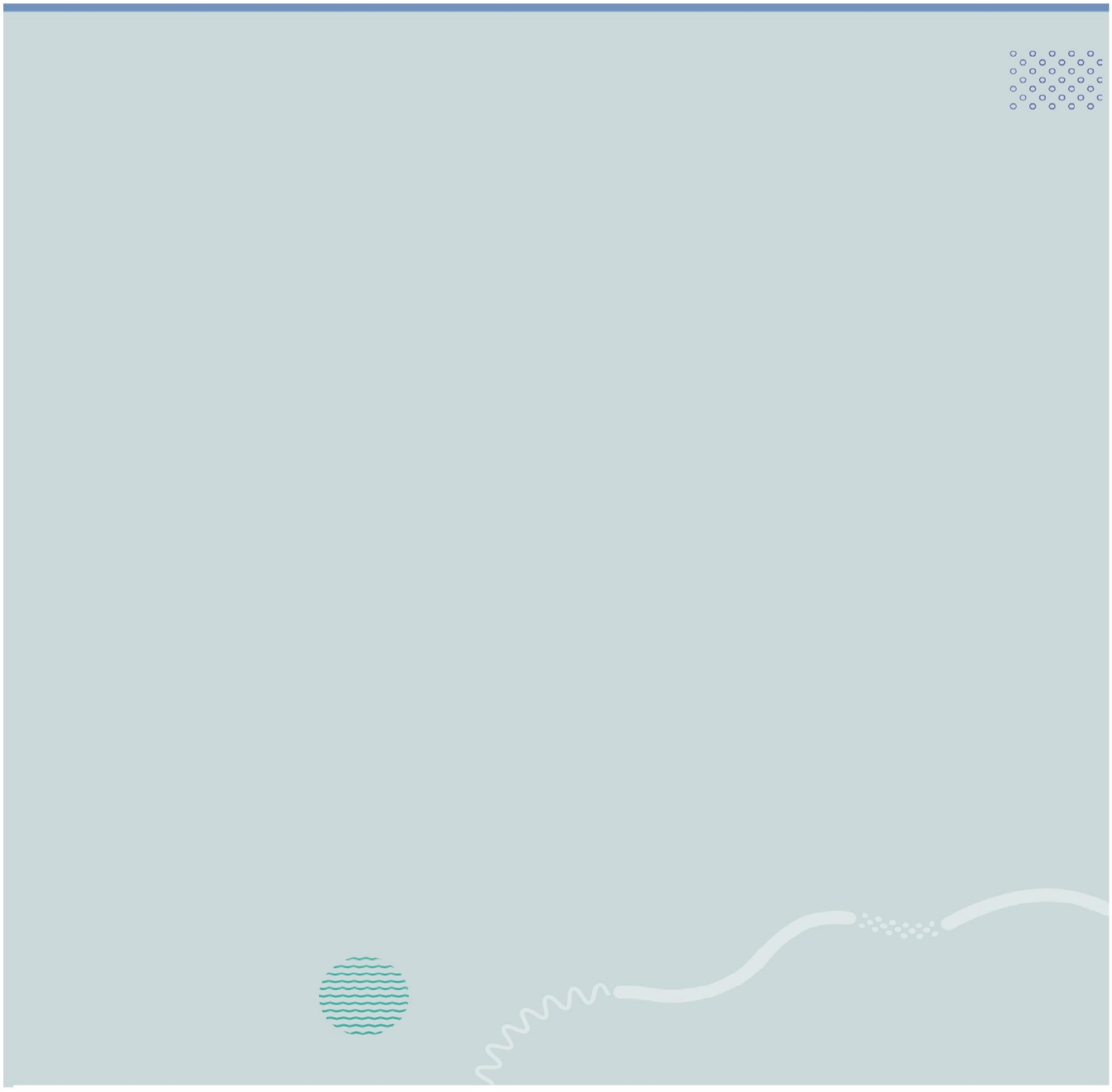
University of South-Eastern Norway
Faculty of Technology, Natural Sciences, and Maritime Sciences
-
Master Thesis in Systems Engineering with Embedded Systems
Department of Science and Industry Systems
June 10, 2021

Victor Johan HANSEN

An Infrared Small Target Detection System for UAVs



Abstract

This thesis is concerned with the task of assisting search and rescue missions by discovering missing people through the use of unmanned aerial vehicles and infrared imaging cameras. Early discovery of a victim is critical and can significantly improve their chances of survival. The thermal radiation emitted by a missing person is referred to as a heat signature and appear brighter than its surroundings in infrared images. Further, infrared imaging cameras are well suited for detection of heat signatures in dark and cloudy conditions. The task of detecting heat signatures is referred to as infrared small target detection. The motivation for this work is to demonstrate the potential value of infrared small target detection in search and rescue missions. This work presents and compares a deep learning approach, and a low-rank and sparse matrix decomposition approach for the task of infrared small target detection. Additionally, research and testing were conducted in order to develop a framework tailored to unmanned aerial vehicles. The resulting infrared small target detection system is capable of detecting heat signatures in images with complex backgrounds. The test results unequivocally demonstrate that an infrared small target detection method based on deep learning is preferable.

Acknowledgements

Above all, I need to express to Professor António L. L. Ramos at the University of South-Eastern Norway (USN) my sincere appreciation for his guidance and encouragement. I am grateful for his suggestion of this subject, which I have enjoyed working on and have gained a great deal of knowledge from.

I would also like to express my profound gratitude to Professor José A. Apolinário Jr. from the Military Institute of Engineering (IME), Brazil, for sharing his valuable insights and timely feedback that helped improving this manuscript.

The collaboration with Professor Apolinário Jr. was made possible owing to the Branortech project (https://app.cristin.no/projects/show.jsf?id=2488728), a collaboration between USN and the Norwegian University of Science and Technology (NTNU), plus IME and the Federal University of Rio de Janeiro (UFRJ) in Brazil. This project is co-funded by the Norwegian Agency for International Cooperation and Quality Enhancement in Higher Education (Diku) and the Coordination for the Improvement of Higher Education Personnel (CAPES), Brazil.

And, I have to thank my fellow students at the University of South-Eastern Norway (USN), especially D. Kazokas and B. Karna for their wise counsel and encouragement.

A special thanks to Y. Dai for giving me the permission to use the *Single-frame InfraRed Small Target* (SIRST) dataset.

Finally, I would like to express my appreciation to my partner, and my immediate family for their encouragement and support during the writing of this thesis.

Victor Johan Hansen
Kongsberg, Norway, June 10, 2021

Contents

List of Figures

List of Tables

List of Abbreviations

APG	Accelerated proximal gradient
ASIC	Application-specific integrated circuit
CNN	Convolutional neural network
CPU	Central processing unit
CUDA	Compute Unified Device Architecture
FPGA	Field-programmable gate array
FPN	Feature pyramid network
FPR	False positive rate
FPS	Frames per second
GPS	Global positioning system
GPU	Graphics processing unit
HOG	Histogram of oriented gradients
IALM	Inexact augmented Lagrange multiplier
IoU	Intersection over union
IPI	Infrared patch-image
IR	Infrared
MCC	Matthews correlation coefficient
PCP	Principal component pursuit
PCA	Principal component analysis
ResNet	Residual neural network
RISC	Reduced instruction set computer
RPCA	Robust principal component analysis
SIRST	Single-frame infrared small target
SVD	Singular value decomposition
SVM	Support vector machine
UAV	Unmanned aerial vehicle

List of Symbols

x	Scalar
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
$\mathbf{X}^*$	Complex conjugate transpose
$\|\mathbf{X}\|_\infty$	Uniform norm, defined as $\max \|\mathbf{X}_{ij}\|$
$\|\mathbf{X}\|_*$	Nuclear norm, the sum of singular values of $\mathbf{X}$
$\|\mathbf{X}\|_1$	ℓ_1 norm, defined as $\Sigma_{ij} \|\mathbf{X}_{ij}\|$
$\|\mathbf{X}\|_2$	ℓ_2 or spectral norm, the largest singular value of $\mathbf{X}$
$\|\mathbf{X}\|_F$	Frobenius norm, defined as $\sqrt{\Sigma_{ij} \|\mathbf{X}_{ij}\|^2}$
∇	Gradient operator
$\cap$	Intersection
$\cup$	Union
$\subset$	Proper subset
$\vee$	Logical disjunction, also known as OR
$\lfloor x \rfloor$	Floor function, defined as $\lfloor x \rfloor = \max \{m \in \mathbb{Z} \mid m \leq x\}$

Chapter 1

Introduction

In recent years, the use of unmanned aerial vehicles (UAVs) in search and rescue missions has gained interest [2, 39, 46, 48]. As of 2020, over 500 missing people have been located by UAVs, according to data collected by DJI [15], a leading company in commercial UAVs. The 500th rescue took place in the United States of America, when a police officer used a UAV fitted with an infrared (IR) imaging camera (also known as a thermal imaging camera) to detect a missing person in a dark field. After looking for the person on the ground without any luck, the officer launched a UAV, which discovered the person in only four minutes.

