HUMAN DETECTION THROUGH SENSOR FUSION AND CONVOLUTIONAL NEURAL NETWORKS

A thesis presented in fulfilment of the requirements for the degree of

Master of Engineering

in

Mechatronics

at Massey University, Manawatu

New Zealand

Isaac Laurie Williams

2023

Summary

Safety practices in industrial environments can be improved by observing patterns of close calls or accidents. Machine detection of humans is one method that can be used to collect data for this purpose, but it has challenges achieving reliable results in complex and varied environments. Sonasafe, a company manufacturing and managing proximity safety systems for industrial environments, is looking to advance their capability to detect people. This thesis is concerned with object detection, with a specific focus on increasing the accuracy of human detection in industrial environments.

Object detection requires the identification of specific, distinguishable features that can be distinctively associated with the object being detected, with the additional challenge of determining its location. The distinctiveness of an object's features is determined by the type of object, its surroundings, and the sensor being used to detect the object.

This study began with an evaluation of sensors that could contribute to distinguishing identifying human features and the mitigation of negative environmental conditions. The senses selected for testing were a standard visual camera for general feature recognition, an infrared camera for temperature features to mitigate the effects of dark environments, along with a millimetre wave (mmWave) radar to enhance location accuracy. This study reviewed the fusion of these sensors in Convolution Neural Networks to determine their respective performance in identifying different features when detecting a person in a complex environment.

The study found that for a simple environment, with little noise and uniform data, single sensors had a higher rate of success, but that as the environment became more complex with more variability, the fusion of all three sensors produced more accurate results. Overall sensor fusion was seen to improve the accuracy of human detection in more complex industrial environments.

Acknowledgments

I would like to thank my supervisors, Morio Fukuoka and Huub Bakker for providing guidance and feedback throughout this project. I would also like to acknowledge SonaSafe and specifically Jacques Johnston as project sponsors and for providing background information on SonaSafe, their existing systems, and operating environments. Thanks also to my wife Mary-Anne who I value for her input in editing my thesis and assisting with data collection. I would also like to acknowledge Linda Evers and Jack Williams for assisting with the data collection and Emily Evers for assisting with a final proofread.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
CNN	Convolution Neural Networks
FOV	Field of View
ML	Machine Learning
mmWave	millimetre wave (radar)
PDU	Personal Detection Unit
RCS	Radar Cross Section
SDK	Software Development Kits
SOP	Standard Operating Procedure

Chapter 1
Introduction

1.1 Background

Sonasafe is a company located in Tauranga, New Zealand that offers an advanced proximity warning system to a range of industrial workplaces, enabling them to "identify and manage risk around mobile plants" [1]. Sonasafe's aims are a combination of providing immediate operational safety and collecting data for the improvement of a client's Standard Operating Procedures (SOPs). The standard Sonasafe product is generally fitted to industrial vehicles such as forklifts and excavators [1].

Sonasafe's system employs detection methods that combine sonar, radio, GPS, and Personal Detection Units (PDUs) to implement a robust 20m safety perimeter around vehicles inside of which any people are detected. The PDUs operate in conjunction with the sonar system to accurately determine the location and provide identification information of the person wearing the PDU. The system can also incorporate programmed areas as no-go or restricted based on GPS location. As a safety system Sonasafe's system requires a high level of accuracy and reliability. However, a current limitation of the Sonasafe system is that it requires personnel to wear a PDU. Pedestrians that enter the worksite without this device, employees not adhering to the system protocols, or faulty PDUs will not be detected, impacting the reliability of the system in some isolated situations [2].

Sonasafe proposed this master's project to investigate and develop technology to advance the detection of people operating in industrial environments without dependence on secondary devices such as PDUs. Sonasafe has made some progress with Machine Learning (ML) using visual cameras for detecting people which was used as the starting point for this research.

An initial review of literature and existing systems considered principles of object detection, the impacts environments have on detection, and how different sensors detect varying aspects of an environment, such as light, sound or range. For example, 2D cameras capture more detailed visual information of an environment, while a sensor like radar will capture spatial proximity. This review indicated that the complex environments Sonasafe operates within negatively impact the performance of conventional single-sensor detection systems. The review concluded that fusing multiple sensor types would likely improve the accuracy of detection in complex environments to bridge the Sonasafe functionality gap.

1.2 Aim of Study

The aim of this research is to determine how a fusion of different sensor types (sensor fusion) with Convolutional Neural Networks (CNN) analysis can improve the accuracy of human detection in complex industrial environments. The effectiveness of different sensors and combinations of sensors is to be investigated by experimental trials.

1.3 Thesis Outline

This thesis is divided into the following chapters: Introduction, Literature Review, Methodology, Results and Discussion, and Conclusion. The thesis begins with a literature review that considers principles of object detection, impacts environments have on detection, and how data types contribute to detection in different environments. This review indicated that the complex environments Sonasafe operates within negatively impact the performance of conventional single-sensor detection systems. The review concludes that fusing multiple sensor types would likely improve the accuracy of detection in complex environments to bridge the Sonasafe functionality gap.

The methodology discusses the processes used to collect the data, train the models and analyse model performance. It includes discussion regarding the selection and configuration of the sensors, capture and processing of three separate data sets of images, as well as the structure and configuration of the models.

The results and discussion section contains the results obtained from three progressive tests conducted throughout this study. Data Set 1 was used to determine the direction of the research through establishing model structures and determining baseline sensor performance and limitations. This provided the foundation for the Data Set 2 tests which primarily focused on enlarging the data set to obtain results more representative of the actual sensor and model performance. These results in turn highlighted aspects that were missing from the data sets. From this, Data Set 3 was developed to more accurately simulate a complex industrial environment to test and evaluate the hypothesis that the accuracy of human detection through CNNs increases through the use of sensor fusion. The test results for each data set were evaluated to see how the sensors performed individually and in fusion, how the different data sets or environments affected the results, and how the fusion models performed compared to the single sensor models. This is followed by a conclusion to summarise the findings.

Chapter 2
Literature Review

2.1 Object Detection

Object detection is knowing both where an object is (location) and what it is (classification). Both functions have their own challenges and are easier or harder to mitigate depending on the application and sensor type. In some applications only one aspect of object detection needs to be known whereas in other cases both are required. For example, an anti-collision system can function well by knowing an object's location but has no need to identify what the object is. This can be seen in systems like the SEEK industrial 3D machine vision cameras for collision avoidance [3], Navtech Radar Terran360 which produces a 3D point cloud map of the environment [4] or the GEOPREVENT Radar designed to detect moving objects in alpine conditions [5]. These systems are either not required to know what the object is, or simple observation can be used to distinguish the objects from their environment. In contrast, Sonasafe requires knowing both aspects: where and what the object is.

Detecting objects, including humans, against a background or environment requires being able to identify characteristics distinctive to that object. The environment in which an object is detected can impact how distinctive and distinguishable these characteristics are in relation to other objects. The sensor, or type of data generated, can influence what specific characteristics can be detected in any given environment. This section looks at the factors of object detection and how they relate to detecting people in an industrial environment.

2.1.1 Object Features

Classification of an object is achieved by identifying characteristics distinct to that object. The distinctiveness of an object's characteristics impacts the ease with which an object can be distinguished from other objects and classified correctly. These distinctive characteristics or features can include colour features, texture features, shape features [6], movement, or temperature features [7-9].

A simple example is identifying apples on a white background. The apple is red; the background is white. To identify the apple, the data is processed to find the groups of red pixels to classify an object as an apple. Locating the apple requires knowing where the group of red pixels are within the image.

2.1.2 Environment

In a controlled environment the specific characteristics of that environment are known. In the example of detecting apples, lighting and backdrop colour can be controlled to emphasise the contrast between the background and the characteristics of the apples [9, 10].

Generally, the less an environment is controlled the harder it is to classify an object due to a reduction of distinctive characteristics. An industrial environment is usually uncontrolled with background clutter, illumination changes, occlusions and other factors [11], and this increases the need for significantly different, or more, characteristics for a given object or person. In the example of detecting apples, the presence of a red ball could result in a false positive. Having a red background, an overly dark setting, or red lighting would reduce the contrast between the distinctive red characteristic of the apple and the background, making it more likely an apple would be missed. The industrial environments this study is concerned with are complex and largely uncontrolled making it more difficult to identify objects.

2.1.3 Sensing Features

The more a feature stands out from its environment the easier it is to detect and classify. Choosing the right sensors or data type is critical to detecting an object's characteristics in contrast to the environment. For example, infrared would be a good choice when trying to sense a person in a cold environment as there is a large contrast in temperature, but would be less effective in a hot environment [12]. Another example is seen in some SEEN safety products using an infrared camera to detect reflective tape on high-visibility safety gear where the reflective tape is the feature distinct from the environment [13]. Each of these sensors collect a specific data type and are selected based on how their function relates to a characteristic distinctive from the environment. In contrast, trying to detect objects with characteristics less distinct from an environment requires an increased amount of data, a greater variety in the types of data, or both, to improve classification [11].

Designing a distinctive characteristic into a system or environment can improve the accuracy of systems that detect people in industrial environments. This characteristic needs to be in such contrast to the environment that it can easily be used to detect a person. The current Sonasafe system is a good example of this where the PDUs transmit information through radio communication [2]. Other examples include using high visibility clothing or reflective stripes that have distinctive characteristics detectable by camera or lidar [14, 15]. The advantage of such systems is that the

characteristics are generally in high contrast to the environment. The limitation with this method is it requires compliance with the system requirements, namely wearing a working PDU or high visibility vest, which is a point of failure as people are not always compliant. Additionally, such systems are restricted to detecting employees and will not detect any other person entering the area (e.g. unauthorised persons or guests) are not wearing a working PDU .

Rather than designing one very distinct feature into a system, an alternative method is including different sensor types, or sensor fusion, to target multiple features [16, 17]. This combination of data improves detection accuracy by requiring more, albeit less distinct, identifiers to be positively detected before returning an object positive result. Each sensor is able to identify a variety of characteristics, with each sensor having their own advantages and disadvantages in relation to the environment [18-21]. A study by Bocu et al. on 3D object detection methods, focused on sensor fusion, asserted that the increased accuracy of detection from fusion makes it an ideal method for object detection [22]. The disadvantage of sensor fusion is that more sensors and technology are required and more data must be processed, resulting in a more complex, more expensive system. However, even when considering the increased computational resources required, Bocu et al. considered sensor fusion an optimal solution. This method also has the benefit of introducing redundancy by having more sensors which minimises the impact of sensor failure. In general, literature indicated sensor fusion offers more accurate and reliable detection in complex environments and applications [22-25].

2.1.4 Human Detection

Safety systems require a high level of reliability and accuracy to meet the standards of safety required when dealing with people. The challenge of detecting a person in complex environments is to distinguish them from a cluttered and busy background.

To achieve a high success rate in an uncontrolled environment the characteristics used to detect a person need to be sufficiently distinctive from other objects in the environment. When comparing a person to an inanimate object, some distinctive characteristics might include movement, warmth, heartbeat, silhouette, or relative size.

The difficulty with the identification of generic characteristics of people is that they are not necessarily specific or consistent. The silhouette, for example, is different front-on to side-on for any individual as well as different between person to person. Body temperature is consistent, but clothing

can have an impact on the infrared signature given off. By chance other objects such as warm engines, vents or animals could give off similar temperature signatures. Likewise, detecting the movement of a person in an uncontrolled environment might be hard to distinguish without false positives or failures from vehicle or animal movements.

The accuracy of detecting characteristics improves through either identifying distinguishable features uncommon to the environment, or through identifying multiple characteristics that must be present to give a positive result.

A review of common methods and sensors applicable to human detection was undertaken as part of a survey of human sensing in a report by Thiago Teixeira [16] as shown in Figure 1.

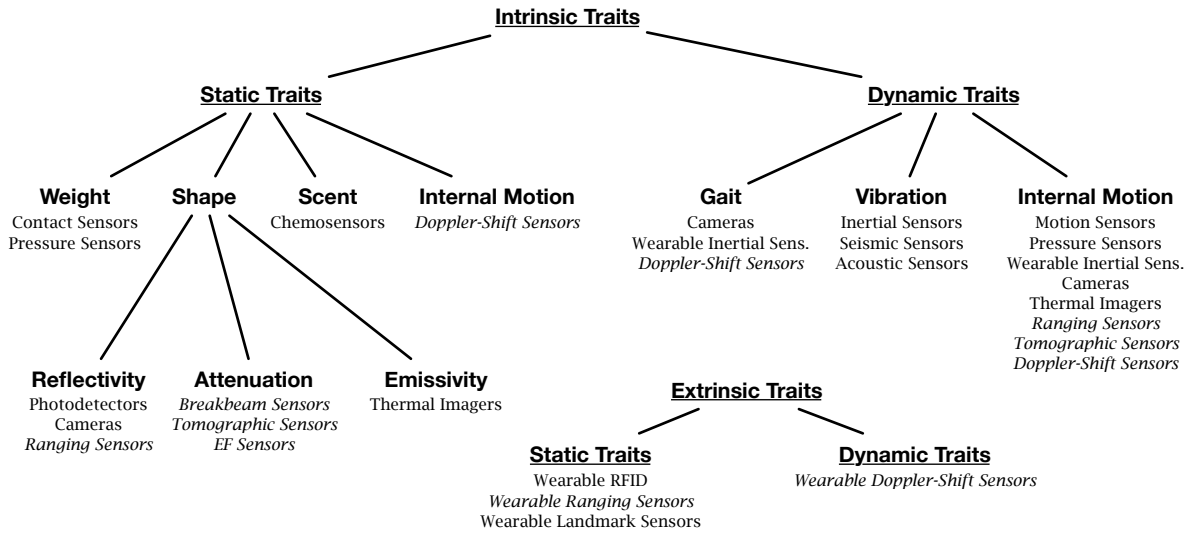

Figure 1: A Survey of Human-Sensing. After [16]

Methods for detecting a person included weight, shape, scent, reflectivity attenuation, emissivity, gait, vibration, and motion. Of these, weight, scent, and vibration can be discounted as sensing methods due to requiring a relatively controlled or stable environment which is in direct contrast to the complexity of industrial environments. Of the remaining methods, shape and motion were the more significant features. The sensors listed that are applicable to this study are motion sensors, scanning range finders and imaging cameras. More specifically these sensors include lidar, sonar, radar, 3D Imaging Cameras, and 2D Imaging Cameras [26].

2.2 Sensors

At a fundamental level, sensors vary in the type of data they collect and the resolution of that data. The types of sensors, their functional capability, and their respective resolutions are critical factors in sensing desired characteristics of people in uncontrolled complex environments, which needs to be balanced against cost and computational requirements.

2.2.1 Sensor Type

The various advantages and disadvantages of each sensor, and the type of data they collect, correlates directly to their ability to locate or classify an object, or both.

Cameras are a prime example of sensors suited to classification functions. Most common cameras operate within the visual spectrum, varying from basic RGB cameras that simulate what we see with the human eye to distinguish colours, through to hyperspectral cameras that can be used to detect characteristics not distinguishable with the human eye, even to the extent of being able to detect diseases in plants [27]. More advanced spectral analysis such as infrared and hyperspectral have the advantage of being able to detect more and different object features; the disadvantage being they are more expensive [28]. Specifically, infrared cameras can measure the temperature of objects, which are potentially effective for detecting warm, living objects [12].

As conventional cameras usually only capture information in 2D, they are less suited for locating an object in real world 3D space. Exceptions are time-of-flight cameras and some 3D cameras, but these still have limitations in range. There is no simple way to translate the intensity of light in the RGB spectrum from a standard camera image to distance [29].

Lidar and radar are accurate at detecting objects' locations in 2D and 3D space [30]. These sensors measure the time-of-flight of sound or light reflecting off an object which has a direct corelation to distance. In contrast to cameras, lidar and radar operate largely based on range measurements and are more suited to location functions and consequently are more limited when it comes to classification.

With regards to radar systems, there are different types, frequencies, and methods to achieve different results depending on the desired application. This review will focus on mmWave radar as this variation has some of the best resolution and accuracy at short range, out to approximately 120m. In the automotive industry, mmWave radar is at the leading edge of radar technology, pushing the

achievable resolution of radar, which makes it one of the better sensors to identify stationary objects in high resolution [31, 32]. Depending on the configuration, some Texas Instruments mmWave radars can operate out to 350m, detecting objects with Radar Cross Section (RCS) of $10m^2$ with an angle of observance up to ±70° at short range. Texas Instruments claims that human RCS objects are detectable out to a distance of 150m though the objects can not necessarily be classified as people [33].

Lidar can capture very accurate 3D data, mapping an environment with a field of view of 360° horizontal to 20° - 45° elevation at a high resolution [24, 31]. Higher accuracy and resolution can also contribute to object classification based on an object's silhouette and shape.

A primary advantage of radar over lidar is that radar is not affected by adverse weather and performs almost equally as well in both rain and clear weather, while lidar is negatively impacted by weather such as snow, fog and rain [19, 24]. Radar also has the advantage of incorporating a velocity component to the data collected due to a doppler shift in the output signals which might be useful to identify additional object features [18, 19]. It also seems that lidar used in the automotive industry is generally less effective inside a range of 30 meters [19].

An advantage of lidar is that it has a higher resolution than radar which can contribute to object classification [34]. High-resolution 3D maps can be captured from ranges of 30 to 200 meters. Understandably, the amount of data this generates can require a large amount of computation to extract the required information [18, 19]. Though lidar is generally more accurate than radar, it is significantly more expensive [30].

Overall, cameras are able to detect a higher variety of features than radar and lidar, which contributes significantly to classification. Radar and lidar are primarily limited to distance and angle measurements, meaning they are more suited for 3D location. The resolution of these sensors is also generally lower than cameras due to the cost and data load of increasing the resolution for these technologies. F. Rosique et al. also identify the effect of weather conditions on cameras as another reason to include radar and lidar in sensor fusion combinations as the additional sensors increase the robustness of the system. While lidar certainly could contribute in sensor fusion, with regards to the issue of weather, radar is the least affected by weather conditions and would be the sensor of choice to mitigate these effects [19, 24]. The varying aspects of each of these sensors makes cameras and lidar or radar complimentary sensors as identified in Zhou et al.'s review of mmWave radar and

camera sensor fusion systems. Zhou et al.'s review focuses on the camera's ability to identify targets and the radar's ability to provide spatial position and velocity information [30].

2.2.2 Resolution

Although higher resolution sensors measure an environment more accurately, they require more data and may cost more due to the additional technology involved. A lower resolution may lose details of distinguishing characteristics needed for accurate detection. An optimal resolution will achieve the required reliability of detection without excess data, as any excess data is a computational drain without benefit. Diversifying the type of data collected, as well as increasing the resolution, could contribute to better identification of characteristics. An example would be using an infrared camera to return data able to identify both a temperature signature and a silhouette. However, the increased resolution required to be able to return a silhouette may not be worth the cost of the more sophisticated camera.

For context: a low-resolution camera that captures an 80x60 pixel image with 50º field of view at 20m, has a horizontal resolution of 0.626º per pixel and vertical resolution of 0.83 º per pixel. At 20m distance each pixel represents an area of 0.22m wide by 0.29m high. For a person 1.8m high by 0.5m wide, they would be represented as 2 pixels wide by 6 high, a total of 12 pixels. This is a very low resolution and would make it difficult to detect what the object is unless the data being measured from the object is significantly different and distinctive to the environment. However, together with data from other sensor types returning information suggesting the presence of a person, it may be sufficiently detailed to extrapolate what the object would be. Clearly a higher resolution camera would generally be a better performing option in this scenario, but the higher resolution would come at a higher cost.

Though resolution capabilities vary between different sensors, some typical resolutions for the following sensors are listed as an overview [35]:

- Lidar: 0.25 - 0.5°
- Sonar: 15°
- Radar: 2 - 5°
- TOF Camera: 0.2 - 0.5°
- Structured light: 0.5°

- Stereo: 0.5°

The resolution of ultrasonic sensors at 15° is not sufficient to provide a level of detail that could be used to determine if an object is a person through straightforward analysis, and therefore will not be discussed further.

2.2.3 Application

The different data types detected by cameras, lidar and radar and the resolutions they offer all have advantages and disadvantages depending on the application and environment.

Industrial environment are varied, and include a range of object types, shapes and sizes, and differing environmental conditions such as ambient lighting, rainfall and mist. As such, using different sensor types such as camera or infrared might work better in different conditions. For example, vision camera might work better during the day while an infrared camera might work better at night.

Due to the complex and uncontrolled environments Sonasafe operates within it is likely that a fusion of two or more sensor types would be a more effective method of detecting people rather than a high-resolution individual sensor type. For a given amount of data that needs processing, there would be an optimal combination of sensor types and respective resolution.

2.3 Convolutional Neural Networks and Sensor Fusion

Prototype Matching, Statistical Classifiers and Neural Networks are all methods that can be used to classify objects from data, primarily for images. These methods all rely on features that are used to classify the image. For Prototype Matching and Statistical Classifiers these features need to be engineered into the system. Neural Networks have the advantage that they are able to generate or learn these features without the need to be specifically engineered [8].

The process to develop and train neural networks has advanced significantly over the past decade with the development of software tools like TensorFlow and Karas that can greatly simplify the process [36, 37]. Due to the complexity of the environments this thesis is evaluating human detection within, the method of Neural Networks is selected for the potential to include many unspecified features in the detection model through machine learning.

2.3.1 Convolutional Neural Networks

CNNs are a form of neural networks commonly used in image-based object detection. A CNN works on the dot product of a grid shaped input with 2D filter arrays or kernels which are convolved or stepped across the whole image to produce feature maps. Though different steps can be used depending on the application, a step of one pixel is common. Kernel sizes commonly range from 1x1 to 5x5 for image-based object detection. Multiple kernels of different weightings are stacked depth-wise which allow extraction of different features such as line or colour gradients [8, 36, 38]. The convolution structure is outlined in Figure 2.

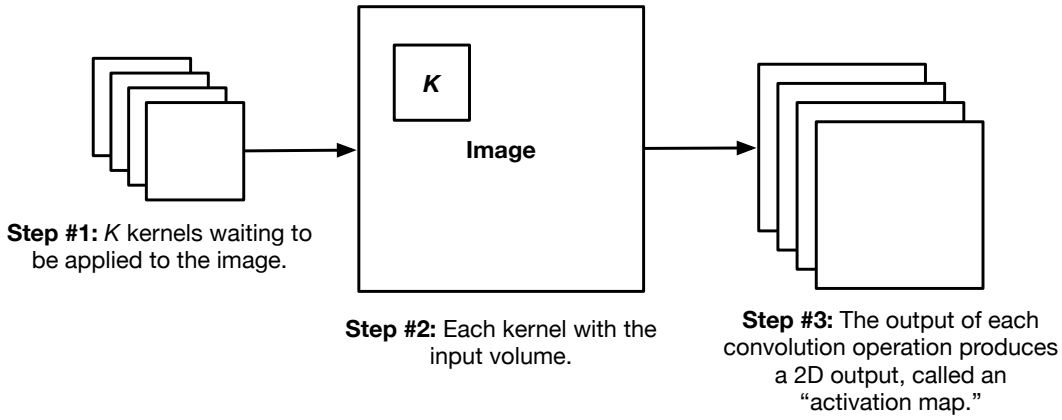

Figure 2: Convolution Steps. After [39]

Convolutions are often paired with pooling layers that assist in filtering out minor features or noise and focus on more significant features. A common pooling method is a max pool which takes the maximum value in a 2D grid while discarding the other values resulting in a smaller resolution feature map consisting of more significant features. For a 2x2 max pool with a step size of 1 pixel the feature map would downsize to half the original dimensions [8, 36, 38]. This process is shown in Figure 3 for a 2x2 max pools with strides of one and two.

