate. The applications of the system will also be concluded to whether it will be useful in crowd-based situations.

8. Designs

Unified Modelling Language (UML) is the common use to produce software designs (Visual Paradigm, n.d). The visual aspect of designs are very important so that programmer(s) have similar perception of the system to follow as a planning tool when developing it (Technwalla, n.d). There are branches that include such as activity diagrams, data-flow-diagrams and more. This project will use these two designs to aid in implementation of image processing and face detection algorithms.

8.1 Activity Diagram: Image processing Design

An activity Diagram was created from the image processing methods to visualise the steps taken in the algorithm realm shown in figure 8. From the start node, the image will be inputted and processed with a series of segmentation methods until the total number of people in the image are found and quantified.

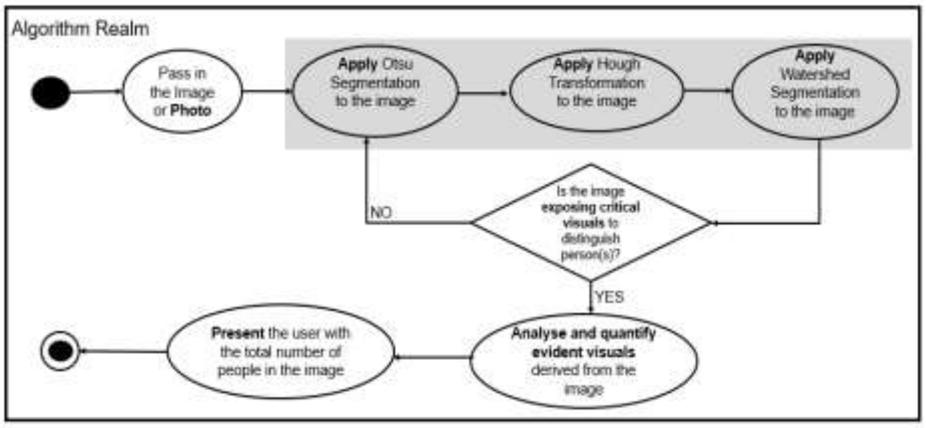
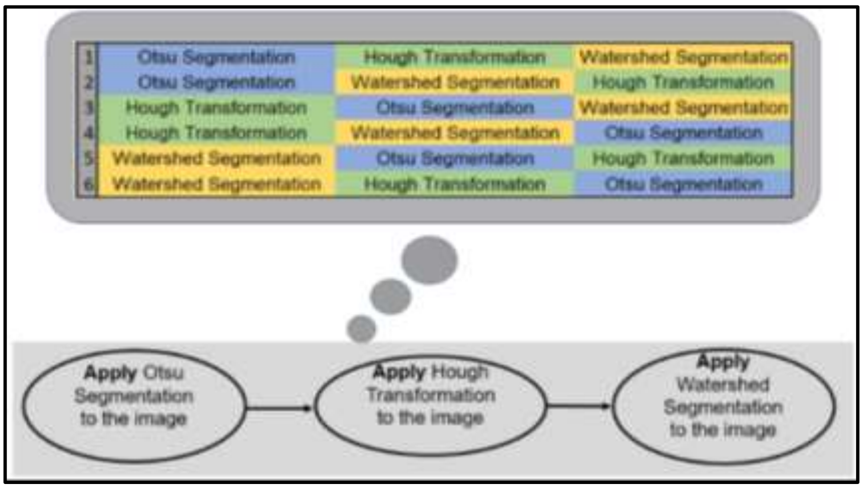

Algorithm Realm
Pass in the Image or Photo
Apply Otsu Segmentation to the image
Apply Hough Transformation to the image
Apply Watershed Segmentation to the image
Is the image exposing critical visuals to distinguish person(s)?
NO
YES
Analyse and quantify evident visuals derived from the image
Present the user with the total number of people in the image

Figure 8: *Activity Diagram for Image Processing*

As the literature review disclosed the success of output images in some of the case studies when using different combinations of segmentation methods on a single image, this project can benefit from this in the hopes of maximising success. The activities in the grey area represent the different



1	Otsu Segmentation	Hough Transformation	Watershed Segmentation
2	Otsu Segmentation	Watershed Segmentation	Hough Transformation
3	Hough Transformation	Otsu Segmentation	Watershed Segmentation
4	Hough Transformation	Watershed Segmentation	Otsu Segmentation
5	Watershed Segmentation	Otsu Segmentation	Hough Transformation
6	Watershed Segmentation	Hough Transformation	Otsu Segmentation

Apply Otsu Segmentation to the image
Apply Hough Transformation to the image
Apply Watershed Segmentation to the image

Figure 9: *Combinations/Different orders of Segmentation*

combinations of techniques applied to an image during procession on the image. Figure 9 illustrates 6 different combinations to be tested and implemented accordingly.

8.2 Data Flow Diagram (DFD): Detection Design

A data flow diagram was used to design the detection within the algorithm realm. This is because the use of storage, processes and external components will be clearer when the process flow takes place (Lucid Chart, n.d). As discovered in the literature review during the facial feature section, that an effective detection relies within the several inner detections. Just like Viola-Jones algorithm used facial feature detection in the correct position of the face, to determine the rate at which a face is detected. Feature detection will be incorporated as part of the design.

Figure 10 is the DFD which begins with the user as the external entity who input the image. Then the process of running the facial detection algorithm is executed as well as the features detection. The detections will be stored to eventually run a new process to remove redundant detections.

This is because they‘re detections that represent the same face. The features list is clustered to then add to the facial detection for a final total to implicate the amount of people detected in the image. This is displayed to the user.

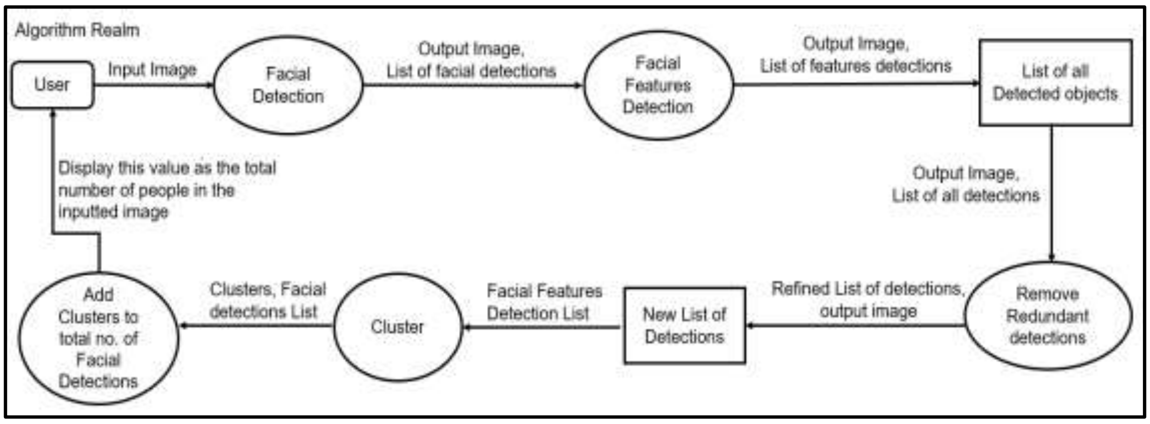

Algorithm Realm
User
Input Image
Facial Detection
Output Image, List of facial detections
Facial Features Detection
Output Image, List of features detections
List of all Detected objects
Output Image, List of all detections
Remove Redundant detections
Refined List of detections, output image
New List of Detections
Facial Features Detection List
Cluster
Clusters, Facial detections List
Add Clusters to total no. of Facial Detections
Display this value as the total number of people in the inputted image

Figure 10: *DFD of Face and Features Detection*

8.3 Abstract Visual Designs: Image Processing Design

Since UML diagrams will not define what output images are expected to look like, abstract visual designs (Odgis, 2017) are created to understand what each process should represent specifically during software testing.

Figure 11 is an abstract diagram that apprehends the outputs of each segmentation method that will hopefully. The Otsu method will detect similarities in regional pixels like people or particularly the colour of their clothes. The Hough method detect the edges of object in the images, like people’s body shapes. The Watershed will detect people through spatial recognition by differences in regional contrasts, like the person in the middle is shown darker indicating that they are in front of the other two people. After the different combinations of segmentation on an image, it is expected that all aspects of the image i.e. the people in the image are detected to then extract and count the total of people.

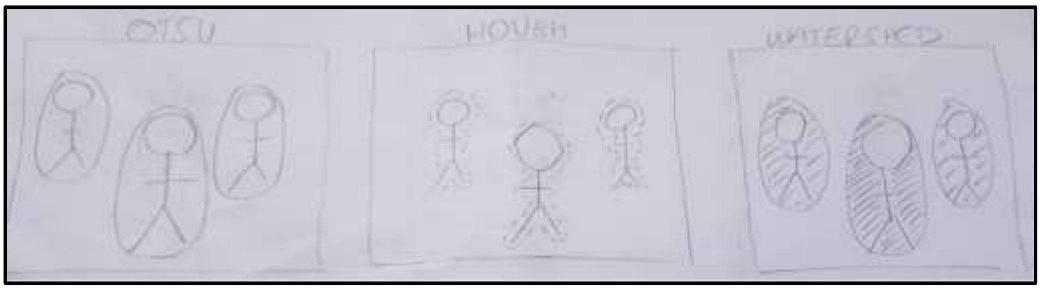
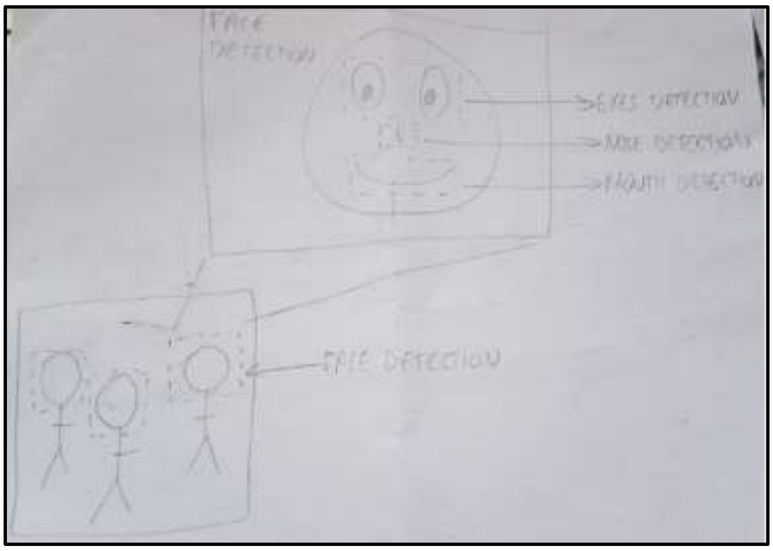


Figure 11: *Sketch of segmentation interpretation*

8.4 Abstract Visual Designs: Detection Design

There was a similar occurrence with the UML diagram for face and features detection with the image processing UML diagram. Processed images need to be understood carefully in order to interpret the data produces afterwards. Figure 12 shows a diagram that is an actual image (similar to the ones drawn for image processing section). That smaller box is the actual image where the face detection is indicated by the dotted line surrounding the faces of people in the image. The larger image is a magnified view of the features selection. The eyes, mouth and nose are detected, indicated similarly to the face detection. Assuming this will form a list of detected items, further analysis on data can be done to then count the people in the image.

Figure 12: *Sketch of Face and Features Detection interpretation*



8.5 Pseudocode: The Viola-Jones Algorithm (Detection)

Pseudocodes help understand a step-by-step process that is imitable when programming (WikiHow, n.d). Viola-Jones algorithm is an open-source face detection algorithm. After studying the structure and inner processes of this algorithm. A pseudocode is created to show how this algorithm can be used for this project:

```
1) READ Input image
2) INITIALISE FDetect = Visual Object Detector Function
3) PROCESS Image with FDetect
4) STORE FaceBoundingBoxes [ ] [ ] = Image with FDetect
5) FOR LOOP i= 1 to size of FaceBoundingBoxes
6)     DRAW rectangle with detected positions stored in FaceBoundingBoxes [ i ] [
i ]
7) END
8) DISPLAY output image
9) SAVE output image
```

The detection will be stored as a 2-dimentional array (list) that saves the pixel co-ordinates used to draw rectangles. The draw function is draws a square around the detected face, the display function displays the image with these squares to produce and output image. The output image will be processed several times within the algorithm realm show in the DFD, therefore save function is used towards the end of the algorithm. The loop is used to draw each square from the detected list. This algorithm will also be used for feature selection, the 'Visual Object Detector Function' will include parameters to look specifically for what is to be detected, like a nose, or mouth or eyes.