Norwegian police located a missing person using a similar technique [18]. Animals in the area made the search difficult due to their human resemblance in terms of thermal radiation, necessitating three battery swaps for the UAV. The police discovered the person in an area a search party had already approached without noticing the person. Generally, a search and rescue mission is limited by time, area coverage, and cost. Other limitations include the availability of human resources, e.g., UAV pilots, and the pilot's mental state and visual perception, which could be exhausted and inattentive from staring at a monitor for hours.

The two dots in the right part of Fig. 1.1 are heat signatures, also known as IR small targets.

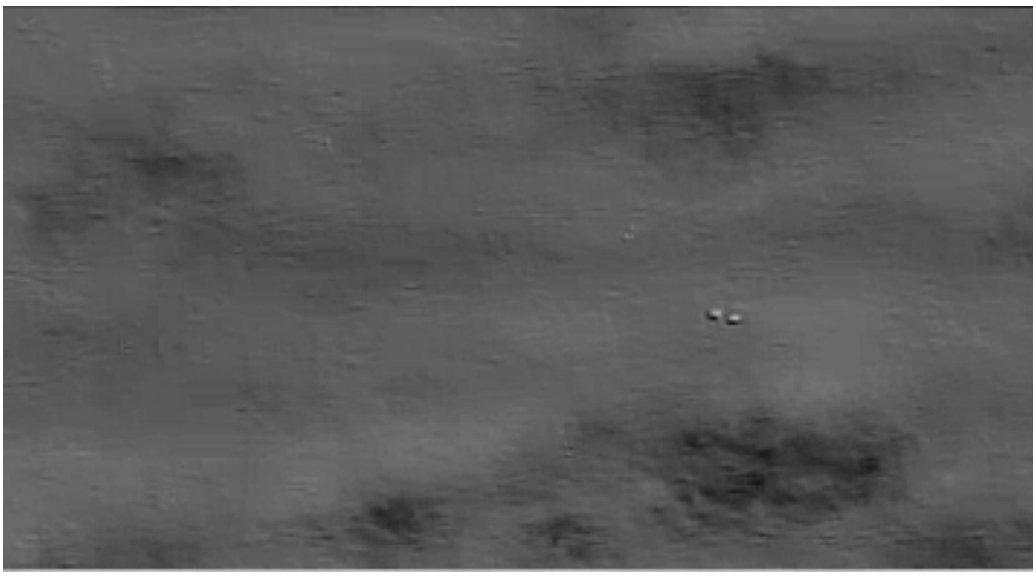


FIGURE 1.1: Feature map created by convolving an image with weights extracted from the modified ResNet used in this thesis. The two dots in the right part of the image are heat signatures, also known as IR small targets.

An IR small target can be defined [12] as having a total size of less than 0.15% of an image. For example, an IR small target can occupy a 9×9 window in an image with a height and width of 256×256 px.

UAVs are usually operated by a pilot. However, autonomous UAVs [1, 24, 34, 55] equipped with thermal imaging cameras are becoming common for applications such as search and rescue missions, by locating people or human-related objects. Further, the use of automated object detection in search and rescue missions can reduce human errors, such as overlooking certain details. Errors are likely to occur if a person has to monitor a video stream for hours. This could lead to the missing person not being found. Compared to automated object detection, human operators have the advantage of understanding the context of a video, and to recognise where to search based on previous experiences. However, it is difficult to detect crucial details in a high-definition video, and the human eye could potentially be focusing only on a small section of the video frame. The task of detecting missing victims is particularly challenging since parts of the victim might be exceedingly small, and sometimes blend in with the surroundings.

1.1 Problem

Time is critical in natural disasters such as avalanches, flooding, and earthquakes; hence, using autonomous UAVs to discover any human activity can save lives. Rescuing a victim starts with locating the victim as quickly as possible. To increase the likelihood of a victim's survival, object detection techniques should be researched and deployed to automatically detect humans and items related to human activity.

UAVs equipped with IR imaging cameras could be effective at detecting humans in low light situations since they do not need to use any external light sources. IR imaging [13, 59, 63] is used in civilian and military applications due to the ability to operate in dark, cloudy and smoke-covered areas. Therefore, making it suitable and widely used in scenarios involving surveillance of distant targets. However, if a search and rescue team is in a situation where they need to find a human in a tropical climate, the IR imaging camera might not be as effective due to the possibility of the target's heat signature (i.e., the thermal radiation emitted by the target) blending in with the surroundings. Challenges will also occur if the target is covered by an object which blocks thermal radiation, or when there are multiple, disturbing heat sources present.

1.2 Proposal

This thesis will focus on researching the use of object detection techniques, and IR imaging to detect IR small targets in remote areas. The goal is to develop a UAV-mounted system which detects IR small targets captured by an IR imaging camera. The system will assist in narrowing down a search area. The combination of IR and colour imaging can provide a fast search time in remote areas (e.g., in the ocean surface) since an IR small target will be distinguishable from its surroundings, i.e., the target will be brighter than the local background. The system is intended to complement search and rescue missions and will not replace ground-level search.

The contributions of this thesis are as follows:

- Investigating various methods for IR small target detection.
- Design of an IR small target detection framework, which shall reduce the need for UAV pilots, and assist search and rescue personnel.
- Evaluation and comparison of different IR small target detection methods on IR images.

1.3 Scope

The system proposed in this thesis is intended to be mounted on a UAV with limited battery capacity and limited payload (e.g., $1\,\mathrm{kg}$), and will likely be based on single-board computers. Typical single-board computers have limited computational power due to their RISC-based CPUs (e.g., ARM) with low clock speeds, and their reliance on SD cards for data storage. A single-board computer's GPU is usually incapable of meeting the requirements of machine learning algorithms. Further, single-board computers equipped with adequate GPUs will increase the power consumption, thus reducing the UAV's total flight time.

Some areas of this work had to be given less focus than others, and a complete and working physical prototype was not possible to create due to limitations during this thesis, such as budget, time, and lack of access to proper hardware. As a result, hardware and physical components are not prioritised, and this thesis will not focus on UAV path planning, or the communication between the UAV and the ground control station. However, a discussion on computation offloading can be found in Section 5.2.2. Additionally, it is assumed that the end-user of this system has the correct permissions to obtain information from the area of operation using IR imaging cameras, or other sensors.

1.4 Outline

The remainder of this thesis is organised in the following manner. Chapter 2 describes the basic concepts of UAVs, search and rescue, IR imaging, object detection and deep learning to equip the reader with the technical knowledge required to fully understand this work. Chapter 3 provides a literature review on related works on IR small target detection, and briefly introduces the problems and solutions associated with IR small target detection, as well as state-of-the-art deep learning-based object detection methods, and low-rank and sparse matrix decomposition methods. Chapter 4 provides an analysis of the IR small target image dataset, and discusses two proposed system architectures. Further, this chapter describes the evaluation metrics in greater detail and outlines the testing process. Chapter 5 provides the test results of the proposed methods and discusses the test results and the performance of the proposed system architecture. Finally, Chapter 6 presents the conclusion and summarises the results of the proposed system. It discusses various future directions for IR small target detection research, as well as additional opportunities for further optimising the proposed system.

