Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Towards Identifying Social Interactions with Thermal Imagery

Eoin Roche
[redacted]

Supervisor: Dr. Jonathan Dukes

Co-Supervisor: Dr. Kevin Koidl

April 30, 2020

A Dissertation submitted in partial fulfilment
of the requirements for the degree of
Master in Computer Science

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: [redacted] Date: 30/04/2020

Abstract

Improvements in Computer Vision methods for people detection and pose estimation, combined with better cameras, has allowed computer scientists to begin exploring real-world social networks and how people interact with each other within these networks. The increased availability of modular sensors presents an opportunity to investigate social interactions using a system that could be deployed in a lightweight, unobtrusive fashion, while also protecting the anonymity of participants, something that is far more difficult to achieve using high-resolution RGB images. Thermal images can provide similar amounts of information as RGB images while also protecting the identity of participants.

This work aims to test the limits of the minimum specifications required in a thermal sensor to enable it to identify social interactions occurring between two people when combined with standard Computer Vision techniques. The interactions must also be measured with respect to time and the attention paid by each participant. To achieve this, this work proposes a system that detects, tracks and estimates the pose of a person found in a small thermal image. Two classes of detection are defined: head, when a person is close to the sensor, and body, when a person is standing a few metres away. A novel method to measure interactions is proposed: each frame a participant spends facing the other person increases their "attention score", showing which participant was more invested in an interaction.

To evaluate the system, a series of pre-recorded scenarios are played back as if it was live data. These scenarios show the best and worst of a system working at the limits of what its components can handle. Ultimately, it demonstrates that such a system could be deployed successfully. This work aims to be a proof of concept that will enable direct or tangential research in the future.

Acknowledgements

I would like to thank my supervisor Dr. Jonathan Dukes, my co-supervisor Dr. Kevin Koidl and Dr. Kristina Kapanova for all their assistance, guidance and patience throughout this research.

I should also thank my family for their unending support, not only for the duration of this research but indeed my entire time in University.

Contents

List of Figures

List of Tables

1 Introduction

For the first time, advancements in Computer Vision (CV) methods and RGB camera quality have allowed computer scientists to explore the dynamics of interactions that occur in the real world. These explorations are limited to fixed environments using large, expensive and stationary cameras. The increased availability of small modular sensors leads this dissertation to question whether interactions can be detected and measured using a single small thermal sensor employing CV methods, in a way that could be deployed with minimal infrastructure and without infringing on the privacy of those who are detected by the system.

1.1 Motivation & Context

An average person tends to have two interconnected social networks: Online Social Networks (OSNs) and real world social networks. OSNs have been a popular area of research since their inception. For computer science, this research has come in many different forms. In the earlier days of OSNs, the research was focused on the structures of these networks [5] and how people and groups operate within them. In recent years, the research has explored the more negative aspects of OSNs, including the propagation of rumours [6], and the increased use of bot accounts [7].

By comparison, computer scientists have had little opportunity to study the real-world social networks that people inhabit. Over the last decade, research has slowly begun to grow in this area. This has been assisted by the improvement in CV methods to detect and track people through video. Combined with better methods of pose estimation, this means that interactions can be detected, social cues recognised and group dynamics understood.

The drawbacks of these systems are their use of RGB cameras and the growth in recognition technology that allows for people to be recognised and tracked even at long distances. If combined with systems that analyse and log their social interactions, the recognition technology could create an even greater privacy breach.

Hence, this project aims to use a different kind of sensor to explore and categorise social interactions: a sensor that can protect the identity of those being observed. There are a

huge variety of sensors available today, including ultrasonic sensors, RFID sensors and thermal sensors. For this project, a thermal imaging sensor will be used.

The social interactions being identified in this project can be characterised as short interactions, lasting anywhere from thirty to sixty seconds, between two people. These kinds of interactions, although logistically simple, can have enough variety to allow this system to be presented as a proof of concept and as a first step in using thermal sensors to detect any form of interaction between people.

1.1.1 Thermal Sensors

There is a huge diversity in the types of thermal sensors, from large handheld sensors like the DT-9875 [8] that cost thousands of euro to small modular cameras like the AMG8833 [9] that cost a fraction of the price and can be interfaced with a Raspberry Pi.

These modular sensors are of particular interest for this project. While they will be inferior in terms of resolution and effective distance, they do have some advantages. They could be deployed in a lightweight fashion that would require only a small amount of power usage and would not be fixed pieces of infrastructure. Instead, they could be deployed more dynamically throughout a space.

Additionally, these modular sensors would help to protect the identity of the people detected. Larger thermal cameras can record at a resolution that makes facial features apparent enough to allow for recognition. The modular cameras are small enough that facial features will not be apparent in the images, while still allowing for a person to be detected.

The hardware considered for this project will test the boundaries of what information can be gleaned from very small sensors using conventional CV methods.

1.2 Aims

This project aims to answer the question of whether social interactions can be identified and categorised using low-resolution thermal imagery. To achieve this goal, a number of aims will be outlined.

Detect People in Low Resolution Images - The system will need to be able to detect people who are close to the sensor, meaning only their head and shoulders are visible, and people who are distant from the sensor, where their whole body is visible. Then a person will need to be tracked from their duration in frame and forgotten once they leave.

Build a Pose Classifier - A pose classifier estimates which way a person is facing using either head pose estimation or body pose estimation. Both are crucial functions in any

system that attempts to identify interactions.

Track Interactions - Once two people have been identified and their poses have been estimated, the system will need to decide whether an interaction is taking place. It does this using the pose information and the location of people in frame, before monitoring the interaction for total duration and measuring the attention of those involved.

1.3 Methodology

The methodology used to achieve these aims can generally be considered a version of the Action Research method [10]. This is an iterative process where, when possible, research and prototype work happen in parallel. This method is particularly applicable to the first two aims: detecting people and classifying poses. There are a plethora of potential methods, as will be outlined in the literature review and throughout prototyping. These methods will be tested and some will be rejected as unsuitable. Meanwhile, research for new methods will be carried out. This will repeat itself until an adequate method is found, at which point the next problem will be addressed using a similar methodology.

A novel approach to test the system will be considered, as currently there is no rigid way to appraise interactions. A series of scenarios will be acted out and recorded by a thermal sensor. These scenarios will be manually measured for duration and metrics related to the attention of the participants. The scenarios will then be played through the system, and its results will be compared to the expected values. These results will form the basis for the conclusion of the dissertation.

1.4 Dissertation Outline

The other chapters in this dissertation are summarised as follows:

Chapter 2, "Literature Review," will examine technologies and methods that have been used in research to achieve similar goals as those outlined above. These include the popular methods in Computer Vision for the detection of people in both RGB and thermal images, methods for tracking people on screen, head and body pose estimation methods and ways to categorise interactions.

Chapter 3, "Design," will discuss the chosen technologies, hardware and methods for the project, outlining the strengths and weaknesses of what was chosen, and comparing and contrasting what was chosen with the other potential options.

Chapter 4, "Implementation," goes in-depth into how the methods described in the previous

chapter are employed in the project. It will also discuss what, if any, deviations from the described methods were made and why.

Chapter 5, "Evaluation," briefly outlines what metrics are taken into account when appraising the performance of firstly the pose classifiers and later the system itself.

Chapter 6, "Results," uses tables, graphs and screenshots from test videos to outline a series of scenarios used to test the system. It concludes by discussing the strengths and weaknesses of the system that become apparent through testing.

Chapter 7, "Conclusion," compares the initial aims set out in this project with what was achieved during the implementation and discusses what worked well within the project and what fell short. Lastly, some potential future work is outlined.

2 Literature Review

This chapter will examine various methods that exist in Computer Vision that can help the project to achieve its aims. These methods will broadly fall into one of these four categories:

- Detect People in images
- Tracking People between frames
- Detecting Pose
- Categorising Interactions

This will give an idea of the potential methods available to this project.

2.1 Detecting People

Detecting people is an important area of research in computer vision, and many different methods are viable. These methods often detect a specific part of the anatomy like the face or head or the entire body. All three of these detection types will be considered here. Although a significant amount of research has gone into people detection in RGB images, the same cannot be said of thermal images. This has led to already proven methods in RGB images being applied to thermal images with minimal changes. Therefore, methods for both RGB images and thermal images will be examined.

2.1.1 Face Detection

One of the most popular methods of object recognition in Computer Vision is the use of Cascade Classifiers. The cascade proposed by Viola & Jones in 2001 [1] chains a set of weak classifiers together to form a strong classifier. In this case, each weak classifier was made of a Haar feature.

The Haar features used in this case are similar to those used in the generic object detector proposed by Freeman [11]. A Haar feature is a function that measures and compares the average intensity of one area of an image to another. These features can be represented by

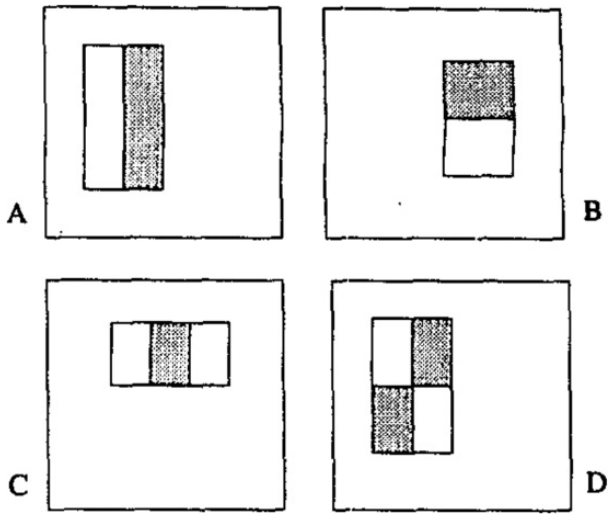

Figure 2.1: Simple Features used in the face detection cascade [1]

rectangles as in fig 2.1. The face classifier uses three different types of features. (1) A two-rectangle feature, which computes the difference in the sum of the pixels within each rectangle, which are both of the same dimensions and area. (2) A three-rectangle feature, which compares the sum of both outer rectangles with the inner rectangle. (3) A four-rectangle feature, that compares the difference in the sum of the pixels in each diagonal pair of rectangles.

At each stage of the cascade, one of these features is used that best describe some aspect of a human face. For example, in photographs, a person's eyes will often appear to be one of the darkest regions of their face. One two-rectangle feature in the cascade may compare the difference in intensity of the eyes versus the cheeks, or a three-rectangle feature could compare the intensity between both eyes and the nose.

Once the best features are chosen, the cascade searches an image with a default window size of 24x24 pixels. Each possible position that the window can search is passed through the cascade. If a sub-image of that window fails any stage of the cascade, it is rejected entirely.

The cascade classifier was considered fast at the time of the original study and is still considered so today. What makes it so fast is the adaptive boosted training algorithm. This algorithm chooses the best feature from a pool of 180,00 potential features at each stage of the cascade. This means that incorrect faces are unlikely to make it through earlier stages of the cascade which speeds up the process. The classifier was tested on thermal images and compared against other face detection methods [12]. The other methods [13, 14] proved to be slower and less accurate than the cascade classifier.

The cascade's speed is of interest to this project, as the system aims to do the classifications in real time. One concern is the search window size of 24x24 pixels given that the size of the

Figure 2.2: The HOG detection method [2]

sensor that this project aims to use is not much larger than the search window.

A similar set of Haar-like features were used to detect people from long range using a camera mounted on a UAV used for search and rescue [15]. This shows that the simple features could potentially be applied to also detect bodies. Although, in this experiment, they were likely to misclassify other objects as people too.

Haar features can be limited by their simplicity. An alternative feature type that can be used with cascade classifiers is Local Binary Patterns (LBP) [16]. Rather than computing the difference between rectangular regions, the LBP searches a 3x3 neighbourhood around each pixel in the search window. If the value of that pixel is higher than the surround pixels it is assigned a 1. Otherwise, it is assigned a 0. This is repeated for the entire search window which creates a histogram that describes the texture of the search window in a binary context.

This kind of feature could be used instead of a Haar features in a cascade classifier, and may perform better on the small image sizes, however the additional noise that exists in thermal images may pose a different kind of problem for the LBP.

2.1.2 Body Detection

Histograms of Oriented Gradients (HOG) was first proposed in 2005 by Dalal & Triggs [2] as a method for people detection in RGB images. HOG breaks images down into small regions or 'cells.' For each cell, it builds a 1-D histogram of gradient directions or edge orientations. These cells are normalised by taking a larger section of the image that would encompass several cells and calculating the 1-D histogram for that block. Combining this grid of normalised cells with a Support Vector Machine-based window classifier forms the person detector.

HOGs have been deployed successfully for use in infrared images [17] to detect pedestrians and to detect people in low-resolution infrared video [18]. HOGs clearly have the ability to work with images outside of the RGB spectrum. However, it is unclear how they will perform with thermal images.

Given the generic nature of how people appear in thermal images, there is the possibility of using template matching [19]. Template matching is a simple method where a search image or template is created, and any target image is searched for that template. Template matching has several limitations. One template only describes one orientation of an object

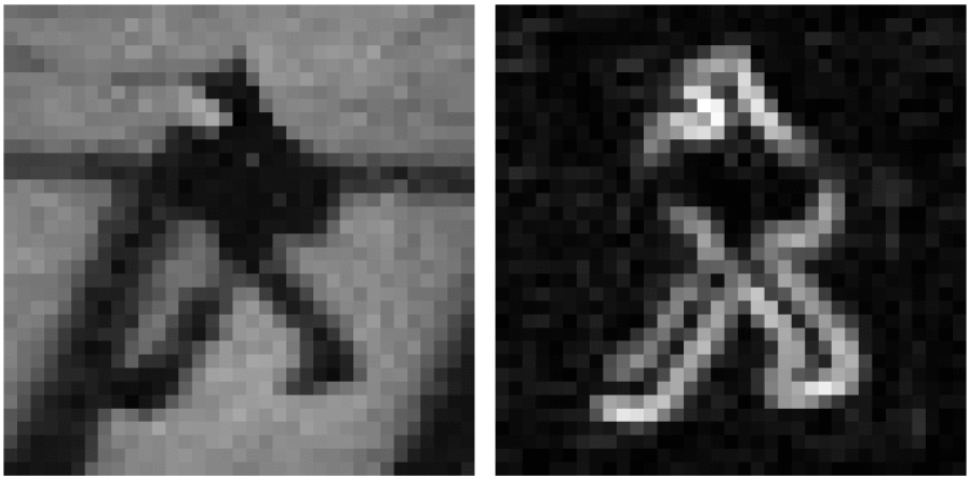

Figure 2.3: How a person appears in a thermal image and in the CSM [3]

or person. The size of the template can be changed but, if not done carefully, that may change how the template looks and render it useless. A template matching result is given a score, and is then thresholded, so an image is likely to include several false positives or negatives depending on the threshold.