8.6 Pseudocode: Removal of Redundant Detection

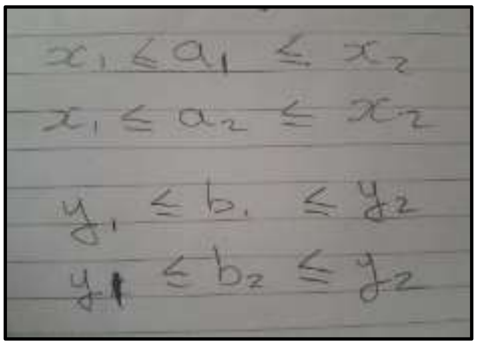
The features detection is used to capture detections that the face detector has not detected. This also means that the detections that have already been found using face detection will also be detected. Meaning these detections need to be removed so they don't not affect the final value that represents the total number of people in the image. The following pseudocode shows a process in which this takes place:

```
RUN FaceDetection (FaceBoundingBox[ ] [ ])
RUN FeaturesDetection (FeaturesBoundingBox[ ] [ ])
INITIALISE AllDetections = (FaceBoundingBox[ ] [ ] + FeaturesBoundingBox[ ] [ ]
FOR LOOP i = 1 to size of AllDetections
         FOR LOOP j = 1 to size of AllDetections
                    IF FeaturesBoundingBox [j ] [j ] is LESS THAN (<)
FacesBoundingBox [i ] [i ]
                    REMOVE FeaturesBoundingBox [i ] [i ]
END
```

Here, the pseudocode is comparing the face detection list against the features detection list. The nested loop processes the subject, a feature detection with all of the face detections. This iteration ensures all detection in the features list are compared against all detections in the face detection list.

If all the pixel co-ordinates of a features detection are within the pixel co-ordinates of a face detection, then it's removed. This is a logical approach as all features in a persons' face, is on a face. This will then alter the features list, to store the remaining features detections that haven't had face detection for that particular person. To ensure that the pixel co-ordinates are within range to successfully compare and taken action, the limits were drawn shown in Figure 13.

Assuming that x,y values represent the face detection and the a,b values represent the features detection (figure 14). If all the values meet those limit inequalities, then it is agreed to remove off the respected list (features) as it means it is contained..



$x_1 \leq a_1 \leq x_2$

$x_1 \leq a_2 \leq x_2$

$y_1 \leq b_1 \leq y_2$

$y_1 \leq b_2 \leq y_2$

Figure 14: *Inequality Equations*

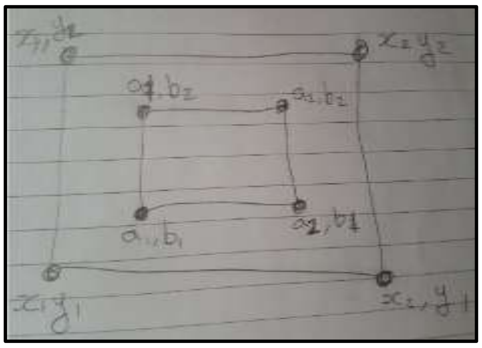


Figure 13: *Sketch of pixel co-ordinate interpretation*

8.7 Pseudocode: Robust Clustering

The literature review explained how robust clustering was useful for achieving agreement between data while partitioning data. After the detection algorithms have been implemented and tested, further analysis takes place to remove redundant detections and a final cluster process takes place on the features list. The amount of clusters generated through this algorithm will be added to the amount of face detections, and this will be the total number of people detected in the image.

This clustering algorithm was analysed through the case studies and a pseudocode was created:

```
 1) Create C = n length vector containing -1 the value of C[x] is the RC that x
is in. I.e. assume all variables not in RCs at the start.
 2) Assume a list exists (R) m by 2. We have m 1's from the matrix, first column
is the first variable, second is the second variable
 3) For x = 1 to m
 4)        a = R[x][1]
 5)        b = R[x][2]
 6)        ca = C[a]
 7)        cb = C[b]
 8)        nc = max(C) +1 [new cluster ID]
 9)        if ca =-1 and cb = -1 then
10)           C[a] = nc
11)           C[b] = nc (put them both in new cluster)
12)     end if
13)     if ca = -1 and cb <> -1 then
14)         C[a] = cb (b already in cluster, a not, thus put a in cluster of b)
15)     end if
16)     if ca <> -1 and cb = -1 then
17)         C[b] = ca (a already in cluster, b not, thus put b in cluster of a)
18)     end if
19)     if ca <> -1 and cb <> -1 then
21)         Relabel everything in C with ca to cb, i.e. replace ca for cb - merge
clusters
               Variables already in RCs, thus merge the two sets of RCs
22)     end if
23) End for
```

The formations of new clusters occur when the elements inside are related. This is a complex algorithm that will need intensive testing to ensure it works, before deploying it to the main prototype.

9. Implementation

9.1 Implementation - Image Processing

From the image processing research during the literature review, the results of the studies that used the segmentation methods achieved success in capturing information they hypothesized beforehand. Individual scripts were created according to the figure 9 in the design section, the Otsu filter, Hough transform, Watershed segmentation. All segmentation techniques were implemented and reviewed against these assumptions that were derived from the abstract design of image processing.

1. The Otsu segmented image shows the 2 different shades (foreground and background)
2. The Hough transformed image reveals the edges of the objects in the image
3. Watershed image produced contrasts of objects (specifically people) according to spatial differences

Firstly the Otsu segmentation was implemented. A script was made to read in the test image, then called the function script which is the initial Otsu filter (open-source) (BiomeCardio, n.d) which took two parameters; the image and the level of segmentation which was kept at 2 by default. The processed image was then shown on screen and saved in to an output folder that saved the called ‘otsuimg.jpg’.

The Hough transformation script was developed in a similar way. The test image was read then was set to grayscale. This made the image able to detect the edges using the ‘canny’ objector. The process of the Hough transform can be viewed on the console/ command window by printing the ‘Hough’, Theta’, and ‘Rho’ values. These represent the pixels that are extracted as edges including the angle, pixel position features (Eddins, 2006). The output image is shown and saved in to an output folder as ‘houghimg.jpg’.

Watershed algorithm also requires prior settings before proceeding through the algorithm, such as greyscale. MATLAB’s resources explained the segmentation of a mathematical figure which provided solutions when the subject is an image instead of a mathematical figure (MathWorks, n.d). First the test image is read in and Sobel filter is used to extract the edges. Leading to the ‘strel’ function which mark the foreground pixels (processed individually) and then re-constructed. This re-construction is the output image shown on screen to review. Also saved into an output directory as ‘watershedIobr.jpg’.

An example of the result images are shown in figures 15-17.

The output folders were used to process the images a combination of segmentations. An example of this process is; that an image would be input, then processed first by the Otsu segmentation, then the output image of the Otsu method will be input to the Hough transformation algorithm, then this output image will be input to Watershed to produce a final output image.

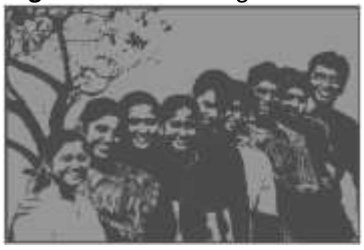


Figure 16: *Otsu Segmentation*

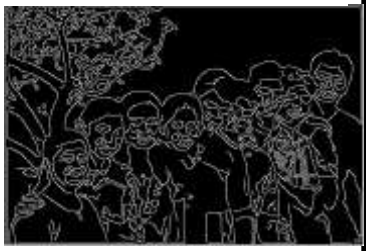


Figure 15: *Hough Transform*

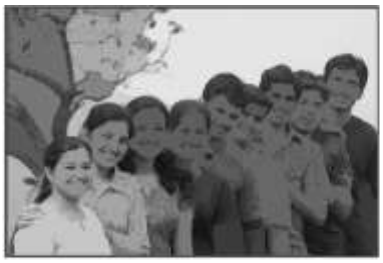


Figure 17: *Watershed Segmentation*

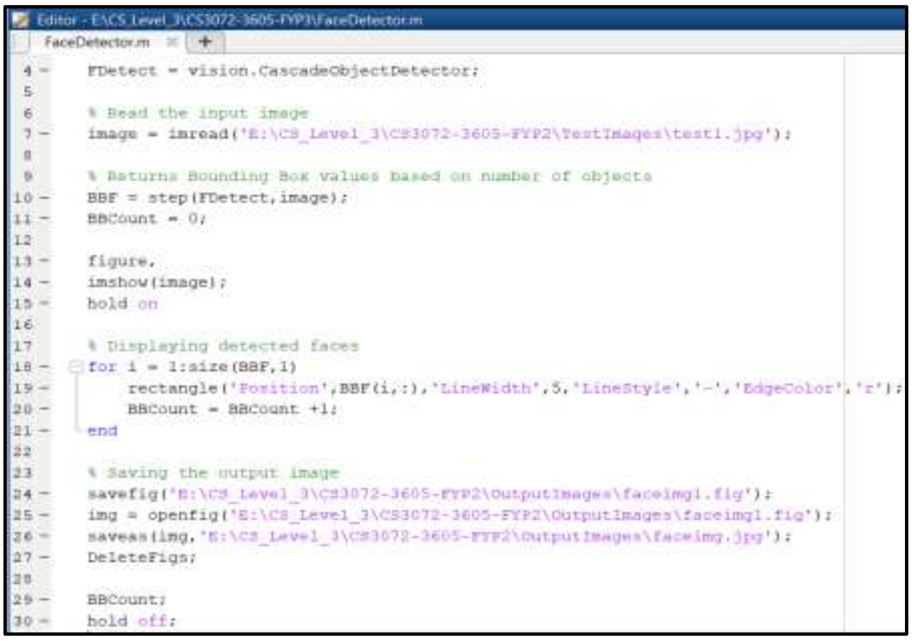
Even though the resulting images came up with remarkable visuals, it is clear to say that the people in this image are indistinguishable by a single segmentation method. The Otsu image captured regions of the image that are unfathomable as the shapes of these regions are improper (both the lighter and darker areas). The Hough transformation brought out intensifying edges including people's facial features, even tree branches. However the repetitive edge detections of the face, or body, all were different shapes and sizes which made it extremely difficult to analyse further. The Watershed segmented image showed different contrasts, especially the people. The person closer to the camera is coated the lightest grey colour while the people behind slowly fade to a darker shade. This may be the closest segmentation that allows a difference in objects, though its still not good enough as the colour of shades can be identified with others with a fine line in between, which is again, extremely hard and perhaps time consuming to establish.

Due to these findings, it is possible that an image processed with a combination of segmentation will also not be useful in revealing information that can be used to extract the amount of people in it.

9.2 Implementation - Face and Features Detection

The face recognition research during the literature review revealed that detection algorithms are very successful in detecting faces, specifically because its derived from neural networks and machine learning concepts. As image processing implementation did not work out well enough for further analysis and implementation to extract the number of people, it has been subject to being throw-away prototype. New scripts will be made for a new form of prototype.

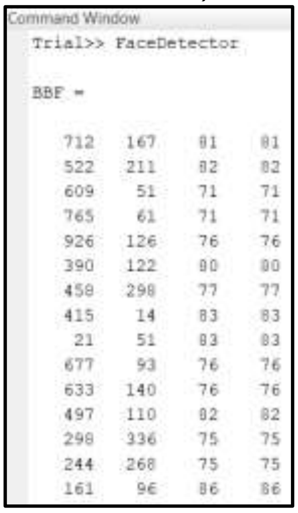
The use of the Viola-Jones open-source algorithm was implemented according from the pseudocode from the design stage.



```
FDetect = vision.CascadeObjectDetector;

% Read the input image
image = imread('E:\CS_Level_3\CS3072-3605-FYP2\TestImages\test1.jpg');

% Returns Bounding Box values based on number of objects
BBF = step(FDetect,image);
BBCount = 0;

figure,
imshow(image);
hold on

% Displaying detected faces
for i = 1:size(BBF,1)
    rectangle('Position',BBF(i,:),'LineWidth',5,'LineStyle','-','EdgeColor','r');
    BBCount = BBCount +1;
end

% Saving the output image
savefig('E:\CS_Level_3\CS3072-3605-FYP2\OutputImages\faceimg1.fig');
img = openfig('E:\CS_Level_3\CS3072-3605-FYP2\OutputImages\faceimg1.fig');
saveas(img,'E:\CS_Level_3\CS3072-3605-FYP2\OutputImages\faceimg.jpg');
DeleteFigs;

BBCount;
hold off;
```

Figure 18: *Face Detection code snippet*

Figure 19: *List of pixel co-ordinates (Face Detections)*



```
Trial>> FaceDetector

BBF =

   712   167    81    81
   522   211    82    82
   609    51    71    71
   765    61    71    71
   926   126    76    76
   390   122    80    80
   458   298    77    77
   415    14    83    83
    21    51    83    83
   677    93    76    76
   633   140    76    76
   497   110    82    82
   298   336    75    75
   244   268    75    75
   161    96    86    86
```

The code in figure 18 is a working program that, when executed will produce an output of an image with red boxes around the detected faces. The process of this algorithm goes as follows, the image is input, the vision objector is called which detects faces by default unless instructed otherwise. The For loop runs to display every detected face (red box) iteratively through the 'BBF' list which is a list of the face detected pixel co-ordinates and shape dimensions.