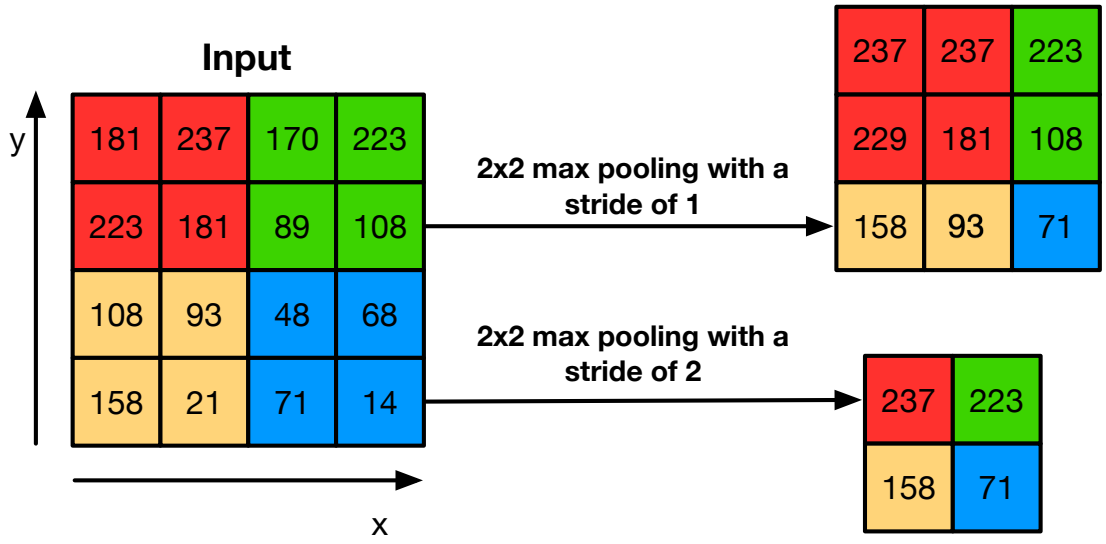

Figure 3: Max Pooling. After [39]

Object detection using CNN with a radar 3D point cloud can be achieved through point cloud sampling or through pre-processing the radar data into a 2D depth map image. For sensor fusion it is likely that models using 2D depth maps will be simpler to implement but could result in a loss of information from the radar data [40-42].

2.3.2 Sensor Fusion

The use of CNNs also allows for relatively easy fusion of image data through modification of a model's structural design prior to training. Different methods can be used such as early or late fusion. In early fusion, images of the same size are stacked, or concatenated, to form a single input image. For example, an RGB image can be concatenated with an infrared (I) image to form a single, four channel RGBI image. Late fusion requires two CNN branches that extract the respective features from each image before fusing the features together and performing the detection [43, 44]. Early fusion has a simpler model structure but is generally not able to compensate as well for data or feature errors while late fusion models can compensate for individual sensor errors separately [45].

2.4 Conclusion of Literature Review

This literature review evaluated object detection against different environments and their impact on the accuracy of detection. Different sensors were also compared against their suitability for different aspects of object detection.

The environments Sonasafe works within are complex, introducing conditions that require distinguishing a person from a myriad of objects and their features. Isolating multiple specific characteristics of people can assist in distinguishing a person from these other objects in such environments.

A review of different sensors used to detect a person showed that the different sensor types tended to suit different aspects of object detection, with cameras being more suited for classification, and radar and lidar being more suited for location. For classification, the use of different data types targets different object features that are more distinctive to the environment and other objects.

This review concludes that a fusion of radar, infrared and visual camera types would most likely provide a good range of distinguishing and location characteristics at a cost that would be reasonable for a commercial application. Distance, angle, and velocity can be provided by the radar at a relatively low resolution, temperature differential can be provided by infrared camera and a more detailed silhouette and colour through the visual camera. Lidar was considered in place of radar, however it was discounted as being too expensive to be able to implement in a viable commercial system of this type.

The data from these sensors will be used with CNNs for human detection, allowing for the CNN models to train on features rather than trying to engineer in specific features. CNNs can be developed for each sensor and sensor combination to evaluate the performance of each separate sensor as well as the fusion of sensors to determine how they complement each sensors' respective strengths and weaknesses. Initial trials will need to be done on the different methods available before deciding on final methodology.

Chapter 3
Methodology

3.1 Method

The method began with selecting and configuring sensors, followed by developing and testing data sets. A preliminary data set, Data Set 1, was created by taking images in various locations, both at night and day. The data was processed, labelled, and used to train CNN models. Model performance was analysed, which informed modifications for following data sets. Data Set 2 was created through mirroring the images from Data Set 1, resulting in a data set doubled in size. The most promising model structures from Data Set 1 were retrained on Data Set 2. These models performed to a higher level of accuracy but insufficiently simulated the industrial environments due to a lack of infrared noise and positional variation of the person within the images. Additional images were taken and added to Data Set 2 to create Data Set 3. The additional images included a motorcycle and a car with warm engines, and the sensor array was moved up and down as well as left and right to create variation in a person's location. Data Set 3 comprised of over 3000 images used to train the final models.

3.2 Sensor Selection

The sensors were selected from the initial categories identified in the literature review based on the features they were likely to be able to identify in order to detect a person. Visual and infrared cameras were evaluated with a focus on classification while radar and lidar were evaluated with a primary focus on range. Equipment was selected with a balance of performance and price being considered. The primary consideration of this research was evaluating the relative sensor performance or contribution to detection models. Sensors with a high accuracy or resolution would not add significant value to this research and consequently the higher cost was not justified.

A visual camera was selected due to its suitability for classifying objects, particularly with current machine learning models [30, 38], and was selected based on resolution and cost. Limitations include a lack of depth perception and being impacted by lighting, obstruction, or other environmental conditions. Some of these limitations could be minimized through using alternative spectrums such as infrared, which is not limited by visual lighting conditions [31]. For this analysis a conventional visual camera was used as a benchmark. A Microsoft LifeCam Studio was selected to offer a good benchmark for testing of an average, low-cost camera.

An infrared camera was also selected to incorporate temperature as an identifiable distinct feature. Infrared provides some system resilience against factors limiting the visual camera such as poor performance in low lighting conditions. The infrared camera options were not as varied and readily available and had to be evaluated by comparing resolution to cost. Higher resolution infrared cameras tended to have a much higher cost. After conducting some tests comparing resolutions of a Flir E60 handheld infrared camera to a Flir Lepton 2.5, a Flir Lepton 3 was selected due to having the highest resolution before a price increase. The Flir Lepton 3 also came with a module that allowed for easy functionality similar to using a web camera.

Radar sensors were evaluated to identify a suitable mmWave radar platform that could be suitably configured for this application and was selected for location data based on resolution and range versus cost. Lidar was briefly considered but was discounted due to generally being more expensive [18, 34] and more affected by weather and adverse conditions compared to radar. Using radar over lidar likely reduces contribution to classification due to lower resolution, however, it was expected classification would be primarily achieved through the visual and infrared cameras. Radar was limited in availability and had a higher cost than the cameras. An evaluation was done on available radar development kits based on resolution, range, ability to customize, support and cost. The Texas instrument AWR1843BOOST was the selected sensor as it met the minimum performance requirements of this trial.

3.2.1 Camera Selection

Selection of the Microsoft LifeCam Studio for the camera was a relatively simple process based on resolution, availability, and ease of use. The Microsoft LifeCam specifications are [46, 47]:

- RGB image
- Resolution up to 1920 x 1080p
- 30 Hz capture rate
- FOV 41.23^{o} high x 67.55^{o} wide

3.2.2 Infrared Selection

An initial evaluation of two different IR systems available on hand to test was done to form a base indication of performance requirements that might be needed for the IR component of the sensor array:

- Flir E60 hand held infrared camera with a resolution of 320x240 pixels and 25°x19° Field of View (FOV) [48]
- Flir Lepton 2.5 IR sensor with a resolution of 80x60 pixels and 52.68°x40.74° FOV diagonal or [49]

Images were taken with each sensor at 10m, 15m and 20m to evaluate images. The images from the E60 handheld camera can be seen in Figure 4 with those from the Flir Lepton in Figure 5.

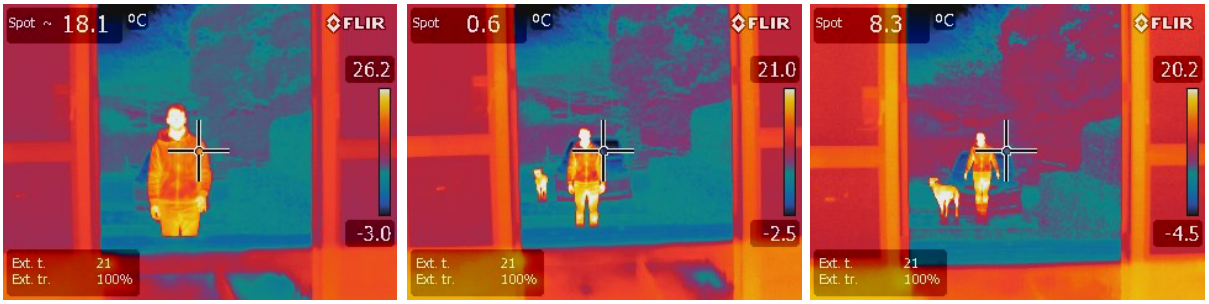

Figure 4: Flir E60 - 10m, 15m and 20m

The images in Figure 4 were taken from inside a room at night looking outside. There is a high level of detail in the Flir E60 image, with enough detail to distinguish features such as arms and legs for the person at 20m, as well as a cold car behind the individual. The contrast with the person against the outside background is different enough that they easily stand out.

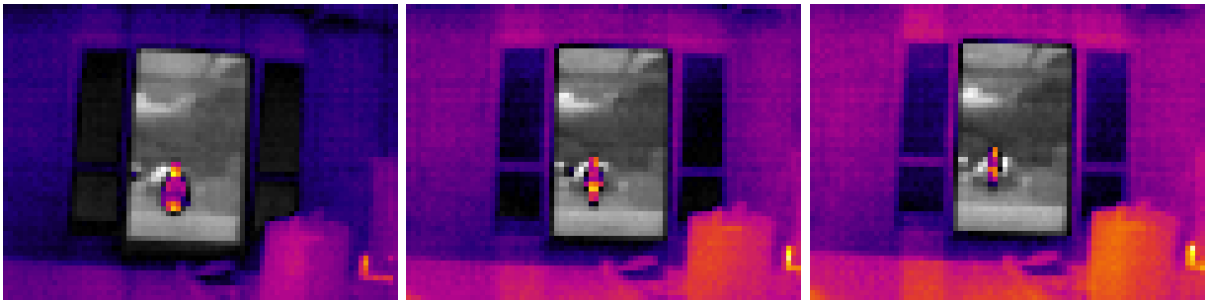

Figure 5: Flir Lepton 2.5 - 10m, 15m and 20m

In comparison, the images taken with the Flir Lepton 2.5 in Figure 5 has little detail of features such as arms and legs, rather showing a warm blob in the centre of the image. When it is understood that the object is a person some shape can be made out. However, in terms of detecting silhouette from this picture there is a significant reduction in clarity. Though temperature differential rather than silhouette is the focus for this sensor a resolution greater than the Flir Lepton 2.5 would be desirable. Images can be post processed to reduce resolution if desired for testing.

	Lipton 2.5	Lipton 3	MLX 90640	D6T-44L-06H	Boson	Tau 2
Pixels	*80x60*	*160x120*	*32x24*	*4x4*	*320x256*	*336x256*
FOV °	*52.5°x40.5°*	*59.5°x46.5°*	*55°x35°*	*44°x45°*	*24° to 75°*	*25° to 46°*
Temp °C	*-10 to 80*	*-10 to 80*	*-40 to 300*	*5 to 50*	*to 140*	*-25 to 100*
Price (NZD)	*$213*	*$330*	*$110*	*$125*	*$2,600*	*$4,463.03*

Table 1: Infrared Cameras

Available infrared sensors were evaluated through market research on resolution, FOV, temperature range and cost. The most relevant of these sensors can be seen in Table 1 [50-55].

Research indicated that as resolution increased above 160x120 pixels the price range rose significantly, possibly due to the increased functionality incorporated in the more advanced, higher resolution cameras. This additional functionality was not required for this study.

The Flir Lepton 3 was selected as the sensor of choice as it had the highest resolution, a FOV similar to the LifeCam and a low cost relative to other infrared options. It was estimated that the Flir Lepton 3 would have a sufficient resolution, based on the initial analysis with the Flir Lepton 2.5.

3.2.3 Radar Selection

The options for mmWave radar were not as extensive as initially expected due in part to limited stock availability, long lead times and generally being a less common sensor type. Software Development Kits (SDK) manufactured by Texas Instruments looked to be the best options based on configurability, stated capability and customer support [56].

The most applicable options were the MMWCAS-RF-EVM and the automotive radar AWR1843BOOST with basic capabilities listed in Table 2.

	MMWCAS-RF-EVM	AWR1843BOOST
Frequency	*76-81GHz*	*76-81GHz*
Field of View (Azimuth x Elevation)	*120˚ x 60˚*	*120˚ x 30˚*
Angular Resolution (Azimuth)	*1.4˚*	*15˚*
Max Distance for person	*150m*	*80m*
Price	*$2270*	*$620*

Table 2: Radar Options [56-59]

Classifying the radar for range, accuracy and FOV was not a straightforward task as this was dependent on many factors including the set-up, configuration and type of object being detected [60]. Reviewed manuals indicated the MMWCAS-RF-EVM was capable of much higher resolution and range [57-59]. Though this would provide better data, it would also increase the data processing load and the price difference was significant. The AWR1843BOOST was selected as it met the minimum requirements of being able to detect a person out to approximately 40m to 80m depending on the manual referenced [61-63].

3.2.4 Configuration

The webcam was set to 640x480 resolution which was a balance between camera frame rate and maximizing the resolution. The infrared camera was set to the maximum resolution of 160x120. The radar output was a point cloud for each detected datapoint which included X, Y and Z position coordinates referenced in this thesis as range (R), as well as lateral velocity (V) and intensity of the return reflection (Q) [64].

The cameras required little in the way of configuration, and were left to automatically configure parameters such as contrast and rate of exposure.

The radar required specific configuration for the task parameters. Firmware and configuration were based on the TI Automated Parking Demo with the configuration file modified for the desired range, resolution and FOV. The radar was configured to achieve a maximum unambiguous range of 19.99m with a FOV of -30^{o} to 30^{o} horizontal by -20.5^{o} to 20.5^{o} vertical. Though this was less than the advertised FOV of the webcam, from observation the actual FOV of the images taken with the webcam was closer to 55^{o} [65].

Parameter	*Config*
Frequency Start (GHz)	*77*
Frequency Slope Constant	*20*
Sampling Rate (kdsps)	*3333*
No. of Samples per Chirp	*640*
Frame Periodicy (ms)	*500*
Idle Time (us)	*124*
ADC Valid Start Time (us)	*7*
Ramp End Time	*200*

Table 3: Radar Parameters [63]

The configuration file for the AWR1843Boost was in part based on the Texas Instruments Online Demo Visualizer tool version 3.5 for SDK version 3.03, platform AWR1843 [63] with design help through the mmWave Sensing Estimator tool. Parameters were set as per Table 3 to achieve the configuration performance listed in Table 4.

Performance	*Value*
Range Resolution(m)	*0.039*
Maximum unambiguous Range(m)	*19.99*
Frame Duration(msec)	*500*
Range Detection Threshold (dB)	*15*
Angle of Arrival FOV	*-30º to 30º*
Range FOV	*-20.5º to 20.5º*

Table 4: Radar Performance [66]

3.3 Generating Data Sets

The data sets were captured in a variety of locations, including a parking lot, a school, and a church complex which were easily accessible to the public, and controlled, safe environments. Effort was made to provide a range of backgrounds including tarmac, trees, buildings, and objects that might be present in or at least simulate the dynamic conditions in industrial locations. Data was captured in both bright daylight and in dark locations at night to ensure varied and complex environmental conditions, again to emulate different industrial locations and conditions. Objects like a hot motorcycle and car were also included in the locations to provide additional representation of some different conditions or objects present in industrial locations.

The data set was limited to one person in a frame to allow for simple True/False classification, requiring only one location prediction. Including more than one person in a frame was initially considered but was discounted in order to evaluate the performance of the sensors with simpler models and because using one person required significantly less data and time required to taken to capture, label and process the data. This allowed many more models to be tested against sensor performance and minimised the likelihood of model or training factors negatively impacting the quality of results.

3.3.1 Capture

The sensors were mounted in a case on a tripod at approximately 1.3m high as seen in Figure 6. One image was taken for each of the cameras while a series of five samples was taken with the radar to generate more dense point cloud clusters. Over 3000 samples were taken, featuring three different people, in eight different locations both at night and during the day.

Figure 6: Sensor Casing

The sensors all captured data with an internal buffer that was requested by the computer and transferred through USB. Efforts were made to synchronize the data through delay of camera capture, clearing camera buffers, flagging images to be captured, and retrieving the radar data both before and after the cameras. Testing revealed the most optimal method was to continually read the radar data and store the latest 5 samples before reading the cameras. The buffers for the cameras were then cleared and the frames marked to be captured for both cameras before reading to the computer. This method resulted in the cameras being reasonably synchronized, but with a delay of 0.5 seconds between capture of radar data compared to capture of the cameras' data. Though this delay was more than desired it was accepted in order to progress the study through to testing. True synchronization would need to be achieved to properly implement a working system in future. This might be achieved through running parallel operations retrieving the camera and radar data. Apart from adjusting the sensor settings and timing of data collection, it was not necessary to calibrate the sensors at this point. Rather, images were calibrated during processing to minimise differences in FOV and viewing angle between the different sensor images.

During capture of Data Set 1, the sensors were slowly moved horizontally to ensure some change in the background. Results from this test indicated that the data set was likely not large enough, and Data Set 2 was generated by mirroring the images to double the number of image samples. Results from Data Set 2 indicated there was not enough variation in position of the person in the images and very little infrared noise leading to unrealistically accurate results from the infrared camera. Data Set 3 was generated to improve these results by moving the sensors vertically as well as horizontally for greater positional variation of a person within the frame. Infrared noise was added by way of including a hot motorcycle and car in the background. Data Set 3 helped to better show true performance of a model with respect to real world environments [67].

3.3.2 Processing Images

Once the data had been captured the images were evaluated for suitability for training models. To minimise factors in the data that might degrade results, images were discarded if they were too blurry, the person was partially out of frame, the person was too close to the camera, or settings caused effects such as overexposure.

The remaining images were processed and labelled ready to train the models. The data was labelled using the programme LabelImg which allows the user to draw a bounding box on an image and saves the box coordinates in a label file [68].

The images from the infrared camera were labelled to get the X (horizontal left & right) and Z (vertical) coordinates of the person. The infrared images were used for consistency as it was visually easier to label low light images. An example of a labelled infrared image is shown in Figure 7.

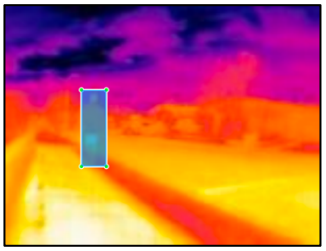

Figure 7: Infrared Label

The bounding boxes from the infrared labels were then drawn on the camera images to evaluate consistency with the infrared images for calibration by observation. This revealed a small offset and difference in FOV between the infrared and camera images which were shifted and cropped so the labels for the infrared were accurate for the camera images. The error left between the infrared and camera labels was estimated to be insignificant relative to the shift caused by the 0.5 second delay between the cameras and radar captures.

The radar was graphed for labelling, also using LabelImg [68]. The 5 radar samples were all plotted to an X (horizontal left & right distance), Y (horizontal range distance) coordinates graph as shown in Figure 8. Through observing respective camera images and changes to consecutive radar plots these graphs were labelled with bounding boxes around the cluster of points representing the person. The units of these bounding boxes were then converted from pixels to range in meters. Due to the resolution of the graphs, it was conservatively estimated that the error of these labels would be in the range of ±0.2m, approximately 1.8% of the radar range.

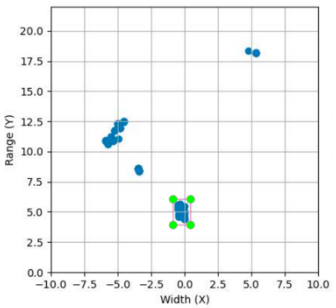

Figure 8: Labelled Radar X, Y Graph

Once the labels for the infrared and radar data had been completed, the two labels were combined into a single XML file for each data sample to include person, no person, and X, Y, Z position. The centre of the bounding box was taken from the infrared labels as the X and Z position coordinates while the range or Y centre of the radar bounding box was taken as the Y position coordinate. No XML file label was made for an image that did not contain a person.

Four approaches were assessed to prepare the radar data for input into CNNs:

- 3D X, Y, Z point cloud with velocity (V) and strength of reflection (Q) as data values
- 2D X, Z point cloud with range (R), velocity (V) and strength of reflection (Q) as data values [69]
- Vector of data X, Z, Y, V and Q [70, 71]
- 2D range map [30, 40, 41]

Initial results indicated that the 3D point cloud required a model too large to train on a conventional computer and was unrealistic. Results for the 2D array showed the model was not able to train. Models trained on the vector data also did not train due to the number of points detected by the radar often being too low and varied. The final method converted the radar data to a 2D range map overlay onto a black background as shown in Figure 9 and were of the same size as the camera images. This method resulted in some of the models training and was selected as the method of choice.

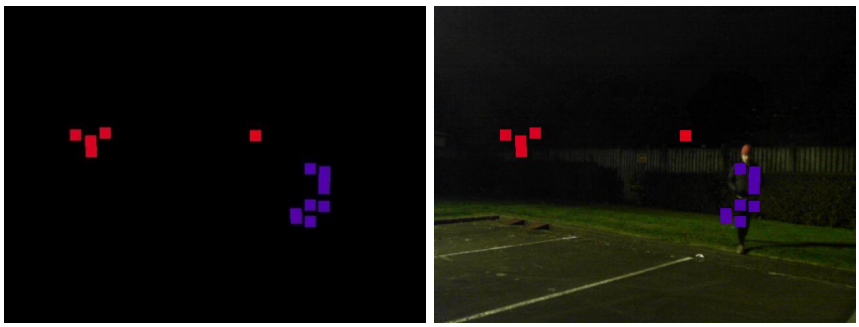

Figure 9: Radar Depth Map and Overlay

The 2D range map was produced by converting the X and Z coordinates to pixel space as shown in Figure 9. The Y axis was represented by the red channel ranging from 0 to 255 for 0m to 22m and the blue channel ranging from 255 to 0 for 0m to 22m. Two colour channels were used to ensure a non-zero value for any detected datapoint regardless of distance. If velocity (V) or intensity of reflection (Q) were incorporated, it was spliced in as additional channels of 0 to 255. Examples of the resulting images produced from each sensor following processing and calibration are shown in Figure 10 and Figure 11. At a glance, these figures demonstrate some of the strengths and weaknesses of each sensor.