Chapter 2

Background

This chapter contains the key concepts which are needed for understanding the topics discussed at a later stage in this thesis.

2.1 Unmanned Aerial Vehicles

The last few years have witnessed an increased interest in unmanned aerial vehicles (UAVs), particularly commercial UAVs. When referring to a UAV, most envision a quadcopter, which is a helicopter with four rotors. This thesis considers a system that will be designed for multi-rotor UAVs (e.g., a quadcopter) as well as other vertical take-off and landing UAVs. In comparison to helicopters, UAVs require less traffic management, are less expensive to operate, are easy to deploy, offer a high degree of design flexibility, and can be equipped with a wide range of sensors. Moreover, UAVs can access confined spaces that helicopters cannot, as well as areas deemed hazardous to humans. Additionally, UAVs can be designed to operate in adverse weather conditions such as high winds, rain, or cold temperatures. Further, UAVs provide a bird's-eye view of the search area. This grants a significant advantage to the search and rescue team and reduces the time required to locate a missing person. However, due to their restricted battery capacity, UAVs have a limited range.

This thesis assumes that the UAV uses an autonomous path planning algorithm to navigate from a ground control station to a search area specified by GPS coordinates. Search and rescue missions conducted in areas far away from the ground control station will require a long battery life, and possibly on-board processing due to the data communication being out of range. This is discussed in greater detail in Section 5.2.2.

2.2 Search and Rescue

Search and rescue operations encompass both the search for missing people and assistance to those in need. Search and rescue missions can be performed by humans with the assistance of certain animals (e.g., dogs), in a variety of climates. There are several methods for locating victims, such as using ground penetrating radar [39] to search for victims concealed beneath a layer of snow, or by using dogs to detect human presence in forests. This thesis will outline a system to be used with UAVs to aid in the search portion of search and rescue missions at sea, air, and land. The rescue procedure, such as retrieving a person from the sea, will not be discussed.

2.3 Infrared Imaging

The system proposed in this thesis will analyse image frames captured by an IR imaging camera attached to a UAV, where the UAV is intended to fly at low heights above ground level, as the height above ground level is inversely proportional to the number of pixels covering the target. In aerial IR images [13, 59], the IR targets occupy just a few pixels on the imaging plane due to the long imaging distance. As a result, the captured heat signature is likely to be weak. This makes the target difficult to recognise, as it lacks obvious shape, size, and texture characteristics. In general, IR imaging cameras have a lower image resolution than colour imaging cameras, which results in insufficient image information to classify a detected object. An appropriate solution to this problem is to use IR and colour imaging cameras in combination. A UAV fitted with an IR imaging camera can detect heat signatures from greater heights, then fly towards the detected heat source, where a colour imaging camera equipped with an external light source can classify whether the object is a human or not.

The electromagnetic spectrum is illustrated in Fig 2.1.

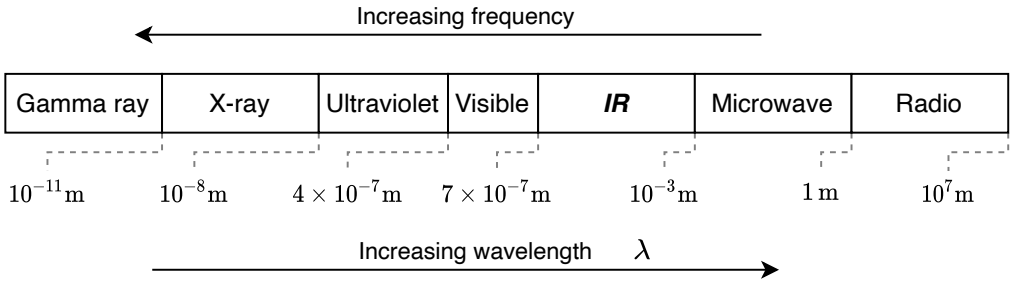


FIGURE 2.1: This diagram of the electromagnetic spectrum depicts types of electromagnetic radiation and their corresponding wavelengths.

An important law in the realm of IR imaging and specifically thermal imaging is **Wien's displacement law** [57]. This law was discovered by the German physicist Wilhelm Wien, and essentially states that an object of high temperature emits strong thermal radiation at short wavelengths λ:

$$\lambda_{max} = \frac{b}{T}, \tag{2.1}$$

where b is Wien's displacement constant $b \approx 2.898 \times 10^{-3}\,\mathrm{m} \cdot \mathrm{K}$, and T is the temperature. Objects with a temperature exceeding $0\,\mathrm{K}$ emit electromagnetic radiation. The amount of emitted radiation is determined by the object's temperature. A *blackbody* [57] is a perfect emitter which absorbs all types of electromagnetic radiation that hits it and is able to emit more radiation than any other object or surface for a given T and λ. A human body emits IR radiation, but cannot be considered as a blackbody. Clothing reduces the skin temperature of a human body, thus causing the body to emit less radiation. IR imaging cameras in the long-wave IR spectral band are suited for detecting radiation emitted by humans. Commercial IR imaging cameras operate in the spectral band or wavelengths [57] shown in Table 2.1. The entire IR spectral band ranges from $700\,\mathrm{nm}$ to $1\,\mathrm{mm}$, but only a small range of the IR spectrum is used for IR imaging.

TABLE 2.1: Types of electromagnetic radiation and their respective blackbody temperature and wavelength range.

Spectral band	**Wavelength** (λ)	**Blackbody temperature** (T)
Ultraviolet (UV)	$0.01\,\mathrm{nm} - 0.4\,\mu\mathrm{m}$	$7245\,\mathrm{K} - 289\,800\,\mathrm{K}$
Visible light (VIS)	$0.4\,\mu\mathrm{m} - 0.7\,\mu\mathrm{m}$	$4140\,\mathrm{K} - 7245\,\mathrm{K}$
Short-wave IR	$0.9\,\mu\mathrm{m} - 1.7\,\mu\mathrm{m}$	$1705\,\mathrm{K} - 3220\,\mathrm{K}$
Mid-wave IR	$3\,\mu\mathrm{m} - 5\,\mu\mathrm{m}$	$580\,\mathrm{K} - 966\,\mathrm{K}$
Long-wave IR	$8\,\mu\mathrm{m} - 14\,\mu\mathrm{m}$	$207\,\mathrm{K} - 362\,\mathrm{K}$

It is worth noting that any IR transmission [57] in the mid-wave IR and long-wave IR spectral ranges can be totally quelled by a $1\,\mathrm{mm}$ thick layer of water. Mid-wave IR and long-wave IR cameras [57] are popularly known as thermal imaging cameras because they are capable of detecting radiation emitted by objects with a low surface temperature, e.g., room temperature (i.e. $25\,^{\circ}\mathrm{C}$). These cameras detect IR radiation and produce a thermal image, which can be used to determine the surface temperatures. Thus, there is no need for an external light source to detect an object. Short-wave IR imaging [57], on the other hand, is comparable to imaging in the visible (VIS) spectral range. Short-wave IR detects scattered radiation coming from external light sources, such as reflected sunlight. Standard short-wave IR cameras are unable to determine object temperature at room-temperature, as short-wave IR radiation is negligible at temperatures below $100\,^{\circ}\mathrm{C}$. Thermal radiation can be detected using short-wave IR imaging cameras at temperatures greater than $300\,^{\circ}\mathrm{C}$. Depending on the technology [42], IR cameras can see in dark and low visibility conditions, such as fog, rain, and snow. High-resolution IR imaging cameras are prohibitively expensive, and not widely available to the general public. The commercially available thermal imaging cameras typically generate low-resolution images, making them less suitable for IR small target detection.

2.4 Generic Object Detection

Object detection can be defined as the process of locating and classifying objects of interest in images. Machine learning, and specifically deep learning, have made tremendous strides in recent years in the fields of computer vision and object detection. The current state-of-the-art object detectors are based on neural networks, especially, convolutional neural networks (CNNs) [27, 42, 67], as the traditional machine learning methods with hand-crafted features cannot achieve comparable results.