However, more complicated methods that incorporate template matching can, in fact, achieve more reliable results. One method proposed is a two-stage approach [3] that was designed to work specifically on thermal imagery. The system detects people from varying distances using a high-resolution thermal sensor. The first step was to create a method to separate foreground objects (people) from the background. This was done by creating a background thermal image of the space, the median or mean image for the space over a period of time. Then, for each pixel in a test image, the minimum of the input gradient magnitude and the background gradient magnitude is chosen. If the input gradient is the minimum then that pixel is set to a 1, otherwise, it is set to a 0. This creates a binary image where only the pixels that are assumed to be the foreground are displayed. This is called a Contour Saliency Map (CSM).

The second step is to create a generalised person template, to do that, several cropped windows of people from sample CSMs were manually extracted, and normalised. The CSM often returns windows containing people, partial people and other foreground objects that are not people. From there an Adaptive Boosted Classifier is created chaining weak classifiers together to form a strong classifier.

A method that could perform both head detection and body detection together is Statistical Pattern Recognition [20]. Pattern recognition has been used in a variety of engineering and computer science areas, including computer vision. There are two forms of pattern matching. Supervised classification is where a given object or pattern is classified into a predefined

group. Unsupervised classification is where a pattern is assigned to an unknown class. The latter is more reminiscent of clustering. In this project, supervised classification could allow for both head and body classification to be completed using the same methodology.

In CV, Statistical Pattern Recognition is applied to features of contours or edges. Edges images are binary images that represent all of the edges found in an image, in the same style as the CSM above. However, these edges can be found through first derivatives, second derivatives or Canny [21]. Edges images are created using Binary Chain Codes (BCC) [11]. A BCC starts with a pixel value (x, y), then each of the possible following neighbouring pixels in the chain is represented by a number between 0 and 7.

Once an edges image is created, each contour can be measured for several features. Examples include width, height, area and circularity. These features can be used to create classes based on the expected value of features for certain classes. This could enable head and body pose to be done together with two different classes defined.

2.2 Tracking People

Once a person is detected, they will need to be tracked in order to maintain continuity across interactions and avoid duplicate detection. Ren [22] uses archive footage (from Casablanca, 1942) to showcase a system of tracking by detection. A cascade classifier [1] is used to detect faces in the video. Once a face is recognised, a track is created for it. As the face is continually recognised, the track is considered 'good' and will be given priority in later frames.

However, there are many reasons a track can fail including low image quality, poor lighting, and the person being tracked turning away from the camera. To account for these instances, a low-level tracking method is implemented. A simple head template is used, and a small window around where the head was last detected is searched. If found, the predicted location is updated accordingly. Something similar can be applied to low-resolution thermal images. A track is created for each person detected and is updated on each frame. In the case of this project, the focus will be on using the tracker to guarantee that the noise in the image does not result in duplicate detection.

2.3 Estimate Pose

Pose Estimation has been an area of research in computer vision for decades. It is seen as an important step in improving the performance of human-robot interactions [23] and in the improving of crowd tracking technology [24]. Pose estimation can be broadly divided into head pose and body pose estimation. Methods of each are discussed below.

2.3.1 Estimating Head Pose

A huge amount of social context is gained through non-verbal actions such as where a person stands or the direction they look. For example, if a group of programmers is having a daily stand up meeting, we can understand who is taking part in the meeting by who is standing. If someone in that meeting is speaking, more often than not, the rest of the participants will be looking in their direction. Therefore, being able to ascertain which direction a person is looking can be crucial to understanding a social interaction. Without the use of eye tracking technology, it is very difficult to know for certain where someone's gaze is directed. Instead, many methods exist to estimate where someone is looking based on their head position.

Many different Head Pose estimation methods have been documented [25] including:

Appearance Template Methods Uses a set of templates for different head positions with matching techniques.

Detector Array Methods Trains several detectors to each recognise a specific pose [26].

Geometric methods Attempts to identify aspects of the face such as the eyes, mouth or nose to discern head position.

Tracking methods Using tracking techniques to note changes to head position between video frames.

Hybrid methods Combines aspects from one or more of these methods to attempt to overcome the individual methods' shortcomings.

The appearance template method would be considered the simplest method. The method has some advantages in terms of flexibility. It works on both high and low resolution images and does not need negative examples to train. However, the templates can struggle to identify small changes in pose and are more likely to be error-prone.

Detector array methods build on the success of face detection algorithms such as [1] and extend these algorithms to train classifiers to recognise specific head poses. Issues arise in training many detectors for a complex set of potential poses. It takes a significant length of time and requires a large number of positive and negative samples. However, for this project, a detector array method may prove the best option due to the well-documented speed of the cascade classifier as discussed above.

2.3.2 Estimating Body Pose

Body pose is a more challenging subject to estimate. Although great strides have been made in recent years in classifying Body Pose in RGB images, little has been done in the thermal

spectrum. Using RGB images, methods are divided by their use to human body models [27], their use of low-level pixel analysis or high-level person detection and their use of one-stage or multistage deep learning.

Even in the Internet of Things (IoT) space [28], there is a heavy reliance on the use of body part and joint detection. This is a potential issue for this project as joints will most likely be undetectable in low-resolution thermal images. Therefore, an adaptation of the detector array methods described above may be the best route forward.

2.4 Categorising Interactions

Categorising interactions has becoming a growing area of research in the last decade. This section will examine two of these systems.

2.4.1 Logging Office Interactions

This system [4] combines several components in order to log the behavioural patterns of people working in a research lab. By tracking people's behavioural patterns the system can create a heat map of the most common positions people sit in throughout the day and who they interact with the most. The system is comprised of 3 components. A RFID Reader notes the identity of anyone who enters the research lab and initiates the tracker. The tracker follows each person as the move around the lab, remembers where they were if they are obscured from the camera and picks them up again once they are back in frame. The Head Orientation Estimator, is a weak system that has 8 possible directions that are manually labelled throughout training samples to build a classifier.

Using these components together can allow the system to identify primitive human interactions connected to head pose. If two or more people are looking at each other, they are considered to be interacting. The system is tested with small interactions of 2 people, up to full group meetings of 5 people, and the interaction classifier achieves an overall precision of 71%.

For this project simplifying the head pose problem can be a viable way forward, focusing on a set number of directions rather than trying to estimate specific angles of pose.

2.4.2 Children's Social Behaviour Classification

Research into children's social behaviour is an import part of developmental psychology. Creating systems that can handle, at least in part, some of the analysis process can save a significant amount of time. This paper [29] divides children's interactions into three categories. (1) Solitary Play - A child plays apart from other children, they are concentrated

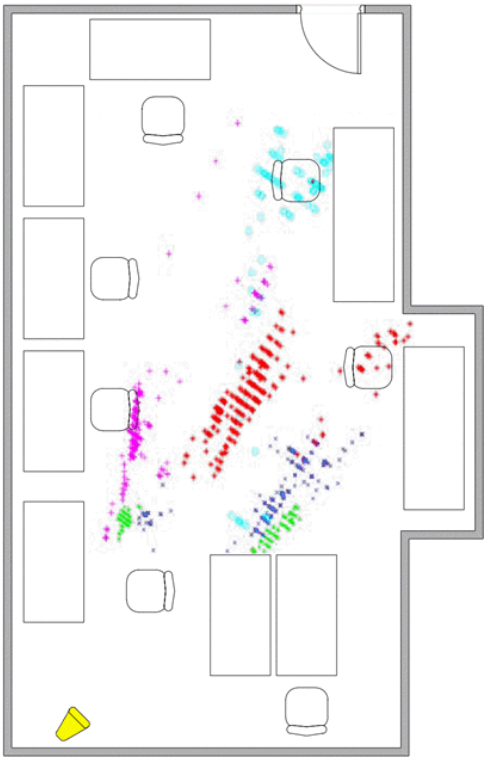

Figure 2.4: Heatmap of office interactions created by the system [4]

on their own activity and pay little or no attention to anyone else. (2) Parallel Play - A child plays independently, while also paying some attention to other children the other children's activities. (3) Group Play - A child is playing with other children and there is a common goal or purpose in the activity.

The system attempts to classify the category of play for 3 children who are standing around a table full of toys. To do this, each child's head pose is estimated. Using this estimation, an attempt is made to understand where each child's attention is directed, e.g. what toys they are looking at. This information allows the interactions to be categorised.

For this project, the idea of attention, and where attention is being directed, will be a valuable metric, as there are few features in thermal images to categorise interactions with.

2.5 Summary

This chapter outlined many methods that could be applied in this project. For person detection, a cascade classifier appears to be a strong method, having been applied to both face detection and body detection. On the other hand, Statistical Pattern Recognition could detect both close up people and distant people using one method, rather than creating several classifiers. Template matching may have too many downsides to be used in either detection or pose estimation, but should still be explored.

A Detector Array appears the best option for pose estimation, especially when limiting the number of possible poses that need to be detected. The detector array should be extendable to consider body pose also, as most of the current body pose methods require limb detection that will not be possible.

To measure interactions, time and attention are the two metrics that have been used elsewhere. How exactly attention can be represented using the thermal images in this project will need to be explored in the design section.

3 Design

This chapter will consider the design of a full system, from data collection to analysis. It will discuss what methods can be used and compare and contrast the chosen methods with other options found during the Literature Review. Lastly, this section will examine the chosen hardware and technologies for this project, and their positives and negatives.

3.1 Hardware

This system would require one or more thermal sensors. These days, a huge variety of thermal sensors exist. They range from large handled and extremely expensive thermal cameras, to sensors that attach to a phone, to small modular sensor arrays that can be connected to a PC or Raspberry Pi. The smaller the sensor that can be used, the less obtrusive the system will be, leading to a focus on modular cameras for this system. Although they will suffer in terms of quality as the images will be very small, there will be benefits in terms of energy usage, privacy and mobility. Ultimately, this project will operate with only one camera.

3.2 Centralised Processing

All data will be sent encrypted over the Local Area Network (LAN) to a central machine that will handle the processing of the raw data. Not requiring the nodes themselves to do any processing will enable them to be physically smaller and have less of an energy footprint.

A central server would need to handle all incoming data, create images and then analyse them. The server would need to be robust enough to handle live data flow from multiple sources. However, as no processing is done at the nodes themselves, the server would not need to send any information back to the nodes, other than necessary messages to exchange data.

Alternatively, a distributed system could be deployed. This would require each node to read and analyse data independently, before sharing that information with neighbouring nodes.

This kind of system has some drawbacks. Requiring nodes to analyse the data means that they will need more processing power than if they were only sending the data to another point. Also sending results between neighbouring nodes would increase the load on the network. Ultimately centralised processing is the stronger option.

3.3 Analysing Interactions

Once the data has been parsed from the nodes in the system, it must be analysed to determine if there are people in the image and whether an interaction is taking place. To determine this the following steps will be taken:

1. Creating Thermal Images
2. Detecting Objects
3. Estimating Pose
4. Categorising Interactions

3.3.1 Creating Thermal Images

As thermal cameras return an array of thermal readings, rather than pixel data, it will be important to be able to take this data and map it to pixel values at a speed that would allow it to be done live. It will also be important to enable this step to be constrained in a way that can help objects stand out, particularly in the case of distant objects that may blend into the background. The system will take values from a CSV and create RBG images.

An alternative would be to take the thermal values and map them straight to a grayscale image. That would cut out several processing steps. However, the advantage of creating RGB images and then processing them is the removal of some of the noise that is generated by thermal images. The way heat radiated creates additional noise around a person detected and creating grayscale images directly from the thermal values will include this noise. This can affect the object detection stage as an image may appear larger or of a different shape because of the noise.

3.3.2 Detecting Objects

Before any object can be detected a number of preprocessing steps will need to take place for each image. Each image must be:

- Isolated to the Red Channel
- Converted to a Grayscale Image

- Converted to a Binary Image
- Converted to a Edge Image

The first three of these are standard Computer Vision methods that will be discussed in further detail during the implementation chapter. Ultimately, these three methods will create a binary image; a 2-D array of white and black pixels.

Extracting edges is a more complex operation. Using the binary image as our edge data, boundary chain codes [11] (BCC) can be created for each edge in the image. An edge is what the system will perceive to be the outline of an object. The BCC gives a visualisation of what the edge looks like, and that can be used to calculate various features about the object.

From here, Statistical Pattern Recognition [20] will be used. This method of classification utilises probability to classify objects discovered in an image.For a set of objects, it defines a set of features that can be obtained through measures of the edge. In this project, there are two features Area and Circularity.

A set of bands are created for each feature with maximum and minimum expected values for an object. Any found object in the image that meets all of the criteria for a particular class will then fall into that class in the system.

The system will have two classes: a Head and a Body. These will be differentiated by the expected area and circularity of their edges.

With a person detected, the system will first draw a bounding rectangle around them. The system will need to track any person found for the duration they are on screen. The tracking will not be overly complex and will serve to avoid duplicate detection while a person moves around the frame. With that in mind, the tracker will use the center point of the bounding rectangle found earlier and store it in memory. In future frames, a new bounding rectangle will be created with a new midpoint. If that new midpoint is within a certain threshold distance of the old midpoint, the system will surmise it to be the same person. Once a person leaves the frame, they will be forgotten, as tracking people between sessions is not part of this project. In addition, these thermal images make recognition highly difficult.

Histograms of Orient Gradients was also considered for this task. However, the method breaks down when working with such small images. The use of a facial classifier as described in [22] was considered but these images do not display enough facial features to allow this form of tracker to operate.