Figure 10: Image Set - Person (Camera, Infrared, Radar)

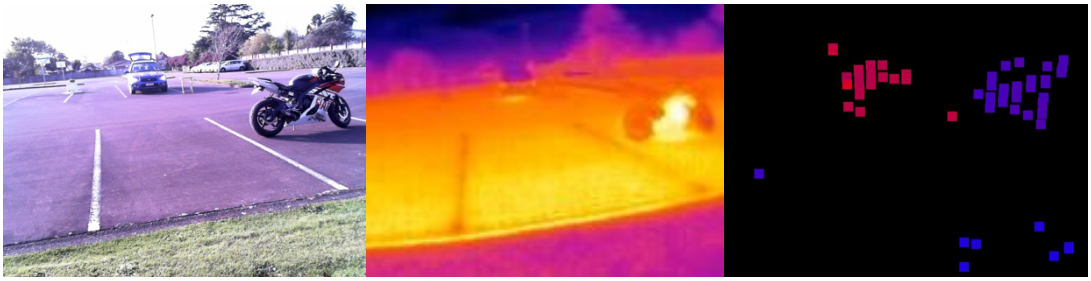

Figure 11: Image Set - Motorcycle and Car (Camera, Infrared, Radar)

In preparation for inputting the images to the different models, the images for all sensors were scaled to the correct sizes. Each camera and radar image was saved to five different image sizes, and the infrared images were saved in three different sizes as shown in Table 5. Though this required more computer memory it allowed for faster training by eliminating the need to process the images each time a data set was used. The image sizes were selected as multiples of two of the image ratios 4:3. This was required because if a max pool is done on an image with an odd number of pixels for width or height, the odd row or column of pixels is lost. For the smaller feature maps one row or column is a large amount of information. For convolutions, same padding is used to ensure the output size of a convolution is the same as the input size. This is common practice with CNNs to preserve information [36].

Camera/Radar		*Infrared*	
Width	*Height*	*Width*	*Height*
512	*384*		
256	*192*		
128	*96*	*128*	*96*
64	*48*	*64*	*48*
32	*24*	*32*	*24*

Table 5: Model Input Size

3.4 Model Design and Training

Models were designed with a basic CNN structure which traditionally performs well when used for image classification tasks [36, 38]. The models were generated and trained on a data set split into two groups: a training set and a test set at a ratio of 9:1. Performance was primarily analysed from the

model accuracy on the test images to ensure results were not based on model familiarity with the training images.

Models were designed using keras.layers library in TensorFlow. This enabled easy programming of the models using the library functions to set up the different model layers. The models were of simple design to ensure the results focused on the effectiveness of the different sensors used and how well they work in fusion.

Creating models begins with defining the width, height, and depth of the input shape. Feature extraction is achieved through paired convolution and max pool layers. The final feature map is flattened to a vector and passed through fully connected dense layers of decreasing size until the desired neural output is achieved. In this case the output was a vector five elements long with the values ranging from 0 to 1. The first two elements relate to the predicted probability for no person present and person present. The last three elements relate to the position of the person in the picture in width, height, and range with [0, 0, 0] being top or left of image at 0m from radar and [1, 1, 1] being bottom or right of image at 22m range from the radar. For example, the correct output for a person present in the centre of the image at 11m would result in an output of [0, 1, 0.5, 0.5, 0.5].

The Keras functions used to create the models were:

- 2D Convolution: Conv2D()
- 2D Max Pool layer: MaxPooling2D()
- Flatten all features to pass to fully connected layers: Flatten()
- Fully Connected Layers: Dense()

Full details on these functions and their operation details are in the Keras API reference guide [72].

Different model structures were trained and evaluated with a variety of:

- Input image size
- Combination of sensors
- Number of max pool and convolution layers from 3 to 6
- Number of convolutional Kernels of 1, 3 and 5 [36]
- Late or early fusion of data

Models were trained on data sets by passing the training images of a data set through the model in batches. The size of these batches was defined at the beginning of training. The accuracy of the model was evaluated by comparing the prediction against the label. The accuracy loss was calculated by way of a loss function.

The loss function for this project was constant for all models and calculated in two parts as classification and location. The models output a prediction for two classes, no person present and person present with a range between 0 and 1 each. The class with the highest confidence was taken as the model prediction.

Classification loss was calculated as the sum of the squared error for the classification prediction.

$$\lambda_{class} = \sum_{c \in classes} (c_t - c_p)^2$$

Location loss was only calculated if there was an person in the image irrespective of whether the model predicted a person.

$$\lambda_{locat} = \sum_{l \in x,y,z} (l_t - l_p)^2$$

Total loss was the sum of both loss values. For example, a basic sum loss for the prediction [0.1, 0.9, 0.4, 0.4, 0.4] against the person present label [0, 1, 0.5, 0.5, 0.5] would result in a 0.2 loss for classification and 0.3 loss in location with a total loss of 0.5. The prediction [0.9, 0.1, 0.4, 0.4, 0.4] against the no person present label [1, 0, 0.5, 0.5, 0.5] would result in a classification loss of 0.2 and total loss of 0.2.

The weights in the convolution and max pool layers were adjusted based on the prediction loss. A new batch would be run until all training images had been used to train the model. The extent of the adjustments could be set to different rates using optimizers and learning rate functions. The optimizer used for study was the keras.optimizers.Adam tested with various learning rates. This process was repeated for a predefined number of iterations or epochs to achieve an optimised level of training. The Python code for making and training the infrared camera models can be viewed in Appendix A.

Initial models from Data Set 1 were evaluated for individual sensors and trained with different learning rates to determine acceptable parameters. Different inclusions of radar data, range (R),

intensity of reflection (Q) and velocity (V), were evaluated to determine what was beneficial to include. The best radar model was selected when testing models for sensor fusion. Models were then generated with all combinations of sensors, with both early fusion and late fusion models being evaluated and compared. The Python code for making and training of the infrared camera, visual camera and radar fusion models can be viewed in Appendix B.

3.4.1 Single Sensor Models

Models were generated for all sensors with variations in input image size, convolutional and max pool layers, and number of fully connected dense layers.

Initial tests indicated that the following model training parameters had some success when trained on Data Set 1 and were the starting parameters used in this study:

- Learning Rate: 10^{-3} and 10^{-4}
- Batch Size: 64
- Epochs: 50
- Kernel:
    - Camera: 3
    - Infrared: 3
    - Radar: 1

Models were generated with the following input data with different combinations of range (R), intensity of reflection (Q) and velocity (V) included for the radar:

- Camera
- Infrared
- Radar (R)
- Radar (RQ)
- Radar (RV)
- Radar (RQV)

The models varied in terms of max pool layers (3 to 6) and input image sizes (ranging from 32x24 to 512x384): The specific combinations for each input size are:

- 512x384 – max pools of 6 and 5

- 256x192 – max pools of 6, 5 and 4
- 512x384 – max pools of 5, 4 and 3
- 512x384 – max pools of 4 and 3
- 512x384 – max pool of 3

The infrared model was only generated with an input image size up to 128x96 as the native resolution was 160x120 and no additional information would be added by scaling up an image. The models' classification and location errors were each graphed to show performance.

Some models were generated to predict both classification and location, while other models were generated to predict either classification or location. Initial results from models only predicting one aspect generally showed a lower performance when compared to models predicting both aspects, so were discontinued.

An example model structure for a camera model is shown in Figure 12. The blue box represents an image input size of 32x24x3. Orange and yellow boxes represent the resultant dimensions of the feature layers output from the respective convolution or max pool computations. The final max pool feature layer (4x3x125) is then flattened to a vector (1500x1) and passed through fully connected dense layers of decreasing sizes, until the final output size is produced (5x1).

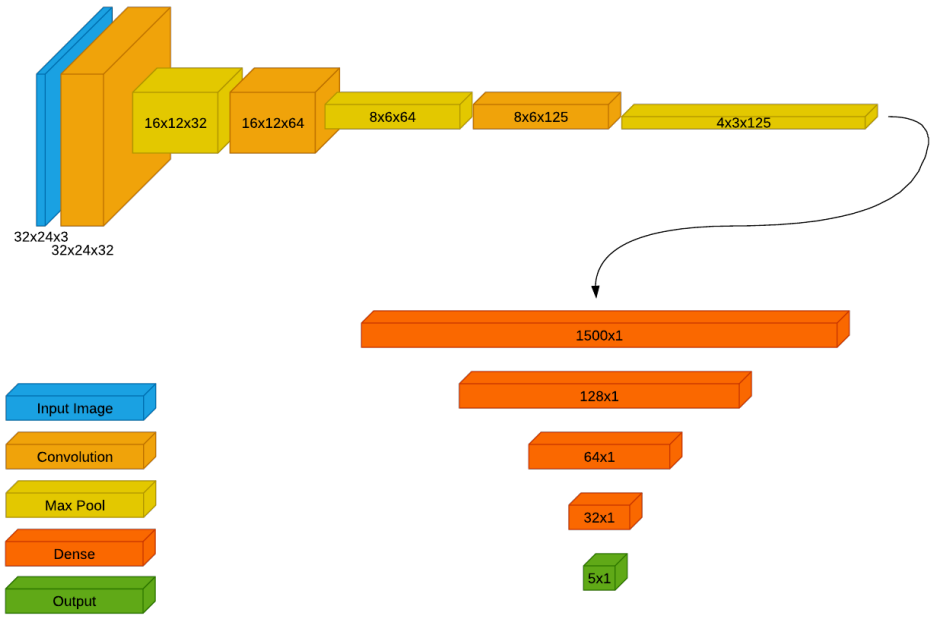

Figure 12: Camera CNN Model Structure

3.4.2 Sensor Fusion

Models for all combinations of sensors were designed and tested to better evaluate what each sensor contributed to model performance. These combinations were:

- Camera & Infrared
- Camera & Radar
- Infrared & Radar
- Camera, Infrared & Radar

Both early fusion and late fusion models were designed and tested to evaluate any advantages in performance of one method over the other.

Early fusion is achieved through stacking or concatenating the input data together before passing it through the model. The early fusion models were generated by concatenating the two input arrays together [43]. For example, a 128x96x3 camera image was concatenated with a 128x96x1 infrared image resulting in an input of 128x96x4, or a 128x96 image with the channels red, green, blue and temperature.

Models were trained to the same input size versus max pool layer combinations as the single sensor models. An example model structure for camera and infrared early fusion with image size of 32x24 is shown in Figure 13. In contrast to Figure 12, Figure 13 includes two blue boxes, representing the concatenated input from two different sensors. The model then follows the same process as the single sensor model.

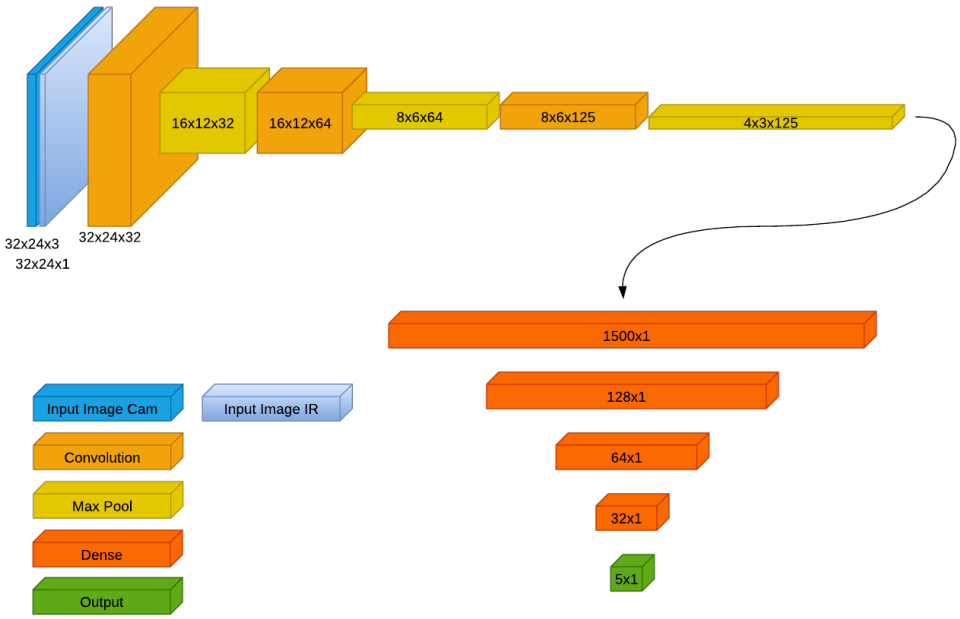

Figure 13: Camera and Infrared Early Fusion CNN Model Structure

Late fusion was achieved by generating a model with multiple branches of input and feature layers with one branch per sensor type [43]. As these branches were identical to those for each individual sensor model convolution and max pool layer, trained weights could be loaded to the respective branches in the late fusion models. Loading these weights meant that the training of the late fusion models could be started with weights that could already perform some feature extraction, making it easier to train the models.

The layers for the loaded weights can be frozen to prevent the layers being trained, maintaining the feature extraction layers that had worked for the single sensor models. Fusion models were trained as both frozen (F) and not frozen (N) models to determine if this had any impact on performance. The last feature layer of each branch were flattened and then concatenated together before feeding through the dense layers to better assist with training. These fully connected dense layers were the only layers trained if the feature layers were frozen.

All combinations of input sizes for the different sensor inputs were generated and evaluated on performance for classification and location. Only the radar RQ and RVQ were used of the radar data as they showed the most promising results for the radar models. Fusion models were trained with same parameters as the single sensor models, only testing different learning rates of 10^{-3} and 10^{-4}, and frozen or not frozen feature layers.

An example of the late fusion model structure for camera and radar with an image size of 32x24 is shown in Figure 14. Each branch represents the input, convolution, and max pool layers for the respective sensors. The feature layers from final max pool outputs (4x3x125) are concatenated together and passed through the fully connected dense layers to get to the final output of 5x1.

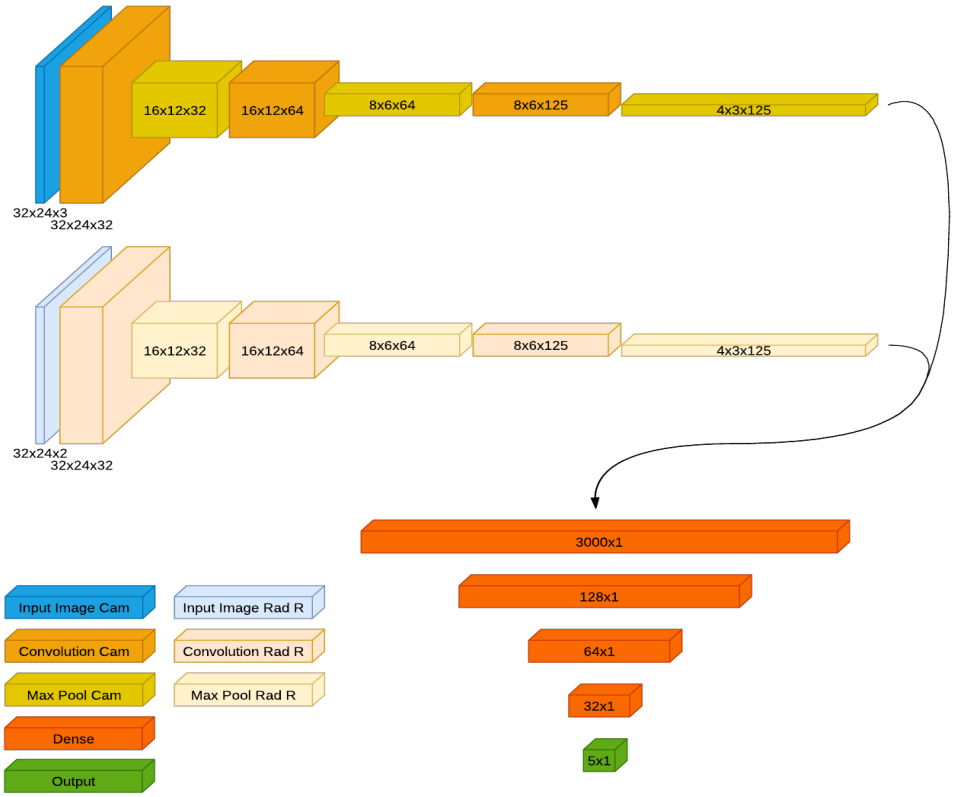

Figure 14: Camera and Radar Late Fusion CNN Model Structure

Early comparison between the early fusion models and the late fusion models revealed that the late fusion models performed better, as the early fusion models often did not train at all. It is unclear if this is due to the relatively small size of the data set or if the early fusion models could not cope with the 0.5 second time delay between the camera and radar data. Generally early fusion is expected to have more difficulty handling sensor errors than late fusion [30, 43, 73]. As a result, the early fusion models were discontinued in favour of the late fusion models.

Chapter 4
Results and Discussion

This chapter evaluates the results from models trained on all the different data sets. In discussion of the results, individual models are identified by serial numbers LxKxSx. L represents defined learning rates: 10^{-2}, 10^{-3}, 10^{-4}; K represents the kernel sizes used for the convolutional computations: 1, 3 and 5; S represents the number of paired convolution and max pool layers, or steps ranging from 3 to 6. For example, L3K5S6 is a model trained at a learning rate of 10^{-3} with kernel size of 5x5 with 6 convolution max pool layers.

Initial results from Data Set 1 did not have promising results, with some models having issues training. When the models were retrained with Data Set 2, the results were much more promising with most of the models training and the more accurate models being close to 100% accurate for classification and within 0.3m average error for location. However, the location accuracy of the infrared camera was unexpectedly more accurate than the radar models, giving the indication this data set was not varied enough to highlight the different sensor strengths and weaknesses. This led to the creation of Data Set 3 with extra images to introduce location error and infrared noise. Data Set 3 was slightly less accurate overall for the single sensors but showed much greater improvement in the sensor fusion models.

Only the optimal models for each sensor type are reviewed in this study as upwards of 360 models were generated and tested. The results are grouped into performance for classification and location with models being considered for average overall performance, classification performance and location performance. Sensor models with the best overall performance were the only models used for fusion with the exception that for the radar both RQ and RVQ fusion models were tested. Where results were closely matched, the classification accuracy held priority for the camera and infrared models while the radar had location priority. This was done as an attempt to emphasize the attributes of each individual sensor in the fusion models.

4.1 Data Set 1 Models

There were a variety of issues with some Data 1 Set models. Some of the larger models trained on Data Set 1 had difficulties training for the camera and radar. This generally occurred with training rate of 10^{-3} and kernel size of 3. The models trained for single camera, infrared or radar sensors gave

the indication that learning rate of 10^{-2} was too high for the models to regress to a minimum loss as a large number did not train at all. Likewise, a kernel size of 5 had a low success rate of training. These parameters were not used for further analysis.

Overall, there was a general trend of models with larger input sizes and number of convolution max pool layers performing better than the smaller models. This was not unexpected as the smaller models were expected to have a larger loss of information from downscaling the images [74]. The trade-off was the time and the size of the data set required to properly train larger models for more precise feature recognition. Data Set 1 may have been too small to effectively train on the larger 512x384 images.

4.1.1 Data Set 1 Camera

The camera graphs indicate that the kernel size of 3 performed marginally better. Although models for a learning rate of 10^{-3} generally performed better on average with more uniform error results around 10% to 15% than for a learning rate of 10^{-4}. The exceptions were that models with learning rate of 10^{-3} and 6 convolution layers were not able to train at all. In contrast, models trained on a 10^{-4} learning rate improved as the number of convolution layers increased and the best singular models were trained at this rate: L4K3S5 256x192 for classification accuracy and L4K3S6 512x384 for location accuracy. The percentage error results for the model L4K3 are included in Figure 15, showing the model through all step and image combinations.

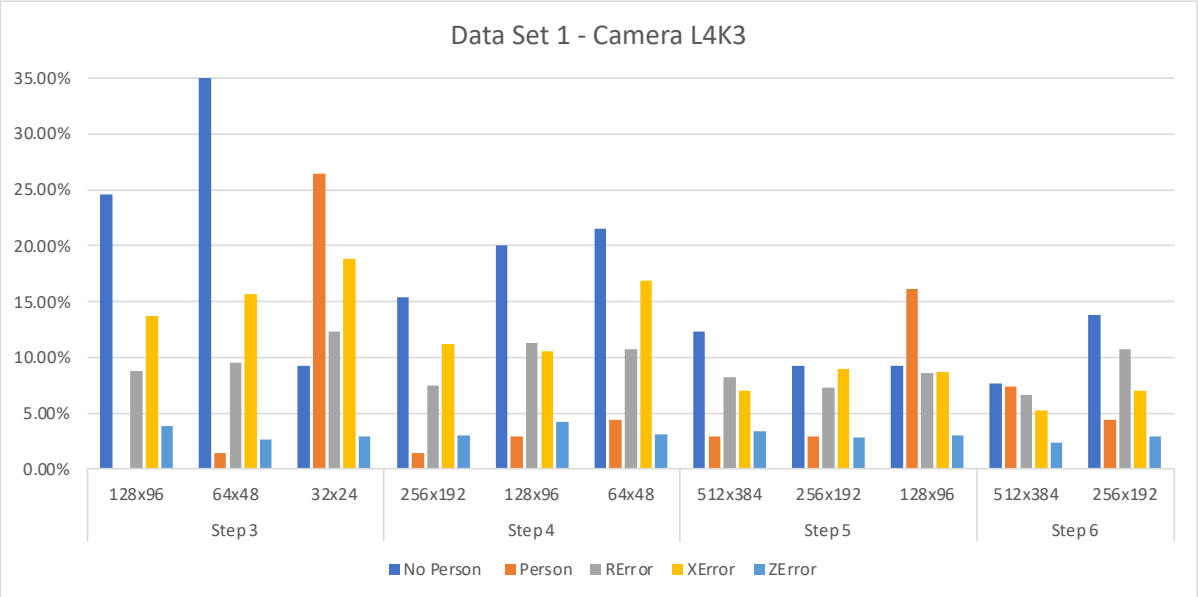

Figure 15: Data Set 1 - Camera Models L4K3

4.1.2 Data Set 1 Infrared

The infrared data had similar results to the camera, with the kernel size of 3 producing more accurate results than the other infrared models. Models trained with a learning rate of 10^{-3} performed well, with multiple models at 100% accuracy for classification and relatively high location accuracy. The best model for both range and accuracy values was L3K3S5 125x96 as shown in Figure 16.