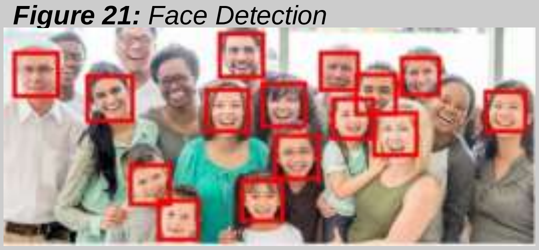
Figure 19 is the list the 'BBF' list. The first two columns represent the bottom left pixel co-ordinates (x, y) of a rectangle which represents a face. The last two columns represent the width and height of the rectangle.

The features detection was implemented in the same way. All detection implementations were developed in individual scripts, meaning 4 scripts were created in total. The pixel co-ordinates and shape dimensions will also be produced of each feature detection, though a single list will be represented for all features (eyes, mouth and nose lists) combined. The vision objector that was used for face detection was altered to detect specific features. This objector pointed to certain attributes of the classifier that enabled the detection of what needed to be detected. Below are the parameters taken by the objector to allow successful detection.

Mouth: 'Mouth, MergeThreshold, 16'
Nose: 'Nose, MergeThreshold, 16'
Eyes: 'EyePairBig'

Figure 20 is the face detection output and figure 21 the output image displaying face and features detections. This is the same image that was used to detect faces on the code previous to this. Note the number of squares found in figure 20 represents the same number of rows in the face detection list (15), meaning 15 detections were found.

Figure 21: *Face Detection*



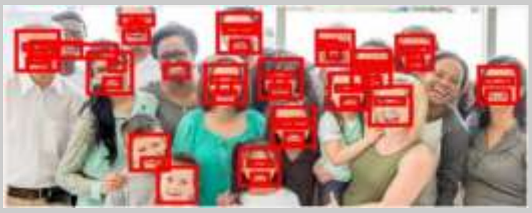
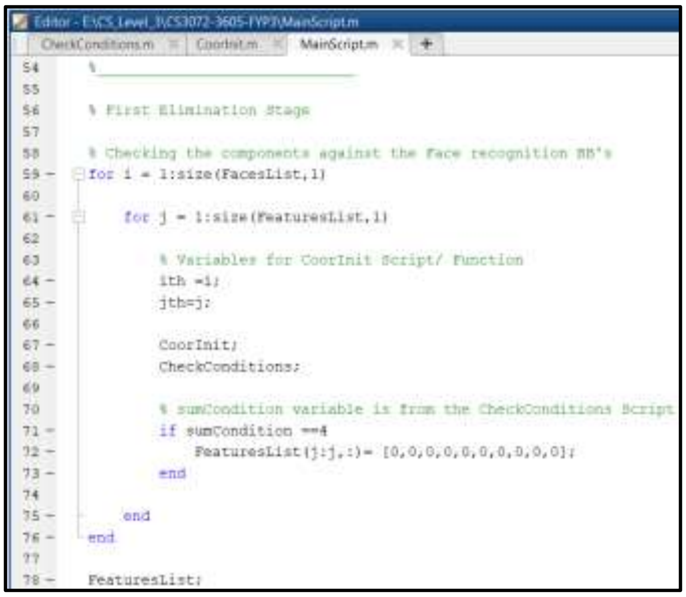


Figure 21: *Face and Feature Detection*

Analysing these two output images, there are 4 people left undetected. From figure 20, 3 out of the 4 people have been detected with features detection (figure 21). This is evidence that features along with face detection is beneficial in detecting people to a higher percentage.

9.3 Implementation - Detections Contained in a Face Detection

The previous implementation stage showed the benefit of using features detection in addition to face detection. However, this makes the output image cluttered and gives a misleading number of detections combined (face & features detection) with the number of detections in the pixel co-ordinates list. The design stage explained that a feature detection contained in a face detection may be removed since it represents the same face, hence only the useful that features that do not associate with a detection face remain as they may represent a face that is undetected. This implementation is built-on the previous prototype as part of an evolutionary stage.



```
%

% First Elimination Stage

% Checking the components against the Face recognition BB's
for i = 1:size(FacesList,1)

    for j = 1:size(FeaturesList,1)

        % Variables for CoorInit Script/ Function
        ith =i;
        jth=j;

        CoorInit;
        CheckConditions;

        % sumCondition variable is from the CheckConditions Script
        if sumCondition ==4
            FeaturesList(j:j,:)= [0,0,0,0,0,0,0,0,0,0];
        end

    end
end

FeaturesList;
```

Figure 22: *Code snippet of removal of feature detection contained in a face detection*

The code in figure 22 runs this process. The nested loop runs to compare each feature detection with all face detections at a time. All detection scripts were run before-hand to produce the lists of pixel co-ordinates and dimesions. 'FacesList' and 'FeaturesList'. Within the loop, two functions are called; 'CoorInit' and 'CheckConditions'. The 'CoorInit' script initialises the pixel co-ordinate values of each row (detection) in the both detections list as 'i' or 'j'. As the detection list stores the width and height, calculations were made so that the pixel co-ordinates of each corner of the detected square is initialised – similar to figure 14 (x1, x2, y1, y2).

The 'CheckConditions' function followed the inequality equations described in the designs (figure 13) to check if the feature detection pixel co-ordinates are within range. Finally the If statement denotes that if all four inequality equations are met, then it means that feature detection is contained within face detection and is then removed.

This was successful in eliminating many feature detections that were redundant, figure 23 compared to figure 21 is a better indication of detections and the amount of people in the image.

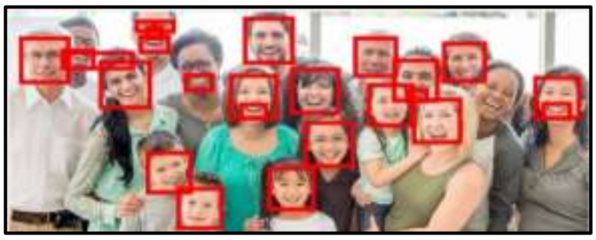


Figure 23: *Output image of reduced feature*

There are few things to notice from the people that have not been detected. The reasons may be that their skin colour have not been learned well as a classifier – 3 of the 4 people with darker skin have not been detected by facial detection. Also, 3 out of 4 people who have not been detected wear glasses, meaning these people have their glasses in the way of detecting their eyes.

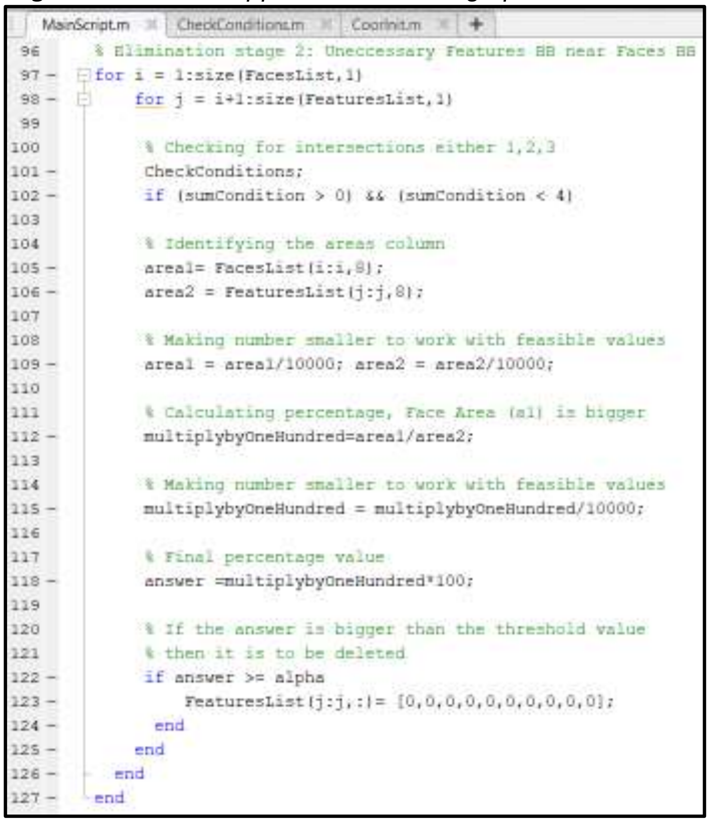
As described during the literature review of face detections, face detection works by making sure relevant feature detections are made in the correct positions. For example, a single pair of eyes should be at the top of the face, a mouth should be detected towards the bottom of the face, and a nose must in between or in the middle of a face. If there is a flaw in one of the detections, for example the mouth is detected at the top of a face for some reason, then the face detection is not triggered.

Even though the result image gave better results compared to an image full of cluttered detections, there are still few feature detections that have not been removed that represent the same face. Further analysis was needed to remove these detections to refine the detections list.

9.3 Implementation - Intersections between Face and Feature Detections

The previous implementation was done to remove feature detections were in containment with a face detection with the use of inequality equations. However there were no conditions for inequality equations that only met 1, 2 or 3 conditions. This stage of the elimination, thresholding will be used to establish a fine line between the decision of whether a feature detection is removed or remained. This implementation is built-on the previous prototype as part of an evolutionary stage. Figure 24 shows the code of how this was implemented.

Figure 24: *Code snippet of thresholding operation*



```
% Elimination stage 2: Uneccessary Features BB near Faces BB
for i = 1:size(FacesList,1)
    for j = i+1:size(FeaturesList,1)

     % Checking for intersections either 1,2,3
     CheckConditions;
     if (sumCondition > 0) && (sumCondition < 4)

     % Identifying the areas column
     area1= FacesList(i:i,8);
     area2 = FeaturesList(j:j,8);

     % Making number smaller to work with feasible values
     area1 = area1/10000; area2 = area2/10000;

     % Calculating percentage, Face Area (a1) is bigger
     multiplybyOneHundred=area1/area2;

     % Making number smaller to work with feasible values
     multiplybyOneHundred = multiplybyOneHundred/10000;

     % Final percentage value
     answer =multiplybyOneHundred*100;

     % If the answer is bigger than the threshold value
     % then it is to be deleted
     if answer >= alpha
         FeaturesList(j:j,:)= [0,0,0,0,0,0,0,0,0,0];
      end
    end
  end
end
```

The nested loop ensures that each feature is compared with the whole list of face detections at a time. The algorithm first checks if the conditions (inequality equations) are between 0 and 4 meaning only some of the inequality equations have been met. This will mean that there are intersections between a feature detection and a face detection and not full containment.

For example figure 25 looks as if the mouth (feature) detection is found within the face detection. However it is most likely to be that the feature detection exceeds the limit of the bottom line, and appears as if only 2 out of 4 pixel co-ordinates were contained within the face detection (since both left and right bottom corners counts as the bottom line - y1, y2,).

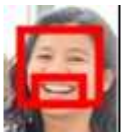


Figure 25*: Feature vs Face Detection*

Afterwards, the detection rectangle shape areas were calculated. The area of each detection rectangle was compared (feature rectangle vs face rectangle). As the pixel areas were too big and were not able to be processed further, smaller representations of them were generated. This lead to generating an overall percentage of how intersected the rectangles are. The threshold called ‘alpha’ which was equalled to a number of 0.052, was used as a decision point whether the feature was removed from the list or kept. If the percentage is higher than his threshold then the feature detection is removed from the list.

Several tests were done to come up with this threshold value by trial and error. The review of output images were what made the value reach to what it is now.

Figure 26 displays the detection of only the features list (for better visualisation and undistracted reviewing). Comparing with previous output images, the features that remained are the crucial detections needed to detect that person, while the feature detections that represented the same face have been completely removed.