In order to perform object detection, it is necessary to first recognise and represent various objects by extracting visual semantic features from digital images. A digital image can be represented as an $m \times n$ matrix of pixels (x, y, u), where (x, y) denotes the location of each pixel, and u denotes the value at that location [25]. For representing locations within the image matrix, a left-hand coordinate system is used, as illustrated in Fig. 2.2. To maintain consistency and reduce complexity, this work will primarily consider 8-bit grayscale images when referring to an image. An 8-bit grayscale image holds pixel values ranging from 0 to 255, with 0 representing the colour black, and 255 representing a completely white pixel.

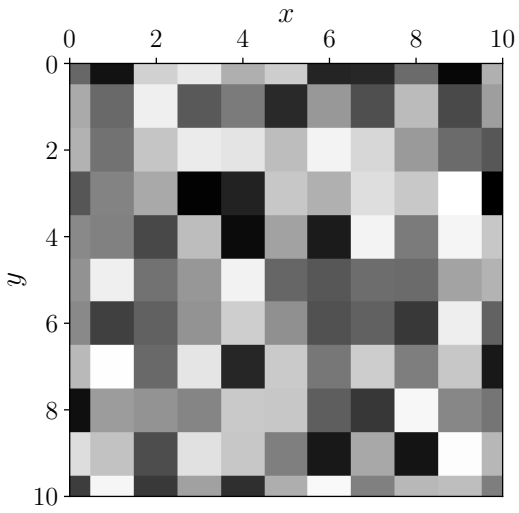


FIGURE 2.2: Left-hand coordinate system used for representing a grayscale image matrix.

Prior to the acclaimed CNNs, object detection relied on low-level feature-based methods. Some of the traditional methods used for object detection in images or videos mentioned in [2, 6, 63, 66, 67] include the scale-invariant feature transform (SIFT) [33] descriptor, the Viola-Jones cascade object detector [56], histograms of oriented gradients (HOG) [14], and local binary patterns [45]. These methods are based on domain-specific feature extractors and perform well on specialised detection tasks. For example, HOG can be used to extract line features [6]. Over time, the performance of these traditional object detectors plateaued, especially in recent years, as large datasets containing images with high intra-class variability makes the design of hand-crafted feature extractors a tedious task.

A classifier is required to distinguish an object from other categories or simply from the background, and to organise the features of an object. Acceptable performance can be achieved by combining a feature extractor (e.g., HOG) with a classifier [64]. Machine learning-based classification [35] involves training a system to produce an appropriate response y to a given input $\mathbf{x}$, where $y \in \{1, \ldots, \mathcal{C}\}$, with $\mathcal{C}$ denoting the number of categories or classes. $\mathcal{C} = 2$ is referred to as a binary classification, i.e., $y \in \{0, 1\}$.

Supervised learning aims to build an algorithm or model which learns from a dataset $\{(\mathbf{x}_i, y_i)\}$ containing training data $\mathbf{x}_i$ and labels y_i, by extracting useful patterns and making predictions. The goal is to train a model that is robust to new input data. In a supervised learning style, the training dataset contains the desired solutions, which are often referred to as labels. These labels are used to correct the system's predictions, specifically to quantify the loss $L(y, \hat{y})$ (also known as error or cost) between the desired solution y and the prediction $\hat{y}$. Examples of supervised learning algorithms [21] used for classification are logistic regression, support vector machine (SVM), k-nearest neighbour, decision tree, naive Bayes, and neural networks. Many traditional object detection algorithms [61] utilise a sliding window algorithm to identify and locate objects within image frames. Each image frame is scanned using a sliding window of size $w \times h$. Then classification is performed in the region covered by the sliding window. This is a computationally inefficient method since it requires

an exhaustive search. Additionally, the sliding window's size will need to be adjusted to accommodate detection of small, medium, and large objects.

2.5 Artificial Neural Networks

Deep learning implements machine learning through artificial neural networks. An artificial neural network is suitable for processing substantial amounts of labelled data since the network becomes more accurate the more data it is being trained on. The artificial neural network consists of an input layer, several hidden layers, and an output layer. A common artificial neural network is the feed-forward network or multi-layer perceptron [21] shown in Fig. 2.3, which learns key features or patterns by adjusting its internal weights $\mathbf{w}$.

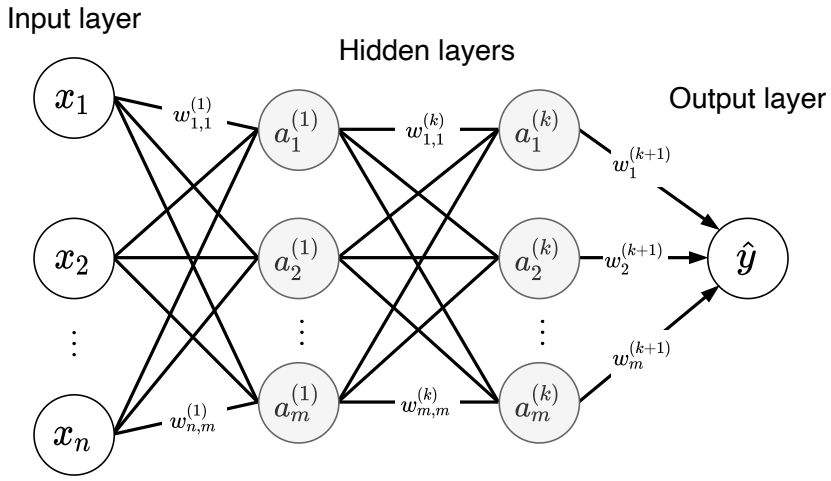


FIGURE 2.3: A multilayer perceptron.

A weight $w_i^{(k)}$ is an adjustable value given to the connection between a neuron in layer $\mathbf{a}^{(k-1)}$ and a neuron in the next layer $\mathbf{a}^{(k)}$. The weights specify how *strong* the connections between the neurons are. The output of $\mathbf{a}^{(k)}$ depends on the values of $\mathbf{w}^{(k)}$, which can be positive or negative. To produce a prediction or output $\hat{y}$, the input $\mathbf{x}$ to the network multiplied with the weight $\mathbf{w}^{(1)}$ flows through a feed-forward network. The input values propagate to the hidden layers of the network, then the hidden layers' output values are forwarded to the output layer which produces a prediction $\hat{y}$.

A perceptron, shown in Fig. 2.4, is an artificial neuron (e.g., a neuron in one of the hidden layers) which can learn over time by reinforcing the weights which lead to the desired output.

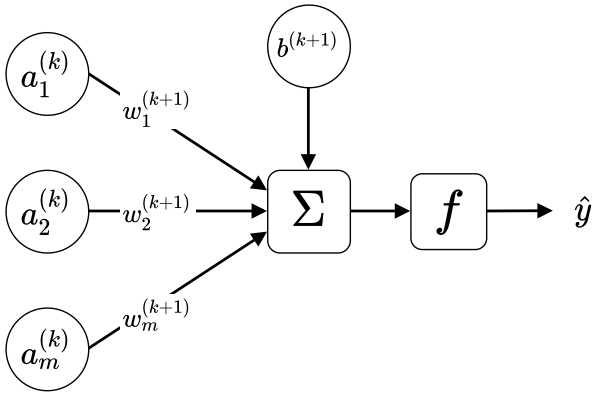


FIGURE 2.4: An artificial neuron.

A perceptron's output $\hat{y}$ is given by the following expression:

$$\hat{y} = f\left(\mathbf{a}^{(k)}\left(\mathbf{w}^{(k+1)}\right)^{\mathsf{T}} + b^{(k+1)}\right), \tag{2.2}$$

where the bias $b^{(k+1)}$ affects learning by giving certain neurons more or less attention. The weights and biases will hereinafter collectively be referred to as θ. The function f in Equation (2.2) can be any activation function [50] from Table 2.2. The operation in Equation (2.2) is implemented by the artificial neural network's dense layer [51].

TABLE 2.2: Activation functions.

Name	Function
Rectified Linear Unit (ReLU)	$\max\{0, x\}$
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
tanh	$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Swish	$x\sigma(\beta x) = \frac{x}{1+e^{-\beta x}}$

Dropout [4, 21], can be used to regularise a neural network, i.e., to reduce the impact of a neuron by randomly removing it from the network's layers. This prevents overfitting, and results in a less computationally expensive network.