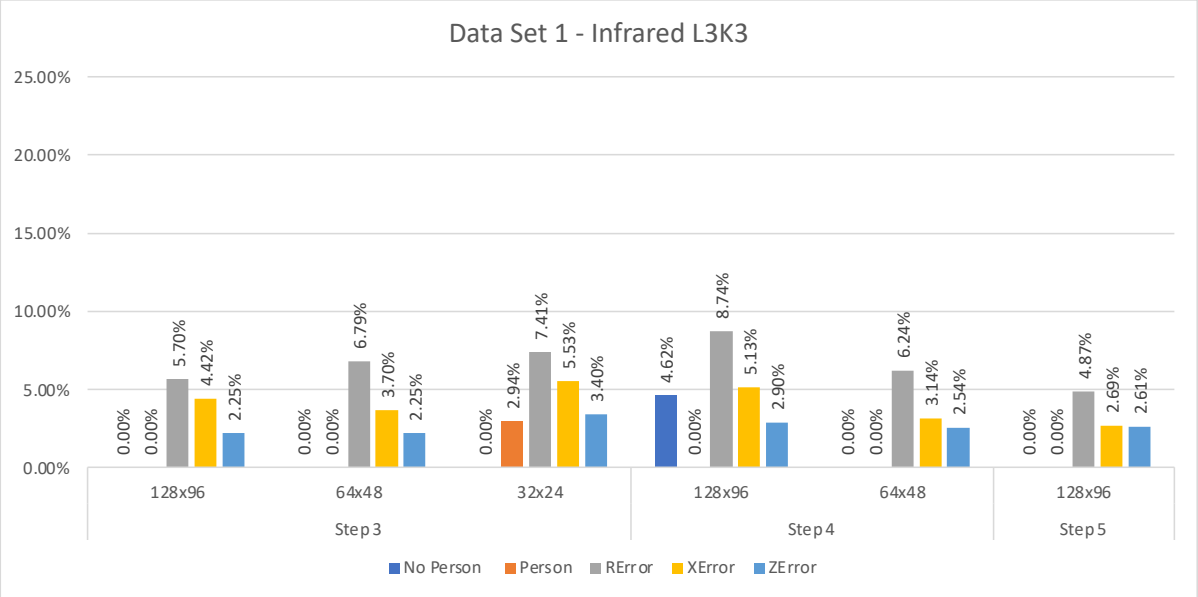

Figure 16: Data Set 1 - Infrared Models L3K3

4.1.3 Data Set 1 Radar

The initial results for the radar indicated that the inclusion of intensity of radar reflection (Q) improved results. Models with a learning rate of 10^{-3} and a kernel size of 3 had difficulties training. Overall, a kernel size of 1 had marginally better results compared to kernel size of 3. Generally, all the different radar data sets looked to have better results for 10^{-3} learning rate. The data showed no definitive results on the best combination of kernel size, learning rate or convolution layers. The main conclusion, other than kernel size 1 generally having trained better, was that there was likely not enough data to adequately train the radar models.

The model with the best overall performance was L3K1S5 256x192; the results are shown in Figure 17.

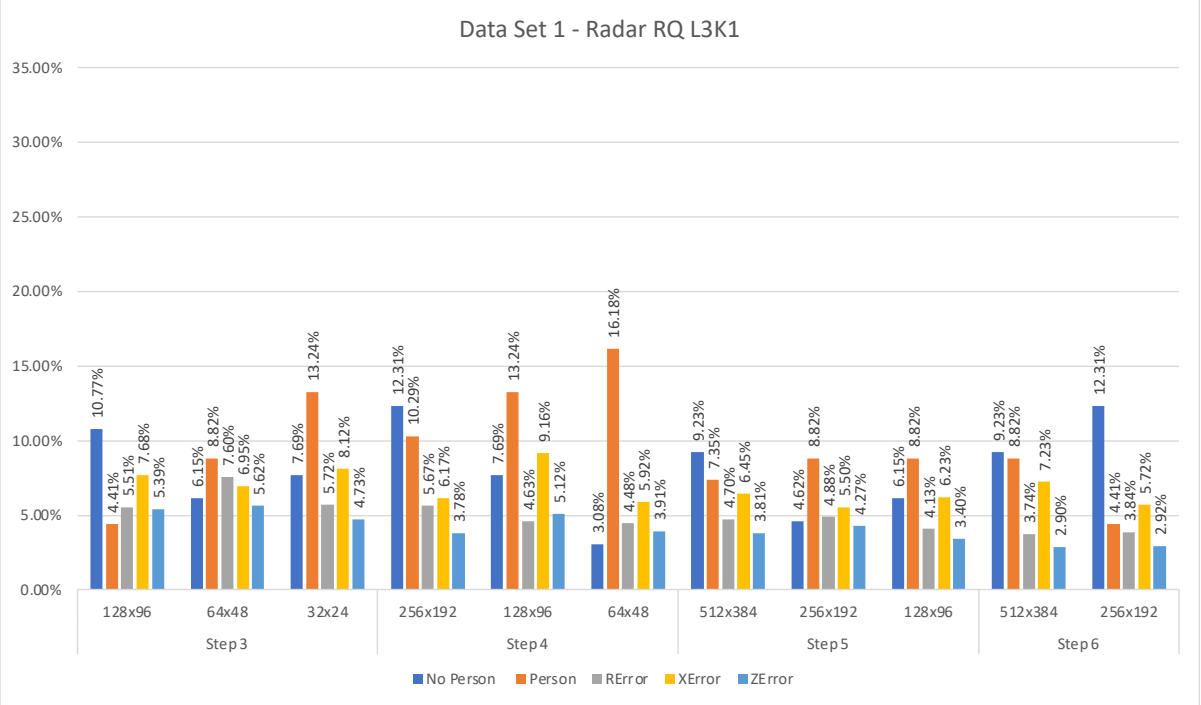

Figure 17: Data Set 1- Radar RQ Models L3K

4.1.4 Conclusion of Data Set 1 Results

Table 6 displays the best models generated for each sensor from Data Set 1. These were identified based on the overall classification and location accuracy with a secondary focus on range error (R Error). Range accuracy was prioritised over classification accuracy when identifying the better radar models as range is the feature radar is expected to contribute most to fusion models. As such, if a radar model had a slightly higher classification error but lower range location error then this model was selected over a model with lower classification but higher range error.

Data Set 1: Single Models	***No Person***	***Person***	***Average***	***R Error***	***X Error***	***Z Error***	***Average***
Camera *L4K3S5 256x192*	7.69%	7.35%	**7.52%**	6.61%	5.26%	2.37%	**4.75%**
Infrared *L3K3S5 128x96*	0.00%	0.00%	**0.00%**	4.87%	2.69%	2.61%	**3.39%**
Radar R *L4K3S6 256x192*	16.92%	10.29%	**13.61%**	8.25%	9.99%	6.24%	**8.16%**
Radar RQ *L3K3S4 64x48*	7.69%	5.88%	**6.79%**	4.21%	5.64%	2.82%	**4.22%**
Radar RV *L3K1S5 128x96*	9.23%	19.12%	**14.18%**	5.20%	9.03%	3.95%	**6.06%**
Radar RVQ *L3K1S5 256x192*	3.08%	10.29%	**6.69%**	2.94%	5.00%	2.73%	**3.56%**

Table 6: Data Set 1 - Sensor Model Accuracy

The infrared models in general performed the best with some of the models being 100% accurate (0% error) for classification. In contrast, the best camera models were around 7% error overall for classification. Accuracy for the location was also better for the infrared models than the camera models by a factor of approximately 1.3 and 2 times for R and X predictions respectively.

Classification accuracy achieved by the radar models varied depending on the type of data included in the model. The radar RQ and radar RVQ were marginally more accurate than the camera models due to the cameras underperforming on the low exposure images which would not have impacted the radar. The location accuracy achieved by the radar models also varied depending on the type of data included, with the best performing models also being the RQ and RVQ models. The location accuracy of the radar was close to that achieved by the infrared models, being less accurate with the X and Z predictions but more accurate in range R predictions.

The good performance of the infrared models when compared to the camera models was not unexpected as camera performance was impaired by low lighting images. The images that were visually hard to determine if a person was present were evaluated against the total person class of images. An example of these images is given in Figure 18 showing a camera and infrared image side by side.

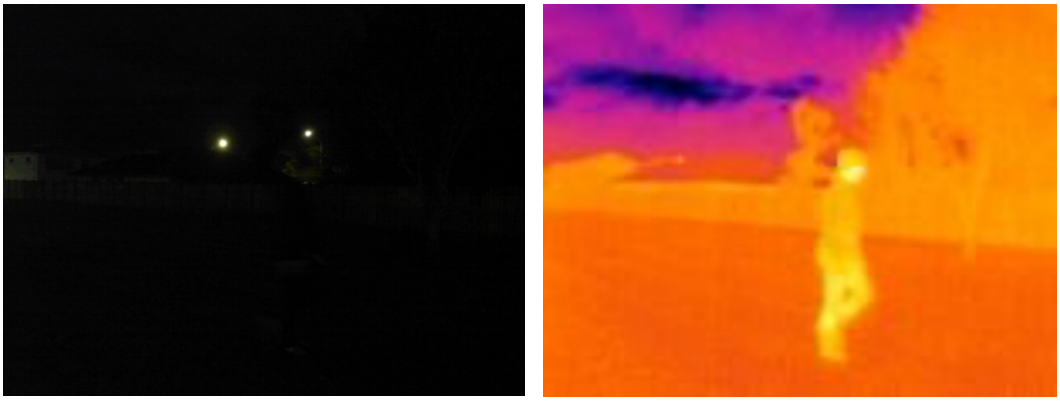

Figure 18: Low Light Camera and Infrared Images

There were only four low light images in the test data set which did not give any statistically meaningful results. The data that was in the training set demonstrated models had greater error on the 65 low-saturation images. Figure 19 shows this in the results for the camera models L4K3 which clearly shows the models were less accurate on low light images.

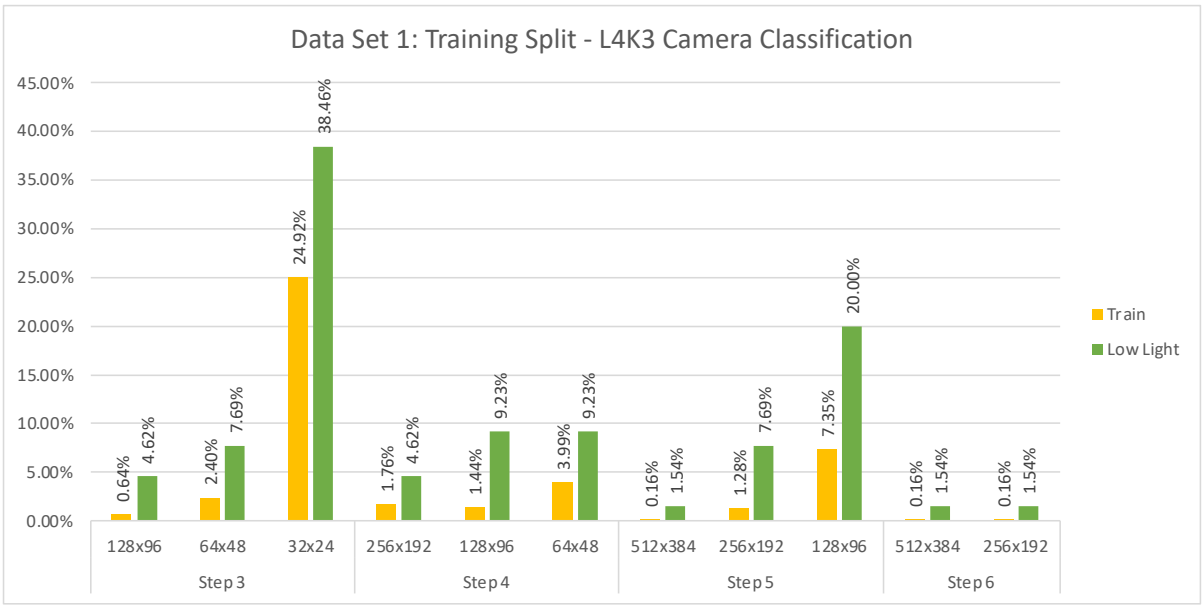

Figure 19: Camera Training vs Low Light

Another question raised was whether the data collected had sufficient noise in the infrared images compared to the noise present in the visual camera images.

4.2 Data Set 2 Models

As the models trained on Data Set 1 showed some difficulties training with certain configurations, it was suspected that the data size was too small. To test this, the models were run again with the larger Data Set 2, which added the mirror of the images in Data Set 1. Only the better models were retrained

for the cameras to test whether the models improved with a larger data set size. Consequently, radar and camera models were only trained with images sizes 256x192 and above.

Models were limited to the kernel size that demonstrated better performance for each sensor type. Camera and infrared models were generated with kernel size of 3 while the radar models were generated with a kernel size of 1. Radar RQ and RVQ were the only combinations used for the radar as results from Data Set 1 indicated they showed the most promising results.

After training the individual sensor models on Data Set 2, the best models for each sensor type were combined in late fusion models to evaluate performance of individual sensors and their contribution to fusion models.

4.2.1 Data Set 2 Camera

Camera models were trained for 5 and 6 convolution layers as these showed the most accurate results in the Data Set 1 tests. Data input size was limited to 512x384 and 256x192 pixels. Kernel size was limited to 3. Models were trained with both 10^{-3} and 10^{-4} learning rates.

The models that produced optimal results were with a learning rate of 10^{-4} with 6 convolution layers. The best overall model was L4K3S6 521x384 which can be seen in Figure 20.

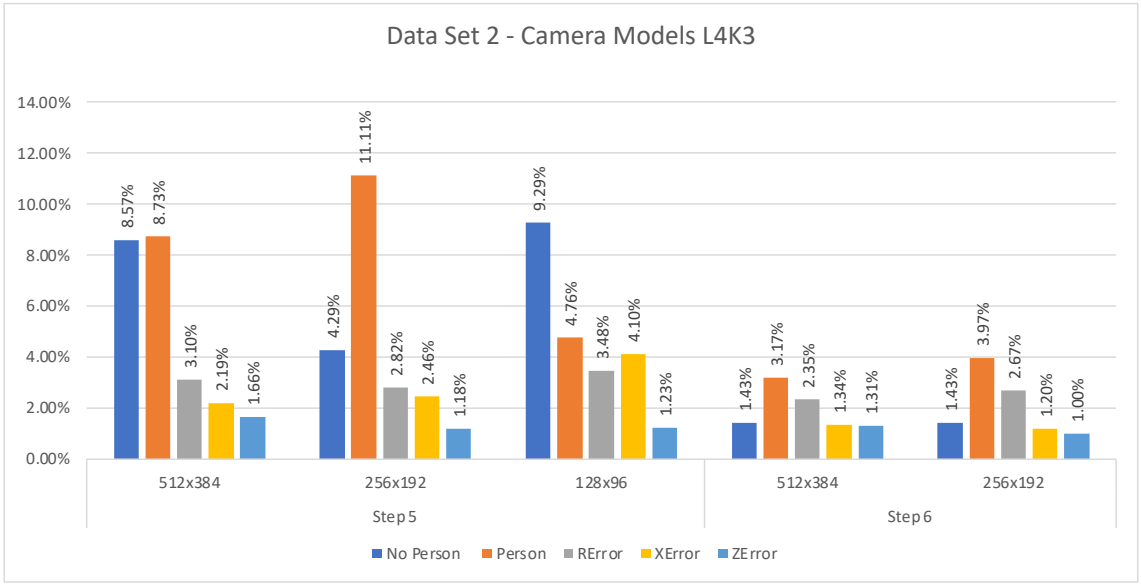

Figure 20: Data Set 2 - Camera Model L4K3

4.2.2 Data Set 2 Infrared

Infrared models were trained for 3 to 5 convolution layers with a maximum data input size of 128x96 pixels. Kernel size was limited to 3 as this size returned the best results in Data Set 1 tests. Models were trained with both 10^{-3} and 10^{-4} learning rates.

The models trained had a high accuracy for both classification and location. The model with the best classification was L3K3S4 128x96 with a classification accuracy of 100% and location error of 1.50% as shown in Figure 21. The R, X and Z predictions were not the most accurate for this model but were still competitive when comparing the overall performance of the models.

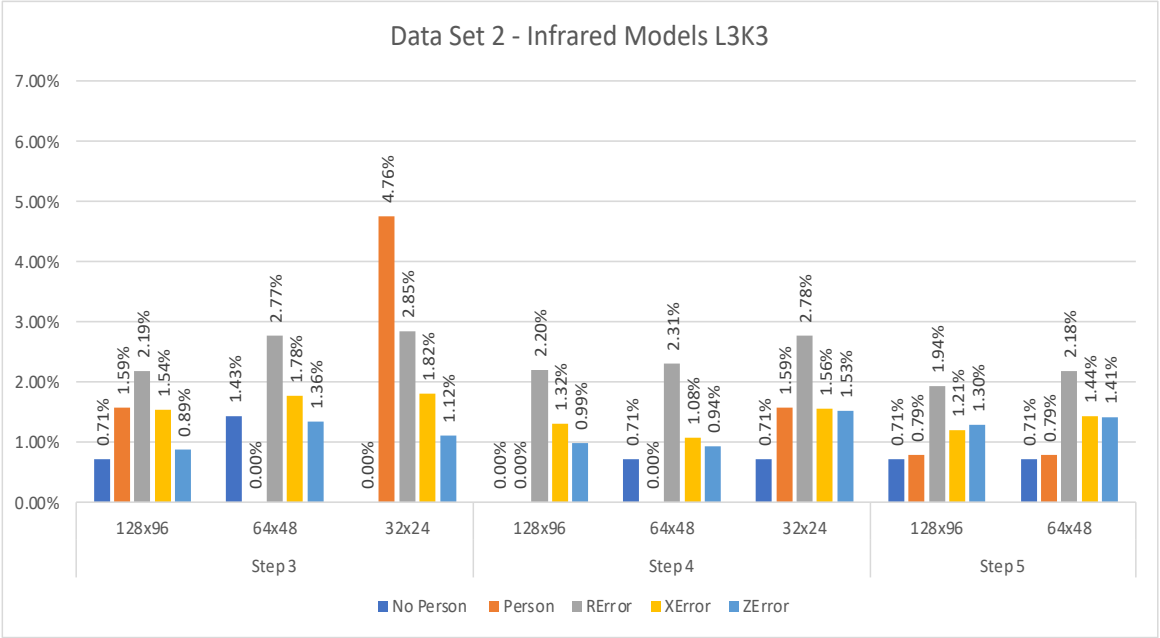

Figure 21: Data Set 2 - Infrared Model L3K3

4.2.3 Data Set 2 Radar RQ

Radar RQ models were trained for 5 and 6 convolution layers with data input sizes of 512x384 down to 128x96 pixels, with a convolution kernel size of 1 as this configuration showed the most promising results from Data Set 1. Models were trained with both 10^{-3} and 10^{-4} learning rates.

These models generally had a lower accuracy for classification and location than the cameras but had marginally better range predictions. The model with the best results was L3K1S5 128x96 with an average classification error of 5.99% and average location error of 2.22% as shown in Figure 22.

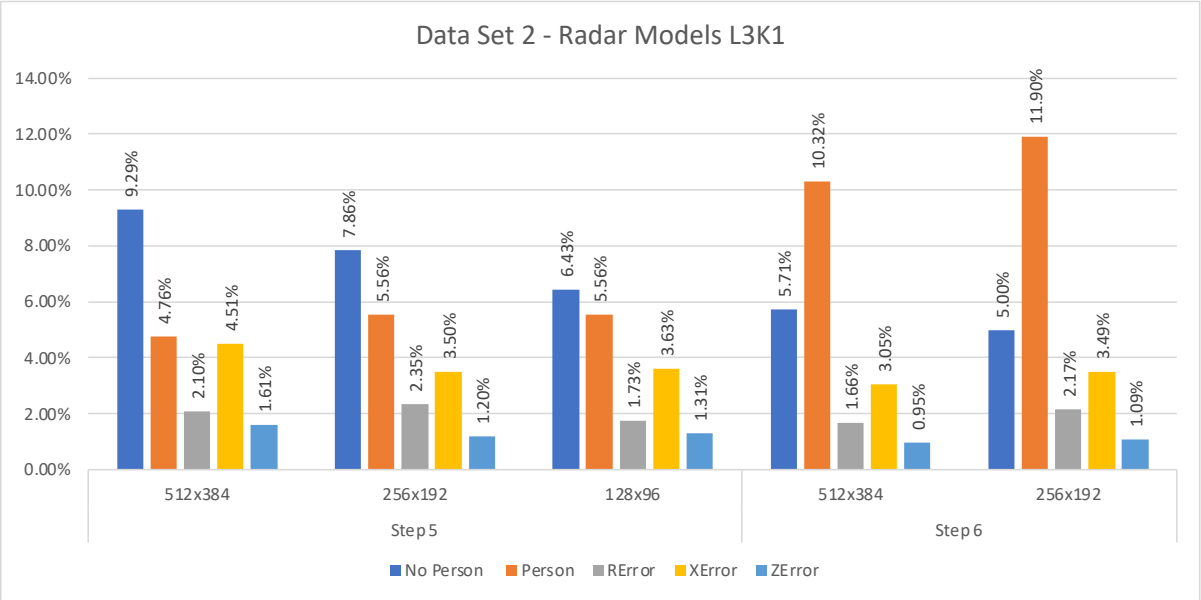

Figure 22: Data Set 2 - Radar RQ Models L3K

4.2.3.1 Data Set 2 Radar RVQ

Radar RVQ models were trained for 5 and 6 convolution layers with data input sizes of 512x384 down to 128x96 pixels with convolution kernel size of 1. Models were trained with both 10^{-3} and 10^{-4} learning rates.

These models had a lower accuracy for classification and location than the cameras. They had a higher accuracy than radar RQ but with no significant difference in accuracy for range R predictions. The model with the best classification was L3K1S6 128x96 with an average classification error of 4.21% and average location error of 2.14% as seen in Figure 23.

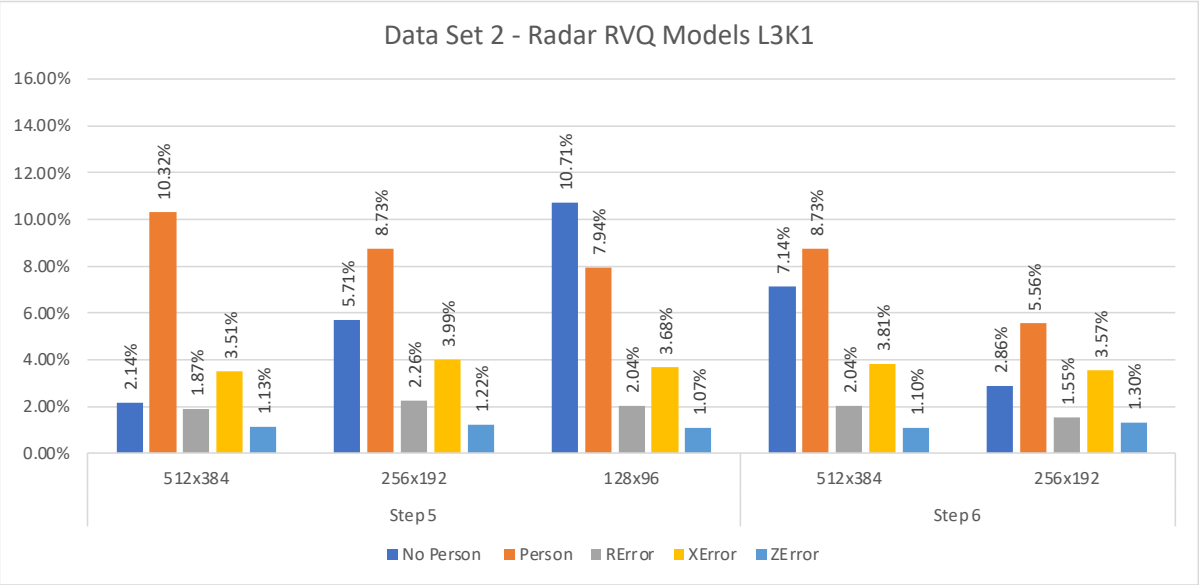

Figure 23: Data Set 2 - Radar RVQ Models L3K1

4.2.4 Data Set 1 Models vs Data Set 2 Models

Data Set 2 camera models greatly improved in both classification and location accuracy compared to Data Set 1 models, with the average classification improving by 5.2 percentage points. This is under half the error of Data Set 1 models, going from 7.52% to 2.3% error. The average location error for the Data Set 2 models showed an improvement of 3.08 percentage points compared to the best models of Data Set 1.