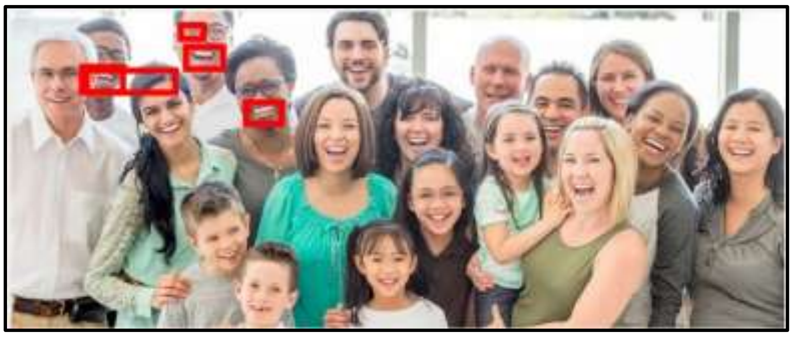
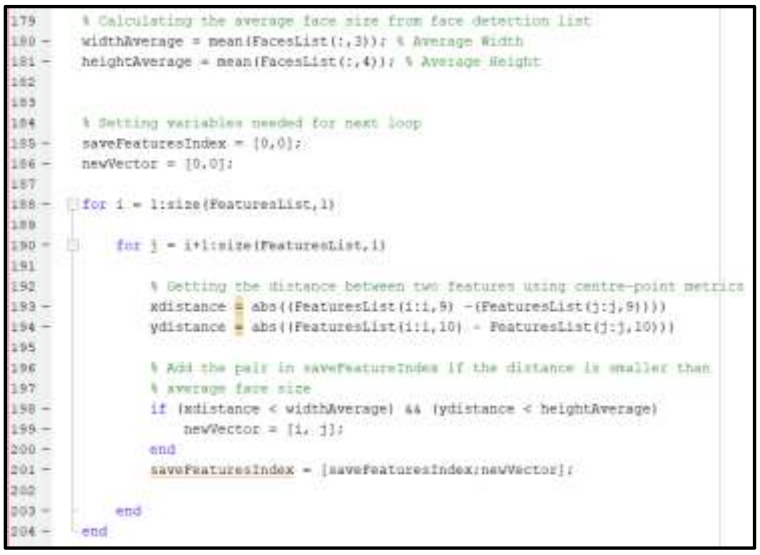


Figure 26: *Output image of remaining feature*

9.4 Implementation - Feature Detections Representing Same Face

Now that the face detection and features list have been refined and majority of the detections represent the same face. It is noticed that there are multiple features that represents that face from the previous output images. Therefore to ensure this last redundancy occurrence does not represent two faces, the features need to be processed against each other to identify similarities. This implementation is built-on the previous prototype as part of an evolutionary stage.

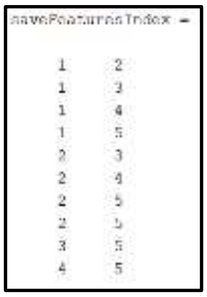
Before the clustering occurs, the pairs of the features that may represent the same face need to be listed. To identify the features that represent the same face, the centre-points generated from the rectangle dimensions were used compare the distance between feature detections.



```
% Calculating the average face size from face detection list
widthAverage = mean(FacesList(:,3)); % Average Width
heightAverage = mean(FacesList(:,4)); % Average Height


% Setting variables needed for next loop
saveFeaturesIndex = [0,0];
newVector = [0,0];

for i = 1:size(FeaturesList,1)

    for j = i+1:size(FeaturesList,1)

        % Getting the distance between two features using centre-point metrics
        xdistance = abs((FeaturesList(i:i,9) -(FeaturesList(j:j,9))))
        ydistance = abs((FeaturesList(i:i,10) - FeaturesList(j:j,10)))

        % Add the pair in saveFeatureIndex if the distance is smaller than
        % average face size
        if (xdistance < widthAverage) && (ydistance < heightAverage)
            newVector = [i, j];
        end
        saveFeaturesIndex = [saveFeaturesIndex;newVector];

    end
end
```

Figure 27*: Code to find extremely close feature detections*

The code in figure 27 goes as follows, the average width and height of the face rectangles are used to generate average values. Within in the for loop, each feature was compared with the rest of the features at a time. The If statement checks if the distance between the two features is smaller than the average face width and height. If the conditions are met, then both feature detection index's are added as a pair to a new list called 'saveFeaturesIndex'. This means that there are two feature detections that are so close to each other (distance smaller than the average size of a face), that it may represent the same face. Figure 28 shows the output of this operation, the feature detection pairs that were stored as a result of potentially representing the same face. Now the robust clustering algorithm takes place.



```
saveFeaturesIndex =

     1     2
     1     3
     1     4
     1     5
     2     3
     2     4
     2     5
     2     5
     3     5
     4     5
```

Figure 28: *‘saveFeaturesIndex’ Table*

9.5 Implementation - Robust Clustering

As the literature review mentioned that this algorithm is very good at achieving high agreement rates, this method was implemented on the previous prototype as part of an evolutionary stage.

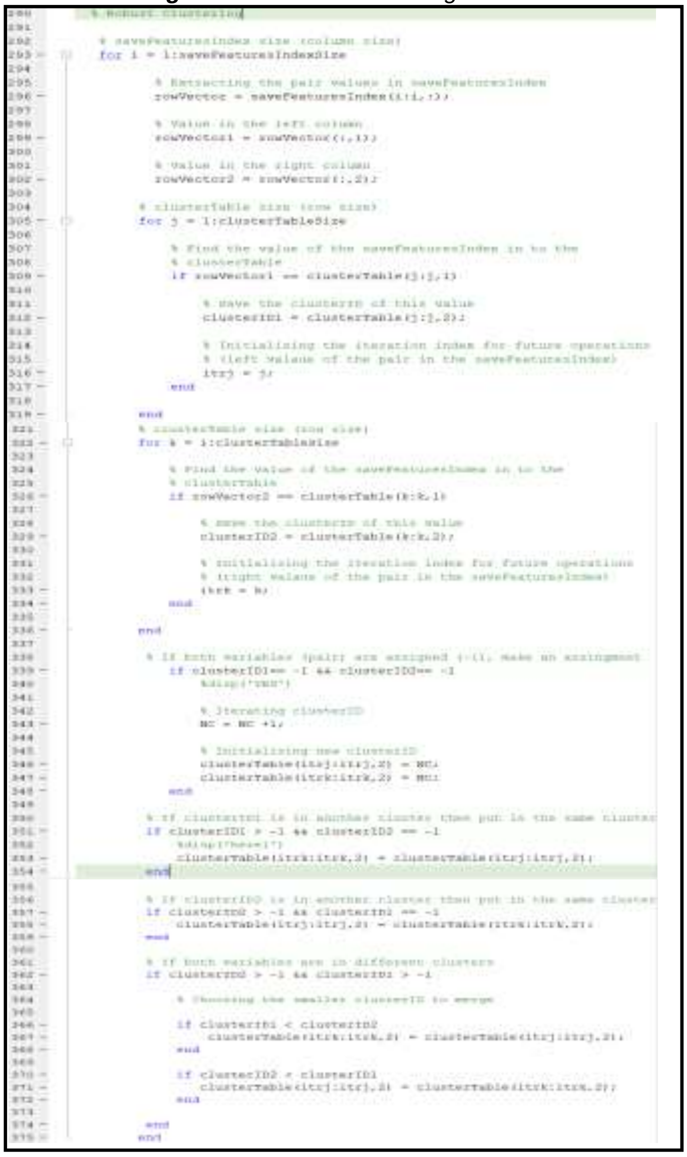
The previous implementation helped create the input for robust clustering. The design stage also includes a pseudocode of this algorithm that was implemented shown in figure 29.

The code runs as follows, the for loop iterates through each row in the saveFeaturesIndex list. Another table was created which listed the variables (index values and their assigned state). The unassigned state at the start was -1, shown in figure 30.

The cluster assignment was searched according to both variables (the pair) in the saveFeaturesIndex. (First two for loops)

The if statement after the two for loops, checks if both variables of the pair has the cluster ID of -1, in which case will change the cluster assignment to an iteration value of 0 using the ‘NC’ variable. The NC iterates for every new cluster ID that needed – unassigned variables (-1)

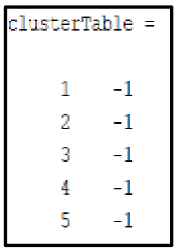
Figure 29: *Robust Clustering Code*



```
% Robust Clustering

 % saveFeaturesIndex size (column size)
 for i = 1:saveFeaturesIndexSize

         % Extracting the pair values in saveFeaturesIndex
         rowVector = saveFeaturesIndex(i:i,:);

         % Value in the left column
         rowVector1 = rowVector(:,1);

         % Value in the right column
         rowVector2 = rowVector(:,2);

     % clusterTable size (row size)
     for j = 1:clusterTableSize

         % Find the value of the saveFeaturesIndex in to the
         % clusterTable
         if rowVector1 == clusterTable(j:j,1)

             % Save the clusterID of this value
             clusterID1 = clusterTable(j:j,2);

             % Initialising the iteration index for future operations
             % (left values of the pair in the saveFeaturesIndex)
             itrj = j;
         end

     end
     % clusterTable size (row size)
     for k = 1:clusterTableSize

         % Find the value of the saveFeaturesIndex in to the
         % clusterTable
         if rowVector2 == clusterTable(k:k,1)

             % Save the clusterID of this value
             clusterID2 = clusterTable(k:k,2);

             % Initialising the iteration index for future operations
             % (right values of the pair in the saveFeaturesIndex)
             itrk = k;
         end

     end

      % If both variables (pair) are assigned (-1), make an assignment
         if clusterID1== -1 && clusterID2== -1
             %disp('TRU')

             % Iterating clusterID
             NC = NC +1;

             % Initialising new clusterID
             clusterTable(itrj:itrj,2) = NC;
             clusterTable(itrk:itrk,2) = NC;
         end

      % If clusterID1 is in another cluster then put in the same cluster
      if clusterID1 > -1 && clusterID2 == -1
          %disp('here1')
          clusterTable(itrk:itrk,2) = clusterTable(itrj:itrj,2);
      end

      % If clusterID2 is in another cluster then put in the same cluster
      if clusterID2 > -1 && clusterID1 == -1
          clusterTable(itrj:itrj,2) = clusterTable(itrk:itrk,2);
      end

      % If both variables are in different clusters
      if clusterID2 > -1 && clusterID1 > -1

          % Choosing the smallest clusterID to merge

          if clusterID1 < clusterID2
             clusterTable(itrk:itrk,2) = clusterTable(itrj:itrj,2);
          end

          if clusterID2 < clusterID1
             clusterTable(itrj:itrj,2) = clusterTable(itrk:itrk,2);
          end

      end
     end
```

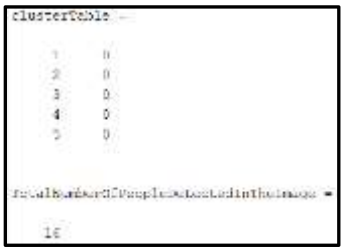
The second and third If statements check if either of the values of the pair is indifferent to a cluster ID of -1. If this is the case then the variable that is unassigned (-1) will be set to the cluster ID that it's pair shares.

The final if statement checks if both values of the pair belong to two different clusters. In which case the smaller cluster ID will be used to assign the other variable which holds the larger cluster ID, i.e. a cluster assignment of 1 will be more useful than a cluster assignment of 2 or 3 or higher.



```
clusterTable =

     1    -1
     2    -1
     3    -1
     4    -1
     5    -1
```

Figure 30: *Cluster Assignment Table*



```
clusterTable =

     1    0
     2    0
     3    0
     4    0
     5    0

totalNumberOfPeopleDetectedInTheImage =

    16
```

Figure 31: *Final Output (Number of People in the Image)*

Figure 31 shows that all variables belong to a cluster ID of 0, which means only 1 cluster was formed. This number of clusters (which is 1 since all variables belong to cluster ID of 0) is added to the face detection become the justifying (detected) number of people in the image.

10. Software Testing

10.1 Software Testing - Image Processing

As each segmentation was created using individual scripts, unit-testing and white-box testing was widely used to make sure each script was running as it should be and producing relevant outputs.