2.5.1 Softmax

The final hidden layer outputs a collection of scores $\mathbf{z}$. To interpret these scores, the softmax function shown in Equation (2.3) can be used:

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^{n}\exp(z_j)}. \tag{2.3}$$

Softmax [21] converts the final prediction scores into a probability distribution over all the possible output classes. Additionally, softmax [61] can be used as a classifier instead of SVM.

2.5.2 Loss Functions

The loss function $L(y, \hat{y})$ (alternatively called the cost or error function) estimates the difference between the predicted output $\hat{y}$ and the desired output y. A low loss indicates an accurate artificial neural network. The following are various types of loss [61] and associated loss functions:

- The **classification loss** represents the difference between the predicted class and the ground truth class. Common classification loss functions include hinge loss, cross-entropy loss, and Huber loss.
- **Localisation loss** represents the difference between an object's predicted location and its ground truth location. Well-known localisation loss functions include mean squared error and mean absolute error.

2.5.3 Gradient Descent

Gradient descent [21] is a technique for minimising a function $f(\mathbf{x})$. A new point $\mathbf{x}_{n+1}$ where f has a lower value is obtained by taking small steps γ in the opposite direction of the gradient:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma \nabla f(\mathbf{x}_n), \tag{2.4}$$

where γ is known as the learning rate. According to Goodfellow et al. [21], a low learning rate can lead to a stagnant training loss. A large learning rate will cause oscillations and an increase in training loss. It is standard practice to begin with a low learning rate and gradually increase it. Gradient descent is used by artificial neural networks to minimise and possibly find the minimum of the loss function, i.e., $L = 0$, by discovering the optimal parameters θ. The first loss is calculated after initialising the weights with random values. Then $\mathbf{x}_{n+1}$ should be a point where the loss function decrease as much as possible. The final θ preferably yields the function's global minimum. However, it is possible to move in a direction that leads to a local minimum.

The performance of a machine learning algorithm is measured by its ability to minimise the training loss and to minimise the difference between the training and test loss. These two factors correspond to machine learning's two primary difficulties [21]: underfitting and overfitting. When a model fails to achieve a small loss on the training data, it is said to be underfitting. If the difference between the training and test loss becomes excessive, the model is overfitting. Deterministic gradient methods use all the available samples in the dataset simultaneously when training the network, which results in a high computational cost. However, most deep learning algorithms use **stochastic gradient descent** [21]. Stochastic gradient descent uses a fixed number of samples from the dataset called a *minibatch* or batch size. This helps reduce the computational cost.

2.5.4 Backpropagation

Backpropagation [21] is a technique for determining the gradient, and it is used in conjunction with another method, e.g., stochastic gradient descent, to facilitate learning with this gradient. Backpropagation is not limited to artificial neural networks; it is able to estimate the derivatives of most functions. In brief, backpropagation is an

algorithm which trains an artificial neural network by propagating the loss L back towards the input layers, and updates the internal parameters θ, thereby strengthening the connections which yields a low loss. This is repeated on the condition that the loss decreases.

2.6 Convolutional Neural Networks

For automatic object detection, deep learning methods, particularly convolutional neural networks (CNN), are preferred. A CNN is an artificial neural network that is primarily used for image classification and object detection. The CNNs are superior to feature extractors such as SIFT and HOG [2] but consume more processing power. The CNN takes an input image and propagates it through multiple hidden layers, returning a probability vector for each class in the process. A CNN's initial layers are capable of extracting simple patterns such as lines and arches. As the network grows deeper, the succeeding layers combine features of previous layers to create more complex patterns. Then, these features are combined into an object resembling the input image's object.

CNNs are essentially classifiers that perform unsupervised feature extraction, in which the network learns by adjusting internal weights based on the assumed importance of certain features. The complexity of training a CNN increases proportionally with the number of hidden layers, as there are more weights to adjust and more feature extraction to perform. For example, if a single input neuron to the CNN is a pixel from a grayscale image (i.e., an image with a single colour channel) with a resolution of 512×512 px. Then the CNN will receive $512 \times 512 \times 1$ input neurons. As a result, training CNNs on high-resolution images will be excessively costly in terms of computational resources required.

2.6.1 Layers

Further, a CNN requires several convolutional layers, several pooling-layers, and a fully connected layer.

Convolutional layers [21] are used to extract features from input images. This is accomplished by sliding a kernel matrix (also known as a filter) over an image with a specified stride, i.e., moving the kernel matrix across an image from left to right for each step. Additional features are obtained by performing the spatial convolutional operation shown in Equation (2.5) on the input image using several different kernel matrices. This is similar to traditional feature extraction, which involves manipulating an input image in order to extract corners or lines. The output from the 2D convolutional layer is a feature map.

$$x(t) * w(t) := \int_{-\infty}^{\infty} x(\tau) w(t - \tau) d\tau. \tag{2.5}$$

Here, x is the input data and w the kernel. Some machine learning software libraries [21], e.g., TensorFlow, implement cross-correlation as shown in Equation (2.6) and refer to

it as convolution.

$$(\mathbf{D} * \mathbf{K})(i,j) = \sum_m \sum_n \mathbf{D}(i+m, j+n)\mathbf{K}(m,n), \tag{2.6}$$

where $\mathbf{D}$ is an input image, and $\mathbf{K}$ is a kernel.

The **pooling** layer is utilised for lowering the computational cost of a network by downsampling (i.e., reducing the size of) feature maps while retaining their main information. The various types of pooling include maximum, average, and sum-pooling. Max-pooling distributes the largest value of a 2×2 filter moved from left to right across a feature map in the following manner:

$$\begin{bmatrix} \mathbf{4} & \mathbf{1} & 2 \\ \mathbf{9} & \mathbf{4} & 5 \\ 6 & 7 & 8 \end{bmatrix} \xrightarrow[\text{Max Pooling}]{2\times2} \begin{bmatrix} \mathbf{9} & 5 \\ 9 & 8 \end{bmatrix}. \tag{2.7}$$

Whereas average pooling calculates the average value for each 2×2 filter moved over a feature map as follows:

$$\begin{bmatrix} \mathbf{4} & \mathbf{1} & 2 \\ \mathbf{9} & \mathbf{4} & 5 \\ 6 & 7 & 8 \end{bmatrix} \xrightarrow[\text{Avg. Pooling}]{2\times2} \begin{bmatrix} \mathbf{4.5} & 3 \\ 6.5 & 6 \end{bmatrix}. \tag{2.8}$$

The **fully-connected** layer is usually the last layer of the network. This layer enables classification by linking the feature maps of the previous layers in order to create a vector of predictions for each class. This vector can be used by the softmax function to generate the final prediction results.

This chapter has briefly introduced the fields of IR imaging, object detection, deep learning and CNNs. The research area of CNN-based object detection is advancing rapidly, with new papers being published daily. The next chapter will discuss object detection methods with an emphasis on the detection of small objects, in addition, the task of extracting an IR small target from a noisy background image will be addressed.

Chapter 3

Object Detection

3.1 Meta-architectures and Feature Extractors

Object detection methods based on Convolutional Neural Network (CNN) are typically composed of a feature extractor and a meta-architecture. The **meta-architectures** can be divided into one-stage and two-stage models. One-stage models [36, 62, 66] such as YOLO [40] and Single Shot MultiBox Detector (SSD) [32] require only one quick pass through the network to estimate the class and location of all objects in an image. Consequently, they are faster and simpler than two-stage models.

A two-stage model [62], such as Faster R-CNN [41], must generate region proposals before the detector can make any predictions. For now, two-stage architectures are considerably slower than the one-stage architectures but achieve better prediction accuracy.

The accuracy of an object detector depends heavily on the **feature extractor** [67]. Some meta-architectures were initially introduced with a certain feature extractor. However, the meta-architecture can be detached from the initial feature extractor, allowing the meta-architecture, with some adjustments, to work with any feature extractor. The feature extractor has a profound effect [22] on the object detector's speed (i.e., the time required to detect objects in an image). According to [61], 90% of computations and memory usage in a CNN-based architecture arises from the feature extractor.