The infrared also showed improvement for location accuracy of 1.89 percentage points.

The radar RQ model for Data Set 2 showed a slight decrease of 1.07 percentage points in accuracy for classification, whereas the radar RVQ model improved by a significant 2.48 percentage points. Both the radar RQ and radar RVQ showed improvement for location accuracy: 2.13 and 1.42 percentage points respectively. Radar again showed a greater increase in range predictions accuracy than the camera and infrared models. These results validated that the size of Data Set 1 was too small to reliably train models.

The infrared models were still significantly more accurate than camera, giving weight to the speculation that there was insufficient infrared noise. Additionally, it seemed the positioning of the people in the image frame was too consistent, leading to the camera and infrared having more accurate range prediction rates than expected.

Table 7 and Table 8 show the full results for the best models in Data Sets 1 and 2.

Data Set 1 Single Models	***No Person***	***Person***	***Average***	***R Error***	***X Error***	***Z Error***	***Average***
Camera *L4K3S6 512x384*	7.69%	7.35%	**7.52%**	6.61%	5.26%	2.37%	**4.75%**
Infrared *L3K3S5 128x96*	0.00%	0.00%	**0.00%**	4.87%	2.69%	2.61%	**3.39%**
Radar RQ *L3K1S5 128x96*	7.69%	5.88%	**6.79%**	4.21%	5.64%	2.82%	**4.22%**
Radar RVQ *L3K1S5 256x192*	3.08%	10.29%	**6.69%**	2.94%	5.00%	2.73%	**3.56%**

Table 7: Data Set 1 - Best Models

Data Set 2 Single Models	***No Person***	***Person***	***Average***	***R Error***	***X Error***	***Z Error***	***Average***
Camera *L4K3S6 512x384*	1.43%	3.17%	**2.30%**	2.36%	1.34%	1.31%	**1.67%**
Infrared *L3K3S4 128x96*	0.00%	0.00%	**0.00%**	2.20%	1.32%	0.99%	**1.50%**
Radar RQ *L3K1S5 128x96*	8.57%	7.14%	**7.86%**	1.40%	3.49%	1.39%	**2.09%**
Radar RVQ *L3K1S6 256x192*	2.86%	5.56%	**4.21%**	1.55%	3.57%	1.30%	**2.14%**

Table 8: Data Set 2 - Best Models

4.3 Data Set 2 Fused Models

The best single sensor models from the Data Set 2, i.e. those listed Table 8, were fused together in all combinations of sensor types to evaluate their effect on model performance. Models are denoted by the serial number specifying which sensors are included: camera (C), infrared (I) and radar types (RQ or RVQ). The learning rate parameters of 10^{-3} and 10^{-4} are specified by L3 and L4, with frozen (F) or not frozen (N) being used to identify if the feature layers were set trainable or not trainable. For example, a model including the camera, infrared and radar RQ, with the feature layers frozen and trained at a learning rate of 10^{-3} would be denoted as CIRQ-L3F.

Fusion models were evaluated to determine the most accurate results for each sensor combination. These models were then selected for further analysis. Models that were close in accuracy when compared singularly on classification or location were further evaluated based on overall performance. If one model was the most accurate in both classification and location, no other models for that sensor combination were included in the results.

The first conclusion was that fused models generally performed better in classification compared to the single sensor camera and radar models, with an average classification error of 0.8% compared to 2.3% and 4.21% respectively. It is also important to note that the most accurate fused model that excluded infrared had a higher classification accuracy than the single sensor camera model did, with a 1.94% error compared to 2.30%, further justifying that the radar can contribute to classification in a fused model.

As previously discussed, the single sensor infrared model had 0% classification error and performed better in both classification and location than any fusion model.

The fused models were less accurate at location predictions than all the other single sensor models, with the best fusion model location error being 2.23% while the camera and infrared models performed at 1.67% and 1.50% respectively.

The factors that caused the classification accuracy to increase but location accuracy to decrease are not clear, though it is likely to be related to the uniformity of the data enabling the single sensors to better predict location than would be achievable in a more complex environment. It is assumed that the additional information included through fusion of different sensors led to the models being unable to learn the specific patterns such as height or position that allowed the single sensors to predict location to a high degree of accuracy. It might also be that the increased classification accuracy required the models to predict locations on more complicated images that would otherwise have been falsely classified as no person.

Other factors that could be contributing to the decrease in accuracy could be the 0.5 second lead the radar data had on the camera and infrared sensor, the lack of infrared noise, the lack of positional variation of the person in the data, or any combination of these factors. There did not appear to be a correlation between higher accuracy, the learning rates, and whether the models were frozen or not.

Though the fusion models generally demonstrated an increase in classification accuracy as was initially hypothesized, the results were inconclusive given the infrared was the overall best performing model, and the fused models were less accurate in location accuracy.

The best fusion models are laid out in Table 9 and in graph form in Figure 24.

Data Set 2 Fusion Models	***No Person***	***Person***	***Average***	***R Error***	***X Error***	***Z Error***	***Average***
CIRVQ-L3F	0.00%	1.59%	**0.80%**	3.37%	2.67%	1.73%	**2.59%**
CIRVQ-L3N	0.71%	1.59%	**1.15%**	2.73%	2.61%	1.73%	**2.36%**
CI-L4N	0.00%	0.00%	**0.00%**	3.60%	1.66%	1.44%	**2.23%**
IRVQ-L3N	0.71%	0.79%	**0.75%**	2.85%	2.76%	1.75%	**2.45%**
IRVQ-L4N	0.00%	4.76%	**2.38%**	2.62%	3.26%	1.82%	**2.57%**
CRVQ-L3F	0.71%	3.17%	**1.94%**	3.59%	3.57%	1.73%	**2.96%**
CRVQ-L4F	0.71%	4.76%	**2.74%**	2.85%	2.76%	1.69%	**2.43%**

Table 9: Data Set 2 - Best Fusion Models

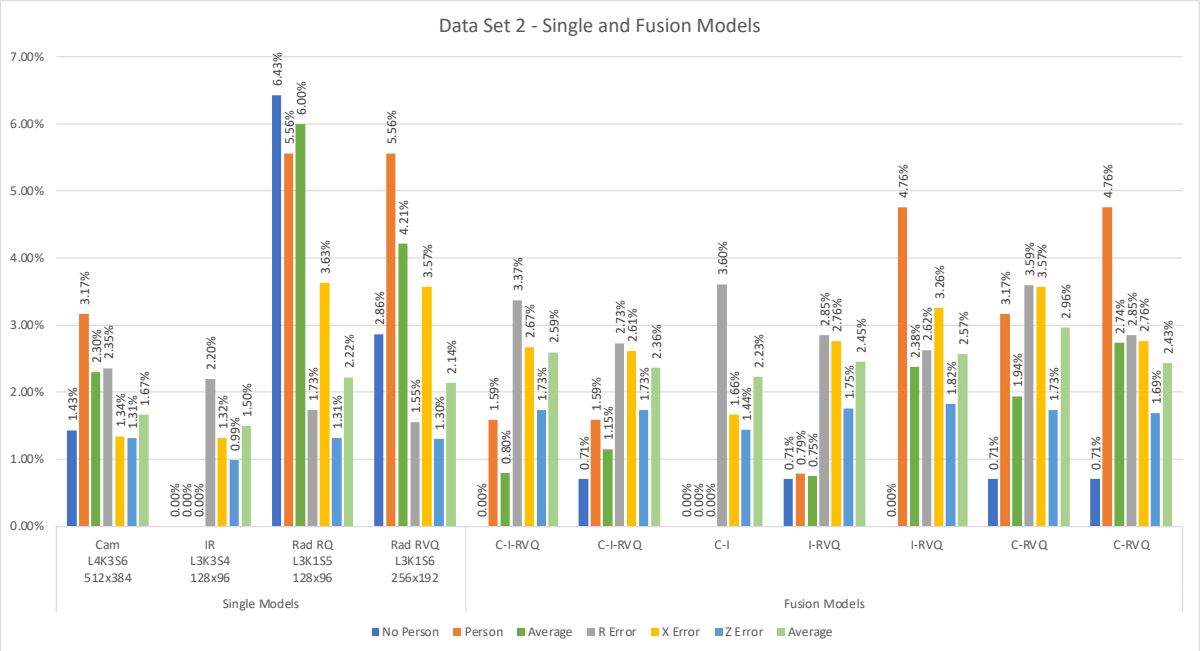

Figure 24: Data Set 2 - Best Models Single Sensor vs Fusion

4.4 Data Set 3

Data Set 3 was created by adding an additional 379 images to Data Set 2, with 118 including a person. These additional images also included hot objects such as a warm car and motorcycle, and ensured a varied horizontal and vertical positioning of the person in the data frame. This data was again mirrored to provide a data set of 3414 samples with 1482 including people.

Models were trained on Data Set 3 for each individual sensor through all data input resolutions from 512x384 down to 32x24, learning rates of 10^{-3} and 10^{-4} and a varied number of convolutional layers ranging from 3 to 6. Only radar RQ and RVQ were trained, as previous models indicated the inclusion of Q had more accurate results.

The most accurate performing models of each sensor were evaluated based on classification accuracy, location accuracy and overall accuracy to identify the most accurate models for each sensor type. These were then combined in sensor fusion models with every combination of sensor being fused, trained, and tested to better evaluate how each sensor contributed to model performance.

4.4.1 Data Set 3 Camera

Camera models were trained for 3 and 6 convolution layers with all image input sizes from 512x384 to 32x24 pixels. Kernel size 3 was used as it showed better results in earlier camera models. Models were trained with both 10^{-3} and 10^{-4} learning rates.

The camera models that produced more accurate results were with an input size of 256x192 pixels, with L4K3S6 256x192 showing the best classification and location results, a classification error of 4.71% and 6.43% for no person and person respectively, and average classification and location errors of 5.57% and 4.41% respectively as shown in Figure 25.

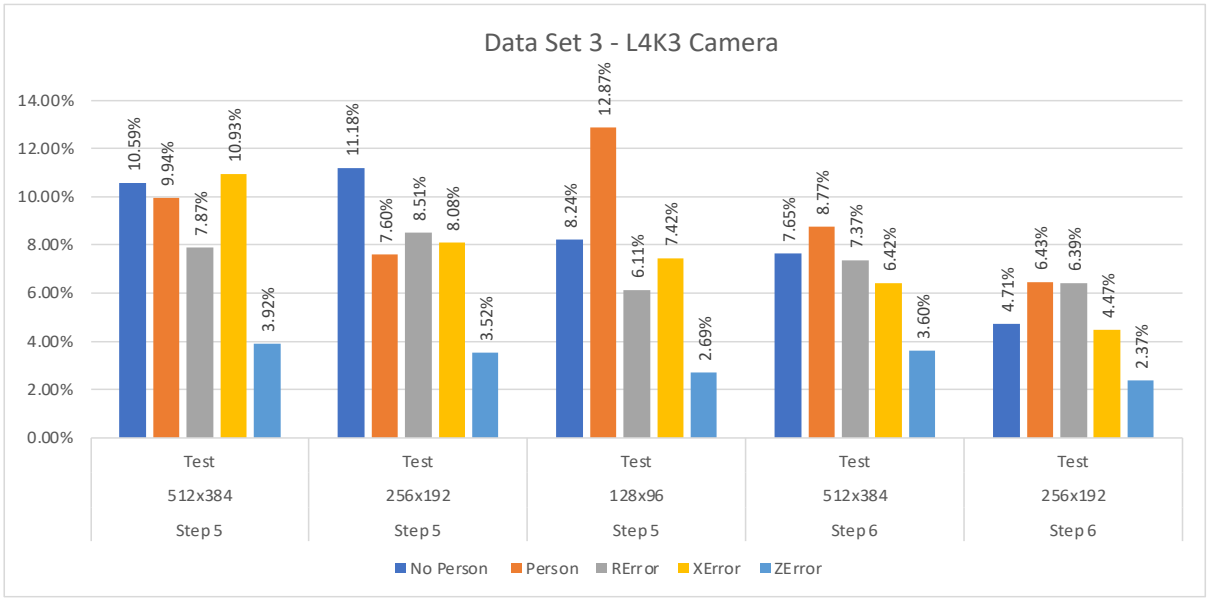

Figure 25: Data Set 3 - Camera Models L4K3

When compared to Data Set 2 results, models showed a drop in range accuracy, giving justification to the idea that in Data Set 2 the uniform positioning of the person in the image frame allowed models to accurately infer the range from a person's height position or aspect ratio. Positional variation introduced in Data Set 3 made it more difficult for the models to make this inference.

For the camera model L4K3S6 256x192, many of the false negatives included images with very low lighting. Other false negatives often had a small-scale person in the background, often with objects in the foreground or background making it harder to distinguish the person. Objects in the background such as a motorcycle or light pole largely looked to be the cause of false positives. Figure 26 shows two examples of false positives on the left and two false negatives on the right.

Figure 26: Data Set 3 - Camera False Classification

4.4.2 Data Set 3 Infrared

The infrared models were trained for 3 to 5 convolutions and max pool layers and a maximum data input size of 128x96 pixels. The kernel size was set at 3. Learning rates of 10^{-3} and 10^{-4} were used.

The model with the best classification was L3K3S5 128x96 with an error of 0.59% and 1.75% for no person and person respectively. The average classification and location errors were 1.17% and 4.48%. Though this model was not the most accurate model for R, X and Z position, it was still competitive when comparing the overall performance with other Data Set 3 infrared models. This can be seen in Figure 27.

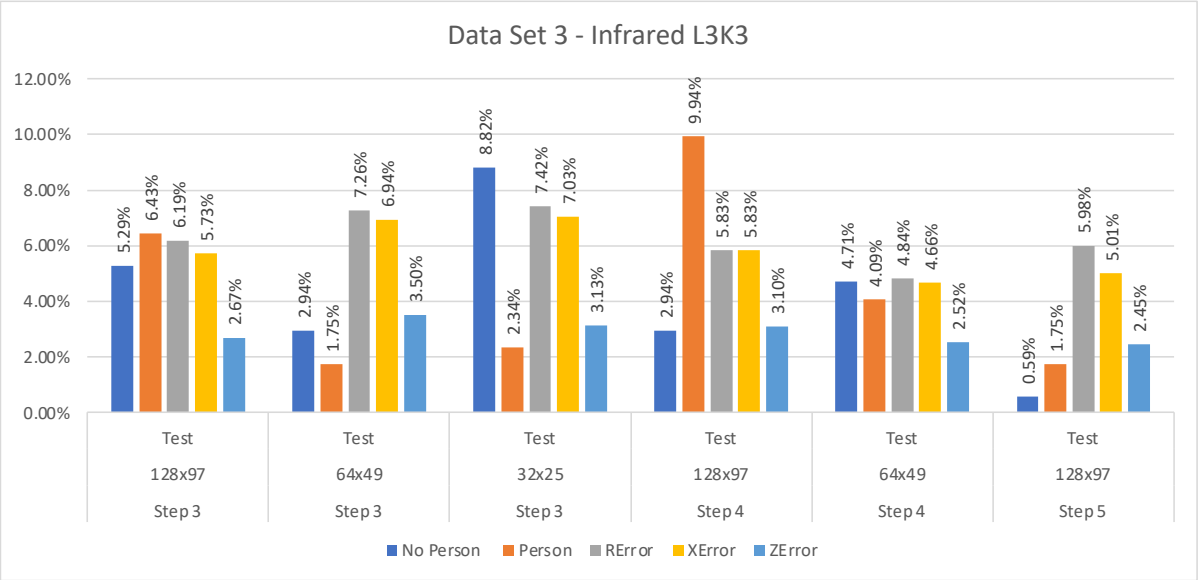

Figure 27: Data Set 3 - Infrared Models L3K3

For the L3K3S5 128x96 IR model, the false positives often included an object in the background that had a sharp infrared contrast to the environment such as a bike or hot spot on the ground. False negatives often included a person in the background being small relative to the size of the image, often also with a warm object in the foreground. Figure 28 shows two false positives on the left and two false negatives on the right.

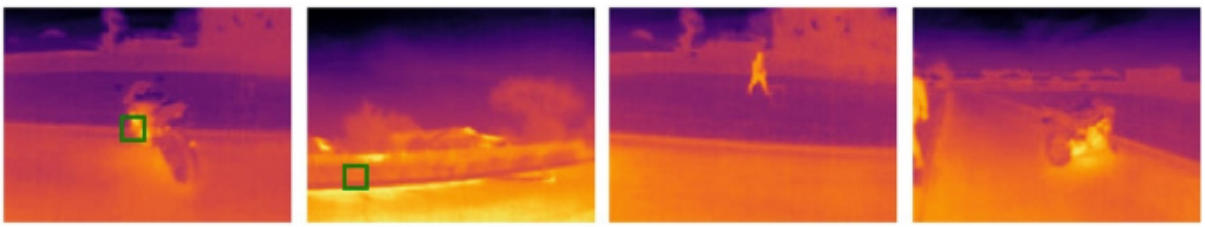

Figure 28: Data Set 3 - Infrared False Classification

Infrared models trained on Data Set 3 were not able to reach the 100% classification accuracy seen in previous models. This was expected and confirms a lack of infrared noise in Data Sets 1 and 2 led to the infrared models having unrealistically accurate results. However, the better models from Data Set 3 still had a reasonably high accuracy for classification and location. Similarly to the Data Set 3 camera models, the infrared models also showed a drop in range accuracy when compared to those in Data Set 2 further reinforcing the idea that for previous data sets the consistency of the positioning of the person in the image frame allowed the models to accurately infer the range of the person.

4.4.3 Data Set 3 Radar RVQ

The radar RVQ models were trained for 3 to 6 convolution and max pool layers with data input sizes from 512x384 pixels down to 32x25. The kernel size was set at 1. Learning rates of 10^{-3} and 10^{-4} were used.

The model with the best classification was L3K1S6 256x192 as seen in Figure 29, with a classification error of 5.88% and 7.02% for no person and person present respectively. The average classification losses were 6.45% and 4.66%. There were models with slightly better range accuracy but with greater overall error.

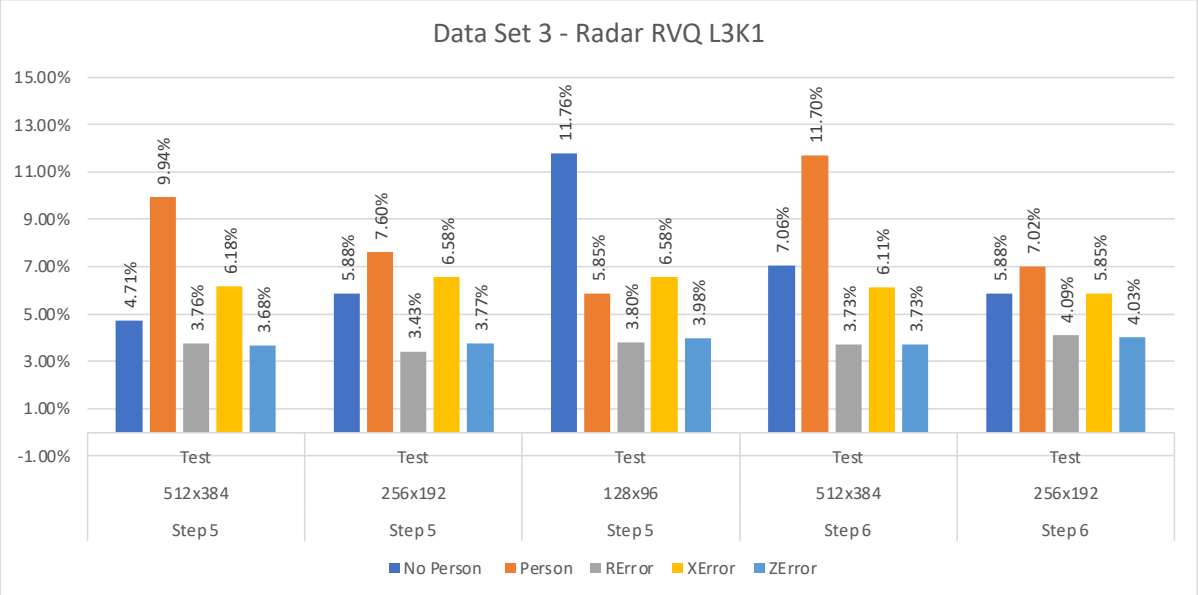

Figure 29: Data Set 3 - Radar RVQ Models L3K1

4.4.4 Data Set 3 Radar RQ

The better radar RQ models had marginally higher classification and location accuracy than the radar RVQ models. The most accurate Data Set 3 RQ models were selected for further analysis.

The radar RQ models were trained for 3 to 6 convolutions layers with data input sizes from 512x384 pixels down to 32x25. The kernel size was set at 1. Learning rates of 10^{-3} and 10^{-4} were used.

The better radar RQ models had lower classification and location accuracy than the camera models except for range location predictions. The model with the best classification was L4K1S6 256x192 with a classification error of 5.88% and 6.43% for no person and person respectively. The average classification and location errors were 6.16% and 4.82%. There were models with better range predictions but they were discounted due to the greater overall error. For perspective the performance for the RQ L4K1 models is shown in Figure 30.