3.3.3 Estimating Pose

Many of the previously discussed pose classifiers were created to work on either RGB images or high-resolution thermal images. The decision to use small thermal sensors in this project created a challenge in choosing the right method for both head pose and body pose estimation. In order to simplify the problem, a decision was made to constrain the possible set of directions the pose classifiers can find. This meant defining a strict set of poses to be recognised rather than a spectrum of angles. For this project, three poses will be recognised with respect to direction towards the sensor, facing forward, facing left or facing right.

The specific method to detect pose is a Detector Array [25]. A detector array creates a number of different detectors, one for each classifiable direction, and then chains them together.

In this case, the detector array will follow the method proposed by Viola & Jones [1]. A cascade classifier will be built for each direction. Given the performance of these classifiers on a set of test data, they will be assigned an order. This will decide in what order each classifier's result is taken on a live image. One big motivator for choosing cascades was the speed. As previously discussed, the cascade classifiers are consistently faster than other detection methods [12]. Unlike Viola and Jones, however, this system will not deploy Haar features. In this case, Haar features are too simple to identify significant differences between pose in low-resolution images. Instead, LBP features are used [16, 26].

Another method that could have been deployed here is template matching [19]. Template matching would also be similarly fast in this system given the size of the images. The featureless nature of how people appear in thermal images also lends itself to the creation of a normalised template. This means that one template could potentially handle a variety of people and distances.However, the increased noise that exists in thermal images hinders the already subpar effectiveness of Template Matching.

Another alternative, in this case, is an Adaptive Boosted Decision Tree. Instead of looking at images' data, the Decision Tree would instead build a model using the raw thermal data. The model would be trained on thermal values found in a Region of Interest (ROI). The ROI would be found during the object detection stage. A line of raw thermal data could be reshaped into the image dimensions, and then the thermal data ROI could be extracted. The ROIs would then be manually labelled, and the model could then build a classifier with multiple possible results. Although this could work in theory, in practice, it would have very limited operating ranges. While a thermal image can be constrained to make a person's face look the same thermal intensity from .5 metres from the camera and 2 metres from the camera, the raw data would not reflect that, and the classifier would rapidly break down.

3.3.4 Categorising Interactions

A key part in the design process is to identify exactly what features can be analysed in these images, so that the system can state whether an interaction is occurring and also monitor aspects of the interaction.

Identifying Interactions

The limitations of using thermal images alone in this project mean the system is restricted to using the pose classifiers to determine when an interaction is taking place. Therefore, an interaction will be deemed to be taking place if at least two people are identified and two people are seen to be facing each other. From the first frame where two people face each other, the interaction begins to take place. An interaction will end if the two people both face forward or one person leaves the frame.

Analysing Interactions

What happens during an interaction must also be considered. In the study into children's behaviour patterns while playing together [29], particular time was dedicated to classifying the type of play; whether a child was playing alone or in a group. To do this, head pose was combined with arm tracking to determine where a child was overall directing their attention. This project can similarly use an understanding of attention to determine the intensity of an interaction.

3.4 Hardware & Technologies

In this project two different pieces of hardware were chosen a Raspberry Pi with AMG8833 Thermal sensor [9] and a MLX90640 Development Kit [30]

The aim when choosing hardware was to find small cameras that required a low amount of power that could be applicable in an IoT environment. Prior research in thermal imagery focused mainly on costly, high-resolution cameras. From the perspective of this project, these large cameras have several drawbacks. Firstly, they require a significant amount of power to operate, meaning they could not be deployed in a standalone or modular fashion. Secondly, the high resolution also poses a challenge to privacy, as these cameras are far more likely to enable the user to discern the facial features of anyone seen by the camera. Additionally, the higher resolution would lead to more data being generated and more of a load being placed on the network.

The hardware chosen for this project is the exact opposite of the cameras described above: small, low resolution, and relatively cheap modular sensors. These sensors will enable this project to test the extremes of low-resolution thermal imagery and examine what the absolute minimum in specifications will allow for in pose classification while maintaining the anonymity of the people being recorded. This section will briefly detail both sets of hardware, the configuration they will be used in and, where applicable, their proprietary software.

3.4.1 Raspberry Pi with AMG8833 Thermal sensor

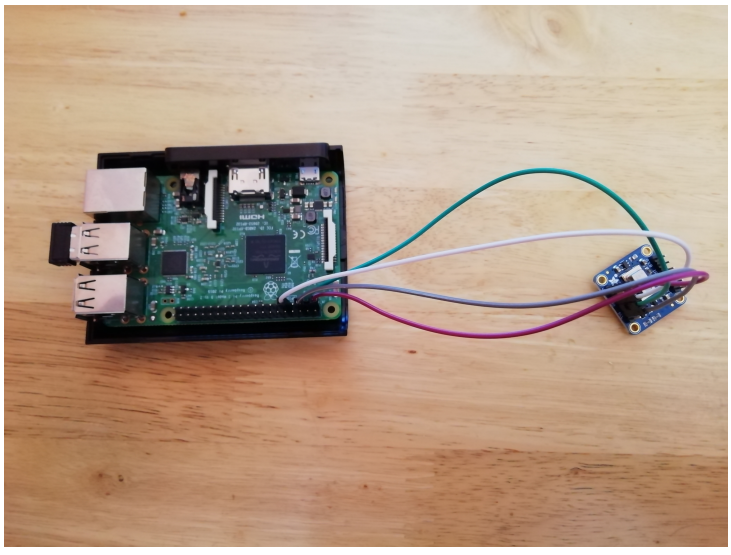

Figure 3.1: AMG8833 with Raspberry Pi

The AMG8833 thermal sensor is a small 8x8 thermal array that can detect temperatures in the range of -40°C and 80°C. It has an acute viewing angle of 60 degrees. Adafruit created a simple python library for use with the AMG8833 that would allow for communication

between the camera and the Raspberry Pi over i2c. This allows for sets of 8x8 thermal readings to be saved in a log or sent over http elsewhere. The AMG8833 integration with a Pi would allow it to be a standalone node. The Pi itself is a standard model 3 with a clean raspbian install.

3.4.2 MLX90640

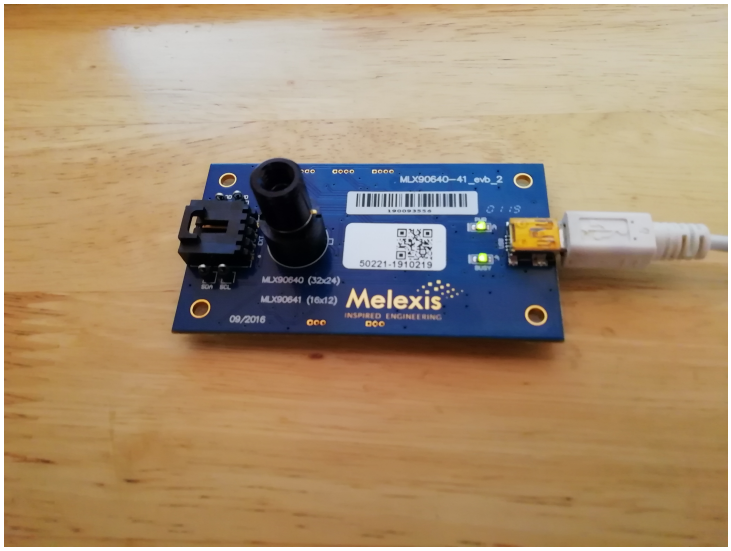

Figure 3.2: The MLX90640 Development Board

The Melexis development board enables quick plug and play functionality between a thermal sensor and machine. The MLX90640 IR array itself has 32x24 pixels, meaning much larger images than the AMG8833. It has a programmable refresh rate between 0.5Hz and 64Hz. It can detect temperatures between -40°C and 80°C. This project will focus on a small temperature range between 14°C and 32°C.

The development board comes with proprietary software that circumvents some of the potential roadblocks that exist within the freeware driver, such as the driver requiring additional libraries that may be unavailable. The software allows a user to control a number of camera features via a UI: The temperature range can be limited to specific higher and lower values, or change dynamically, the frame rate the sensor records at, the interpolation of software's displayed image and the status of the log file.

The software can display the current readings in the log. Alternatively, it allows you to see the current measurement in the form of an image. Although this image cannot be seen outside of the software, so it is not very useful, it was used as a benchmark when creating images later.

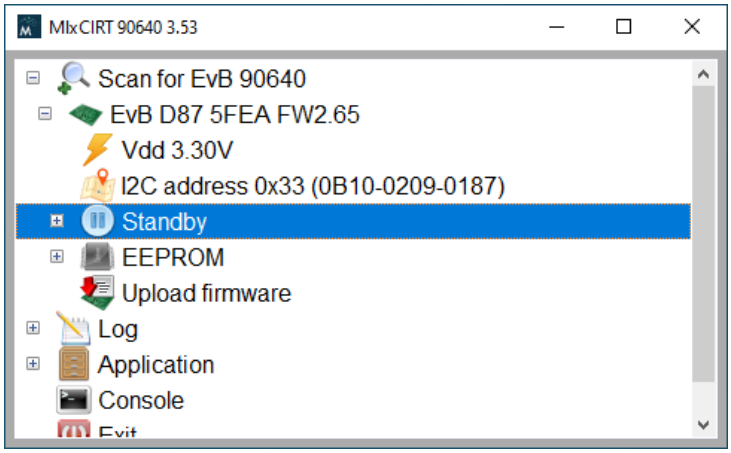

Figure 3.3: The Development Board software

3.4.3 Technologies

For image manipulation, OpenCV [31] was the library of choice given its extensive history and large variety of both prebuilt functions and helpers to create custom classifiers. OpenCV has implementation on both CPP and Python. For this project, the Python implementation was used due to its ease of install across multiple platforms. More specifically, in this case, Windows and Raspbian. Various Python utilities were accessed throughout the project.

3.5 Privacy & Security Considerations

The system proposed in this design section comes with inherent privacy and security concerns that will be outlined here, along with the mitigation techniques and the decisions taken to reduce these risks.

3.5.1 Security

This system will require a disparate group of nodes sending data simultaneously over the LAN to a central point. This kind of system is vulnerable to a variety of potential attacks, including both active and passive attacks [32].

Passive Attacks

Passive attacks involve an attacker reading packets off the network, without attempting any modification. In this project, thermal data is being sent from nodes such as raspberry pis with modular thermal sensors or small standalone sensors to a central point over the network. If an attacker was to passively monitor the network, they would be able to see the frequent transfer of packets from the these nodes and through additional header data in the messages would eventually conclude this was thermal data.

From there, an attacker could build images and attempt to recognise people. This would enable them to break the confidentiality of the messages and also potentially the confidentiality of the people. This could reveal some simple but significant information about the number of people who share an office or attend a particular meeting at a certain date and time.

Active Attacks

Once an attacker has passively monitored the network for enough time to understand the contents of the packets that are being sent from a node to the central machine, there are a number of potential avenues of attack. Active attacks involve writing packets onto the network. Without IPsec, it is simple to spoof the source address needed to send a packet to the desired destination.

There are multiple methods of attack that can all lead to the same thing, forcing a particular system state that is not the real expected state.

Message Insertion An attacker creates a new message with a specific set of thermal values that will represent a scenario they wish to display, this message would be disguised to appear as if it originates at a source node.

Message Modification An attacker takes a message from the wire, modifies it and sends it back into the system, this could allow an attacker to edit the thermal readings to add or remove a person or object from an image and affect the system state.

Man-In-The-Middle Uses a mixture of methods to simultaneously pose as the message sender to the receiver and the receiver to the sender at the same time. This would allow an attacker to still receive the live feed of data from the node while controlling what the central machine sees too.

Ultimately, these methods of attack all surmount to some form of tricking the system to be in another state than reality. This could, depending on the other systems using this status to choose actions, cause tangible damage.

In order to mitigate the risk of an active attack, all the packets across the LAN could be encrypted to help protect against passive attacks. The attacker would then need to brute force the key over time or attempt to socially engineer their way into the system to get access to the messages.

3.5.2 Privacy

Similar systems of this type use colour images [4, 29]. Colour images are easier to process, and to build head or body pose classifiers for, however the use of RGB images makes the

recognition of a person's identity far easier. This means that an attacker with access to such a system using RGB images would be able to track the behavioural patterns of a person throughout multiple sessions whether the system itself had intended that to be done or not.

The choice of low-resolution thermal images in this project seeks to protect the anonymity [33] of those people founds by the system. Facial features are completely unrecognisable in the images created from the MLX90640 data. This means they are protected from facial recognition. However, other methods can be used to identify a person. For example, a person's gait can be unique enough to identify them with sufficient training data. An attacker with access to the system for a prolonged period could build up a model of a person's gait over time and still attempt to identify them. However, to accurately assess a person's gait, high frame rate video is preferable. Therefore, limiting the frame rate of the thermal sensors where possible helps to mitigate this as a potential threat.

Thermal imagery similarly makes the tracking of people more difficult across sessions. In one particular session, a person in a thermal image can be tracked like any object in an image. To avoid any possibility of the person being tracked in a later session, no memory of the objects previously tracked should be retained.

4 Implementation

This chapter delves into the detail of implementing the 'Categorising Interactions' section of the design chapter. The implementation will discuss how the stages outlined there were built to work in a live environment with a single the MLX90640 Thermal sensor.

As listed previously, the four stages in categorising interactions are:

1. Creating Thermal Images
2. Detecting Objects
3. Estimating Pose
4. Categorising Interactions

4.1 Create Images

The MLX90640 thermal sensor returns an array of thermal values. The MLX90640 Development board writes thermal values to a local CSV file at a specified location. Each frame of readings from the sensor is represented by a 1-D line of 768 values written into the CSV file. The sensor also writes additional information into the file. The first processing step is to remove the additional data and leave the file with just the 768 values per line.

Using the Python Color library, a bank of colour is constructed with 1024 colours between red and blue. Algorithm 1 outlines how an image is created from a line from the CSV.