3.2 Feature Pyramid Networks

Lin et al. proposed the feature pyramid network (FPN) [28] in 2017. FPN takes an image as input and returns proportionally scaled feature maps at multiple levels. A bottom-up pathway, a top-down pathway, and lateral connections as shown in Fig. 3.1 are used to create a *pyramid*. The combination of feature maps in an FPN creates new feature maps with high- and low-level features at different resolutions. Where the low-resolution feature maps contain the high-level features. Further, the FPN operates separately from the network's feature extractor.

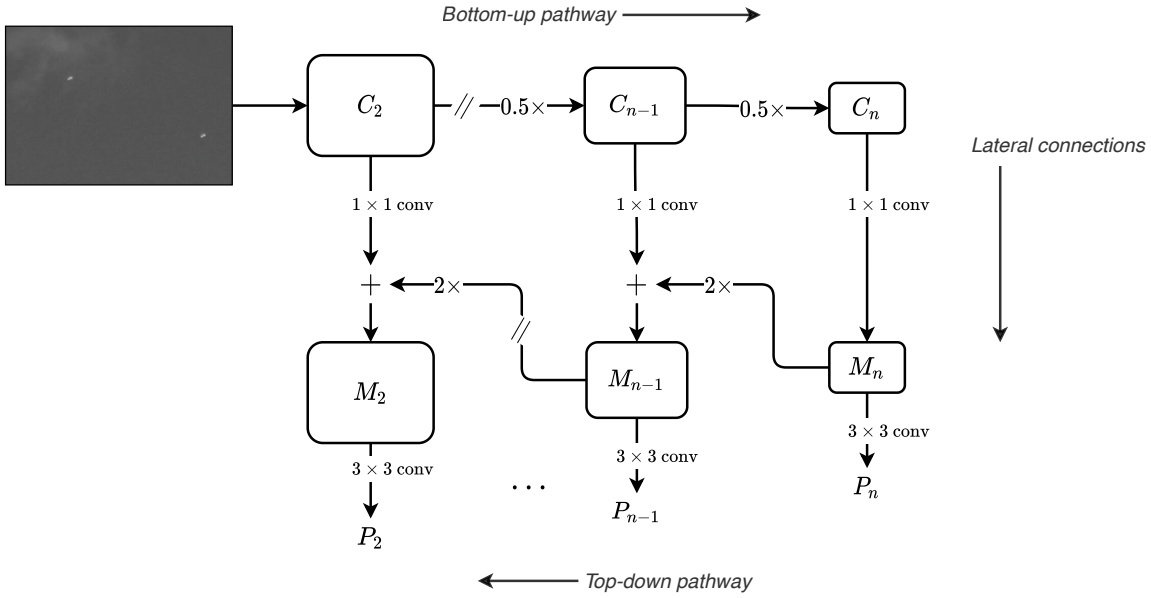


FIGURE 3.1: A feature pyramid network. $\{C_2, \cdots, C_n\}$ are feature maps created by a feature extractor, $\{M_2, \cdots, M_n\}$ are merged feature maps, and $\{P_2, \cdots, P_n\}$ are the final feature maps.

Several deep learning-based object detectors [67] perform detection at the top layer of the network. While the deeper layers of a CNN are useful for classification, they are ineffective at object localisation. FPNs [28] employ a top-down design with lateral connections for creating high-level feature maps at all sizes. The FPN makes improvements in object detection at assorted sizes because the forward propagation in a feature extractor inherently builds a feature pyramid. The bottom-up pathway of the FPN is created by the model's feature extractor (e.g., ResNet), which extracts features at several scales, followed by a down-scaling step. One pyramid level for each stage of the feature extractor is created, and the final layer from each stage is used as feature maps in the bottom-up pathway, as they contain the strongest features. The lateral connections merge the feature maps (C) from the bottom-up pathway with the top-down pathway. Fig. 3.1 shows how a top-down feature map (M) is created by upsampling a smaller top-down feature map by a scale of two. Then, this upsampled top-down feature map is summed with a bottom-up map of the same scale. C is first convolved with a 1×1 kernel to reduce channel depth. The first top-down feature map M_n is created by adding a 1×1 convolution layer to C_n. The final set of feature maps (P) are created by adding a 3×3 kernel to each M. Lin et al. states that C_1 is not included in the pyramid due to its large memory size. Comparing FPN in Fast R-CNN with the baseline Fast R-CNN, the FPN method improves small object average precision by 2.1% [28].

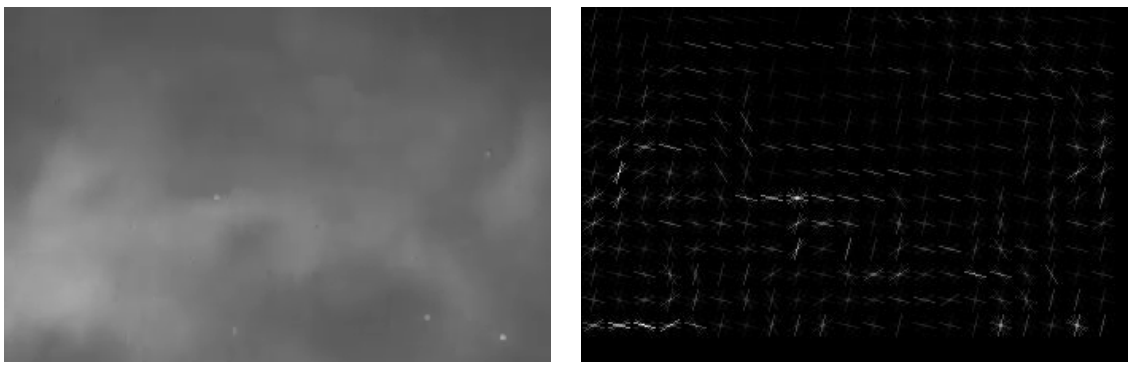
3.3 Infrared Small Target Detection

Due to their size, IR small targets commonly does not exhibit any prominent features which could help classify them. However, an IR image model can be formulated as [20, 31]:

$$f_D = f_L + f_S + f_N, \tag{3.1}$$

which states that the original image f_D is composed of a background layer f_L, a target layer f_S, and a noise layer f_N. As already stated, an IR small target [12] can be defined as an object having a total size of less than 0.15% of an image. This implies that an object detection method relying on a sliding window algorithm, e.g., a traditional object detector, will be quite slow at locating IR small targets.

In long-wave IR images, the combination of a feature extractor such as HOG and a machine learning-based classifier such as SVM is non-successful [63], as low resolution and poor contrast influences the performance. HOG depends on feature descriptors based on the local gradients of an image, which is obtained from the edges or contrast changes within an image. This will not yield satisfactory results in IR images, as shown in Fig. 3.2(b).



(a) (b)

FIGURE 3.2: HOG feature descriptor on IR small target image. (a) Raw image from the SIRST dataset [12]. Image courtesy of Yimian Dai, College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China. (b) HOG descriptors of original image.

3.3.1 Infrared Small Target Detection Methods

The field of IR small target detection has been dominated by model-driven methods. One of the best performing [12] non-learning model-driven methods using low rank and sparse matrix decomposition is the infrared patch-image (IPI) model [20]. This method is described in Section 3.4.1.

CNNs, specifically FPNs, outperform non-learning model-driven methods [13], indicating that learning from data can lead to high accuracy in IR small target detection. However, the majority of CNNs learn high-level features by downsampling the feature maps. As a result, the IR small targets become engulfed by the background features in the deepest layers. To ensure adequate detection results, a specialised network design is required [12, 13]. While calibrating a pre-trained network can help with the problems created by training the feature extractor on general classification images (i.e., colour images), such as the Common Objects in Context (COCO) [9] dataset, the IR images

are significantly different, and the fine-tuning has a negligible effect on performance. As a result, it is not advised [59] to use pre-trained networks (i.e., transfer learning) for the task of IR small target detection, but rather to train the CNN's weights from zero using only IR small target images. In addition, feature maps at different scales are important for generic datasets which contain large, medium, and small objects. However, these feature maps are unnecessary when the network is only utilised for detection of IR small targets.