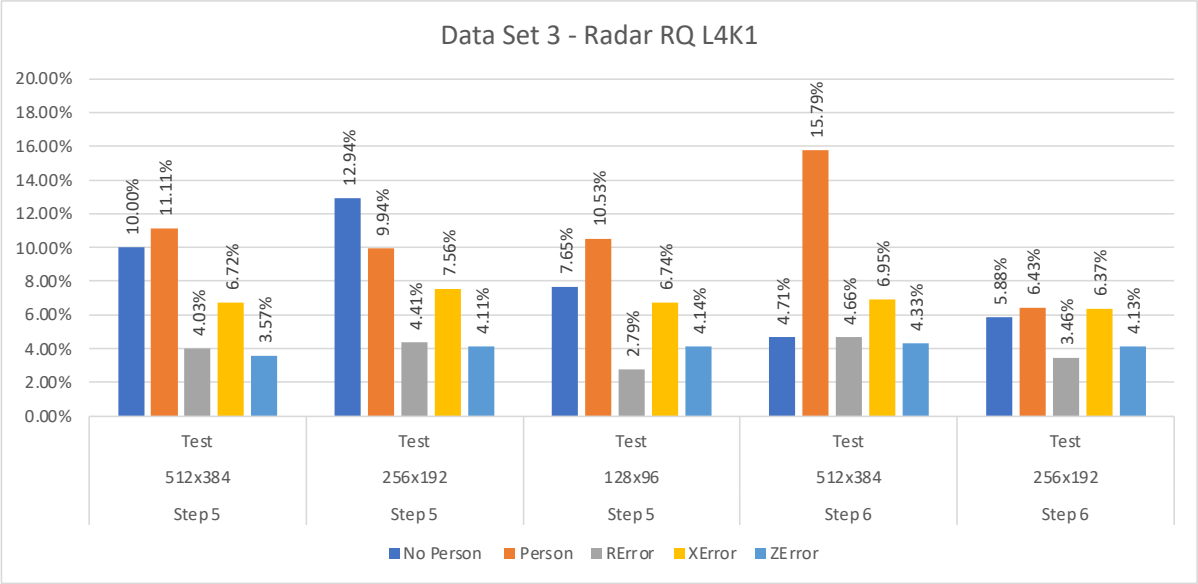

Figure 30: Data Set 3 - Radar RQ Models L4K1

The radar RQ gives the indication that for both false positive and false negative classifications the radar is less accurate when point clouds are highly or sparsely populated. Figure 31 shows two false positive examples on the left and two false negative examples on the right.

Figure 31: Data Set 3 - Radar False Classification

4.4.5 Data Set 2 vs Data Set 3 Single Models

The results for the best single models for Data Set 2 and Data Set 3 models are compared in Table 10, Table 11 and Figure 31.

Data Set 2 Single Models	***No Person***	***Person***	***Average***	***R Error***	***X Error***	***Z Error***	***Average***
Camera *L4K3S6 512x384*	1.43%	3.17%	**2.30%**	2.36%	1.34%	1.31%	**1.67%**
Infrared *L3K3S4 128x96*	0.00%	0.00%	**0.00%**	2.20%	1.32%	0.99%	**1.50%**
Radar RQ *L3K1S5 128x96*	8.57%	7.14%	**7.86%**	1.40%	3.49%	1.39%	**2.09%**
Radar RVQ *L3K1S6 256x192*	2.86%	5.56%	**4.21%**	1.55%	3.57%	1.30%	**2.14%**

Table 10: Data Set 2 - Best Sensor Models

Data Set 3 Single Models	***No Person***	***Person***	***Average***	***R Error***	***X Error***	***Z Error***	***Average***
Camera *L4K3S6 256x192*	4.71%	6.43%	**5.57%**	6.39%	4.47%	2.37%	**4.41%**
Infrared *L4K3S5 128x96*	0.59%	1.75%	**1.17%**	5.98%	5.01%	2.45%	**4.48%**
Radar RQ *L3K1S6 256x192*	5.88%	6.43%	**6.16%**	3.46%	6.37%	4.13%	**4.65%**
Radar RVQ *L3K1S6 256x192*	5.88%	7.02%	**6.45%**	4.09%	5.85%	4.03%	**4.66%**

Table 11: Data Set 3 - Best Sensor Models

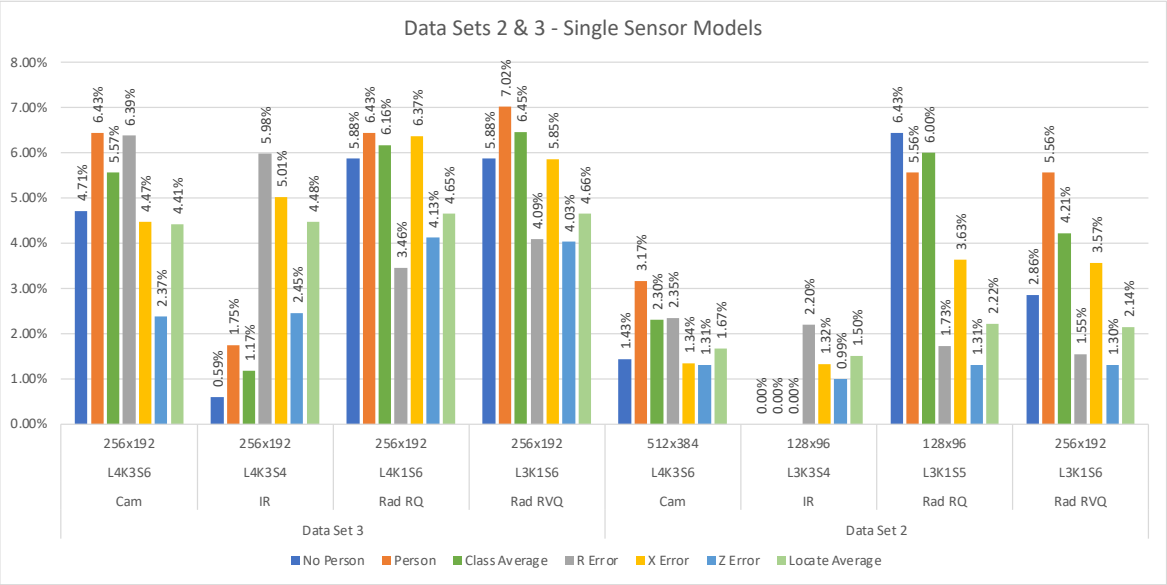

Figure 32: Data Sets 2 and 3 - Fusion Models

Results show a general decrease in accuracy across all sensor models including infrared. Infrared models were still much more accurate in classification than the camera, but only marginally better in location accuracy. For comparison, Data Set 2 infrared models had a much higher location accuracy in comparison to the other sensors. This can be seen in Table 12 which shows the comparative ratio of Data Set 3 errors to Data Set 2 errors.

Model Location Error Ratio (Data Set 3 / Data Set 2)	***Classification Error Ratio***	***Location Error Ratio***	***Range (R) Error Ratio***
Camera	2.42	2.64	2.72
Infrared	DIV/0	2.99	2.72
Radar RQ	1.03	2.09	2.00
Radar RVQ	1.53	2.18	2.64

Table 12: Model Location Error Ratio

The decrease in accuracy between the radar models, and the camera and infrared models, was much less in Data Set 3 compared to Data Set 2. This further justifies the idea that the earlier data was too uniform, allowing the camera and infrared models to predict the location from information that would likely not be sufficient in a real scenario. It also corroborates infrared models being unusually accurate due to lack of vertical variation in the positioning of the person in the image as well as there being a lack of infrared noise in the original data set.

4.5 Data Set 3 Fused Models

Data Set 3 fusion models were trained with the structure and weights from the single sensor models that had the best performance for each individual sensor. These models are listed as follows, as per Table 11:

- Camera - L4K3S6 256x192
- Infrared - L4K3S5 128x96
- Radar RQ - L3K1S6 256x192
- Radar RVQ - L3K1S6 256x192

These models were combined in a late fusion method. Fusion models were trained with learning rates of 10^{-3} and 10^{-4}, with the convolution feature extraction layers both not frozen and frozen.

All combinations of the individual sensors were trained. Fusion models were generated with both radar RQ and RVQ respectively to evaluate if there was a difference in performance between the data types in fusion models as radar RQ was only more accurate than the RVQ model by 0.29 percentage points in classification and 0.01 percentage points in location.

4.5.1 Data Set 3 Fusion: Camera, Infrared and Radar

The best results for the camera, infrared and radar fusion were the RQ-L4F and RQ-L4N for classification and location respectively. RQ-L4N was the more accurate by an average error of 1.86% compared to 2.09%.

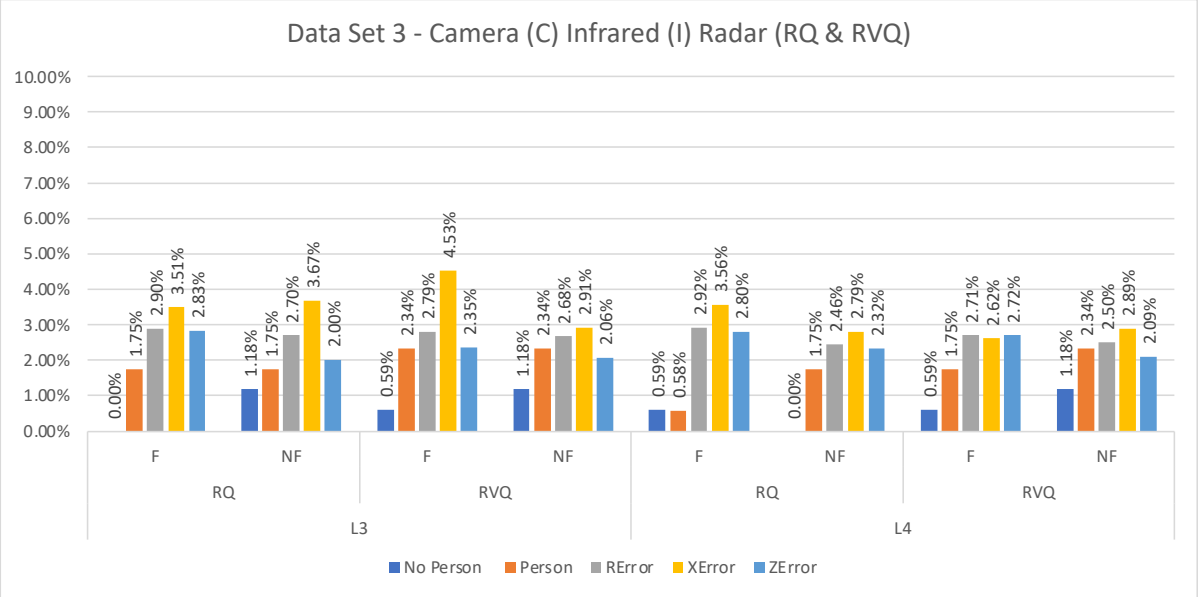

Figure 33: Data Set 3 - Fusion Models (Camera, Infrared & Radar)

4.5.2 Data Set 3 Fusion: Camera and Infrared

The best results for the camera and infrared fusion were the L4F and L4N models for location and classification respectively. Overall L4N was the more accurate with average error of 2.80% compared to 2.95%.

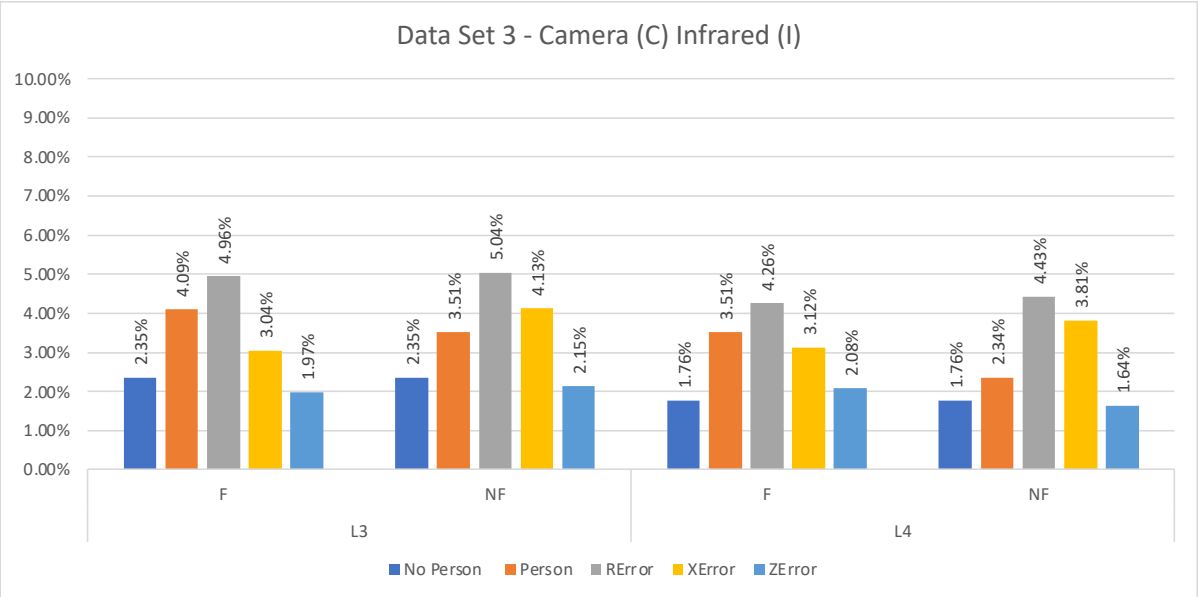

Figure 34: Data Set 3 - Fusion Models (Camera & Infrared)

4.5.3 Data Set 3 Fusion: Infrared and Radar

The best results for the infrared and radar fusion were the RQ-L4N and RVQ-L4N. RVQ-L4N had a marginally better range prediction but was less accurate overall. While the models were quite close in overall accuracy, RQ-L3 was slightly more accurate.

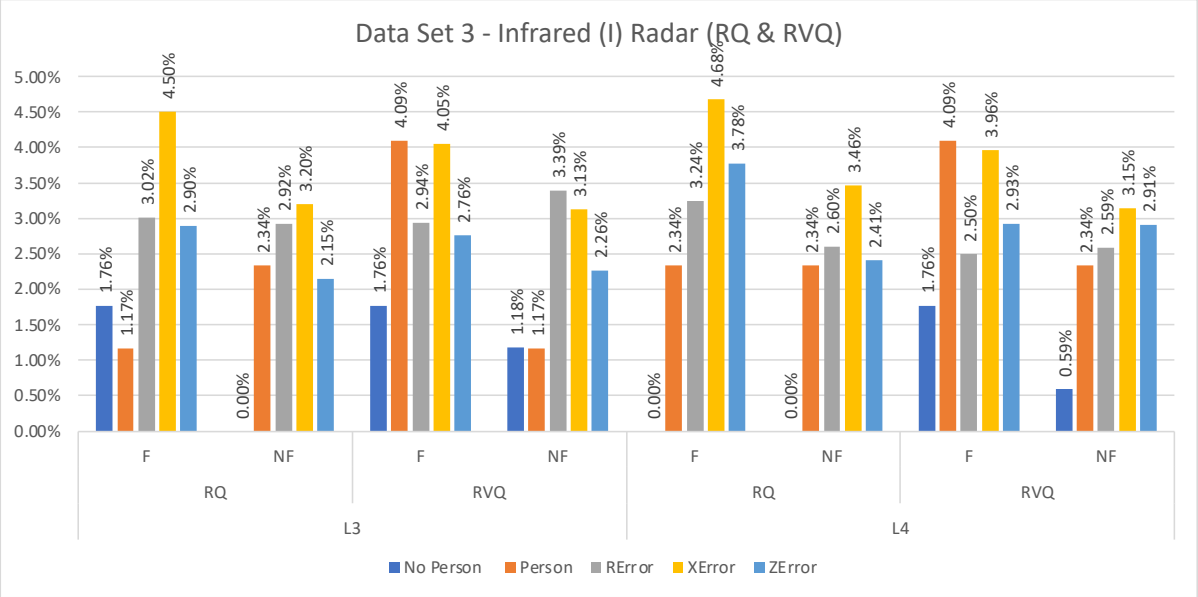

Figure 35: Data Set 3 - Fusion Models (Infrared & Radar)

4.5.4 Data Set 3 Fusion: Camera and Radar

The best results for the infrared and radar fusion were the RQ-L4N and RVQ-L4F for location and classification respectively. RVQ-L4F was the more accurate model overall with an average error of 2.57% compared to 2.65% for RQ-L4N.

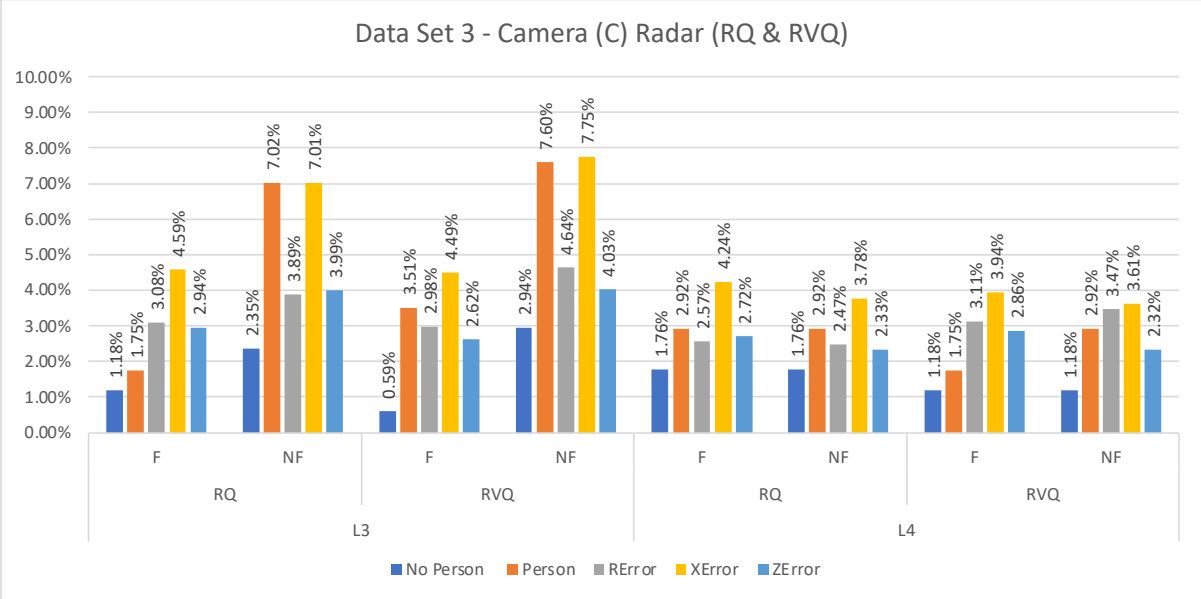

Figure 36: Data Set 3 - Fusion Models (Camera & Radar)

4.5.5 Data Set 3 Fusion vs Data Set 3 Single Models

Table 13 summarises the single sensor models used in the fusion.

Data Set 3 Single Models	***No Person***	***Person***	***Average***	***R Error***	***X Error***	***Z Error***	***Average***
Camera	4.71%	6.43%	**5.57%**	6.39%	4.47%	2.37%	**4.41%**
Infrared	0.59%	1.75%	**1.17%**	5.98%	5.01%	2.45%	**4.48%**
Radar RQ	5.88%	6.43%	**6.16%**	3.46%	6.37%	4.13%	**4.65%**
Radar RVQ	5.88%	7.02%	**6.45%**	4.09%	5.85%	4.03%	**4.66%**

Table 13: Data Set 3 - Best Single Sensor Models

The most accurate fusion models of each type are listed in Table 14 with a graphical comparison in Figure 37. The models showed a unanimous improvement on the single models in both classification and location accuracy. The lowest average classification error of the single models was 1.17% for the infrared model, with the other single models being above 5.5% error. The worst average classification error for the fusion models was for the camera-infrared L4 Fixed model, at 2.64%. The best fusion model was CIRQ-L4F, which included all sensors with an average classification error of 0.59%.

Data Set 3 Fusion	*No Person*	*Person*	*Average*	*R Error*	*X Error*	*Z Error*	*Average*
CIRQ-L4F	0.59%	0.58%	**0.59%**	2.92%	3.56%	2.80%	**3.09%**
CIRQ-L4N	0.00%	1.75%	**0.88%**	2.46%	2.79%	3.32%	**2.86%**
CI-L4F	1.76%	3.51%	**2.64%**	4.26%	3.12%	2.08%	**3.15%**
CI-L4N	1.76%	2.34%	**2.05%**	4.43%	3.81%	1.64%	**3.29%**
IRQ-L4N	0.00%	2.34%	**1.17%**	2.60%	3.46%	2.41%	**2.82%**
IRVQ-L4N	0.59%	2.34%	**1.47%**	2.59%	3.15%	2.91%	**2.88%**
CRVQ-L3F	1.76%	2.92%	**2.34%**	2.47%	3.78%	2.33%	**2.86%**
CRVQ-L4F	1.18%	1.75%	**1.47%**	3.11%	3.94%	2.86%	**3.30%**

Table 14: Data Set 3 - Best Fusion Models

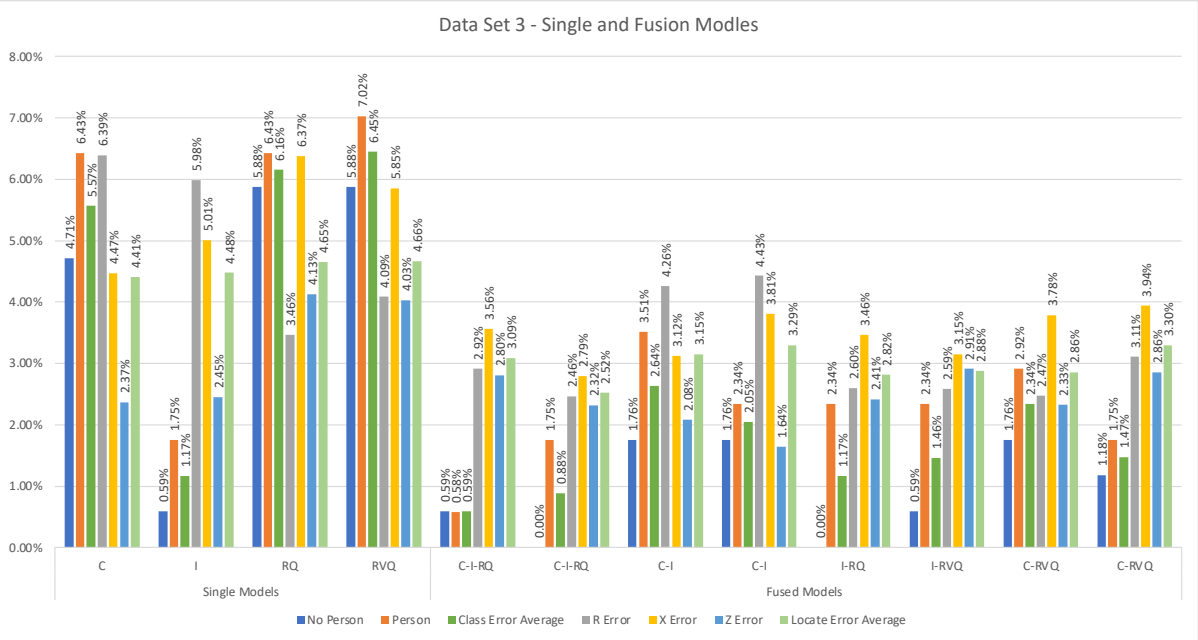

Figure 37: Data Set 3 - Fusion and Sensor Models

The best average location error for Data Set 3 single models was 4.41% while the worst of the fused models was 4.12%. The best fused model was CIRQ-L4N with an average location error of 2.52%. Looking at the five data samples that were classified in error from these two models, the sample images generally contained cases of infrared noise, a person in the background, noisy radar or a combination of these. Figure 38: Camera Infrared Radar False Classification details samples of false positive (top) and false negative (bottom). For images with a false positive, a red or green box has been drawn over the image where the model predicted the person.