The ability to constrain the maximum and minimum values is an important. With a fixed maximum and minimum, distant objects are likely to get lost in the noise as the difference in temperature between an object and the background drops significantly with distance. Similarly, when dealing with close up images, the increased temperature creates a lot of noise in the surrounding area. A flexible maximum and minimum enables the greatest amount of information to be retrieved from any one image. It should be noted that all thermal images shown throughout this section and in later sections have been scaled up 8 times from their original size.

```
Specify minimum and maximum temperatures, either the highest and lowest value read
 by the sensor or custom values;
Create an Empty Array of dimensions (imageX, imageY, 3) where imageX and imageY
 are the camera's resolution and 3 represents a value for each colour in an RGB image;
for Each Row do
    Reshape the row to match the camera dimensions ;
    for value in row do
        Map the value to a specific colour using the max and min temperatures as the
         upper and lower bounds for the bank of colours;
        Set the corresponding pixel in the empty image array to the new colour value
    end
end
```

Figure 4.1: Algorithm 1: Create Images

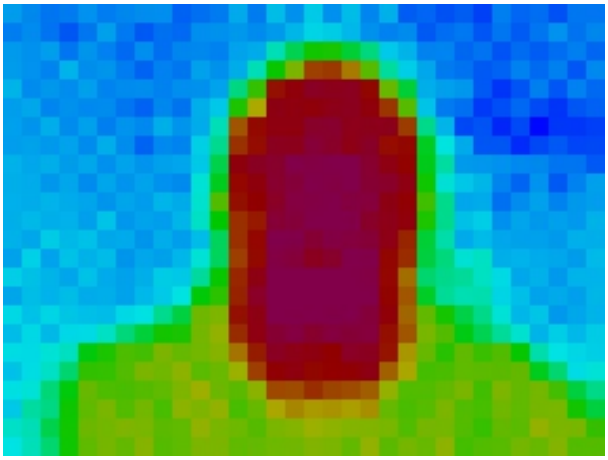

Figure 4.2: Initial image created from thermal readings

4.2 Detecting Objects

To detect objects in the thermal images that have been created, Statistical Pattern Recognition [20] is implemented using features calculated from the edges found in the images. Before the edges can be found, the following pre-processing of the images must be completed.

4.2.1 Isolate Red Channel

RGB Images are represented in code by 2-D Arrays where each element in the array contains three values, one for each of the primary colours that mix to create the colour that is displayed. Given that heat radiates, a thermal image will always contain a large amount of noise. As a person will always be the hottest object that is seen in these low-resolution images, isolating the red channel removes a large amount of the noise around the edges of a detected person. Isolating the red channel also removes all the background pixels as well as

the shoulders in the case of close range facial images.

To get the red channel for each pixel, the first two channels that represent the green and blue values are set to zero.

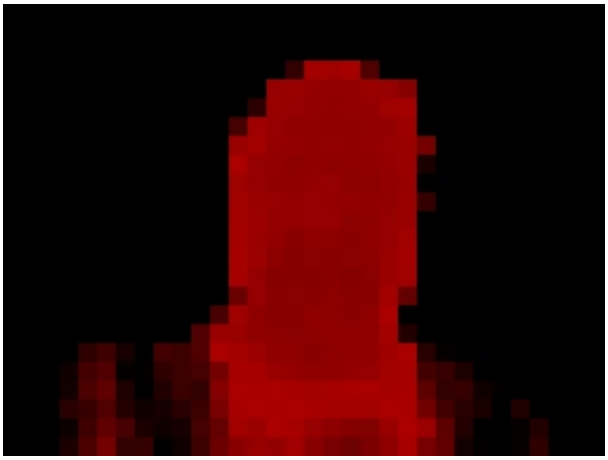

Figure 4.3: Red Channel of a thermal image

4.2.2 Grayscale

The images must then be grayscaled. Grayscaling is a process that collapses the 3 channels of an RGB image into one. This is done using a weighted formula so that each channel is accurately represented.

$$\text{RGB[A] to Gray:} \quad Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Figure 4.4: Grayscale formula

Many useful Computer Vision applications have been designed to work with grayscaled images due to the increased speed and simplicity of having only one channel of data. This includes the Cascade Classifier that will be implemented later.

4.2.3 Create Binary Image

The last step in preprocessing is to create a binary image. A binary image further simplifies the grayscale image into an array of binary values, either 1 or 0. This is represented in an image by either white pixels or black pixels. A grayscale image is converted to a binary image with a short threshold algorithm:

```
for Each Pixel do
    if Pixel.value > Threshold then
        Binary.Pixel = 1
    else
        Binary.Pixel = 0
    end
end
```

Figure 4.5: Algorithm 2: Thresholding

Figure 4.6: A Binary Image

4.2.4 Edge Detection

The newly created binary image acts as our edge data. Edge data is used by boundary chain codes (BCC) [11] to visualise edges in an image. Each edge begins with a specific (row, column) starting pixel in the image. Each subsequent pixel in the chain is represented by a value between 0 and 7, signifying one of the eight possible neighbouring pixels. Drawing each of these edges on a blank image creates an edge image.

4.3 Statistical Pattern Recognition

In Statistical Pattern Recognition [20], a group of N classes is defined based on a set of d-features. In this system, there are two classes: Faces and Bodies. The features that define the classes are Area and Circularity. The area of an object can be calculated by counting all the pixels that encompass the object. Circularity is the measurement of how close to a perfect circle an object is, with 1 being a perfect circle. Circularity requires the area as calculated before and requires the length of the perimeter. Circularity is calculated by the formula:

$$circularity = (4 * pi * area)/d^2, d = perimeterlengthoftheobject$$

As the images created are small, these two features are sufficient to classify the objects discovered. This means that an object in the images can either be a face, a body or nothing of interest. To do this, upper and lower bands are specified for the expected area and circularity of each class.

	Area		Circularity	
Class	Min	Max	Min	Max
Head	135	285	0.6	0.8
Body	35	115	0.13	0.85

With these features, the objects in the thermal images can be classified. Once an object is classified, a bounding rectangle is created and stored. The bounding rectangle will help with both the tracking of an object throughout its time on screen and with the creation of samples for the cascade classifier later.

4.4 Object Tracking

Once an object is identified and classified with Statistical Pattern Recognition, it needs to be tracked for the duration it is on screen. This is to avoid duplicate detection in later frames and potential duplicate discovery in the cascade classifiers.

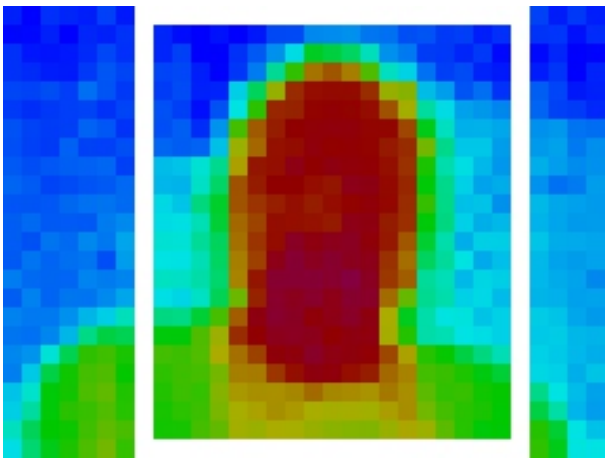

Figure 4.7: A Head Image

To do this two different Python classes are created, a Person class and a Tracker class. These classes have several attributes and functions which will be outlined below.

4.4.1 Person Class

The person class has the following attributes with corresponding getter and setter classes.

Midpoint This is the midpoint of the object found after Statistical Pattern recognition is carried out on the edge data from the binary image.

Position This is the generalised location of the person on screen, if row value of the midpoint is less than half the total image width a person would be considered to be on the left, other wise they are on the right.

Status A persons status is their current pose, can be either forwards, left or right.

Rect A tuple that represents the top left and bottom right points of the bounding rectangle, is used to draw the rectangle on the displayed image.

Label A generic name for this person, which is a number depending on what order the person appeared on screen.

```
for person in tracker.people do
    for mid in newMidpoints do
        if distance(person.midpoint, mid) < Threshold then
            Update person with newest midpoint, position and bounding rectangle ;
            Set mid to 0 ;
            break;
        end
    end
    if person.midpoint != updated then
        remove person
    end
end
for mid in newMidpoints do
    if mid != 0 then
        Create a new Person Object;
        Add tracker.people;
    end
end
```

Figure 4.8: Algorithm 3: Update People

4.4.2 Tracker Class

When the tracker is initialised it contains an empty list of People. And has the following functions:

addPerson Creates a new Person object with the attributes as specified above and adds this object to the trackers list of People

findPosition Finds the location of a given point with respect to the center of the image, if the points row value is less than the half the total length of the image it returns Left, otherwise returns Right

distance Given two points returns the distance between them

updatePeople Given a new list of midpoints and bounding rectangle points, updates the trackers list of people using the following algorithm:

4.5 Cascade Classifier

To act as the pose estimator in this project, a series of cascade classifiers are created, one for each possible direction this system classifies: Forward, left and right. These cascades are then strung together to form a cascade of cascades. The cascades are created in the style of Viola & Jones [1], however these cascades will use Local Binary Pattern (LBP) features [16] as these proved to be more robust than the simpler Haar features. Both the face and body pose classifiers were created with the same methodology. This section will explain the creation of the body pose classifier in detail.

4.5.1 Data Creation

Each cascade requires a large amount of positive samples and negative images. To create a bank of data to be used, thousands of readings were taken with the sensor and the pose in the created image was manually labelled. This allowed for easy division of the data into positive and negative images for training.

To create positive samples, the object detection process as outlined in section 3.2 is used. Rather than drawing the bounding rectangle around the object in a display image, a sub-image of the object is created. It is then grayscaled and saved to a folder. Simultaneously, a file is created that lists the name of every positive sample file, the number of samples found in the image (always 1 in this case), the starting location of its bounding rectangle and the rectangle's width and height. However, as the sub-images only contain the object and have no pixels to be cut by a rectangle, the file specifies the origin as (0,0) and the width and height as the dimensions of the image. Each line in the file would appear in the form:

$$\textit{filename.jpg}, 1, 0, 0, 14, 20$$

Similarly all negative samples are grayscaled and saved in full to a folder, while a file is created with all the filenames.

4.5.2 Cascade Creation

OpenCV has built-in utilities that can create custom cascade classifiers using the data created in the previous section. The 'Create Samples' utility takes the positives samples and creates a vector file that represents each positive sample in a binary format, which will be needed to train the cascade.

Each cascade classifier is a chain of weak classifiers. In each stage, one feature is used to test an image. If the image fails to meet the requirements of the feature, the image fails the

entire cascade. The OpenCV utility "Train Cascade" takes the newly created positive sample vector file, the negative samples and the type of feature that should be used. In this case, LBP. The utility then chooses the best possible feature for each stage to get the highest accuracy. Given the size of these images, the majority of the classifiers had 5 stages.

This process is repeated for all three classifications and the classifiers are chained together based on their accuracy to form a strong cascade of cascades. The cascades can then be stored somewhere in the file system and can be loaded by OpenCV as a custom cascade.

4.6 Interaction

The interaction class is to represent a discovered interaction. To do this, the interaction class measures the length of the conversation and the attention score of both participants in the conversation.

The interaction has three possible statuses, (1) Not started - the people in the interaction have not been classified as facing each other. (2) In progress - an interaction has started, and the system is actively tracking the time of the interaction and the attention scores of both participants. (3) Interaction Over - Both participants have faced forward again or one of them has left the frame, indicating the end of an interaction. With this status, the system starts looking for the start of another interaction.

The start time of the interaction is saved as a time stamp once the interaction begins. When it finishes, the end time is noted, and the total duration of the interaction is calculated by subtracting the start time from the end time. For each frame of the interaction, the pose of each person is taken. If one person is looking directly at the other person in the interaction, the first person's interaction score goes up by 1. This is measured for each frame until the interaction is over.

4.7 System in Action

This section will summarise how all of the components and functions described in this chapter operate together. As the implementation for this project focuses on the interaction categorisation described in the design section, some steps were taken to do pose classification in a live environment by mocking incoming data from an external sensor. The MLX90640 development board used for this project wrote thermal sensor readings to a CSV file on the local machine. To mock the arrival of new data, the CSV was read once a second and checked for new data. The number of lines read on the previous pass would be remembered and for each new pass the difference in total lines against the previously

counted total was calculated and these new lines were read.

For each new line, which represents one frame, an image is created and the preprocessing is carried out as described above, ending with a binary image. That image is used as edge data to create an edge image of all the objects found in the image.

Statistical Pattern Recognition is then used to decide whether each object found in the image is a head, a body or nothing of interest. Following that, the tracker either adds new people, updates existing people or removes anyone who has left the frame.

Next, the image is passed to the pose classifier and each person's pose is classified. Those results are then given to the Interaction class which sets the status of the interaction. If the interaction has already started, the system appraises whether they are facing each other. If the interaction is in progress, then the system checks for the end of an interaction, and then updates the attention scores. If the interaction is over, the system looks for the start of a new interaction.

5 Evaluation

This chapter provides an overview of the metrics that are used to measure the accuracy of the pose classifiers and the metrics used to categorise interactions. This chapter also briefly outlines the mocked scenarios that are used to test the system.

5.1 Evaluation Issues

Given the restrictions on movement and work that have been in place for much of the duration of this project due to the outbreak of COVID-19 [34], it should be noted that testing was limited to only the author as a participant. In order to emulate social interaction in the coming results section, short videos were spliced together to create mocked interactions. These were then played back to the system in real time to act like newly arrived live data.

5.2 Evaluation Metrics

There are two sets of evaluation metrics that will be covered in this section. Firstly, the cascades that make up the pose classifiers need to be evaluated for accuracy. This will allow them to be chained together in the most effective way that will give the most accurate pose estimations.

Secondly, the metrics that are used to categorise an interaction are discussed. These metrics will be used to measure the effectiveness of the system in monitoring an interaction as it takes place.

5.2.1 Cascade Evaluation

Initially, for each pose classifier, the component cascades that evaluate each of the three pose directions must be tested for accuracy. This is to ensure that the overall classifier works as effectively as possible. To evaluate them, each cascade will be tested with a set of labelled test images. From that testing, the following metrics will be calculated:

1. True Positive Rate
2. False Positive Rate
3. True Negative Rate
4. False Negative Rate
5. Accuracy

5.2.2 Interaction Evaluation

As mentioned previously, an interaction will be seen by a system if two people face each other. The interaction will end once both people face forward or if one person leaves the frame. Within that period of interaction, two metrics can be calculated: (1) Length of Interaction - The time in seconds of the interaction, from the first frame two people face one another until the first frame where they both face forward or one person leaves. (2) Attention Score of each participant - The interaction score is the number of frames from the start of the interaction to the end where a participant is directly facing the person they are interacting with.