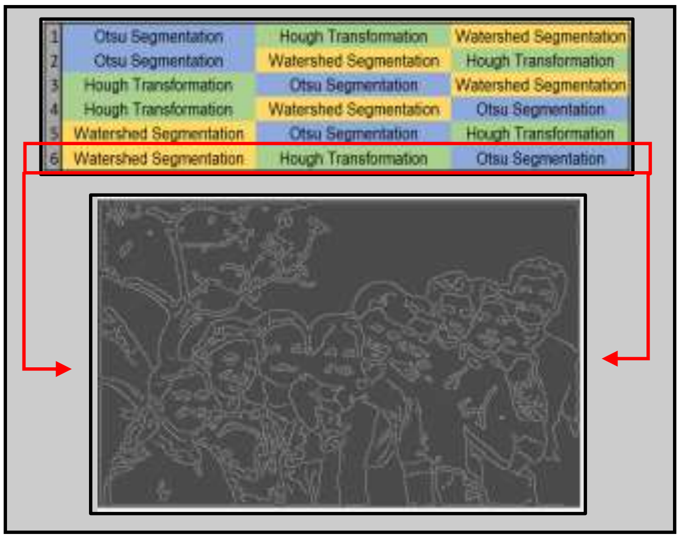
Otsu segmentation was the first script that was implemented and tested. As the original Otsu method was called as a function had a maximum of 10 lines of code, therefore did not involve many testing phases. The input instructions were provided in the function script and was followed accordingly, which was the image and the threshold level of 2. The goal of this script was to produce a segmented image that matches the designs derived from the literature that was reviewed about what the Otsu method should produce. The image produced the correct results after reviewing the output images.

The Hough transformation was tested similarly. This script was also a small method which included the input of the test image. The conversion from RGB to grey-scale, then the 'canny' detector reader. The two parts of the code were tested individually by displaying the output image and reviewing it to ensure it was grey-scaled and that the edge detector was working. The research on MATLAB demonstrations for displaying the Hough transformed image encouraged this script to thrive towards an outputted segmented image. The output image was compared the designs and the MATLAB demonstrations to ensure this script was running and processing the image effectively.

The Watershed segmentation involved a series of black-box testing. This was because it involved many morphological transformations, as well as the demonstration on this method specifically using a mathematical model instead of an inputted image. The first was the conversion of the colour image (RGB) was converted to grayscale. This was tested by displaying the image to make sure it was grey-scaled. Then a few operations were made that was similar to the demonstration on the MATLAB website of the Watershed segmentation. These lines of code were used to portray the same flow of the of the data that was used to segment the mathematical model. As the image was inputted, the variable that represented the mathematical model was replaced with the image variable. White-box testing was then used to ensure the output image matched the designs and output images of the mathematical model from the Watershed method demonstration.

As the individual segmentation methods did not produce output image for further analysis. Though a combination of segmentation was to deliver better results, according to studies in the literature reviews. Just as the study about counting the eggs in a packet used layered combinations of Otsu and Hough to accurately count how many eggs there are (Baygin et al, n.d), the Otsu, Hough and Watershed segmentation methods were run one after the other on the test images to see if they revealed information that can be used to count the amount of people there are in an image.

The test images were processed one after the other using the images stored in the output folder. All different orders of the techniques were run on each test image. Unit-testing was used to take the image processing step by step. Meaning each of the combinations in figure 32 were, for example tested first with watershed, then the processed image is then processed using the Hough method and the same for the Otsu segmentation. Some of the errors included the image not grey-scaled, in which case a new line of code needed to be added.



1	Otsu Segmentation	Hough Transformation	Watershed Segmentation
2	Otsu Segmentation	Watershed Segmentation	Hough Transformation
3	Hough Transformation	Otsu Segmentation	Watershed Segmentation
4	Hough Transformation	Watershed Segmentation	Otsu Segmentation
5	Watershed Segmentation	Otsu Segmentation	Hough Transformation
6	Watershed Segmentation	Hough Transformation	Otsu Segmentation

Figure 32: *Combination of segmentations and output image*

Unfortunately the results were not useful to move forward to the next stages that were intended according to the designs. Neither of the 6 resulting images produced recognisable objects to identify as people after complementary segmentation. The output image that came close to further analysis is shown in figure 32 This is image shows a lesser intense edge detection from the Hough transformation. This means that the watershed eliminated some of the tiny detections by morphing in to the common regional areas. The Otsu segmentation afterwards didn’t add any value apart from changing the brightness of the image.

10.2 Software Testing - Face and Features Detection

The testing of the Face and Features implementation began with the white-box testing strategy. To ensure the expected output image was delivered, the process was done by inputting an image and running the script to review the displayed output image. This process occurred several times with different test data (images).

After the output images were delivering successful detections, the black-box testing strategy was used to ensure the detection lists were tested.

The detection list is widely used during further development of the prototype. However the purpose of this implementation stage, it was a secondary expected output in contrast to the output display of detected faces. Therefore considered as a separate branch of this script.

The for loop that iteratively draws the rectangles, uses the detection list to draw rectangles denoting the detected faces. If the faces were correctly detected, it means that the objector is producing the correct pixel co-ordinates and dimensions. The detection list was displayed analyse and review what the values meant.

Unit-testing strategy was used to test the feature detections. As explained during the implementation, the objector needs parameters to detect particular objects on an image otherwise it detects a face (by default). The unit-testing strategy encouraged to input these parameters one by one to review the displayed output images to ensure the objector is detecting what is intended. In addition, the detection lists were displayed at the same time, and the amount of row in the list were compared to the amount of detections to ensure all detections consistent.

10.3 Software Testing - Detections Contained in a Face Detection

The development of the prototype occurred when the detections were analysed and manipulated according to visual reviews. The goal of this script was to reduce the amount of feature detections by removing the detections that were contained within a face detection- which meant they represent the same face. White-box testing strategy was used to ensure this goal was met. However unit and black-box testing were complementarily used when developing the code.

The structure of the code involved the comparing of each feature against the whole list of face detections. It was important to get this process right, therefore the ‘i’ and ‘j’ values were printed to ensure the correct iterative process took place.

Afterwards the ‘CoorInit’ was created. It is used to initialise the corner values of rectangle using the detection list that stored pixel co-ordinates and dimensions. The assignments were displayed in the command window and compared against the original pixel co-ordinated to check if they matched.

The ‘CheckConditions’ method compares the dimensions of the feature with the face detections. This was tested by displaying every dimension of both detections with the decision of whether to remove or not. This also tested the if statement so ensure correct deletions were being made to the features list, only ones that were contained within a face detection.

After all tests were complete, the goal of this part of the code was tested. The refined features detection list was displayed with the face detection and reviewed visually. Majority of the feature detections, especially ones that were contained were removed and this stage of the software testing was complete and ready take the prototype forward.

10.4 Software Testing - Intersections between Face and Feature Detections

The for loop in during this part of the prototype is the same order as it was in the previous part, therefore testing was not needed for this part. The same went for 'CheckConditions' method. A big part of this implementation included the feasible representation of the percentage.

As original percentage values were used and compared against the alpha initially, the command window output 'infinite number' which meant that it was not possible to process a very large number. The area of pixel co-ordinates represent very precise values which then produced large area sums. Due to this, black-box testing was used to debug this issue and appropriate action was taken afterward. Smaller representations were generated by dividing all values by 10,000. Unit-testing was used to ensure the percentages were then being calculated by displaying them on the command window.

After this, white-box testing aided in testing the overall goal of this part of the code. Which was to only keep the features that had a higher percentage meaning the faces that had a very small intersection and were deleted. The refined features list were displayed and reviewed according to this goal.

10.5 Software Testing – Feature Detections Representing Same Face

The goal for this part of the prototype was to initially develop the input for robust clustering. However this aspect is similar to previous implementation in the sense that it is separating data, specifically, the features. Robust clustering requires pairs of variables that are associated to each other, to then cluster. To find a similarity between two features, the distance between two features and an average face sizes were calculated from the face detection list to keep close detected featured and store them (as pairs) into a new list

MATLAB's built-in functions were used for majority of this implementation. Though unit-testing was used to ensure the correct column and data was being processed. For example, the averages of the face detection (width and height) dimensions were used. Therefore the detection list was displayed to view the column index and ensuring it was the correct one being processed in the code. The same occurred with the feature variables when calculating the distances between the x and y values. The absolute value was displayed on the command window to ensure there were no negative values.

Black-box testing was used to test the if statement. Whether the average face size compared to the distance between features were smaller and therefore added to the new list. These values were displayed iteratively and reviewing to ensure the process ran correctly.

White-box testing enabled the goal to be tested and reviewed. The output 'saveFeaturesIndex' was output onto the command window to view the indexes and ensure a suitable structure was formed for the robust clustering algorithm use as input.

10.6 Software Testing – Robust Clustering

Robust clustering was part of the most intensive software testing. This was because there were many loops and conditions that were taking place within a single loop. The data structures were also carefully selected and tested before moving forward. The overall goal of this final part of the code was to generate the amount of clusters and add it to the amount of face detections to finalise the total number of detection people in the image.

Unit-testing was strictly followed when initialising columns or particular indexes of data. The rows and columns were displayed on the command window to ensure the correct information is what was being extracted for a particular purpose. For example, the code below is an extraction of the columns in the saveFeaturesIndex. The left and right columns were displayed in the command window and made sure the correct rows were being passed.

```
% Value in the left column
rowVector1 = rowVector(:,1);

% Value in the right column
rowVector2 = rowVector(:,2);
```

Black-box and unit-testing was used to test the conditions that were responsible for assigning the cluster ID's. Majority of the time, the conditions were not being triggered and testing due to almost all pairs belonging to the same cluster. Since it is not necessary that all conditions will be executed, different test-data/ inputs were used to test these conditions. Firstly the condition that will always be triggered is the condition that checks if both variables are un-assigned, therefore this was tested by displaying the 'clusterTable' to the command window to view the results. The expected output that all pairs had a value of 0 since the 'NC' new cluster variable is the cluster ID that is iterated and assigned.

The second and third conditions (if only 1 variable was un-assigned) were tested with newly fromed test-data. The test-data was developed from a manual run of the algorithm, trial and error pairs of numbers were used check if these conditions would be triggered. The expected results were displayed on the command window and reviewed to ensure the same cluster ID was shared to the un-assigned variable. After these conditions were producing expected results. The final condition was to net a situation where both variables belonged to two different clusters. In which case the smaller cluster ID is assigned to the variable with a larger cluster ID. Just like the test-data that was developed during manual runs of this algorithm, the same occurred to ensure this condition was triggered.

The cluster table was displayed in an iterative manner to review the transformation of the clusters as a final review of all conditions. Finally, white-box testing was used to ensure to the correct amount of clusters were taken in account and added to the total amount of face detections.

11. Exploratory Data Analysis

The exploratory data analysis stage occurred through-out and in duration of the implementation. With the literature review providing insightful ideas, these ideas were considered to be replicated to produce some of the results the case studies produced. For example applying different segmentation one after the other on an image so each segmentation will reveal information for its own purpose. Image processing was not useful to take the solution further, so it became a throw-away prototype. Majority of the data analysis was done when implementing the detection algorithms.

11.1 Face vs Feature Detections

After the feature and face detections were implemented. It was clear which detections were feature-based and which were face detections. However further analysis was done to review the accuracy of the feature detections (as the detection was combined (eyes, nose and mouth).

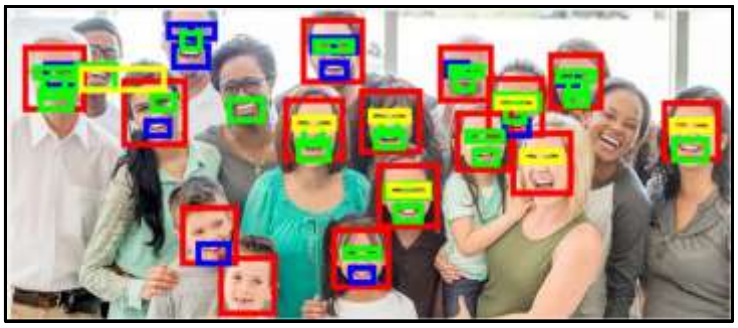


Figure 33: *Colour Visualisation of detections*

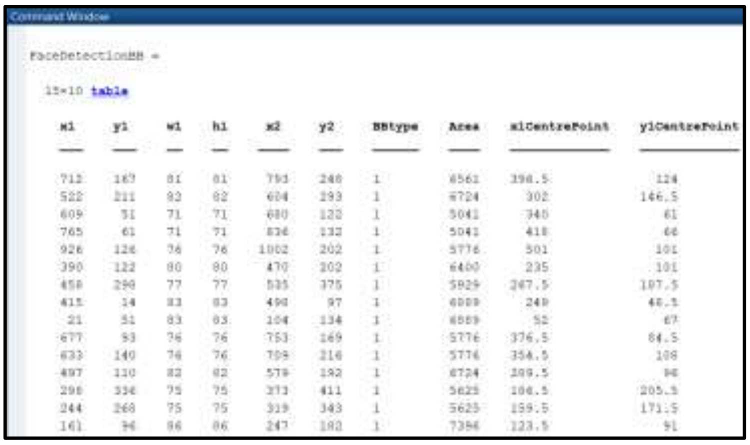
Figure 33 shows the same image used for implementation and testing, it shows the 4 different coloured rectangles that represent a different detection. The red is a face detection, the blue is the eyes detection, green is the mouths detection and yellow is the nose detection. After reviewing the detections, it's clear that the feature detections are not as accurate as the face detection. Another noticeable aspect of this visualisation is that the undetected faces have feature detections negatively implicates whether it is a face or not. For example a face that detected the eyes and a mouth as 3 eye detections (blue) is obviously not a face in reality therefore means that it does not qualify for a face detection. From this point, it was clear that the feature detections (lists) were needed to be altered to aid the face detections in finalising the amount of people detected in an image.