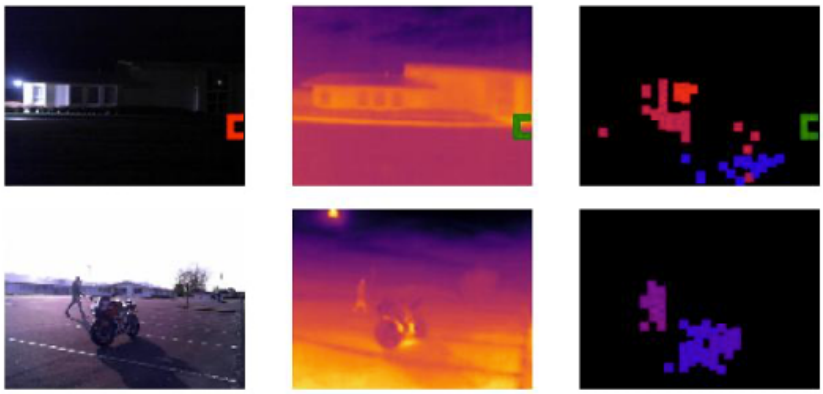

Figure 38: Camera Infrared Radar False Classification

When comparing the infrared-radar and camera-radar models for classification, the infrared-radar models generally performed better than those with the camera. This is expected given the classification for the infrared-radar model had a better accuracy with an error of 3.23% compared to 3.82% for the camera model and given the visual camera did not perform as well during the low saturation samples. Figure 39 shows a false positive set on the left and a false negative on the right for the low saturation images the camera incorrectly classified.

Figure 39: Camera Radar False Classification

The infrared classification error came from one picture with the person half out of the frame and three pictures with a hot motorcycle or car in the picture as shown in Figure 40.

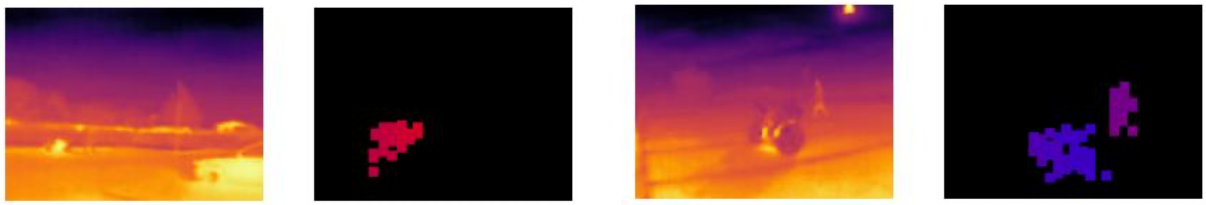

Figure 40: Infrared Radar False Classification

The camera-infrared fusion model performed worse in classification and location compared the other fusion models. As can be seen from samples of false classification shown in Figure 41, this model had difficulty classifying a person in the background when there was an object in the foreground that

also introduced infrared noise. This may have been due to the architecture of the models being more tailored for one particular scale, i.e. to detect larger objects in the foreground rather than a small object in the background [75-77]. The model also had some difficulty classifying a person half out of frame which is understandable. Both issues might be mitigated by using a larger, more complex data set.

Figure 41: Camera Infrared False Classification

Against initial expectations the camera-infrared fusion model was less accurate for classification than the infrared-radar fusion models, given that the camera model was more accurate for classification than the better radar model, with classification errors of 5.57% and 6.16% respectively.

When also considering models outside of the best performing ones, a trend was observed of models with high classification accuracy tending to have a lower location accuracy. Though the best models often had good accuracy in both location and classification, it might be that classification could be compromised due to the increased difficulty of trying to optimize the location prediction, with little helpful information for range from the camera and infrared sensors.

The fusion models trained on Data Set 3 showed a marked improvement overall over the performance of Data Set 3 single sensor models in all aspects to the point that the fusion models were almost comparable to the classification accuracy of the single sensor models trained on the less complex Data Set 2.

4.6 Data Set 3 vs Data Set 2 Fusion Models

Though fusion models trained on Data Set 3 were not as accurate overall compared to those from Data Set 2 (Table 15), the Data Set 3 (Table 16) fusion models showed a significant improvement in both classification and location accuracy over the single sensor models. Data Set 2 fusion models had showed only a little improvement in classification with a degradation in location performance.

It appears that accurate predictions require the additional data provided by more complex fusion models, as well as the introduction of infrared noise, to compensate for the increased complexity of detecting a person in more varied positions within an images. For the Data Set 2 models, the single sensor models were able to extrapolate information from the uniform positioning to accurately predict the classification and location of the person, whereas the more complex data did not enable this extrapolation.

Data Set 2 Fusion Models	*No Person*	*Person*	*Average*	*R Error*	*X Error*	*Z Error*	*Average*
CIRVQ-L3F	0.00%	1.59%	**0.80%**	3.37%	2.67%	1.73%	**2.59%**
CIRVQ-L3N	0.71%	1.59%	**1.15%**	2.73%	2.61%	1.73%	**2.36%**
CI-L4N	0.00%	0.00%	**0.00%**	3.60%	1.66%	1.44%	**2.23%**
IRVQ-L3N	0.71%	0.79%	**0.75%**	2.85%	2.76%	1.75%	**2.45%**
IRVQ-L4N	0.00%	4.76%	**2.38%**	2.62%	3.26%	1.82%	**2.57%**
CRVQ-L3F	0.71%	3.17%	**1.94%**	3.59%	3.57%	1.73%	**2.96%**
CRVQ-L4F	0.71%	4.76%	**2.74%**	2.85%	2.76%	1.69%	**2.43%**

Table 15: Data Set 2 - Best Fusion Models

Data Set 3 Fusion	*No Person*	*Person*	*Average*	*R Error*	*X Error*	*Z Error*	*Average*
CIRQ-L4F	0.59%	0.58%	**0.59%**	2.92%	3.56%	2.80%	**3.09%**
CIRQ-L4N	0.00%	1.75%	**0.88%**	2.46%	2.79%	3.32%	**2.86%**
CI-L4F	1.76%	3.51%	**2.64%**	4.26%	3.12%	2.08%	**3.15%**
CI-L4N	1.76%	2.34%	**2.05%**	4.43%	3.81%	1.64%	**3.29%**
IRQ-L4N	0.00%	2.34%	**1.17%**	2.60%	3.46%	2.41%	**2.82%**
IRVQ-L4N	0.59%	2.34%	**1.47%**	2.59%	3.15%	2.91%	**2.88%**
CRVQ-L3F	1.76%	2.92%	**2.34%**	2.47%	3.78%	2.33%	**2.86%**
CRVQ-L4F	1.18%	1.75%	**1.47%**	3.11%	3.94%	2.86%	**3.30%**

Table 16: Data Set 3 - Best Fusion Models

4.6.1 Overall Discussion

Results indicate there is potential in the application of sensor fusion to Sonasafe's need to detect people in a complex industrial environment, with some limitations. Given that safety systems require a very high degree of reliability and accuracy, this system might be unsuitable for application on

vehicles as similar to Sonasafe's existing proximity warning system. Further research would be required to determine if sensor fusion could reach the required level of accuracy and consistency. However this system would likely be a viable option as a system to collect data, logging events or infringements by people without a PDU. It is highly likely that a person who enters the frame of reference would be detected given the fused models demonstrated a 98.83% accuracy for detection at a rate of 12ms to 14ms per frame.

Though the results of this study indicated the use of specific different types of sensors fused together can contribute to the accuracy of using CNNs for human detection, there were several factors that impacted model accuracy.

The use of different data sets gave a good indication that the larger data sets greatly increased the accuracy and reliability of the models. The more complex the environment, the larger a data set would need to be. Ideally, the size of the data set would be significantly larger than any used in this study. The data set should also be specifically captured or tailored to the environment the system would be used in. The size of the data set required would depend on the noise and diversity of the environment and the number of categories, or classes, being detected at any one time.

During this study the visual and infrared camera settings automatically adjusted depending on the conditions. This led to inconsistent imaging, resulting in some blurring in low lighting conditions and variability in saturation. Manually configuring the settings might lead to more consistent imaging, potentially resulting in more consistent data which would help the models train, producing more accurate results. The camera would likely benefit from a set exposure rather than leaving this setting to automatically adjust. The infrared camera might benefit from having the temperature range set to specific values to better focus on people and exclude other objects or noise.

The radar module proved to be somewhat problematic with the lack of density in the point cloud it returned. Based on the initial research of the product the point cloud was expected to be denser, with no clear indication of if the performance seen in this study was due to the specific application, the configuration used in this project, or limitations of the development kit. Work to further increase the resolution of the radar and better configure it for the specific range and FOV requirements would likely provide more accurate radar results.

If this system was to be utilised in industry, the hardware would need to be selected to allow for better synchronization of the sensors. In this study the radar feed was delayed by approximately 0.5 seconds

from infrared and camera feeds. During the study this factor was somewhat minimised in the collection of the data by having people walk slower. However, it is likely that results would be improved with more accurate sensor synchronisation and calibration. This would become increasingly critical in faster moving environments.

Further research would be needed to determine if the inclusion of velocity data in the radar would improve classification results to determine if the inclusion of the additional data would increase the accuracy of predictions.

The structure of the models used in this study were designed to be simple and give a good indication of the advantages and disadvantages of each sensor without having to use a significantly larger data set. This simplicity limited models to a specific scale and to single person detection only. If this system was to be further developed, detection capability would need to be expanded to detecting multiple people in a frame. This might be achieved through a 2D output with class and location predicted for image segments such as used in YOLO models. Additionally, the models trained in this study had greater difficulty detecting a person further in the background. It would be worth looking into adjusting model architecture to classify objects at different scales to improve detection accuracy.

Overall, this study demonstrated sensor fusion is a viable option for human detection in complex industrial environments. The results indicate an infrared camera effectively balances limitations of a visual camera in low light conditions. It also established that using radar improves the classification accuracy in addition to providing range measurements for which the radar was initially included in this study.

Chapter 5
Conclusion

From analysis on models trained on different data sets with different combinations of camera, infrared, and radar in both single sensor and fusion models, it is clear that fusion is a viable option to improve human detection accuracy in complex industrial environments for both classification and location accuracy. Results indicated that as an environment gets more complex and includes more adverse effects the use of sensor fusion becomes more beneficial to the accuracy of CNN models for human detection.

Areas for future development include increasing the size and complexity of data sets, tailoring to the specific environment is it intended to be used within; using hardware that can be manually configured rather than relying on the hardware to automatically adjust settings; increase the resolution or improve the configuration of the radar; ensure hardware is fully synchronised to return images without delay between sensors; further testing the benefits of including velocity in the radar data; and increasing model complexity to cope with detection of multiple persons in a frame and over different scales.

Appendices

Appendix A. Python Code for Camera Models

```
from keras.models import Model
from keras.layers import Dense, Flatten, Input, Reshape, Conv2D, MaxPooling2D, UpSampling2D,
          / Concatenate, LeakyReLU, BatchNormalization, ZeroPadding2D, add, concatenate
from keras import backend as K, optimizers
from keras.utils import Sequence

from tensorflow import print as tf_print, expand_dims

import os
import numpy as np
import xml.etree.ElementTree as ET
import cv2
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from sklearn.model_selection import train_test_split

x_train = []
y_train = []

directory = 'FileDirectory/'
img_ir = directory + 'IR_Cam/Crop/IR-%s.jpg'
lables = directory + 'Labels/label-%s.xml'

Rad_q = directory + 'Radar/q_val_%s_%s.npy'
Rad_v = directory + 'Radar/v_val_%s_%s.npy'
Rad_x = directory + 'Radar/x_val_%s_%s.npy'
Rad_y = directory + 'Radar/y_val_%s_%s.npy'
Rad_z = directory + 'Radar/z_val_%s_%s.npy'

model_history = {'loss': [], 'Detect_1_loss': [], 'Detect_2_loss': [], 'Detect_3_loss': []
                   'Detect_1_accuracy': [], 'Detect_2_accuracy': [], 'Detect_3_accuracy': [],
                   'val_loss': [], 'val_Detect_1_loss': [], 'val_Detect_2_loss': [],
                   'val_Detect_3_loss': [], 'val_Detect_1_accuracy': [],
                   'val_Detect_2_accuracy': [], 'val_Detect_3_accuracy': []}

train = True
epoch = 50
lrn_rt = 1e-3
batch_sz = 64
nxt_w = 0
t_cycle = 1
x_test = {}
x_train = {}
kernel = 3

nb_classes = 1
nb_classes += 1

adam = optimizers.Adam(learning_rate=lrn_rt, beta_1=0.9, beta_2=0.999, decay=(lrn_rt / epoch))

perform = 'Performance/result%s-%sx%s.txt'
weights = 'Model Weights/Model_Weights%s_%sx%s.h5'
Backbone = 'Model Weights/BackBone%s_%sx%s.h5'
history = 'History/History%s_%sx%s.npy'

data_I_IR = directory + 'Numpy Data/Flipped/IR Crop/%sx%s/data-%s.npy'

data_L = directory + 'Numpy Data/Flipped/Labels Crop/label-%s.npy'
data_F = directory + 'Numpy Data/Flipped/train_file_crop.npy'
data_F_invis = directory + 'Numpy Data/Flipped/file_invis_crop.npy'



def make_model():
    input_tensor = Input(shape=(img_h, img_w, 1))
    base = Conv2D(32, kernel, strides=1, activation='relu', padding='same')(input_tensor)
    base = MaxPooling2D()(base)
    base = Conv2D(64, kernel, strides=1, activation='relu', padding='same')(base)
```

```
    base = MaxPooling2D()(base)
    base = Conv2D(128, kernel, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)

    if nmb >= 1:
        base = Conv2D(256, kernel, strides=1, activation='relu', padding='same')(base)
        base = MaxPooling2D()(base)

    if nmb >= 2:
        base = Conv2D(512, kernel, strides=1, activation='relu', padding='same')(base)
        base = MaxPooling2D()(base)

    if nmb >= 3:
        base = Conv2D(1024, kernel, strides=1, activation='relu', padding='same')(base)
        base = MaxPooling2D()(base)

    if nmb >= 4:
        base = Conv2D(2048, kernel, strides=1, activation='relu', padding='same')(base)
        base = MaxPooling2D()(base)

    base = Flatten()(base)

    if nmb == 4:
        top = Dense(2048, activation='relu')(base)
        top = Dense(1024, activation='relu')(top)
        top = Dense(512, activation='relu')(top)
        top = Dense(256, activation='relu')(top)
        top = Dense(128, activation='relu')(top)

    if nmb == 3:
        top = Dense(1024, activation='relu')(base)
        top = Dense(512, activation='relu')(top)
        top = Dense(256, activation='relu')(top)
        top = Dense(128, activation='relu')(top)

    elif nmb == 2:
        top = Dense(512, activation='relu')(base)
        top = Dense(256, activation='relu')(top)
        top = Dense(128, activation='relu')(top)

    elif nmb == 1:
        top = Dense(256, activation='relu')(base)
        top = Dense(128, activation='relu')(top)

    elif nmb == 0:
        top = Dense(128, activation='relu')(base)

    top = Dense(64, activation="relu")(top)
    top = Dense(32, activation="relu")(top)
    top = Dense(3 + nb_classes, activation="sigmoid", name='Detect')(top)

    return Model(inputs=input_tensor, outputs=top), Model(inputs=input_tensor, outputs=base)


def train_model():
    global x_train, x_test, model

    x_train = np.load(data_F)

    (x_train, x_test) = train_test_split(x_train, test_size=0.1, random_state=42)

    no_train = np.shape(x_train)[0]
    no_test = np.shape(x_test)[0]

    train_batch = Generator(x_train, batch_sz)
    val_batch = Generator(x_test, batch_sz)

    model, backbone = make_model()
    print('--Model Generated')

    model.compile(optimizer=adam, metrics=['accuracy'], loss={'Detect': loss},
                  loss_weights={'Detect': 1.})
    print('--Model Compiled')

    if train:
```

```
        print('--Beginning Training')
        train_history = model.fit(train_batch,
                                  steps_per_epoch=int(no_train // batch_sz),
                                  epochs=epoch,
                                  verbose=1,
                                  validation_data=val_batch,
                                  validation_steps=int(no_test // batch_sz))
        print('--Finished Training cycle')
        model.save_weights(weights % (nmb, img_w, img_h))
        backbone.save_weights(Backbone % (nmb, img_w, img_h))
        np.save(history % (nmb, img_w, img_h), train_history)
        print('--Weights Saved\n--History Saved')

    else:
        try:  # Try to load model weights
            model.load_weights(weights % (nmb, img_w, img_h))
            print('--Weights Loaded')
        except:
            print('--Unable to load Weights')
            exit('--Terminated')
    return


def loss(result_true, result_pred):
    const_obj = 1
    obj = result_true[:, 1]
    obj = expand_dims(obj, -1)

    clas_loss = K.square(result_true[..., 0:2] - result_pred[..., 0:2])
    coord_loss = K.square(result_true[..., 2:5] - result_pred[..., 2:5]) * obj * const_obj

    clas_loss = K.sum(clas_loss)
    coord_loss = K.sum(coord_loss)

    return clas_loss + coord_loss

def performance(trn, tst):
    global model
    invis = np.load(data_F_invis)
    data = np.concatenate([trn, tst, invis])

    with open(perform % (nmb, img_w, img_h), 'w+') as f:
        f.write('Image DataGroup Class ClassPred Detected RError XError ZError\n')
        g = 0
        for j in range(len(data)):
            i = data[j]
            if j == len(trn) + 1:
                g = 1
            elif j == len(tst) + len(trn) + 1:
                g = 2

            img = np.load(data_I_IR % (img_w, img_h, i))
            img = np.expand_dims(np.array(img), axis=0)

            t_clas = np.load(data_L % i)

            P = model.predict(img)
            p = P[0]
            p_clas = np.argmax(p[0:2])

            if p_clas == 1 and t_clas[1] == 1:
                rp, xp, zp = p[2:5]
                rt, xt, zt = t_clas[2:5]

                xp = round((xp - xt), 3)
                zp = round((zp - zt), 3)
                rp = round((rp - rt), 3)
                f.write('%s %s %s %s %s %s %s %s\n' % (i, g, int(t_clas[1]), p_clas,
                   round(p[1], 3), rp, xp, zp))
            else:
                f.write('%s %s %s %s %s\n' % (i, g, int(t_clas[1]), p_clas, round(p[1])))

    return
```

```
class Generator(Sequence):

    def __init__(self, image_filenames, batch_size):
        self.image_filenames = image_filenames
        self.batch_size = batch_size

    def __len__(self):
        return (np.ceil(len(self.image_filenames) / float(self.batch_size))).astype(int)

    def __getitem__(self, idx):
        batch = self.image_filenames[idx * self.batch_size: (idx + 1) * self.batch_size]
        batch_x = []
        batch_y = []

        for file_num in batch:
            batch_x.append(np.load(data_I_IR % (img_w, img_h, file_num)))
            batch_y.append(np.load(data_L % file_num))

        batch_x = np.asarray(batch_x)
        batch_y = np.asarray(batch_y)

        return batch_x, batch_y

train = True

img_w = 32
img_h = 24

nmb = 0
train_model()
draw_images(x_test)

nmb = 1
train_model()
draw_images(x_test)


img_w = 64
img_h = 48

nmb = 0
train_model()
draw_images(x_test)

nmb = 1
train_model()
draw_images(x_test)

nmb = 2
train_model()
draw_images(x_test)


img_w = 128
img_h = 96

nmb = 0
train_model()
draw_images(x_test)

nmb = 1
train_model()
draw_images(x_test)

nmb = 2
train_model()
draw_images(x_test)


img_w = 256
img_h = 192

nmb = 1
train_model()
draw_images(x_test)
```

```
nmb = 2
train_model()
draw_images(x_test)

nmb = 3
train_model()
draw_images(x_test)


img_w = 512
img_h = 384

nmb = 2
train_model()
draw_images(x_test)

nmb = 3
train_model()
draw_images(x_test)
```