To test this system, six scenarios will be mocked: three focusing on the head pose classifier and three focusing on the body pose classifier. In each scenario, the time of interaction and the attention scores will be compared against the expected values that have been recorded by manually annotating the video.

The six scenarios will be briefly outlined below, and they will be explained in more detail in the results section.

Scenario 1 - Fixed Poses Two people will be staring at each other, from the entire duration of the interaction.

Scenario 2 - One Fixed, One Changeable One person will stare intensely at another person whose pose keeps changing.

Scenario 3 - Both Changeable - Both peoples poses change throughout the scenario and they briefly interact.

Scenario 4 - Bus Stop Interaction Two people arrive at a bus stop and after initially not seeing each other turn to interact, before ending the interaction by both facing forward again

Scenario 5 - Repeatedly Leaving Frame Two people are interacting until one of them leaves frame, ending the interaction, then they (or someone else, the system does not care) return and leave twice more throughout the scenario.

Scenario 6 - Social Distancing This scenario demonstrate how a system like this could be used to monitor whether people in public spaces are social distancing by measure the distance between them.

6 Results

This chapter will outline the results of the cascade classifiers and the results of the systems performance on 6 test scenarios.

6.1 Cascade Results

This section will cover the specific results of each classifier based on the metrics as outlined in the evaluation section.

6.1.1 Head Pose Cascades

Each of the cascades were tested on the same test data, that began in the form of thermal readings in a CSV file with the last column representing the expected classification result.

Below each cascade is scored on each metric as outlined above and the results are illustrated in a tabular format, with the totals for each metric also displayed in a graph. Each set of results will be briefly discussed.

Forward Cascade	
Metric	Score
True Positive Rate	4%
False Positive Rate	6%
True Negative Rate	99%
False Negative Rate	95%
Accuracy	64%

Table 6.1: Head Pose Forward Cascade Results

The front pose cascade has the lowest accuracy of the 3 head pose cascades at 64%, see Table 6.1. Therefore, it's classification will be the last considered in the classifier, although it's very low false negative rate means that it should avoid misclassifying right and left facing heads.

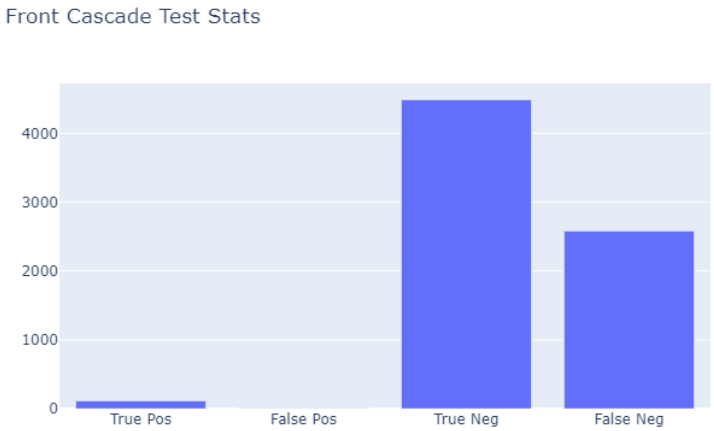

Figure 6.1: Totals for each metric in the forward head pose cascade

Left Cascade	
Metric	Score
True Positive Rate	84%
False Positive Rate	24%
True Negative Rate	88%
False Negative Rate	16%
Accuracy	87%

Table 6.2: Head Pose Left Cascade Results

The left head pose cascade is far stronger in comparison with the forward facing cascade (see Table 6.2). With a high accuracy, low false positive and false negative rates, it is likely to return the correct classification and will have first priority in the classifier.

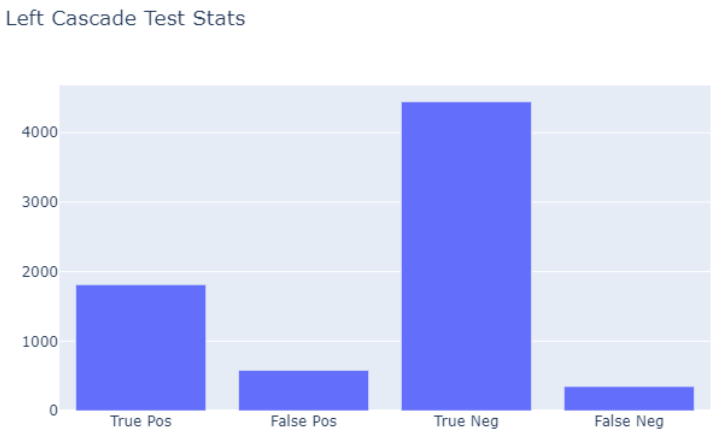

Figure 6.2: Totals for each metric in the left head pose cascade

Right Cascade	
Metric	Score
True Positive Rate	40%
False Positive Rate	1%
True Negative Rate	99%
False Negative Rate	60%
Accuracy	80%

Table 6.3: Head Pose Right Cascade Results

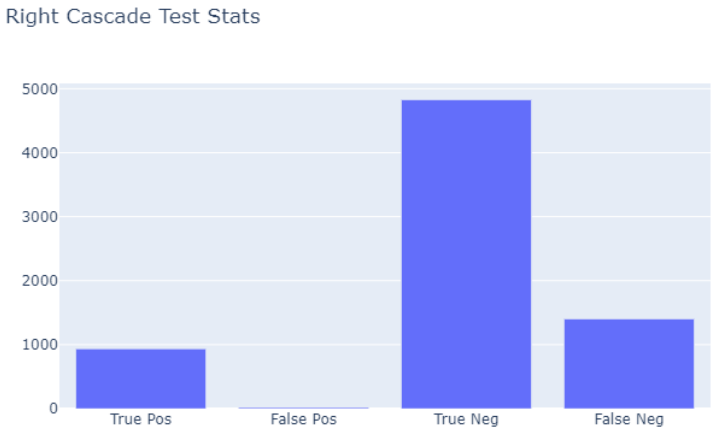

Figure 6.3: Totals for each metric in the right head pose cascade

Similarly the right cascade performs well with a high accuracy (see Table 6.3), however the false negatives mean that it is likely to reject many positive images and if they fall through to the forward cascade are likely to be falsely classified as forward facing.

6.1.2 Body Cascade

Forward Cascade	
Metric	Score
True Positive Rate	89%
False Positive Rate	42%
True Negative Rate	67%
False Negative Rate	11%
Accuracy	74%

Table 6.4: Body Pose Front Cascade Results

The front facing body pose cascade has the opposite issue to it's head pose counterpart, a large amount of false positives (see Table 6.4) mean it is likely to misclassify right or left facing people as front facing. To combat this, the left and right facing cascades are given a higher priority.

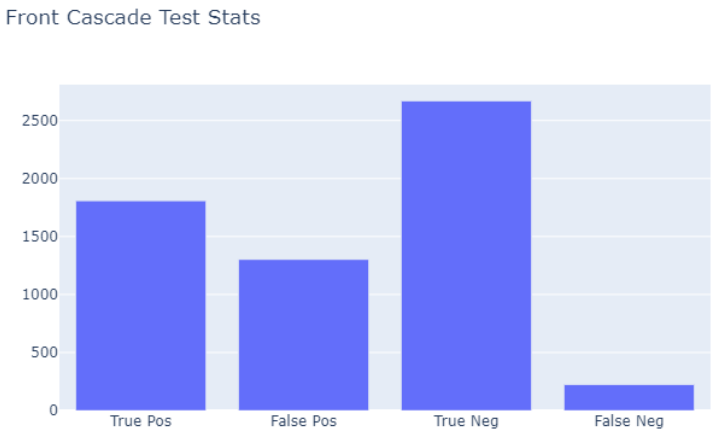

Figure 6.4: Totals for each metric in the front body pose cascade

Left Cascade	
Metric	Score
True Positive Rate	74%
False Positive Rate	49%
True Negative Rate	64%
False Negative Rate	26%
Accuracy	67%

Table 6.5: Body Pose Left Cascade Results

The left pose performs better than the front facing, but is likely to return false positives (see Table 6.5), meaning that any incorrectly classified front facing images will fall through and be misclassified as left facing.

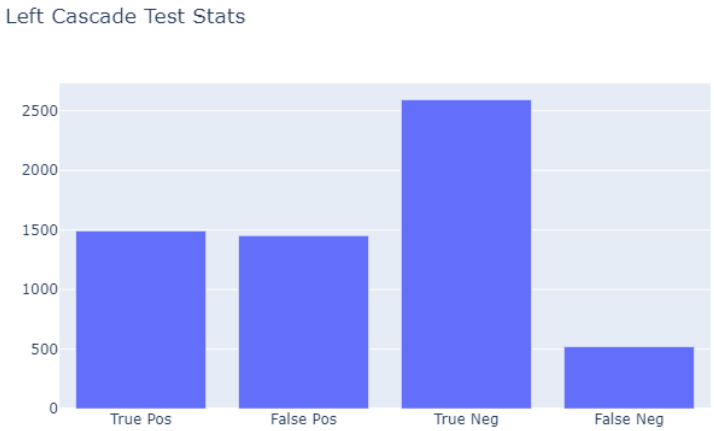

Figure 6.5: Totals for each metric in the left body pose cascade

Right Cascade	
Metric	Score
True Positive Rate	12%
False Positive Rate	29%
True Negative Rate	98%
False Negative Rate	88%
Accuracy	69%

Table 6.6: Body Pose Right Cascade Results

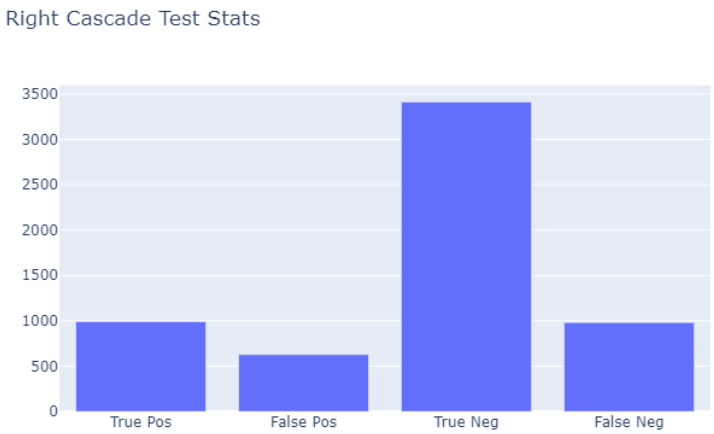

Figure 6.6: Totals for each metric in the right body pose cascade

The right body pose cascade performs the best of the three and first cascade that will be used in the classifier (see Table 6.6).

6.2 Interaction Experiment Results

In this section a series of mocked interaction will be described and the expected results in both time of interaction and the attention score of the participants will be compared with the final result. The description will be accompanied by a series of frames from the test videos that will help to illustrate the scenarios.

6.2.1 Scenario 1 - Fixed Poses

The first scenario is a basic test of the head pose classifier and the attention scores. Two people begin the interaction in intense eye contact and remain this way for the duration of the scenario.

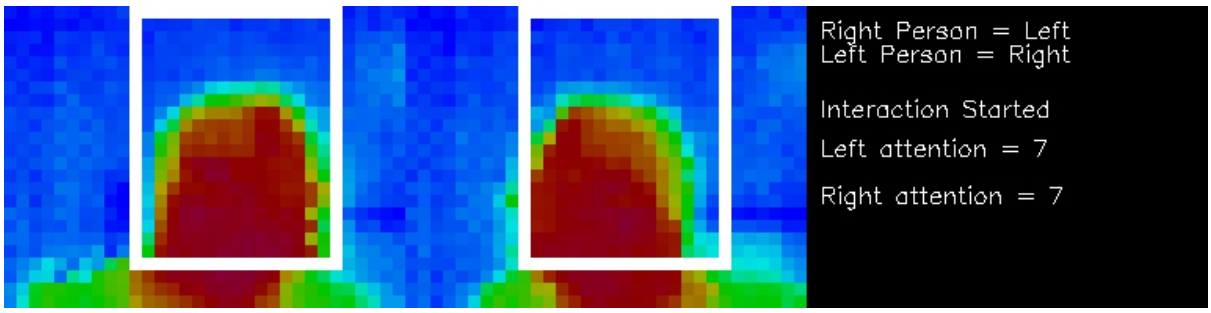

Figure 6.7: Scenario 1 Stage 1 - Interaction started

In the first stage the interaction is immediately detected by the system following the correct pose estimation of both people in the image. The system begins to tally the attention scores for both participants.

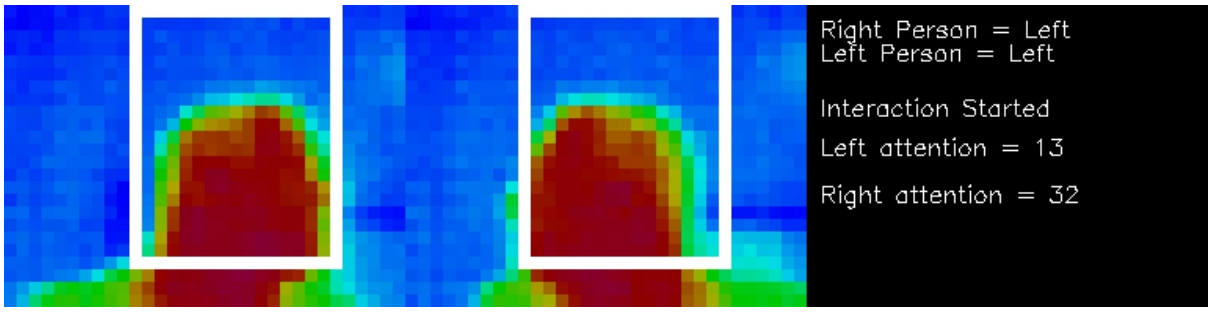

Figure 6.8: Scenario 1 Stage 2 - Attention Score tallies

In this scenario the attention scores should be the same for both participants throughout the scenario as both are facing the same direction for the duration. However as the pose classifier begins to incorrectly classify the attention scores begin to separate. Lastly the system categorises the interaction as over, just before the end of the scenario due to two misclassifications marking both participants as facing forward.