Meta-architectures such as Faster R-CNN and YOLO only use the last layer's feature map to localise objects and make predictions. These models are ineffective at localising small objects due to the absence of low-level features [6] at the last layer. The Single Shot MultiBox Detector is also not adept at detecting small objects [61]. The problem of detecting small objects can be alleviated by using a more fitting feature extractor (e.g., ResNet) [64]. According to [17], the main drawback of employing a CNN for the task of IR small target detection is that feature learning will become particularly challenging, as an IR small target generally does not have any prominent shape. Further, it is difficult to extract features from low resolution images, and the IR small targets may disappear in the deep layers of a network due to their small size. Dai et al. [13] states that a prediction map of high resolution is crucial for detecting IR small targets, and thus propose the attentional local contrast network (ALCNet). ALCNet achieves better results than the completely data- and model-driven methods on the SIRST dataset (the dataset is discussed in Section 4.1), indicating that when detecting IR small targets, one should prioritise combining CNNs with domain-specific knowledge, e.g., methods for measuring local contrast. To conserve small targets and extract feature maps, ALCNet employs a modified ResNet as its feature extractor.

Wang et al. [59] suggests restricting the number of downsampling operations, thus gaining a sufficiently large feature map which conserves features of the IR small targets. Further, Wang et al. employs residual connections to mitigate the network's *degradation* problem. This results in higher object detection accuracy. The degradation problem and residual connections are discussed in Section 4.2.1.

3.4 Low-Rank and Sparse Matrix Decomposition

Low-rank and sparse matrix decomposition methods [58] try to separate an image into a foreground component $\mathbf{S}$ and a background component $\mathbf{L}$. A sparse matrix [4] has a considerable majority of elements equal to zero. Thus, the IR small targets are often the non-zero values in the sparse matrix, making them easily identifiable. A low-rank matrix [4] has a small number of linearly independent rows and columns compared to the matrix's size. The background patch of an IR image has a low rank, and the IR small targets of the foreground are sparse when compared to the background.

3.4.1 Infrared Patch-Image Model

Evaluation of a model-driven method is quite straightforward because it does not require any training data. The IR patch image (IPI) model proposed by Gao et al. [20] is an example of a model-driven method. The IPI model is a technique for IR small target detection based on low-rank and sparse matrix decomposition, which is capable of producing accurate results even when confronted with extremely complex scenes [58].

However, background edges, corners, or blobs infiltrate the sparse matrix, resulting in multiple discrepancies that the IPI model could treat as targets. In the IPI model, image patches from an IR image are rearranged using a sliding window to form a data matrix $\mathbf{D}$. $\mathbf{D}$ is then decomposed into a low-rank matrix $\mathbf{L}$ and a sparse matrix $\mathbf{S}$ using the robust principal component analysis (RPCA) algorithm in conjunction with principal component pursuit (PCP) [5]. Principal component analysis (PCA) [4] is a widely used technique for dimensionality reduction in situations where one needs to isolate the core of a dataset. The aim of PCA is to fit an ellipsoid to a dataset, with each axis of the ellipsoid representing a *principal component*, and the axis size indicating the amount of variance.

Continuing with the IPI model, $\mathbf{D}$ can be decomposed into two components:

$$\mathbf{D} = \mathbf{L} + \mathbf{S}. \tag{3.2}$$

PCP can recover $\mathbf{L}$ and $\mathbf{S}$ from $\mathbf{D}$ by solving the following optimisation problem [20], i.e., finding the $\mathbf{L}$ and $\mathbf{S}$ that minimises the following expression:

$$\min_{\mathbf{L},\mathbf{S}} \left(\|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1 \right), \text{ subject to } \mathbf{D} = \mathbf{L} + \mathbf{S}. \tag{3.3}$$

However, this decomposition does not consider the noise $\mathbf{N}$ within the image, i.e.:

$$\mathbf{D} = \mathbf{L} + \mathbf{S} + \mathbf{N}. \tag{3.4}$$

Thus, the following problem needs to be solved [20]:

$$\min_{\mathbf{L},\mathbf{S}} \left(\|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1 + \frac{1}{2\mu} \|\mathbf{D} - \mathbf{L} - \mathbf{S}\|_F^2 \right), \tag{3.5}$$

where μ and λ are positive-valued parameters. The problem in Equation (3.5) can be solved through accelerated proximal gradient (APG) described in Algorithm 1.

Algorithm 1 RPCA-PCP via APG [20, 30].

Require: IR patch-image matrix $\mathbf{D} \in \mathbb{R}^{m \times n}$, λ.

$\mathbf{L}_0 = \mathbf{L}_{-1} = 0; \mathbf{S}_0 = \mathbf{S}_{-1} = 0; t_0 = t_{-1} = 1; \bar{\mu} > 0; \eta = 0.99;$

while not converged **do**

$\mathbf{Y}_k^L = \mathbf{L}_k + \frac{t_{k-1}-1}{t_k}(\mathbf{L}_k - \mathbf{L}_{k-1})$.

$\mathbf{Y}_k^S = \mathbf{S}_k + \frac{t_{k-1}-1}{t_k}(\mathbf{S}_k - \mathbf{S}_{k-1})$.

$\mathbf{G}_k^L = \mathbf{Y}_k^L - \frac{1}{2}(\mathbf{Y}_k^L + \mathbf{Y}_k^S - \mathbf{D})$.

$(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}) = \text{svd}(\mathbf{G}_k^L)$.

$\mathbf{L}_{k+1} = \mathbf{U}\mathcal{S}_{\frac{\mu_k}{2}}[\mathbf{\Sigma}]\mathbf{V}^*$.

$\mathbf{G}_k^S = \mathbf{Y}_k^S - \frac{1}{2}(\mathbf{Y}_k^L + \mathbf{Y}_k^S - \mathbf{D})$.

$\mathbf{S}_{k+1} = \mathcal{S}_{\frac{\lambda\mu_k}{2}}\left[\mathbf{G}_k^S\right]$.

$t_{k+1} = \frac{1+\sqrt{4t_k^2+1}}{2}$, $\mu_{k+1} = \max(\eta\mu_k, \bar{\mu})$.

$k \leftarrow k + 1$.

end while

return $\mathbf{S}_k$. ▷ Return the sparse matrix

Gao et al. [20] use $\mu_0 = \sigma_2$, $\bar{\mu} = 0.05\sigma_4$, and $\lambda = \frac{1}{\sqrt{\max(m,n)}}$, where σ_2 and σ_4 are the 2nd and 4th largest singular values from the singular value decomposition (SVD) of $\mathbf{D}$. And $\mathcal{S}$ is the soft-thresholding operator [30]:

$$\mathcal{S}_\epsilon[x] \doteq \begin{cases} x - \epsilon, & \text{if } x > \epsilon, \\ x + \epsilon, & \text{if } x < -\epsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

However, RPCA-PCP via APG requires a significant amount of time to converge (i.e., over $6\,\text{min}$ for a single IR small target image when tested on the machine detailed in Table 4.6). Fortunately, several algorithms exist for solving PCP [3]. A PCP solver known as the inexact augmented Lagrangian method (IALM) is at least $5\times$ faster than APG. IALM is described in detail in Section 4.4.

Singular value decomposition (SVD) [4] is used by Algorithm 1 and IALM. SVD is employed to decompose an $m \times n$ data matrix $\mathbf{D}$ into unitary[1] matrices which contain information on the column $\mathbf{U}$ and row $\mathbf{V}^*$ space of $\mathbf{D}$, including a diagonal matrix $\mathbf{\Sigma}$ as shown in Equation (3.7).

$$\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* = u_1\sigma_1 v_1^* + u_2\sigma_2 v_2^* + u_m\sigma_m v_m^*, \tag{3.7}$$

where $\mathbf{U}$ is an $m \times m$ matrix, $\mathbf{\Sigma}$ is an $m \times n$ matrix, and $\mathbf{V}^*$ is an $n \times n$ matrix. The diagonal entries of $\mathbf{\Sigma}$ are known as the singular values of the data matrix $\mathbf{D}$, and indicate the importance of $\mathbf{U}$ and $\mathbf{V}$. The number of non-zero singular values is equal to the rank of $\mathbf{D}$, i.e., a low-rank matrix has a small number of singular values. For example, a repeating background in an image has a low rank. However, since the $\text{rank}(\mathbf{D}) = m$ at most, the SVD of $\mathbf{D}$ yields:

$$\mathbf{D} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\mathbf{V}^*, \tag{3.8}$$

where $\hat{\mathbf{U}}$ and $\hat{\mathbf{\Sigma}}$ are the first m columns of $\mathbf{U}$, and first $m \times m$ block of $\mathbf{\Sigma}$, respectively. This is known as economy SVD and can be implemented in Python using the NumPy library function [10] shown in Listing 3.1. Economy SVD [4] is suited for matrices with a significantly larger number of columns than rows.

```
1 import numpy
2 U, Sigma, V = numpy.linalg.svd(matrix, full_matrices=False)
```

LISTING 3.1: NumPy library function for implementing economy SVD in Python.