Appendix B

Python Code for Camera, Infrared and Radar Fusion Models

```
import os
from keras.models import Model
from keras.layers import Dense, Flatten, Input, Conv2D, MaxPooling2D, Concatenate

from keras import backend as K, optimizers
from keras.utils import Sequence

from tensorflow import expand_dims

import numpy as np
import cv2
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from sklearn.model_selection import train_test_split


directory = 'FileDirectory/'
data_L = data_directory + 'Labels/label-%s.npy'
data_F = data_directory + 'train_file.npy'
data_F_invis = data_directory + 'file_invis.npy'

InData = '%sx%s/data-%s.npy'

data_I_Cam = data_directory + 'Cam/' + InData
data_I_IR = data_directory + 'IR/' + InData
data_I_RadRQ = data_directory + 'Rad RQ/' + InData
data_I_RadRVQ = data_directory + 'Rad RVQ/' + InData

SavedWeights = 'Model Weights/Backbone%s_%sx%s.h5'

models_directory = '/Models_Directory/'

Cam_weights = models_directory + 'Model_Cam 4/' + SavedWeights
IR_weights = models_directory + 'Model_IR 3/' + SavedWeights
RadRQ_weights = models_directory + 'Model_RadRQ 4/' + SavedWeights
RadRVQ_weights = models_directory + 'Model_RadRVQ 3/' + SavedWeights

model_history = {'loss': [], 'Detect_1_loss': [], 'Detect_2_loss': [], 'Detect_3_loss': []
                 'Detect_1_accuracy': [], 'Detect_2_accuracy': [], 'Detect_3_accuracy': [],
                 'val_loss': [], 'val_Detect_1_loss': [], 'val_Detect_2_loss': [],
                 'val_Detect_3_loss': [], 'val_Detect_1_accuracy': [],
                 'val_Detect_2_accuracy': [], 'val_Detect_3_accuracy': []}


epoch = 50
batch_sz = 64

nb_classes = 3


def Cam_Model():
    input_tensor = Input(shape=(img_h, img_w, 3))
    base = Conv2D(32, 3, strides=1, activation='relu', padding='same')(input_tensor)
    base = MaxPooling2D()(base)
    base = Conv2D(64, 3, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)
    base = Conv2D(128, 3, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)

    base = Conv2D(256, 3, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)

    base = Conv2D(512, 3, strides=1, activation='relu', padding='same')(base)
```

```
    base = MaxPooling2D()(base)

    base = Conv2D(1024, 3, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)

    base = Flatten()(base)
    return Model(inputs=input_tensor, outputs=base)


def IR_Model():
    input_tensor = Input(shape=(ir_h, ir_w, 1))
    base = Conv2D(32, 3, strides=1, activation='relu', padding='same')(input_tensor)
    base = MaxPooling2D()(base)
    base = Conv2D(64, 3, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)
    base = Conv2D(128, 3, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)
    base = Conv2D(256, 3, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)
    base = Conv2D(512, 3, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)

    base = Flatten()(base)
    return Model(inputs=input_tensor, outputs=base)


def Rad_Model():

    if RQ:
        rad_in = 3
    else:
        rad_in = 4

    input_tensor = Input(shape=(rad_h, rad_w, rad_in))
    base = Conv2D(32, 1, strides=1, activation='relu', padding='same')(input_tensor)
    base = MaxPooling2D()(base)
    base = Conv2D(64, 1, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)
    base = Conv2D(128, 1, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)
    base = Conv2D(256, 1, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)
    base = Conv2D(512, 1, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)
    base = Conv2D(1024, 1, strides=1, activation='relu', padding='same')(base)
    base = MaxPooling2D()(base)

    base = Flatten()(base)
    return Model(inputs=input_tensor, outputs=base)


def make_model():
    cam_base = Cam_Model()
    ir_base = IR_Model()
    rad_base = Rad_Model()

    cam_base.load_weights(Cam_weights % (cam_nmb, img_w, img_h))
    ir_base.load_weights(IR_weights % (ir_nmb, ir_w, ir_h))
    rad_base.load_weights(Rad_weights % (rad_nmb, rad_w, rad_h))

    cam_input = cam_base.input
    ir_input = ir_base.input
    rad_input = rad_base.input

    output = Concatenate()([cam_base.output, ir_base.output, rad_base.output])
    output = Dense(4096, activation='relu')(output)
    output = Dense(2048, activation='relu')(output)
    output = Dense(1024, activation='relu')(output)
    output = Dense(512, activation='relu')(output)
    output = Dense(256, activation='relu')(output)
    output = Dense(128, activation='relu')(output)
    output = Dense(64, activation="relu")(output)
    output = Dense(32, activation="relu")(output)
    output = Dense(3 + nb_classes, activation="sigmoid", name='Detect')(output)
    return Model(inputs=[cam_input, ir_input, rad_input], outputs=output)
```

```
def train_model():
    global x_train, x_test, model, frozen, perform
    if lrn_rt == 1e-4:
        lr = 4
    else:
        lr = 3

    if frozen:
        perform = 'Performance/result_L%sF-C%s-I%s-R%s.txt'
        history = 'History/History_L%sF-C%s-I%s-R%s.npy'
        weights = 'Model Weights/Weights_L%sF-C%s-I%s-R%s.h5'

    else:
        perform = 'Performance/result_L%sN-C%s-I%s-R%s.txt'
        history = 'History/History_L%sN-C%s-I%s-R%s.npy'
        weights = 'Model Weights/Weights_L%sN-C%s-I%s-R%s.h5'

    x_train = np.load(data_F)

    (x_train, x_test) = train_test_split(x_train, test_size=0.1, random_state=42)

    no_train = np.shape(x_train)[0]
    no_test = np.shape(x_test)[0]

    train_batch = Generator(x_train, batch_sz)
    val_batch = Generator(x_test, batch_sz)

    model = make_model()

    if frozen:
        for i, layer in enumerate(model.layers):
            if layer.name == 'flatten':
                break
            layer.trainable = False

    print('--Model Generated')
    adam = optimizers.Adam(learning_rate=lrn_rt, beta_1=0.9, beta_2=0.999, decay=(lrn_rt / epoch))
    model.compile(optimizer=adam, metrics=['accuracy'], loss={'Detect': loss},
                  loss_weights={'Detect': 1.})
    print('--Model Compiled')

    if train:
        print('--Beginning Training')
        train_history = model.fit(train_batch,
                                  steps_per_epoch=int(no_train // batch_sz),
                                  epochs=epoch,
                                  verbose=1,
                                  validation_data=val_batch,
                                  validation_steps=int(no_test // batch_sz))
        print('--Finished Training cycle')
        model.save_weights(weights % (lr, img_w, ir_w, rad_w))
        np.save(history % (lr, img_w, ir_w, rad_w), train_history.history)
        print('--Weights Saved\n--History Saved')

    else:
        try:  # Try to load model weights
            model.load_weights(weights % (lr, img_w, ir_w, rad_w))
            print('--Weights Loaded')
        except:
            print('--Unable to load Weights')
            exit('--Terminated')
    return


def loss(result_true, result_pred):
    const_obj = 1
    obj = result_true[:, 1]
    obj = expand_dims(obj, -1)

    clas_loss = K.square(result_true[..., 0:2] - result_pred[..., 0:2])
    coord_loss = K.square(result_true[..., 2:5] - result_pred[..., 2:5]) * obj * const_obj

    clas_loss = K.sum(clas_loss)
```

```
    coord_loss = K.sum(coord_loss)

    return clas_loss + coord_loss


def performance(trn, tst):
    global model
    invis = np.load(data_F_invis)
    data = np.concatenate([trn, tst, invis])

    if lrn_rt == 1e-4:
        lr = 4
    else:
        lr = 3

    if RQ:
        data_I_Rad = data_I_RadRQ
    else:
        data_I_Rad = data_I_RadRVQ

    with open(perform % (lr, img_w, ir_w, rad_w), 'w+') as f:
        f.write('Image DataGroup Class ClassPre Detected RTrue XTrue ZTrue RPre XPre ZPre\n')
        g = 0
        for j in range(len(data)):
            i = int(data[j])
            if j == len(trn) + 1:
                g = 1
            elif j == len(tst) + len(trn) + 1:
                g = 2

            cam_img = np.load(data_I_Cam % (img_w, img_h, i))
            ir_img = np.load(data_I_IR % (ir_w, ir_h, i))
            rad_img = np.load(data_I_Rad % (rad_w, rad_h, i))

            cam_img = np.expand_dims(np.array(cam_img), axis=0)
            ir_img = np.expand_dims(np.array(ir_img), axis=0)
            rad_img = np.expand_dims(np.array(rad_img), axis=0)

            ir_img = np.expand_dims(ir_img, axis=-1)

            t_clas = np.load(data_L % i)
            Mod_input = (cam_img, ir_img, rad_img)

            P = model.predict(Mod_input)
            p = P[0]
            p_clas = np.argmax(p[0:2])

            if p_clas == 1:
                rp, xp, zp = p[2:5]
                rt, xt, zt = t_clas[2:5]

                f.write('%s %s %s %s %s %s %s %s %s %s %s\n' % (i, g, int(t_clas[1]), p_clas,
                 round(p[1], 3), rt, xt, zt, rp, xp, zp))
            else:
                f.write('%s %s %s %s %s\n' % (i, g, int(t_clas[1]), p_clas, round(p[1])))

    return

class Generator(Sequence):

    def __init__(self, image_filenames, batch_size):
        self.image_filenames = image_filenames
        self.batch_size = batch_size

    def __len__(self):
        return (np.ceil(len(self.image_filenames) / float(self.batch_size))).astype(int)

    def __getitem__(self, idx):
        batch = self.image_filenames[idx * self.batch_size: (idx + 1) * self.batch_size]
        Cam_batch_x = []
        IR_batch_x = []
        Rad_batch_x = []
        batch_y = []

        if RQ:
```

```
            data_I_Rad = data_I_RadRQ
        else:
            data_I_Rad = data_I_RadRVQ

        for file_num in batch:
            Cam_batch_x.append(np.load(data_I_Cam % (img_w, img_h, file_num)))
            IR_batch_x.append(np.load(data_I_IR % (ir_w, ir_h, file_num)))
            Rad_batch_x.append(np.load(data_I_Rad % (rad_w, rad_h, file_num)))

            batch_y.append(np.load(data_L % file_num))

        Cam_batch_x = np.asarray(Cam_batch_x)
        IR_batch_x = np.asarray(IR_batch_x)
        Rad_batch_x = np.asarray(Rad_batch_x)

        batch_y = np.asarray(batch_y)

        return [Cam_batch_x, IR_batch_x, Rad_batch_x], batch_y

train = True
RQ = True

cam_nmb = 3
rad_nmb = 3
ir_nmb = 2

ir_w = 128
ir_h = 96
img_w = 256
img_h = 192
rad_w = 256
rad_h = 192

train = False
frozen = True
lrn_rt = 1e-3
train_model()
draw_images(x_test)

lrn_rt = 1e-4
train_model()
draw_images(x_test)

frozen = False
lrn_rt = 1e-3
train_model()
draw_images(x_test)
lrn_rt = 1e-4
train_model()
draw_images(x_test)

train_model()
performance(x_train, x_test)
```

Bibliography

[1] Sonasafe. "Sonasafe - Features." https://www.sonasafe.co.nz/sonasafe-features (accessed 11/03, 2022).

[2] Sonasafe, "Sonasafe info pack," in *Isaac Williams – Master's project 2022*, 1st ed, 2022.

[3] SICK. "3D machine vision." https://www.sick.com/at/en/machine-vision/3d-machine-vision/visionary-b/c/g348851 (accessed 16/03, 2022).

[4] Navtech Radar. "Terran360." https://navtechradar.com/explore/single-sensor-solution/ (accessed 16/03, 2022).

[5] GEOPREVENT. "PERYX® People radar." https://www.geopraevent.ch/technologies/radar-people-detection/?lang=en (accessed 24/05, 2022).

[6] S. Li, H. Chen, Q. Wang, J. An, and J. Li, "Summary of Object Recognition," *Journal of Physics: Conference Series,* vol. 1651, no. 1, pp. 1-4, 2020/11/01 2020, doi: 10.1088/1742-6596/1651/1/012159.

[7] R. Jain, R. Kasturi, and B. Schunck, *Machine Vision*. 1995.

[8] R. C. W. R. E. Gonzalez, *Digital Image Processing*. 2018.

[9] D. S. Gajbhar. "Detecting and Counting Apples in Real World Images using OpenCV and Python." https://shrishailsgajbhar.github.io/post/OpenCV-Apple-detection-counting (accessed 07/06, 2022).

[10] A. Hurlbert, "Colour vision: Putting it in context," *Current Biology,* vol. 6, no. 11, pp. 1381-1384, 1996/11/01/ 1996, doi: https://doi.org/10.1016/S0960-9822(96)00736-1.

[11] Y. Li, X. Zhang, H. Li, Q. Zhou, X. Cao, and Z. Xiao, "Object detection and tracking under Complex environment using deep learning-based LPM," *IET Computer Vision,* https://doi.org/10.1049/iet-cvi.2018.5129 vol. 13, no. 2, pp. 157-164, 2019/03/01 2019, doi: https://doi.org/10.1049/iet-cvi.2018.5129.

[12] Heliguy. "In-Depth Guide to Thermal Imaging." https://www.heliguy.com/blogs/posts/guide-to-thermal-imaging (accessed 24/05, 2022).

[13] S. S. Limited. https://www.seensafety.com/seen-products (accessed 16/01, 2023).

[14] Seen Safety Ltd. "IRIS 860 sensor pack." https://www.seensafety.com/seen-products/p/sensor (accessed 16/03, 2022).

[15] SICK. "Personnel detection and machine safety." https://www.sick.com/fi/en/industries/industrial-vehicles/mobile-platforms/automated-guided-vehicles/personnel-detection-and-machine-safety/c/g513976 (accessed 16/03, 2022).

[16] T. Teixeira, G. Dublon, and A. Savvides, "A Survey of Human-Sensing: Methods for Detecting Presence, Count, Location, Track, and Identity," in "ACM Computing Surveys," 01/01 2010, vol. 5.

[17] C. Pao, "Powering Human Presence Detection with Sensors." [Online]. Available: https://www.electronicdesign.com/industrial-automation/article/21140143/powering-human-presence-detection-with-sensors

[18] AUTOCRYPT. "Camera, Radar and LiDAR: A Comparison of the Three Types of Sensors and Their Limitations." https://autocrypt.io/camera-radar-lidar-comparison-three-types-of-sensors/ (accessed 29/03, 2022).

[19] A. Neal. "LiDAR vs. RADAR." https://www.fierceelectronics.com/components/lidar-vs-radar (accessed 29/03, 2022).

[20] S. Boral. "LiDAR vs. Radar: Which Is Better for Autonomous Vehicles." https://www.iottechtrends.com/lidar-vs-radar-for-autonomous-vehicles/ (accessed 30/03, 2022).

[21] T. Grey. "Lidar vs. camera: driving in the rain." https://ouster.com/blog/lidar-vs-camera-comparison-in-the-rain/ (accessed.

[22] R. Bocu, D. Bocu, and M. Iavich, "Objects Detection Using Sensors Data Fusion in Autonomous Driving Scenarios," *Electronics,* vol. 10, no. 23, 2021, doi: 10.3390/electronics10232903.

[23] J. Mendez, M. Molina, N. Rodriguez, M. P. Cuellar, and D. P. Morales, "Camera-LiDAR Multi-Level Sensor Fusion for Target Detection at the Network Edge," *Sensors (Basel),* vol. 21, no. 12, p. 3992, 2021, doi: 10.3390/s21123992.

[24] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research," *Sensors,* vol. 19, no. 3, 2019, doi: 10.3390/s19030648.

[25] Y. Wang and J. Ye, "An Overview Of 3D Object Detection," 29/10/2020 2020. Accessed: 30/03/2022. [Online]. Available: https://arxiv.org/pdf/2010.15614.pdf

[26] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*, 2nd ed. (Intelligent robotics and autonomous agents). Cambridge, Mass.: MIT Press, 2011.

[27] F. Arias, M. Zambrano, K. Broce, C. Medina, H. Pacheco, and Y. Nunez, "Hyperspectral imaging for rice cultivation: Applications, methods and challenges," *AIMS Agriculture and Food,* vol. 6, no. 1, pp. 273-307, 2021, doi: 10.3934/agrfood.2021018.

[28] TELOPS. "Gas Detection and Identification." https://www.telops.com/applications/gas-detection-and-identification/ (accessed 23/05, 2022).

[29] A. Zaarane, I. Slimani, W. Al Okaishi, I. Atouf, and A. Hamdoun, "Distance measurement system for autonomous vehicles using stereo camera," *Array,* vol. 5, p. 100016, 01/03 2020, doi: https://doi.org/10.1016/j.array.2020.100016.

[30] Y. Zhou, Y. Dong, F. Hou, and J. Wu, "Review on Millimeter-Wave Radar and Camera Fusion Technology," *Sustainability*, vol. 14, no. 9*,* doi: 10.3390/su14095114.

[31] K. Yoneda, N. Suganuma, R. Yanase, and M. Aldibaja, "Automated driving recognition technologies for adverse weather conditions," *IATSS Research,* vol. 43, no. 4, pp. 253-262, 01/12 2019, doi: https://doi.org/10.1016/j.iatssr.2019.11.005.

[32] A. Benjamin. "Imaging radar: one sensor to rule them all." https://e2e.ti.com/blogs_/b/behind_the_wheel/posts/imaging-radar-using-ti-mmwave-sensors (accessed 29/03, 2022).

[33] T. Instruments, "Imaging Radar Using Cascaded mmWave Sensor Reference Design," in "Design Guide: TIDEP-01012," TexasInstruments, 2020. Accessed: 30/03. [Online]. Available: https://www.ti.com/lit/ug/tiduen5a/tiduen5a.pdf?ts=1648554666812&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FTIDEP-01012

[34] D. Hani. "LiDAR vs RADAR: Detection, Tracking, and Imaging." https://www.wevolver.com/article/lidar-vs-radar-detection-tracking-and-imaging (accessed 29/03, 2022).

[35] K. Hedenberg, "Obstacle Detection for Driverless Trucks in Industrial Environments," Licentiate thesis, comprehensive summary, Halmstad University Dissertations, Halmstad University Press, Halmstad, 7, 2014. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:hh:diva-25349

[36] S. R. V. Mirjalili, *Python Machine Learning*, 2nd ed. Packt Publishing, 2017.

[37] TensorFlow. "Create production-grade machine learning models with TensorFlow." https://www.tensorflow.org (accessed 8/12, 2022).

[38] Y. B. A. C. I. Goodfellow, *Deep Learning*. The MIT Press, 2016.

[39] A. Rosebrock. "Convolutional Neural Networks (CNNs) and Layer Types." https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/ (accessed 7/12, 2022).

[40] L. J. W.-Y. Lee, W. Philips, "Semantic-guided radar-vision fusion for depth estimation and object detection," BMVC, the 32nd British Machine Vision Conference, 2021. [Online]. Available: http://hdl.handle.net/1854/LU-8713974

[41] Y. Long, D. Morris, X. Liu, M. Castro, P. Chakravarty, and P. Narayanan, "Radar-Camera Pixel Depth Association for Depth Completion," doi: 10.1109/cvpr46437.2021.01232.

[42] D. Griffiths. "Point cloud classification with PointNet." https://keras.io/examples/vision/pointnet/ (accessed 8/12, 2022).

[43] S. Ahmed, M. N. Huda, S. Rajbhandari, C. Saha, M. Elshaw, and S. Kanarachos, "Pedestrian and Cyclist Detection and Intent Estimation for Autonomous Vehicles: A Survey," *Applied Sciences,* vol. 9, 06/06 2019, doi: 10.3390/app9112335.

[44] M. Seeland, M. Rzanny, N. Alaqraa, J. Wäldchen, and P. Mäder, "Plant species classification using flower images—A comparative study of local feature representations," *PLOS ONE,* vol. 12, p. e0170629, 02/24 2017, doi: 10.1371/journal.pone.0170629.

[45] H. Tibebu. "Introduction to Data Fusion." Medium. https://medium.com/haileleol-tibebu/data-fusion-78e68e65b2d1 (accessed 8/12, 2022).

[46] P. Tech. "Microsoft LifeCam Studio." https://www.pbtech.co.nz/product/CAMMST2061570/Microsoft-LifeCam-Studio---Silver-Black---USB-20?qr=GShopping&gclid=EAIaIQobChMIs8DMzcm6-AIVXppmAh3ZFw-jEAQYAiABEgI52fD_BwE (accessed 20/06, 2022).

[47] B&H. "Microsoft LifeCam Studio." https://www.bhphotovideo.com/c/product/893197-REG/Microsoft_q2f_00013_Lifecam_Studio_PL2.html (accessed 20/06, 2022).

[48] F. Systems. *Technical Data - FLIR E60*. (2012). [Online]. Available: https://www.globaltestsupply.com/pdfs/cache/www.globaltestsupply.com/flir_systems/thermal_imager/e60/manual/flir_systems_e60_thermal_imager_manual.pdf

[49] F. Systems. "FLIR Lepton 2.5: Lwir Micro Thermal Camera Module." https://www.flir.eu/products/lepton/ (accessed 25/05, 2022).

[50] Digi-Key. "FLIR Lepton 2.5." https://www.digikey.co.nz/en/products/detail/flir-lepton/500-0763-01/6250105 (accessed 25/05, 2022).

[51] Digi-Key. "FLIR Lepton 3." https://www.digikey.co.nz/en/products/detail/flir-lepton/500-0726-01/6163867 (accessed 25/05, 2022).

[52] Digi-Key. "PIM365." https://www.digikey.co.nz/en/products/detail/pimoroni-ltd/PIM365/9606191 (accessed 25/05, 2022).

[53] Element14. "D6T-44L-06H Thermal IR Sensor." https://nz.element14.com/omron/d6t-44l-06h/thermal-ir-sensor-5-200deg-c-i2c/dp/3406470?ost=d6t-44l-06 (accessed 25/05, 2022).

[54] GroupGets. "FLIR Boson 320 - Consumer - 50° " https://store.groupgets.com/products/flir-boson-320?variant=6082160689179 (accessed 25/05, 2022).

[55] GroupGets. "Tau 2 336 60Hz - Performance Grade." https://store.groupgets.com/products/tau-2-336 (accessed 25/05, 2022).

[56] T. Instruments. "mmWave radar sensors." https://www.ti.com/sensors/mmwave-radar/overview.html (accessed 25/05, 2022).

[57] T. Instruments. "MMWCAS-RF-EVM: mmWave cascade imaging radar RF evaluation module." https://www.ti.com/tool/MMWCAS-RF-EVM (accessed 25/05, 2022).

[58] T. Instruments. "AWRx Cascaded Radar RF Evaluation Module (MMWCAS-RF-EVM)." https://www.ti.com/lit/ug/swru553a/swru553a.pdf?ts=1649816962363&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FMMWCAS-RF-EVM (accessed 25/05, 2022).

[59] T. Instruments. "Design Guide: TIDEP-01012 Imaging Radar Using Cascaded mmWave Sensor Reference Design." https://www.ti.com/lit/ug/tiduen5a/tiduen5a.pdf?ts=1649884712882 (accessed 25/05, 2022).

[60] S. R. Cesar Iovescu, "The fundamentals of millimeter wave radar sensors," 2020, vol. 2022. [Online]. Available: https://www.ti.com/lit/wp/spyy005a/spyy005a.pdf?ts=1651627694418&ref_url=https%253A%252F%252Fwww.google.com%252F

[61] T. Instruments. "Start development with our automotive mmWave radar sensors." https://www.ti.com/design-resources/embedded-development/automotive-mmwave-radar-sensors.html#Development (accessed 25/05, 2022).

[62] A. Nguyen, "mmWave Radar Sensors: Object Versus Range," 2021. Accessed: 25/05/2022. [Online]. Available: https://www.ti.com/lit/an/swra593a/swra593a.pdf?ts=1651629274921&ref_url=https%253A%252F%252Fe2e.ti.com%252F

[63] T. Instruments. "Introduction to mmWave radar sensing: FMCW radars." https://training.ti.com/node/1139153?context=1128486-1139153 (accessed 25/05, 2022).

[64] T. Instruments. "Understanding the Out of Box Demo Data Output." Texas Instruments Resource Explorer. https://dev.ti.com/tirex/explore/node?node=AH-2SVYArR7wfP9cuNOm0w__VLyFKFf__LATEST (accessed 15/06, 2022).

[65] T. Instruments. Automated Parking Demo [Online] Available: https://dev.ti.com/tirex/explore/node?node=A__AH4G8WbiDK4IbnEcvleAzg__com.ti.mmwave_automotive_toolbox__AocYeEd__LATEST

[66] T. Instruments, "mmWave Sensing Estimator " vol. Mac, 1.4.0d ed, 2022, p. Google Chrome.

[67] TensorFlow. "Overfit and underfit." https://www.tensorflow.org/tutorials/keras/overfit_and_underfit (accessed 14/11, 2022).

[68] J. Nelson. "LabelImg for Labeling Object Detection Data." https://blog.roboflow.com/labelimg/ (accessed 12/05, 2022).

[69] W.-Y. Lee, et al., "Semantic-Guided Radar-Vision Fusion for Depth Estimation and Object Detection.," In: BMVC, the 32nd British Machine Vision Conference, Proceedings, 2021. Accessed: 06/10/2022. [Online]. Available: http://hdl.handle.net/1854/LU-8713974

[70] D. Griffiths. "Point cloud classification with PointNet." https://keras.io/examples/vision/pointnet/ (accessed 16/08, 2022).

[71] S. P. Soumik Rakshit. "Point cloud segmentation with PointNet." https://keras.io/examples/vision/pointnet_segmentation/ (accessed 15/08, 2022).

[72] Keras. "Keras API reference." https://keras.io/api/ (accessed 08/12, 2022).

[73] H. Tibebu. "Introduction to Data Fusion " https://medium.com/haileleol-tibebu/data-fusion-78e68e65b2d1 (accessed 14/11, 2022).

[74] J. Luke, R. Joseph, and M. Balaji, "Impact of Image Size on Accuracy and Generalization of Convolutional Neural Networks," vol. 6, pp. 70-80, 02/01 2019.

[75] A. Kathuria. "What's new in YOLO v3?" https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b (accessed 16/11, 2022).

[76] U. Almog. "YOLO V3 Explained." https://towardsdatascience.com/yolo-v3-explained-ff5b850390f (accessed 16/11, 2022).

[77] W. Li, K. Liu, L. Yan, F. Cheng, Y. Lv, and L. Zhang, "FRD-CNN: Object detection based on small-scale convolutional neural networks and feature reuse," *Scientific Reports,* vol. 9, no. 1, p. 16294, 2019/11/08 2019, doi: 10.1038/s41598-019-52580-0.