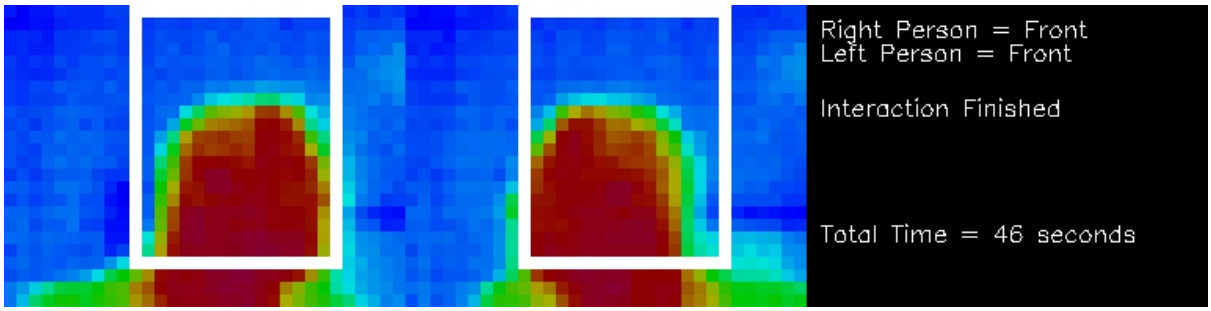

Figure 6.9: Scenario 1 Stage 3 - Interaction ended

Metric	Expected	Recorded
Time Of Interaction	56 seconds	46 seconds
Left Person Attention Score	112	31
Right Person Attention Score	112	57

Table 6.7: Scenario 2 Results

6.2.2 Scenario 2 - One Fixed, One Changeable

Scenario 2 builds on scenario 1, one person will remain facing the same direction for the entire session while the other person moves there to different poses every few seconds. In the first stage the two people are not interacting so the system only estimates their pose.

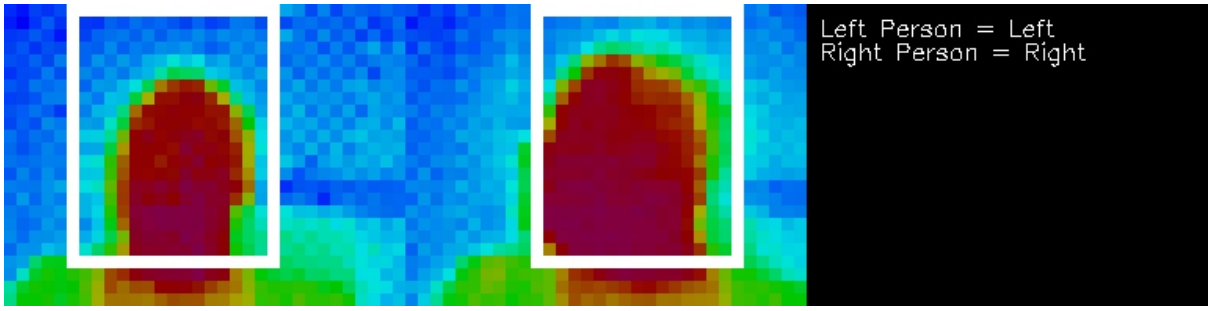

Figure 6.10: Scenario 2 Stage 1 - No Interaction

Next the system detects the interaction starting, however it makes the decision a few frames early, it correctly begins to count the head pose and the difference between the non moving persons attention score compared to the moving head is accurately reflected.

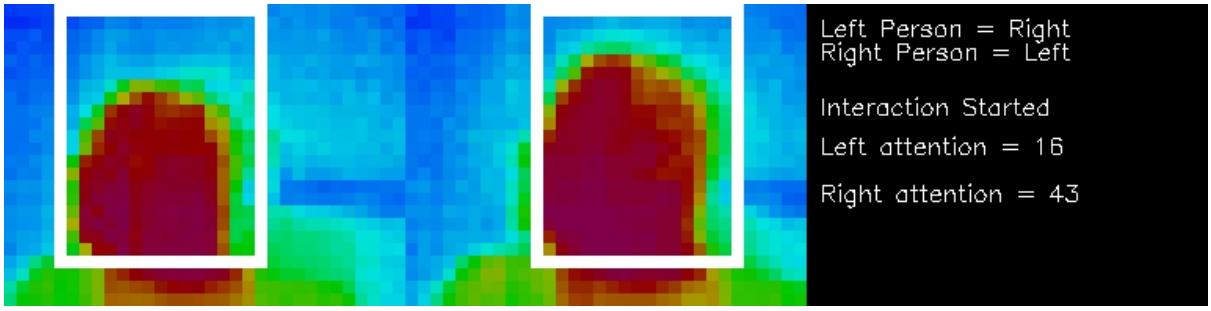

Figure 6.11: Scenario 2 Stage 2 - Attention Score tallies

In this scenario the interaction does not end, a both participants remain in frame and never both look forward at the same time.The system correctly does not stop measuring the attention scores.

Metric	Expected	Recorded
Time Of Interaction	No Ending	No Ending
Left Person Attention Score	40	45
Right Person Attention Score	15	16

Table 6.8: Scenario 2 Results

6.2.3 Scenario 3 - Both Changeable

In the final head pose scenario, both people are freely looking around. They quickly enter into an interaction.

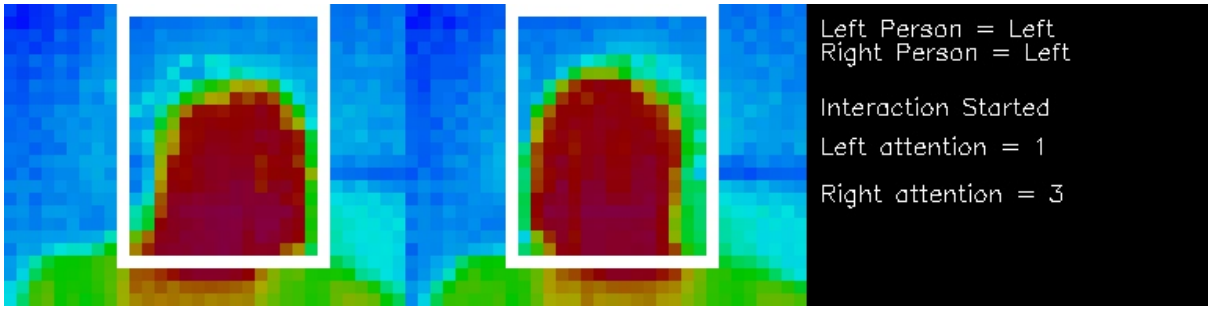

Figure 6.12: Scenario 3 Stage 1 - Interaction Started

However the interaction quickly ends as both people look forward again.

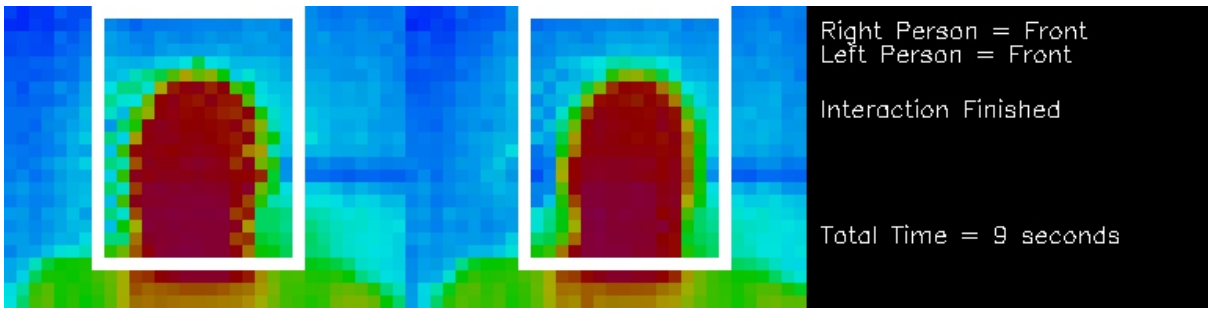

Figure 6.13: Scenario 3 Stage 2 - Interaction Ended

For the remainder of the scenario, both people continue to change head pose, however they never look at each other again and the system correctly reflects that by not registering any further interactions.

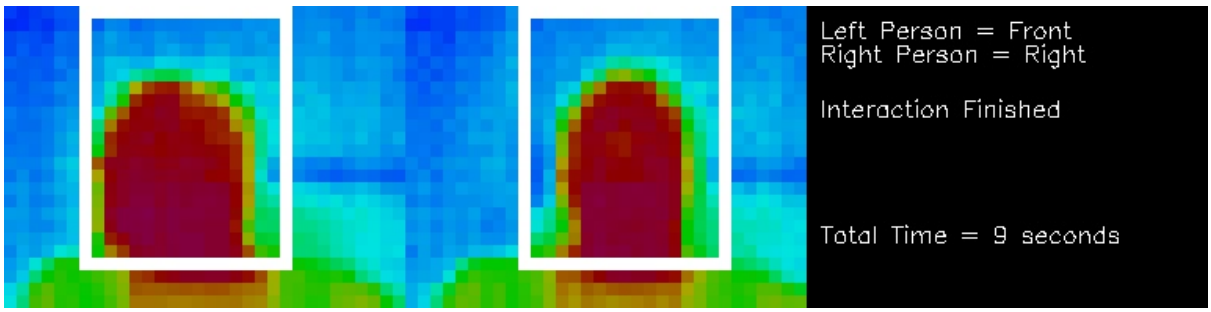

Figure 6.14: Scenario 3 Stage 3 - No further interactions

Metric	Expected	Recorded
Time Of Interaction	9 seconds	9 seconds
Left Person Attention Score	6	4
Right Person Attention Score	12	11

Table 6.9: Scenario 3 Results

6.2.4 Scenario 4 - Bus Stop Interaction

In this scenario two people arrive at a bus stop. Initially both people are facing forward as they have not recognised each other. The system detects both people and tracks them.

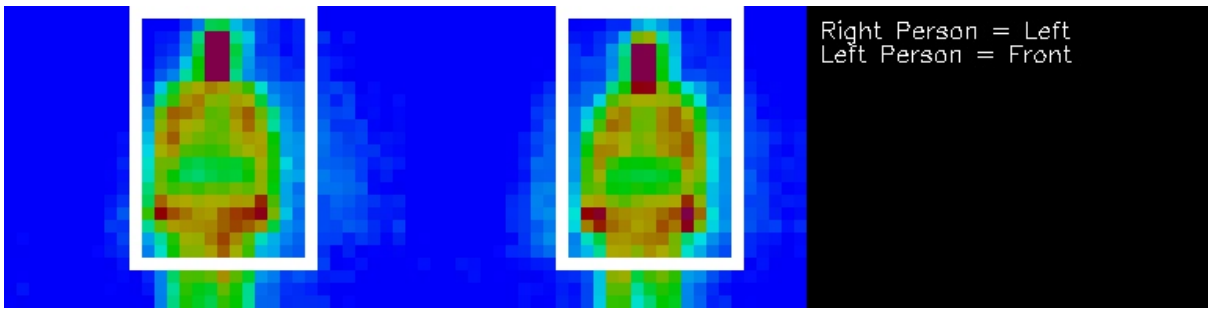

Figure 6.15: Scenario 4 stage 1 - No interaction

Next both people turn to face one another, they system detects both the pose of each person and starts to measure the interaction in terms of time and attention score.

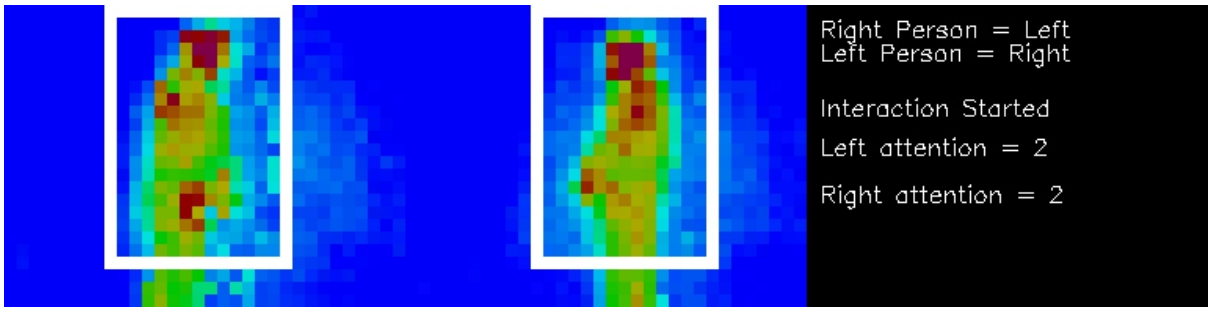

Figure 6.16: Scenario 4 stage 2 - Interaction Started

The two people continue to face each other for 37 seconds, during that time the attention score is increased for a participant for every frame the system classifies them as facing the other person. Once both people return to face forward, the interaction ends.

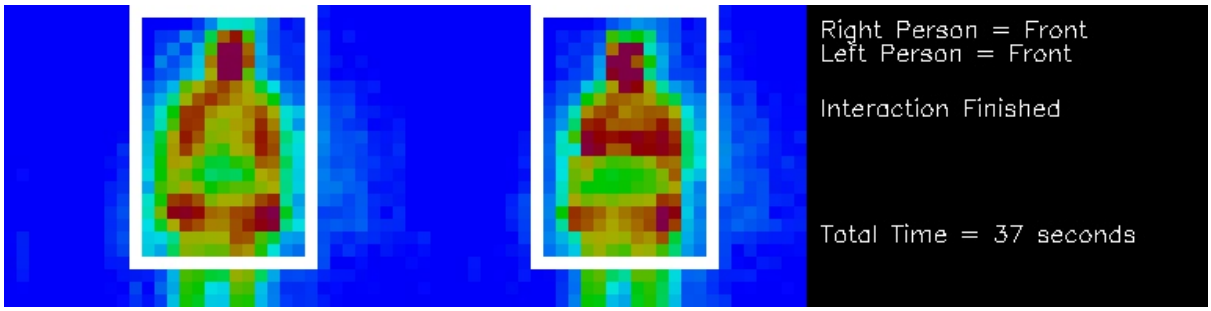

Figure 6.17: Scenario 4 stage 3 - Interaction Ended

Metric	Expected	Recorded
Time Of Interaction	37 seconds	37 seconds
Left Person Attention Score	67	62
Right Person Attention Score	57	58

Table 6.10: Scenario 1 Results

6.2.5 Scenario 5 - Repeatedly Leaving Frame

This scenario test the systems ability to recover to a default state once an interaction has ended, and detect the following interactions. The scenario starts with two people who are interacting almost immediately.