11.2 Expanding Data Metrics

After the contained feature detections within face detections were removed, there was a need to expand the range of data to manipulate to a new break-through. The pixel co-ordinates and dimensions were not enough alone, but they were useful in creating new metrics for each detection.

Figure 34 shows the shows the new metrics derived from the pixel co-ordinated and dimensions. The first two columns are the original metrics derived from the objector (pixel co-ordinates and dimensions). The 5th and 6th columns are the x2 and y2 columns that were used in the third implementation (containment) representing the top right corner pixel co-ordinates (shown in designs, figure 14). These values were calculated by adding the x1 with the width for x2 and y1 with the height for y2.

Figure 34: *Detection Metrics (Face Detection List)*



```
FaceDetectionBB =

  15×10 table

    x1     y1     w1    h1     x2      y2     BBtype    Area    x1CentrePoint    y1CentrePoint
    ___    ___    __    __    ____    ___    ______    ____    _____________    _____________

    712    167    81    81     793    248      1       6561        396.5             124
    522    211    82    82     604    293      1       6724          302           146.5
    609     51    71    71     680    122      1       5041          340              61
    765     61    71    71     836    132      1       5041          418              66
    926    126    76    76    1002    202      1       5776          501             101
    390    122    80    80     470    202      1       6400          235             101
    458    298    77    77     535    375      1       5929        267.5           187.5
    415     14    83    83     498     97      1       6889          249            48.5
     21     51    83    83     104    134      1       6889           52              67
    677     93    76    76     753    169      1       5776        376.5            84.5
    633    140    76    76     709    216      1       5776        354.5             108
    497    110    82    82     579    192      1       6724        289.5              96
    298    336    75    75     373    411      1       5625        186.5           205.5
    244    268    75    75     319    343      1       5625        159.5           171.5
    161     96    86    86     247    182      1       7396        123.5              91
```

The 'BBtype' is an indication of the type of detection it is. Since this is a face detection all values are 1, the 'BBtype' for the features detection lists is 2. This would have been useful if detection lists were combined and way of identifying the type was needed. The area was useful for the fourth implementation stage (implementation 9.3) which led the prototype to process the intersections between feature and face detections. The area was calculated by multiplying the width and height. The final two metrics are the x and y pixel co-ordinates of the centre-point of the detected rectangle. This was derived through using (x2+w1)/2 for the x-value and (y2+h1)/2 for the y-value. All of these processes took place before the manipulation of the detection lists took place (before implementation 10.2) using the 'tableMatrix' function.

11.3 Failure of Merging Rectangles Exercise

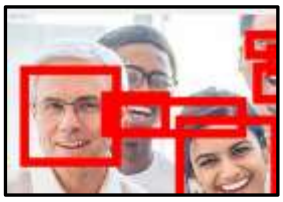
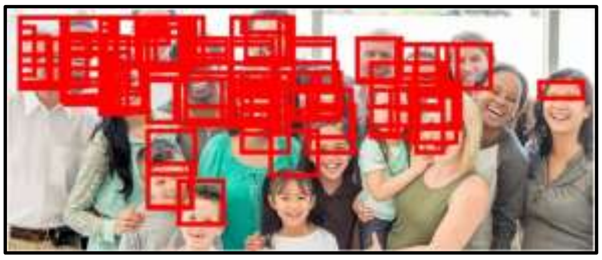


Figure 35: *Incorrect detection Example*

During the implementation stage, there were many different ideas including one that involved the merging and new formations of detections. It was the idea that the contained features detections within a face were removed (implementation 9.3), then the intersecting rectangles were in the way of progress. It was intended to create new dimensions by drawing a whole-some rectangle around the two intersecting rectangles by using the maximum dimensions between the two detections. This was to remove the detections that weren't not needed as they represented the same face, or it was simply a wrong detection. Figure 35 shows an incorrect detection where the rectangle stretches from the persons mouth to the head of the person on the right. Figure 33 shows the detection colour of yellow which means that it is detected as a pair of eyes. This makes one realise that the actual eyes of two separate people may be detected, meaning this detection will count as a single person's pair of eyes. This is extremely misleading which is why this merging exercise was hoped to remove such detections and merge to another detection shares the highest intersection percentage (measured by the comparing the areas).

This is didn't work for many reasons. The loop quite literally merged with many rectangles and in place merged with new rectangles that were formed during the loop. This created a massively cluttered and meaningless output, therefore this idea was not built-on upon (shown in figure 36).

Figure 36: *Merging of rectangles (BB)*



11.4 Potentially Applying Training Concepts to new Test Data

Regardless of the most bizarre detections, more attention was payed as to why the face detections were not made for certain people in the image. In this particular image which was used to for the development of the prototype, however many other non-detections may arise that have various reasons. This particular test-image did not detect 4 people. (Implementation 10.2) The reasons noticed reasons were unfortunate that the skin colour was un-recognised (darker-skinned people), or that visibly framed glasses obstructed the detection of eyes or a face in general. In order to address this, the literature research mentioned specific machine learning concepts like Adaptive Boosting which could be used to train a new set of data to strengthen classifiers in the hopes that the objector will detect these faces. The test-data would be images of people with glasses and darker skinned people (effectively black people).

The literature review mentioned that this method is only successful with a big range of training data which was difficult to access especially because all test-images are un-copyrighted. MATLAB has a very well demonstrated way of setting up the application for training (MathWorks, n.d). Though time was limited and access to a large set of un-copyrighted data meant that this was not a step worth taking.

11.4 Purpose of the Exploratory Data Analysis

The overall purpose of this stage was to understand what was possible with the data the was being produced with detection algorithm. Since this project was mainly an experimental process, many ideas and explorations needed to ensure the correct route is being taken to further develop the prototype.

This creative process discovered many discrepancies of detection algorithms and potential practical issues that should be addressed in order to evolve the prototype for a better application. As the prototype was developed with the tools and literature knowledge, it will now be evaluated.

12. Evaluation – Final Prototype

The aim of the study was to accurately detect the amount of people in an image or photograph. This is exactly what the intentions of this chapters are. Firstly the each person in every test image was manually counted and used to compare with derived analysis of the outputs produced from the prototype.

12.1. Prototype Accuracy

Figure 37 compares the difference between the prototype that detects the number of people against the actual number of people. The orange indication is the how far off the prototypes detected output is from the correct value. From this, it seems that in general there aren't too many differences, specifically for the images that have a lower number of people to be detected. The smallest number of people in an image has the smallest difference (image no.2), whilst image 11 and 12 have the highest number of people where image 11 has the highest difference. The actual total number of people for both images was 36, the highest in the whole data set. This makes one realise if the differences are formed due to a larger mass of detections required from the prototype. It also worth noticing that the 13th image was the detected perfectly as there are no differences shown.

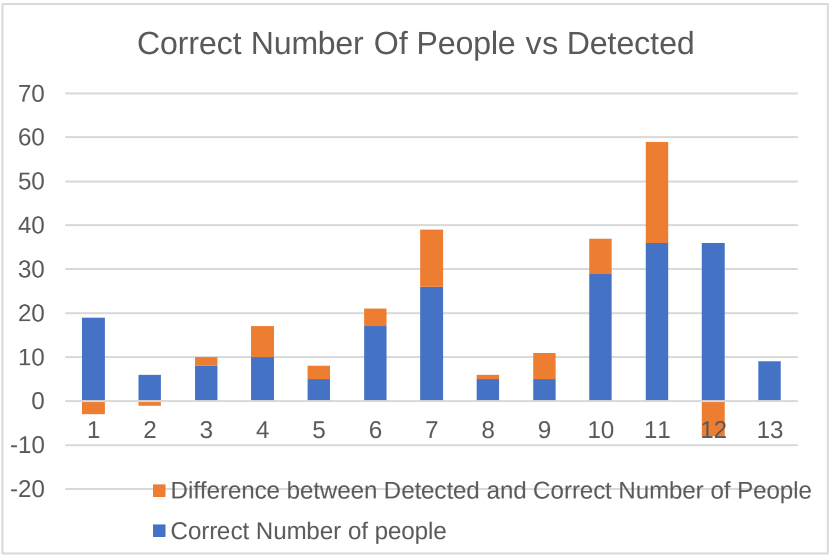


Figure 37: *Correct Number of People vs Prototype Output*

12.2. Data Set Comparisons

A new chart was produced to better understand the theory of why there are bigger differences in the correct number of people compared to the outputs of the prototype. Figure 38 shows the differences between the feature detections and what the proportionate feature detections should have been. According to the correct number of people, an individual has 4 features to be detected (2 eyes, 1 mouth and a nose), therefore multiplied accordingly. The images that have a

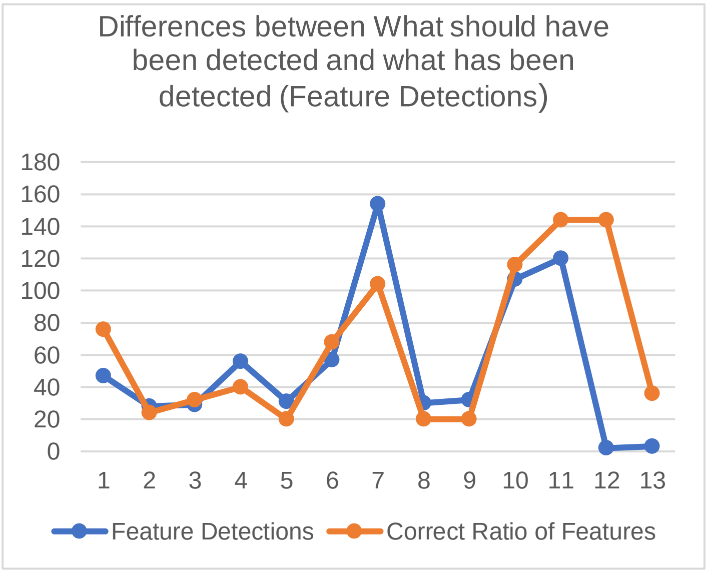


Figure 38: *Comparison of the ratio of Feature Detections*

massive gap begin with images 11 and 12, which are the ones that affected the detection of total the number of people outputted from the prototype (figure 37). This means that the larger the number of people to be detected in an image, the higher the difference i.e. error rate. Due to this, a comparison between face detection and the prototype was done, to then find out if a larger number of detections required affect the error rate than a smaller number of people that require to be detected from an image.

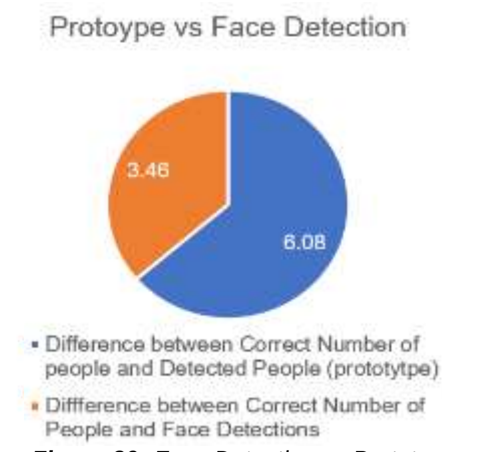


Figure 39: *Face Detection vs Prototype Results*

12.3. Face Detection VS Prototype

To understand which detection method is more effective, a pie chart was produced which displayed the average difference between the correct number of people and the outputs from the prototype and face detection. If the difference is greater, then it means that there is more error in when detection the correct number of people.

Figure 39 shows the findings that the prototype did not do better than face detection as the difference is greater from its output compared to the smaller averaged difference produced by the face detection. As figures 38 and 37 reveal that the larger the number of detected people in an image, the higher the error rate (difference from the actual number of people). So the data was split into two data sets. One included 6 test images with the total number of people to be detected being less than 10. While the other set included 6 images with the amount of people to be detected being higher than 10.