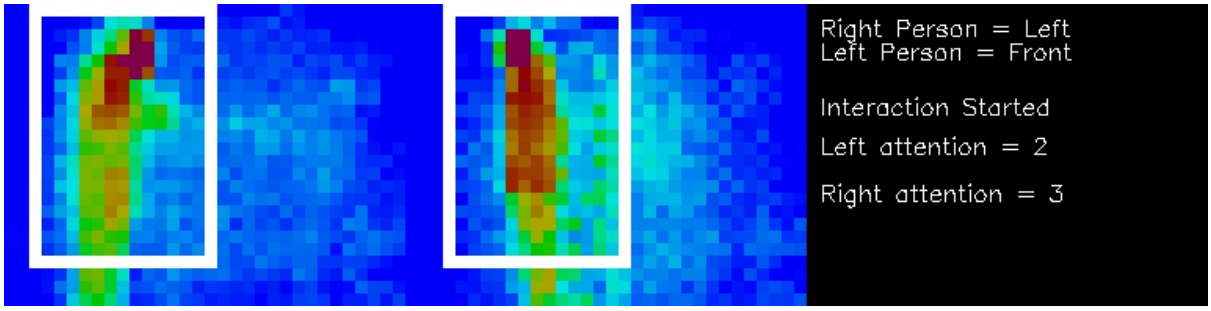

Figure 6.18: Scenario 5 stage 1 - First Interaction Started

This interaction ends quickly when one of the participants leaves frame. The system forgets the person and ends the interaction.

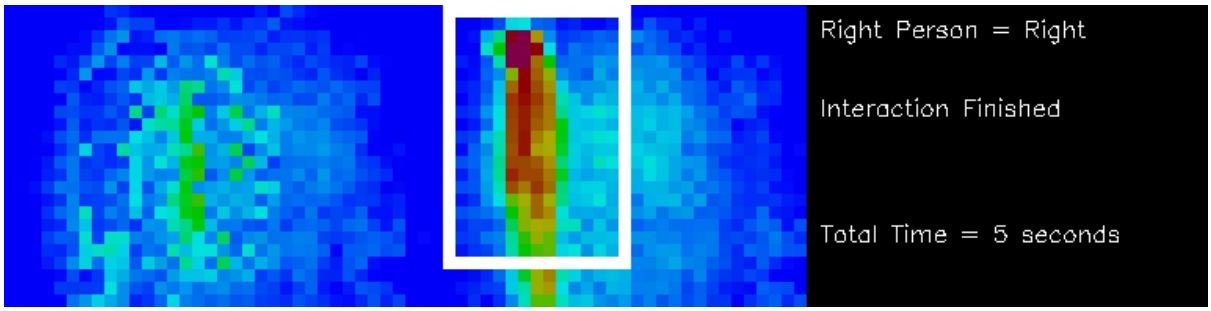

Figure 6.19: Scenario 5 stage 2 - First Interaction Ended

Next the person (or a different person, the system doesn't know) returns and is detected. And an interaction starts again. The system catches the interaction near the end, before one of the participants leaves again.

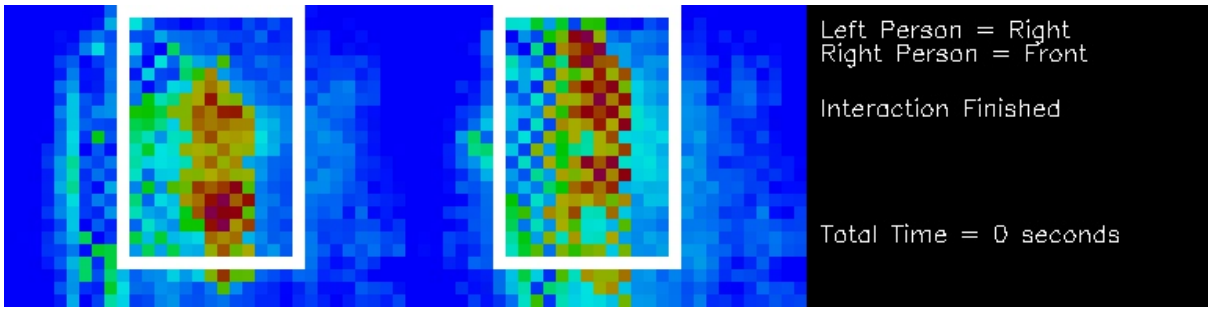

Figure 6.20: Scenario 5 stage 3 - Second Interaction Ends as one person leaves

Another person comes into frame and is detected, and a third interaction starts. Again the interaction is caught right at the end, meaning most of the attention score is missed.

	Metric	Expected	Result
Interaction 1	Time Of Interaction	6 seconds	5 seconds
	Left Person Attention Score	15	3
	Right Person Attention Score	18	5
Interaction 2	Time Of Interaction	5 seconds	0.5 seconds
	Left Person Attention Score	15	1
	Right Person Attention Score	20	1
Interaction 3	Time Of Interaction	6 seconds	1 second
	Left Person Attention Score	17	1
	Right Person Attention Score	22	2

Table 6.11: Scenario 5 Results

6.2.6 Scenario 6 - Social Distancing

At the time of writing, the COVID-19 [34] pandemic has put the subject of social distancing [35] in the forefront of everyone's mind. This scenario mocks what a system to monitor social distancing could look like in simple terms. To do this, the system is configured slightly different and an additional assumption is made. The assumption is that an interaction is occurring from the time it is detected and the system will not attempt to find the end of the interaction, instead the system will measure the distance between the participants at each frame and state whether they are social distancing.

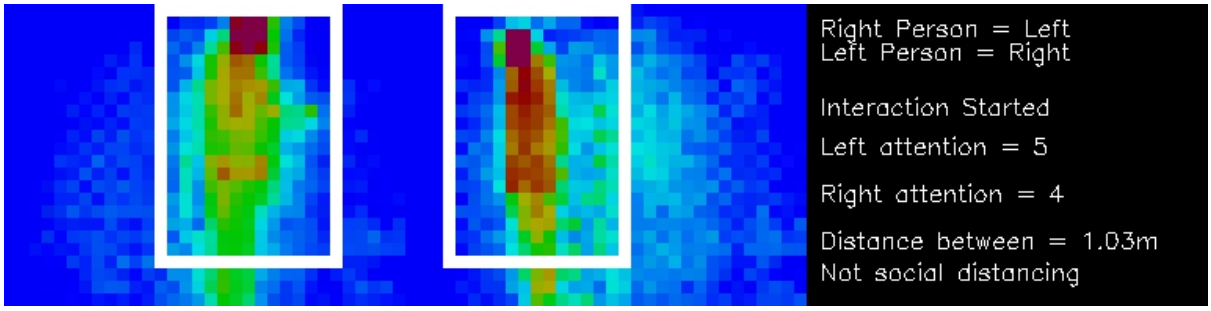

Figure 6.21: Scenario 6 stage 1 - Interaction starts, not social distancing

At stage one the interaction has started, and both people are standing just over 1 metre from each other and not following social distancing guidelines, which is noted by the system.

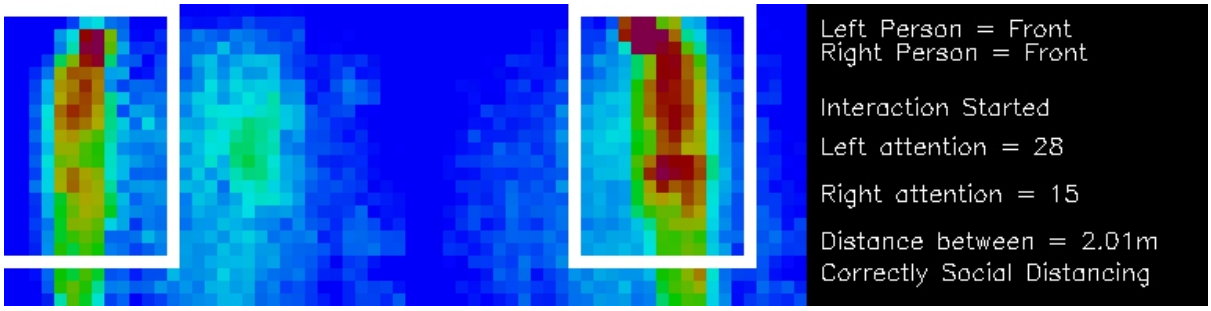

Figure 6.22: Scenario 6 stage 2 - Interaction continues, now participants are social distancing

After 30 seconds, each participant takes a step backwards, extending the distance between them to over 2 metres and the systems notes that they are now social distancing.

6.3 Discussion

These results show the best and worst of the system in action, with some scenarios being broadly considered successful and others not. This section will divide the scenarios by their success in the metrics discussed in the evaluation and see what aspects of the system work well and what aspects leave more to be desired.

6.3.1 Unsuccessful Scenarios

Scenarios 1 can be considered an unsuccessful scenario. Despite it being the simplest scenarios, it shows the shortcomings of the pose detectors clearly. The total interaction scores for both users recorded is only 40% of the expected values. This scenario shows the likelihood that the the pose classifier will misclassify, even on images with very minor pose variations.

6.3.2 Partially Successful Scenarios

Scenario 5 is a partially successful scenario. It shows the trackers working as intended, following people while they appear in frame and as they move around, but forgetting them once they leave frame. It also shows the system returning to a default state once and interaction ends, ready to detect another interaction. The system ultimately correctly detects the 3 interactions that take place in the scenario.

What lets Scenario 5 down is the attention scores. Scenario 5 only records 12% of the total expected attention score over all 3 interactions, which shows that despite all interactions being successfully identified, nothing substantive could be taken from them.

6.3.3 Successful Scenarios

Scenarios 2, 3, 4 and 6 can be considered successful scenarios. Scenarios 2, 3, and 4 all record total attention scores of within 20% of the total expected values, while also reporting correct interaction times (or in the case of scenario 2 correctly not detecting an end to the interaction. Scenario 4 is probably the scenario that best showcases the system. Two people arrive in frame, are tracked until they start interacting, the system records 96% of the expected attention score in total and records the correct interaction time of 37 seconds.

The parameters for success in scenario 6 were different to previous scenarios. The attention score and interaction time were not accurately measured here as the system never checks for the end of an interaction, instead, the system measures the distance between the participants. This relies more on the consistency of the tracker to correctly monitor the

position of both people as they move across the frame to a socially distant position.

The Statistical Pattern Recognition and the tracker work almost flawlessly in these scenarios, which is due in part to the simplicity of the images and the stringent parameters of the scenarios.

It is hard to say which of the two pose classifiers work best as each have good example (Scenarios 2, 3 and 4) and bad examples (Scenarios 1 and 5). Despite the head pose classifier having better accuracy for each individual cascade, it does appear to struggle more in the scenarios than the body pose classifier, which suggests the way the cascades are chained together may be the issue. One thing that can be said is both could do with improvement to increase consistency.

7 Conclusion

At the beginning of this project, three aims were outlined to ascertain whether social interactions could be detected in low-resolution thermal images: detecting people in low resolution images, building pose classifiers and identifying and tracking the interactions.

Detecting objects was accomplished using standard Computer Vision methods such as grayscaling, binary thresholding and visualising edges with boundary chain codes. Statistical Pattern Recognition allowed the system to classify objects found in the images into two types of classes: heads or bodies.

Two pose classifiers were built, each made of three-component cascades to represent three possible poses: forward, left and right. The component cascades were then chained together to form a classifier. These classifiers had mixed results. This was due, in part, to the minimal amount of data available in small images. A flaw in cascades of this type also contributed: when the search object takes up a significant portion of the image, the cascade is more likely to misclassify.

Interactions were defined as two people facing one another. An interaction lasted until both participants faced forward or one person left the frame. They were evaluated on a set of test scenarios, where the time of interaction was recorded and compared to the expected time. Additionally, during the interaction, each frame a participant spent looking at the other person increased their attention score. This gives an overall sense of how intense the conversation was and how invested each participant was during it. Although the overall attention scores often failed to match the expected values, a general impression of which participant was more active in the conversation comes across in the difference between each person's attention score.

Overall, this system stands as a proof of concept demonstrating that it is possible to identify social interaction with low-resolution thermal imagery.

7.1 Future Work

There is a plethora of future work that can be undertaken using the theory presented in this project as discussed below. Some possibilities involve improving the existing system while others discuss how the project can be extended.

7.1.1 Deploy Higher Resolution Cameras

This project explored the limits of what could be done with small modular hardware in the form of the MLX90640 thermal sensor. An extension of this project would be to deploy a higher resolution camera. This could provide more features of a person's body or head to be used in pose classification. However, a higher resolution camera could also jeopardise the privacy of the people detected by the system, as having more features could increase the ability to recognise them. This means there is also an upper limit to be discovered as to what resolution of camera can lead to the most accurate pose classification without compromising the anonymity of the people detected. The use of higher resolution is the gateway to considering many other improvements and adaptions.

7.1.2 Adapt Deep Learning Methods

Many of the body pose methods proposed in research use body part detection combined with deep learning models to suggest a person's orientation. Using a higher resolution camera may enable enough features to be extracted from a detected person to adapt a deep learning model, perhaps simplifying it if needed, to improve the accuracy of the pose classifier and the system as a whole.

7.1.3 Consider Multiple Interactions and Group dynamics

This project limited the scope of interactions to between two people. A minor increase in camera resolution could allow the system to consider more than two people involved in interactions. This would change the potential system states from a binary operation, where an interaction is either occurring or not, to a more dynamic environment where multiple interactions are occurring or a group of people are interacting with each other. This would also increase the dimensions of the attention scores, so the attention paid to each member of the group could be identified and some group dynamics could be inferred.

7.1.4 Increase Hardware Diversity

An alternative would be to deploy more sensors and sensors of different types. Deploying more cameras could create a more 3-dimensional view of the space. It would improve pose

estimation, as there would be multiple sources to draw a conclusion from. More cameras would also improve the system's ability to estimate a person's location and the distance between people.

An ultrasonic sensor could also be deployed. Modular ultrasonic sensors are now available in very similar setups and price ranges as thermal sensors. Combining a thermal sensor with an ultrasonic sensor on one raspberry pi, for example, would enable better object detection, location and tracking without the need to deploy a second camera.

A motion detector could be deployed to improve the tracker. When the motion detector is not active, then the trackers may not need to actively update, which would improve fringe cases where the Statistical Pattern Recognition fails. Alternatively, the system could only search for interactions involving people moving across frame.

7.1.5 Social Distancing

At the time of writing, the outbreak of COVID-19 [34] has placed an emphasis on social distancing [35]. This is the concept whereby people need to stay at least 2 metres from each other while outdoors and in social spaces. Although a simple version of how the system could detect social distancing was implemented and test in Scenario 6, combined with the ground truth of the space it is deployed in, or other sensors that would help to measure the dimensions of the space, a more comprehensive social distancing system could be created.

Bibliography

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, Conference Proceedings, pp. I–I.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, Conference Proceedings, pp. 886–893 vol. 1.

[3] J. W. Davis and M. A. Keck, "A two-stage template approach to person detection in thermal imagery," in *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*, vol. 1, 2005, Conference Proceedings, pp. 364–369.

[4] C. Chen, R. C. Ugarte, C. Wu, and H. Aghajan, "Discovering social interactions in real work environments," in *Face and Gesture 2011*, 2011, Conference Proceedings, pp. 933–938.

[5] M. Trier and A. Bobrik, "Social search: Exploring and searching social architectures in digital networks," *IEEE Internet Computing*, vol. 13, no. 2, pp. 51–59, 2009.

[6] L. Cui, G. Xie, S. Yu, X. Zhai, and L. Gao, "An inherent property-based rumor dissemination model in online social networks," *IEEE Networking Letters*, vol. 2, no. 1, pp. 43–46, 2020.

[7] J. Zhang, R. Zhang, Y. Zhang, and G. Yan, "The rise of social botnets: Attacks and countermeasures," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1068–1082, 2018.

[8] RS-PRO. Rs pro dt-9875 thermal imaging camera datasheet. https://docs.rs-online.com/2f31/0900766b81692851.pdf.

[9] Adafruit. (2020) Infrared array sensor "grid-eye" datasheet. https://cdn-learn.adafruit.com/assets/assets/000/043/261/original/Grid-EYE_SPECIFICATIONS%28Reference%29.pdf?1498680225.

[10] K. Lewin, "Action research and minority problems," *Journal of social issues*, vol. 2, no. 4, pp. 34–46, 1946.

[11] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 2, pp. 260–268, 1961.

[12] K. Reese, Y. Zheng, and A. Elmaghraby, "A comparison of face detection algorithms in visible and thermal spectrums," in *Int'l Conf. on Advances in Computer Science and Application*. Citeseer, 2012, Conference Proceedings.

[13] H. Sahoolizadeh, D. Sarikhanimoghadam, and H. Dehghani, "Face detection using gabor wavelets and neural networks," *World Academy of Science, Engineering and Technology*, vol. 45, pp. 552–554, 2008.

[14] Y. Zheng, "A novel thermal face recognition approach using face pattern words," in *Biometric Technology for Human Identification VII*, vol. 7667. International Society for Optics and Photonics, 2010, Conference Proceedings, p. 766703.

[15] P. Rudol and P. Doherty, "Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery," in *2008 IEEE Aerospace Conference*, 2008, Conference Proceedings, pp. 1–8.

[16] M. Chao, T. Ngo Thanh, U. Hideaki, N. Hajime, S. Atsushi, and T. Rin-ichiro, "Adapting local features for face detection in thermal image," *Sensors (14248220)*, vol. 17, no. 12, p. 2741, 2017.

[17] F. Suard, A. Rakotomamonjy, A. Bensrhair, and A. Broggi, "Pedestrian detection using infrared images and histograms of oriented gradients," in *2006 IEEE Intelligent Vehicles Symposium*, 2006, Conference Proceedings, pp. 206–212.

[18] R. Miezianko and D. Pokrajac, "People detection in low resolution infrared videos," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2008, Conference Proceedings, pp. 1–6.

[19] R. Brunelli, *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

[20] A. K. Jain, R. P. W. Duin, and M. Jianchang, "Statistical pattern recognition: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[21] K. a. Dawson-Howe, "A practical introduction to computer vision with opencv / kenneth dawson-howe," pp. 0–0, Wed Jan 01 00:00:00 GMT 2014 2014.

[22] X. Ren, "Finding people in archive films through tracking," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, Conference Proceedings, pp. 1–8.

[23] R. W. A. Saputra, B. S. B. Dewantara, and D. Pramadihanto, "Human body's orientation estimation based on depth image," in *2019 International Electronics Symposium (IES)*, 2019, Conference Proceedings, pp. 266–271.

[24] T. Golda, T. Kalb, A. Schumann, and J. Beyerer, "Human pose estimation for real-world crowded scenarios," in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2019, Conference Proceedings, pp. 1–8.

[25] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 607–626, 2009.

[26] M. Bingpeng, Z. Wenchao, S. Shiguang, C. Xilin, and G. Wen, "Robust head pose estimation using lgbp," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 2, 2006, Conference Proceedings, pp. 512–515.

[27] Y. Chen, Y. Tian, and M. He, "Monocular human pose estimation: A survey of deep learning-based methods," *Computer Vision and Image Understanding*, vol. 192, p. 102897, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314219301778

[28] A. Cosma, I. E. Radoi, and V. Radu, "Camloc: Pedestrian location estimation through body pose estimation on smart cameras," in *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2019, Conference Proceedings, pp. 1–8.

[29] L. Tian, D. Duan, J. Cui, L. Wang, H. Zha, and H. Aghajan, "Video based children's social behavior classification in peer-play scenarios," in *2013 2nd IAPR Asian Conference on Pattern Recognition*, 2013, Conference Proceedings, pp. 770–774.

[30] Melexis. (2020) Mlx90640 datasheet. https://www.melexis.com/en/documents/documentation/datasheets/datasheet-mlx90640.

[31] OpenCV. (2020) Opencv. https://opencv.org.

[32] E. Rescorla and B. Korver. (2003) Guidleines for writing rfc text on security considerations. https://tools.ietf.org/html/rfc3552.

[33] A. Cooper, H. Tschofenig, B. Aboba, J. Peterson, J. Morris, M. Hansen, and R. Smith. (2013) Privacy considerations for internet protocols. https://tools.ietf.org/html/rfc6973.

[34] W. H. Organisation. (2020) Coronavirus disease (covid-19) pandemic. https://www.who.int/emergencies/diseases/novel-coronavirus-2019.

[35] ——. (2020) Coronavirus disease (covid-19) advice for the public. https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public.

A1 Appendix

A1.1 Python Classes

A1.1.1 Person Class

```
class Person:

    def __init__(self, midpoint, position, status, rect, label):
        self.midpoint = midpoint
        self.position = position
        self.status = status
        self.rect = rect
        self.label = label

    def setMidpoint(self, midpoint):
        self.midpoint = midpoint

    def getMidpoint(self):
        return self.midpoint

    def setPosition(self, position):
        self.position = position

    def getPosition(self):
        return self.position

    def setStatus(self, status):
        self.status = status

    def getStatus(self):
        return self.status
    def setRect(self, rect):
        self.rect = rect

    def getRect(self):
        return self.rect

    def setLabel(self, label):
        self.label = label

    def getLabel(self):
        return self.label
```

Figure A1.1: Person Class

A1.1.2 Tracker Class

```
class tracker:

    def __init__(self, imgX, imgY):
        self.imgX = imgX
        self.imgY = imgY
        self.People = []

    def addPerson(self, midpoint, position, status, rect, label):
        newPerson = Person(midpoint, position, -1, rect, label)
        self.People.append(newPerson)

    def distance(self, point1, point2):
        dist = (point2[0]- point1[0])**2 + (point2[1] - point1[1])**2
        dist = math.sqrt(dist)
        return dist


    def findPosition(self, midpoint):
        if midpoint[0] < (self.imgX/2):
            return "Left"
        elif midpoint[0] > (self.imgX/2):
            return "Right"

    def getPeople(self):
        return self.People

    def updatePeople(self,midpoints, rects):
        for i in range(len(self.People)):
            removeList = []
            updated = False
            for j in range(len(midpoints)):
                if updated == False and midpoints[j] != 0:
                    dist = self.distance(self.People[i].midpoint, midpoints[j])
                    if int(dist) <= 50:
                        #same Person
                        self.People[i].setMidpoint(midpoints[j])
                        newPos = self.findPosition(midpoints[j])
                        self.People[i].setPosition(newPos)
                        self.People[i].setRect(rects[j])
                        midpoints[j] = 0
                        updated = True
            if updated == False:
                removeList.append(i)
        for i in removeList:
            self.People.pop(i)
        for i in range(len(midpoints)):
            if midpoints[i] != 0:
                pos = self.findPosition(midpoints[i])
                status = ""
                label = ""
                newPerson = Person(midpoints[i], pos, -1, rects[i], label)
                self.People.append(newPerson)

    def getDistanceBetween(self):
        pixelFrameLength = 64
        cmFrameLength = 280
        person1 = self.People[0]
        person2 = self.People[1]
        dist = self.distance(person1.midpoint, person2.midpoint)
        dist = (dist/pixelFrameLength) * cmFrameLength
        return dist/100
```

Figure A1.2: Tracker Class

A1.1.3 Create Images

```
from colour import Color
import numpy as np
import math
import time
import cv2
import csv
import sys

class createImages:
    global MINTEMP, MAXTEMP, COLORDEPTH, colors
    def __init__(self, minTemp, maxTemp, imgX, imgY):

        self.MINTEMP = minTemp

        self.MAXTEMP = maxTemp

        self.imgX = imgX

        self.imgY = imgY

    COLORDEPTH = 1024
    blue = Color("indigo")
    colors = list(blue.range_to(Color("red"), COLORDEPTH))
    colors = [(int(c.red * 255), int(c.green * 255), int(c.blue * 255)) for c in colors]

    def map_value(self, x, in_min, in_max, out_min, out_max):
        return (float(x) - float(in_min)) * (float(out_max - out_min)) / (float(in_max) - float(in_min)) + float(out_min)

    def get_img_data(self, row):
        img = np.zeros((self.imgY, self.imgX, 3), np.uint8)
        row = np.reshape(row, (self.imgY, self.imgX))
        for x in range(len(row)):
            for y in range(len(row[x])):
                val = row[x][y]
                if float(val) < float(self.MINTEMP):
                    val = self.MINTEMP
                elif float(val) > float(self.MAXTEMP):
                    val = self.MAXTEMP
                colourIndex = self.map_value(val, self.MINTEMP, self.MAXTEMP, 0, COLORDEPTH - 1)
                img[x, y] = colors[(COLORDEPTH-1) - int(round(colourIndex))]
        return img

    def scale_img(self, img, scale):
        width = int(img.shape[1] * scale / 100)
        height = int(img.shape[0] * scale / 100)
        dim = (width, height)
        resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
        return resized
    def show_img(self, img):
        cv2.imshow('Img', img)
        cv2.waitKey(1000)
        cv2.destroyAllWindows()
```

Figure A1.3: Create Images Part 1

```
def find(self, contours, imgCopy):
    imgY, imgX, dim = imgCopy.shape
    objectList = []
    midpoints = []
    objectType = -1 #0 = head, 1 = body
    headMinArea = 135
    headMaxArea = 285
    headMinCirc = 0.6
    headMaxCirc = 0.8

    bodyMinArea = 18
    bodyMaxArea = 115
    bodyMinCirc = 0.13
    bodyMaxCirc = 0.85
    for cnt in contours:
        img = np.zeros((imgY, imgX, 3), np.uint8)
        img = cv2.drawContours(img, [cnt], 0, (0,255,0), 3)
        area = cv2.contourArea(cnt)
        if headMinArea < area < headMaxArea:
            perimeter = cv2.arcLength(cnt, True)
            circularity = (4*math.pi*area)/(perimeter*perimeter)
            if headMinCirc < circularity < headMaxCirc:
                objectType = 0
        if bodyMinArea < area < bodyMaxArea:
            perimeter = cv2.arcLength(cnt, True)
            circularity = (4*math.pi*area)/(perimeter*perimeter)
            if bodyMinCirc < circularity < bodyMaxCirc:
                objectType = 1
        if objectType != -1:
            x, y, w, h = cv2.boundingRect(cnt)
            rectangleArea = w*h
            midpoint = ((x+(x+w))/2, (y+(y+h))/2)
            duplicate = False
            for mid in midpoints:
                dist = self.distance(midpoint, mid)
                if dist < 20:
                    duplicate = True
            if duplicate == False:
                rectWidth = 8
                if objectType == 0:
                    rectWidth = 7
                midpoints.append(midpoint)
                rect1 = (midpoint[0] - rectWidth , 0)
                rect2 = (midpoint[0] + rectWidth , imgY-4)
                objectList.append((rect1, rect2))
    return (objectType,objectList, midpoints)
```

Figure A1.4: Create Images Part 2

```
def get_objects(self, img):
    imgCopy = img.copy()
    #get red channel
    img[:,:,0] = 0
    img[:,:,1] = 0
    redCopy = img.copy()
    gImg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret,bImg = cv2.threshold(gImg, 35, 255, cv2.THRESH_BINARY)
    bImgCopy = bImg.copy()
    contours, hierarchy = cv2.findContours(bImg, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    if len(contours) > 0:
        result = self.find(contours, imgCopy)
        return result
    return ([],[], [])
```

Figure A1.5: Create Images Part 3