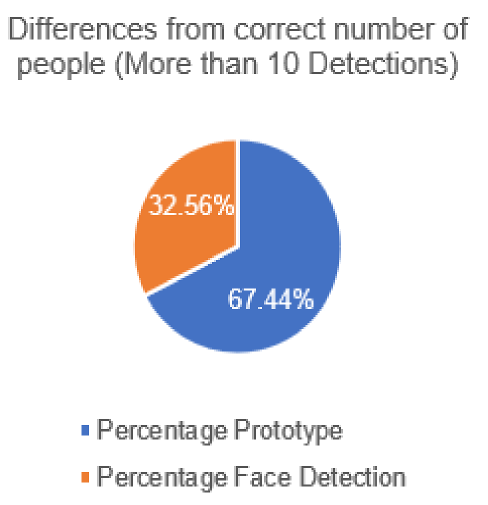


Figure 40: *Differences with a larger data set*

After the average differences of all images in each data set were calculated, the percentages were generated for a better understanding of the which difference was higher than the other.

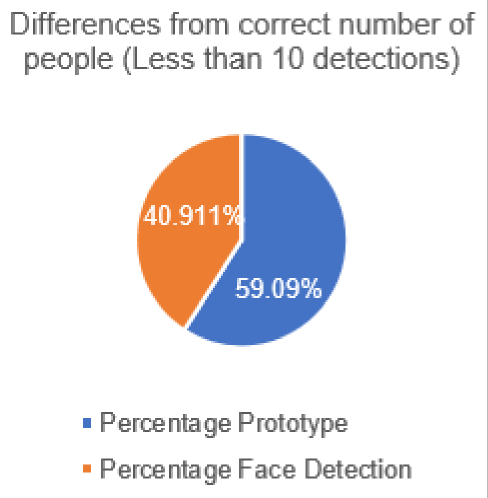


Figure 41: *Differences with a smaller data set*

Figure 40 shows the percentage difference of the dataset that included the images requiring more than 10 detections. The prototype has a larger difference than the face detection for the data that included more people. This means there were more errors than there were with the face detection. Which was similar to the case in figures 37 and 38.

Figure 41 shows the same metrics but with the data set that involved fewer detections (less than 10). The prototype has less of an error rate compared to the face detection difference. This means that images with the smaller number of people has a fewer error rate than images with a larger number of detections required.

Overall, the prototype produced that results that were similar to the face detections to at least less than 50%. This means that the prototype is not terrible. It can be applicable to crowd-based situations, however it will not be very useful for larder crowds involving sports events, or counting the number of students in an assembly during school. Alternatively, it may be useful for smaller groups of people. This may mean a group that is feasible for the prototype to detect the correct number of people and also requires the manual counting. So the prototype may be useful for a small task that the human may choose the prototype to produce the total number of people.

This is also suggests that artificial intelligence technology may not always be useful and in turn requires more data to test and review to continue raising the accuracy of its purpose.

13. Conclusions

The evaluations revealed quite a few weaknesses of the prototype, including having a higher error-rate compared to face detection. Even though feature detections were used and manipulated using clustering algorithm. However the error rate dropped as smaller number of people were to be detected in an image, specifically under 10. The prototype may be a small step to a better solution as it's accuracy shifts from the different ranges of number of detections required.

13.1. Meeting the Aim

The aim of the project was to develop a technical solution that accurately counts the total number of people in an image. It is fair to say that this aim was not fully met but a lot of information regarding face detection, and image processing came together in an attempt to form innovative technology to detect faces better than technology that already exists. This technology can definitely be expanded with bigger data-sets, more detection inputs like body shapes and other applications of neural networks to help refine a more accurate reading of the total number of people in an image. All objectives were strictly followed and helped the successfully closure of this project.

13.2. Applicability of Prototype

The prototype at one point, did deliver a result that was unexpected, one image was detected perfectly. This shows that the prototype is not completely useless and can perform perfect detections. Perhaps if this detection was reverse engineered as well as noting the image characteristics including lighting, number of people, colour schemes, more discoveries can be made, and the prototype can be further developed and increase its accuracy. The application of this prototype can still be applicable for teachers who take their students to school trips or event based situations. However it is recommended that this is not the only technology which is relied upon.

13.3. Improvements given the Time and Resources

According to the time and resourcing available at this given time, the technical solution was developed accordingly. Many improvements and error-nets could have been made, for example collecting a larger data set, instead of 13, maybe 30 or 40. Another improvement would be to train classifiers to better detect people and facial features like people with different skin colours. More tests could have been done to take into account of the type of images, including black and white or photographs taken in specific scenes like forests, or desserts. Regardless of this, the information absorbed during the literature review and the learning during the implementation stages encouraged future ventures involving Artificial Intelligence and Data Science.

14. Personal Reflection & Future Ventures

Overall this project has given me insightful knowledge in the field of Computer Science that I enjoy the most. The exploratory data analysis in particular, which included the analysis of image processing and face detection algorithms during implementation enabled me to explore my creativity in ways that programming, and visualisation can inherently demonstrate. It was fascinating to gain deeper knowledge of how neural networks began and the scholars that discovered interesting concepts like trigger of a decision and how it is now used for systems that were use daily. This provided me with a new perception and way of thinking when I use the technology around me, I am constantly eager to peek within the processes of a particular system and how it runs. I hope this curiosity will be long-term and to ensure it is, my future venture may welcome opportunities relating to Artificial Intelligence and Data Science. I also aspire to further study Artificial Intelligence or Data Science if the opportunity comes by.

The objectives of this project ensure a proper execution of this project which is primarily the reason why this project became a success (regardless of the prototype results). The following of the objectives enabled me to remain organised during all stages of the project. This became a learning curve that I will carry to future projects and ventures to ensure they are properly planned to reach success and eventually advance in my career.

Some tasks that were difficult included the programming of a new language. It was the first time I implemented a major project in MATLAB and it was imperative that I got practise especially because it was useful in many areas of the implementation and creative-processes. I am glad this software was used to for the execution the prototype as it was easily able to transform data structures and open opportunities to learn new machine learning operations like the use of 'trainCascadeObjector' tool. MATLAB was my introduction to a commercial environment that I may face in the future which will help align my programming skills well enough to learn quickly.

15. Appendices

Appendix A: Project Directory

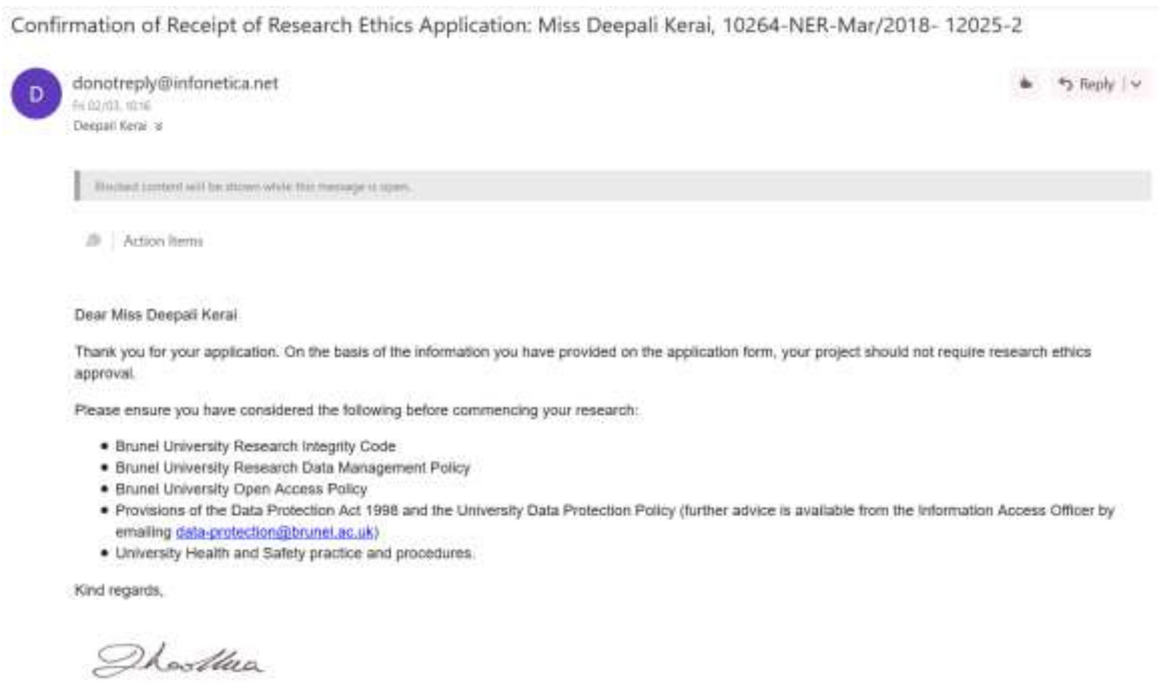
https://github.com/DeepaliCS/FinalYearProject-2018-

The link above is leads to a GitHub repository which include all test data, Segmentation scripts (Otsu, Watershed, and Hough Transform).

The prototype called (MainScript), Aiding scripts (CoorInit, CheckConditions, tableMatrix). And the detection algorithms (Face detection, Mouth detection, Nose detection, and Eye detection). All of these scripts need to be downloaded and saved in the same directory in order for the execution of the prototype to compile successfully. The test image path also may need to be altered (in MainScript). The test-data may be used.

The material uploaded on Wiseflow as appendix also a .zip file that contains all files mentioned above.

Appendix B: Ethical Documentation



Confirmation of Receipt of Research Ethics Application: Miss Deepali Kerai, 10264-NER-Mar/2018- 12025-2

donotreply@infonetica.net
Fri 02/03, 10:16
Deepali Kerai

Blocked content will be shown while this message is open.

Action Items

Dear Miss Deepali Kerai

Thank you for your application. On the basis of the information you have provided on the application form, your project should not require research ethics approval.

Please ensure you have considered the following before commencing your research:

- Brunel University Research Integrity Code
- Brunel University Research Data Management Policy
- Brunel University Open Access Policy
- Provisions of the Data Protection Act 1998 and the University Data Protection Policy (further advice is available from the Information Access Officer by emailing data-protection@brunel.ac.uk)
- University Health and Safety practice and procedures.

Kind regards,

[illegible]

Table of Figures

References

1. Kearns, H. (2012) ‘Crowd Control gone wrong: Hillsborough’, *Event Operation in Context*, 11 December. Available at: https://hannahkearns.wordpress.com/2012/12/11/crowd-control-gone-wrong-hillsborough/ (Accessed: 24 March 2018).
2. Ripley, A. (2008) ‘How to Prevent a Crowd Crush’, *TIME*, 6 December. Available at: http://content.time.com/time/nation/article/0,8599,1864855,00.html (Accessed: 24 March 2018).
3. Williams, A. (2015) *Furious parents slam ‘shambolic’ school trip after bungling teachers left Own, five, on his own in safari park for two hours*. Available at: http://www.dailymail.co.uk/news/article-3100707/Boy-5-abandoned-safari-park-school-trip-teachers-noticed-missing-arrived-nursery-hour-later.html (Accessed: 24 March 2018).
4. Newland, A. (2012) *The day I lost a child on the Tube*. Available at: https://www.theguardian.com/teacher-network/teacher-blog/2012/apr/21/lost-child-on-tube (Accessed: 24 March 2018).
5. Hoodies4Schools (n.d) 8 facts about kids Hi Viz Vests Everyone Should Know. Available at: http://hoodies4schools.co.uk/8-facts-about-kids-hi-viz-vests-everyone-should-know.html (Accessed: 24 March 2018).
6. Bleicher, A. (2017) ‘Teenage Whiz Kids Invents an AI System to Diagnose Her Grandfather’s Eye Disease’, *IEEE Spectrum*, 3 August. Available at: https://spectrum.ieee.org/the-human-os/biomedical/diagnostics/teenage-whiz-kid-invents-an-ai-system-to-diagnose-her-grandfathers-eye-disease (Accessed: 24 March 2018).
7. Marr, B. (2018) ‘The Key Definitions Of Artificial Intelligence (AI) That Explain Its Importance’, *Forbes, 14 February.* Available at: https://www.forbes.com/sites/bernardmarr/2018/02/14/the-key-definitions-of-artificial-intelligence-ai-that-explain-its-importance/#4cd1b6644f5d (Accessed: 24 March 2018).
8. Marr, B. (2016) ‘What Is The Difference Between Artificial Intelligence And Machine Learning’, *Forbes, 6 December.* Available at: https://www.forbes.com/sites/bernardmarr/2018/02/14/the-key-definitions-of-artificial-intelligence-ai-that-explain-its-importance/#4cd1b6644f5d (Accessed: 24 March 2018).
9. Albright, D. (2016) ’10 Examples of Artificial Intelligence You’re Using in Daily Life’ 26 September. Available at: https://beebom.com/examples-of-artificial-intelligence/ (Accessed: 24 March 2018).
10. Bhave, A. (2003) Biometrics Presentation [PowerPoint Presentation].
11. Oswal, N. (2018) ‘How AI is transforming business intelligence’, *Dataconomy*, 19 February. Available at: http://dataconomy.com/2018/02/ai-transforming-business-intelligence/ (Accessed: 24 March 2018).
12. Artificial Neural Networks Technology (n.d) *History of Neural Networks*. Available at: http://www.psych.utoronto.ca/users/reingold/courses/ai/cache/neural4.html (Accessed: 24 March 2018).
13. O’Riordan, A.P. (2005-6) ‘An Overview of Neural Computing: Part 1: The First Wave’, *Perceptron*, pp.(n.p).
14. Tucker, A. (2017) Neural Networks [Lecture to Computer Science students, Level 3: Artificial Intelligence], [Image]. Brunel University
15. Hubel, D.H, Wiesel, T.N, Physiol, J. (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. 160():106-54.
16. Christian G. (2013) ‘Hubel & Wiesel 1: Intro’ [Online Video] Available at: https://www.youtube.com/watch?v=y_l4kQ5wjiw (Accessed: 24 March 2018).
17. Steve Lehar (n.d) Hubel & Wiesel. Available at: http://cns-alumni.bu.edu/~slehar/webstuff/pcave/hubel.html (Accessed: 24 March 2018).
18. (n.n). (n.d) Automatic Thresholding. [PowerPoint Presentation] Available at: http://www.math.tau.ac.il/~turkel/notes/otsu.pdf (Accessed: 24 March 2018).
19. Hart, R. P. (2009) ‘IEEE Signal Processing Magazine: How the Hough Transform Was Invented’ pp. 18-22.

20. Sinha, U. (2010) 'The Hough Transform', *AI Shack*. [Image] Available at: http://aishack.in/tutorials/hough-transform-basics/ (Accessed: 24 March 2018).
21. Baxi, A. et al (2013) 'A Review On Otsu Image Segmentation Algorithm: Introduction', International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), volume 2, Issue 2. pp.387.
22. The Lab Book Pages (2010) *Otsu Thresholding*. [Image], Available at: http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html (Accessed: 24 March 2018).
23. Baygin, M., Karakose, M., Sarimaden, A., and Akin, E. (n.d) 'An Image Processing based Object Counting Approach for Machine Vision Application', [Image], *International Conference on Advances and Innovations in Engineering (ICAIE).*
24. MathWorks (2002) The Watershed Transform: Strategies for Image Segmentation. Available at: https://uk.mathworks.com/company/newsletters/articles/the-watershed-transform-strategies-for-image-segmentation.html (Accessed: 24 March 2018).
25. Hodneland, E., Tai, C., Weickert, J., Bukoreshtliev, V. N, Lundervold, A. and Gerdes, H-H., (n.d) 'Level set methods for watershed image segmentation', [Image], *University of Bergen Department of Sciences.*
26. Culurciello, E. (2017) 'The History of Neural Networks', Dataconomy, 19 April. Available at: http://dataconomy.com/2017/04/history-neural-networks/ (Accessed: 24 March 2018).
27. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998) 'Gradient-Based Learning Applied to Document Recognition', Proc. *Of The IEEE,* pp. 5-6.
28. Srivastava, N., Hinton, G., Krizhevsky, A., and Salakhutdinov, R. (2014) 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting' pp. 1929-1958.
29. Le. J, (n.d) 'The 10 Algorithms Machine Learning Engineers Need to Know' *KD Nuggets*, (n.d). Available at: https://www.kdnuggets.com/2016/08/10-algorithms-machine-learning-engineers.html (Accessed: 24 March 2018).
30. Deshpande, A. (2016) 'A Beginner's Guide To Understanding Convolutional Neural Networks', CS Undergrad at UCLA ('19), 20 July. Available at: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/ (Accessed: 24 March 2018).
31. Clark, L., (2012) 'Google's Artificial Brain Learns To Find Cat Videos', 26 June. Available at: https://www.wired.com/2012/06/google-x-neural-network/ (Accessed: 24 March 2018).
32. Khalajzadeh, H., Mansouri, M., Teshnehlab, M. (2012) 'Face Recognition using Convolutional Neural Network and Simple Logistic Classifier', *2012 Online Conference on Soft Computing in Industrial Applications.*
33. Geitgey, M., (2016) 'Machine Learning is Fun! Part: Modern Face recognition with Deep Learning' Medium Corporation, 24 July Available at: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78 (Accessed: 24 March 2018).
34. Ex-Sight (n.d) *Face recognition – Technology Overview*. Available at: http://www.ex-sight.com/technology.htm (Accessed: 24 March 2018).
35. MathWorks, (n.d) Object Detectors. Available at: https://uk.mathworks.com/help/vision/object-detectors.html (Accessed: 24 March 2018).
36. Y, Wang. (2014) 'An Analysis of the Viola-Jones Face Detection Algorithm', Image Processing On Line, pp.131-133.
37. Nazari, M., Majid, Seyyed., and Sayadiyan, A. (n.d) 'Face Detection Using AdaBoosted SVM-Based Component Classifier', *Electrical Engineering Department, Amirkabir University of Technology*, pp.2-3.
38. MathWorks (n.d) 'Train cascade object detector model'. Available at: https://uk.mathworks.com/help/vision/ref/traincascadeobjectdetector.html (Accessed: 24 March 2018).
39. Chen, Z., Lin, W., Xu, L., Li, L., Wang, S., (2017) 'Image Quality Assessment Guided Deep Neural Networks Training', pp.5-7.

40. Maghraby, M., Abdalla, M., Enany, O., and Nahas, M., (2014) 'Detect Analyze Face Parts Information using Viola-Jones and Geometric Approaches', *International Journal of Computer Applications*, pp.1-2.
41. GeeksForGeeks (n.d) *Introduction to Hill Climbing | Artificial Intelligence.* Available at: https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/ (Accessed: 24 March 2018).
42. StackExchange (2017) *When to choose Stochastic Hill Climbing over Steepest Hill Climbing*. Available at: https://ai.stackexchange.com/questions/4000/when-to-choose-stochastic-hill-climbing-over-steepest-hill-climbing (Accessed: 24 March 2018).
43. Woodford, C., (2018) 'Neural Networks' *ExplainThatStuff*, 14 March. Available at: http://www.explainthatstuff.com/introduction-to-neural-networks.html (Accessed: 24 March 2018).
44. WaveMetrics (n.d) *Image Threshold: Thresholding methods*. Available at: https://www.wavemetrics.com/products/igorpro/imageprocessing/thresholding.htm (Accessed: 24 March 2018).
45. Preiswerk. F. (n.d) 'Shannon entropy in the context of machine learning', *Medium Corporation – The Startup*, 4 January. Available at: https://medium.com/swlh/shannon-entropy-in-the-context-of-machine-learning-and-ai-24aee2709e32 (Accessed: 24 March 2018).
46. Dummies (n.d) Basics of Data Clusters In Predictive Analysis. Available at: http://www.dummies.com/programming/big-data/data-science/basics-of-data-clusters-in-predictive-analysis/ (Accessed: 24 March 2018).
47. Guest Writer (2015) 'What is Clustering In Data Mining?' *Big Data Made Simple*, 15 April. Available at: http://bigdata-madesimple.com/what-is-clustering-in-data-mining/ (Accessed: 24 March 2018).
48. S, Swift., Tucker, A., Vinciotti, V., Martin, N., Orengo, C., Liu, X., Kellam, P. (2004) Concensus clustering and functional interpretation of gene expression data, *Bio-Medical Central – Genome Biology*, section: Results.
49. QA Symphony (n.d), Agile Methodology: *The Complete Guide to Understanding Agile Testing.* Available at: https://www.qasymphony.com/blog/agile-methodology-guide-agile-testing/# (Accessed: 24 March 2018).
50. Powell-Morse, M. (2016) 'Waterfall Model: What is it and when should you use it?', *Airbrake.* 8 December, Available at: https://airbrake.io/blog/sdlc/waterfall-model (Accessed: 24 March 2018).
51. Extreme Programming Practises (2006), *Extreme Programming Enabling Chart.* [Image] Available at: http://wiki.c2.com/?ExtremeProgrammingEnablingChart (Accessed: 24 March 2018).
52. Ghahrai, A. (2016) 'Pros and Cons of Test Driven Development' *Testing Excellence* 29 November, Available at: https://www.testingexcellence.com/pros-cons-test-driven-development/ (Accessed: 24 March 2018).
53. Partogi, J. (2016) '5 Reasons why extreme Programming isn't popular', *Medium Corporation*, 22 June. Available at: https://medium.com/agility-path/5-reasons-why-extreme-programming-isnt-popular-83790418b901 (Accessed: 24 March 2018).
54. Rouse, M. (2016) 'Rapid Application Development (RAD)', *Search Software Quality* (n.d). Available at: http://searchsoftwarequality.techtarget.com/definition/rapid-application-development (Accessed: 24 March 2018).
55. I.T.Weld.(n.d) *Application Development*. Available at: http://itweld.net/development/ (Accessed: 24 March 2018).
56. Activo Hub (2016) *Rapid Application Development (RAD) Model – Pros and Cons*. Available at: http://notes.activohub.com/rapid-application-development-rad-model-pros-and-cons/ (Accessed: 24 March 2018).
57. Brunel University London (n.d) *Research Ethics*. Available at: https://www.brunel.ac.uk/study/postgraduate-study/graduate-school/Researcher-Development/Research-Ethics (Accessed: 24 March 2018).

58. MathWorks (n.d) *Image Processing Toolbox*. Available at: https://uk.mathworks.com/products/image.html (Accessed: 24 March 2018).
59. OpenCV (n.d) Available at: https://opencv.org/ (Accessed: 24 March 2018).
60. Engineering Statistics Handbook (n.d) *What is EDA?* Available at: https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm (Accessed: 24 March 2018).
61. Universal Teacher (n.d) *Disadvantages of Exploratory Research*. Available at: http://universalteacher.com/1/disadvantages-of-exploratory-research/ (Accessed: 24 March 2018).
62. Guru 99 (n.d) *What is WHITE box Testing? Techniques, Example and Types*. Available at: https://www.guru99.com/white-box-testing.html (Accessed: 24 March 2018).
63. Software Testing Fundamentals (n.d) *Black Box Testing*. Available at: http://softwaretestingfundamentals.com/black-box-testing/ (Accessed: 24 March 2018).
64. Rouse. M, (n.d) 'unit testing', *Search Software Quality,* (n.d). Available at: http://searchsoftwarequality.techtarget.com/definition/unit-testing (Accessed: 24 March 2018).
65. Visual Paradigm (n.d) *What is Unified Modelling Language (UML).* Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/#activity-diagram (Accessed: 24 March 2018).
66. Techwalla (n.d) *List of Advantages of UML.* Available at: https://www.techwalla.com/articles/list-of-advantages-of-uml (Accessed: 24 March 2018).
67. Lucid Chart (n.d) What is a Data Flow Diagram. Available at: https://www.lucidchart.com/pages/data-flow-diagram (Accessed: 24 March 2018).
68. Odgis, J. (2017) 'What is Abstract Visual Language', *Huffington Post,* 6 December. Available at: https://www.huffingtonpost.com/janet-odgis/what-is-abstract-visual-l_b_12428396.html (Accessed: 24 March 2018).
69. WikiHow (n.d) *How to Write Pseudocode.* Available at: https://www.wikihow.com/Write-Pseudocode (Accessed: 24 March 2018).
70. BiomeCardio (n.d) Damien Garcia Research. Available at: http://www.biomecardio.com/en/index.html (Accessed: 24 March 2018).
71. Eddins, S. (2006) 'Hough Transform coordinate system', MathWorks, 19 October. Available at: https://blogs.mathworks.com/steve/2006/10/19/hough-transform-coordinate-system/ (Accessed: 24 March 2018).
72. MathWorks (n.d) watershed. Available at: https://uk.mathworks.com/help/images/ref/watershed.html (Accessed: 24 March 2018).