[1] A unitary matrix is defined as $\mathbf{X}^*\mathbf{X} = \mathbf{X}\mathbf{X}^* = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

Chapter 4

Methodology

This chapter outlines the methods used to collect and analyse the image dataset employed in this work. The methods used to develop the system, and the rationale for selecting these methods are discussed. In this work, two different methodologies for detection of IR small targets are proposed and tested, a data-driven CNN-based method and a model-driven method using low-rank and sparse matrix decomposition. Further, this chapter details the testing process.

4.1 IR Small Target Dataset Analysis

A dataset of IR small target images is required to train the data-driven method, and test both the model- and data-driven methods. However, there are few publicly available datasets of IR small targets. The datasets for IR small target detection that exist are not easily accessible and contain a small number of samples. The primary reason for this is that IR imaging cameras are expensive and not widely available to the public. As a result, creating a dataset of thermal (i.e., long-wave IR) images is an expensive and time-consuming task for those without access to an IR imaging camera.

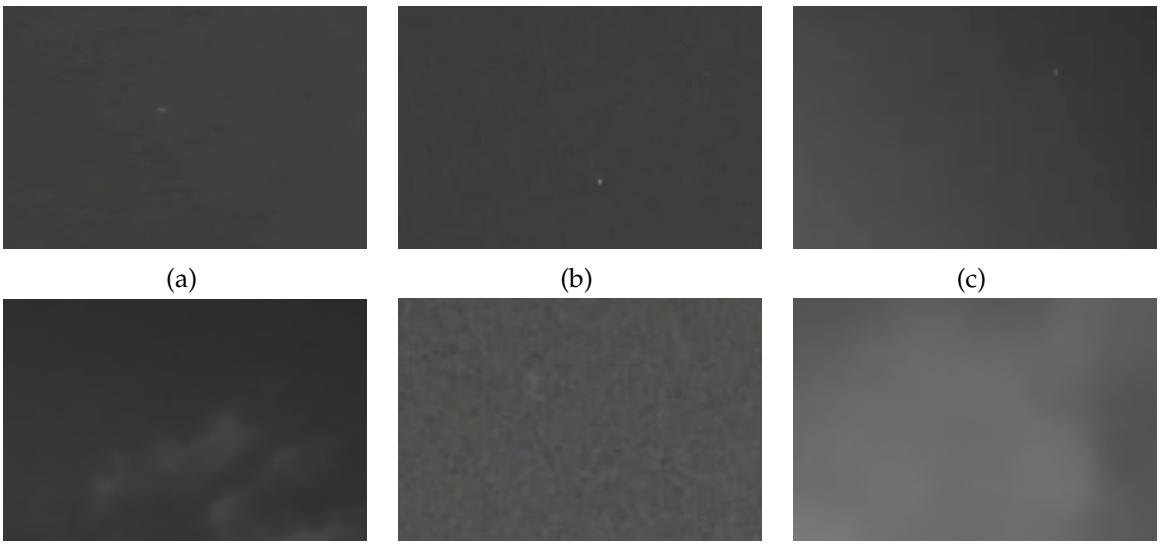
The **single-frame IR small target (SIRST) dataset** [11], which contains 427 short-wave IR and mid-wave IR images, is used for training and testing the proposed methods. Short-wave IR and mid-wave IR are suited for objects with temperatures in the range $300\,^{\circ}\mathrm{C}$ to $3000\,^{\circ}\mathrm{C}$. However, the final system is designed to utilise long-wave IR images, which is suited for targets with temperatures in the range $-70\,^{\circ}\mathrm{C}$ to $90\,^{\circ}\mathrm{C}$. In IR images, hot objects will consist of white pixels, and cold objects will appear as black pixels. Images in the SIRST dataset have a width and height ranging from $150\,\mathrm{px}$ to $440\,\mathrm{px}$. The sample size of the dataset was increased since the model will perform better with more samples and because transfer learning cannot be utilised. Further, scarcity or low variance in the training data will result in a model that performs poorly on new data. The dataset was augmented with 642 images by manipulating the original images. The images were mirrored around their x-axis, rotated 90° clockwise, and modified by shifting an image horizontally and zooming in on specific areas. Also, the brightness of the images was adjusted. This resulted in 210 images which did not contain any targets. For example the images shown in Fig. 4.1(d), 4.1(e) and 4.1(f). These images will be used as negatives for testing the final methods. The original SIRST dataset does not include any negative samples.

The content of the new, augmented SIRST dataset is listed in Table 4.1. As suggested by Goodfellow et al. [21], about 80% of the positive samples in the dataset is used for training and 20% is used for validation and testing.

TABLE 4.1: Augmented SIRST dataset.

Stage	Number of images	Number of objects
Training	855	1030
Testing	214	268
Negative	210	0
Total	1279	1298

Fig. 4.1 shows an excerpt from the dataset. It is obvious that the targets shown in Fig. 4.1(a), 4.1(b) and 4.1(c) are dim and absorbed by the noisy background. Certain targets are difficult for humans to discover as they require one to perform a focused and thorough search and consider whether they are a target or just noise. In short, the classification task of IR small target detection is binary [12], and as most of the targets in the images lack any definite features, all of them are placed into a general class called "Target".



(a) (b) (c)

(d) (e) (f)

FIGURE 4.1: Samples from the modified SIRST dataset. (a) – (c) contains IR small targets. (d) – (f) contains no IR small targets.

The new dataset is labelled using the annotation tool LabelImg [29]. A rectangular box is drawn around each object discovered in the dataset. These boxes are known as ground truth bounding boxes. The annotations are saved as XML files in the PASCAL VOC format as shown in Listing 4.1. The XML files are then converted into the TensorFlow record format [54], which are binary files optimised for TensorFlow.

```
<annotation>
  <folder>dataset</folder>
  <filename>img.png</filename>
  <path>/dataset/img.png</path>
  ...
  <size> <!--Image size-->
    <width>202</width>
    <height>289</height>
    <depth>1</depth>
  </size>
  ...
  <object>
    <name>Target</name>  <!--Object class-->
    ...
    <bndbox>  <!--Bounding box: object location-->
      <xmin>90</xmin>
      <ymin>135</ymin>
      <xmax>109</xmax>
      <ymax>152</ymax>
    </bndbox>
  </object>
</annotation>
```

LISTING 4.1: Annotated dataset sample in PASCAL VOC format.

4.2 Data-driven Approach

Numerous CNN models are available online, and it is difficult to differentiate between them. A model from the *TensorFlow 2 Detection Model Zoo* [49] is selected to be the data-driven method, as these models are readily available. A model with a good trade-off between accuracy and speed (i.e., inference time for a single image) is desired. Table 4.2 shows an excerpt from the TensorFlow Model Zoo. Based on the proposals from the literature review, the final choice fell on CenterNet ResNet50 V1 FPN (512×512). This model achieves an accuracy of 31.2% mAP (mean average precision) on the COCO dataset at an average speed of $27\,\mathrm{ms}$ per image. A model with a higher accuracy can be deployed if edge computing is utilised, see Section 5.2.2 for a discussion on local and edge processing.

TABLE 4.2: Object detection models from TensorFlow 2 Detection Model Zoo. The speed is the average inference time required for a single image.

Model	**Speed**	**COCO mAP**
CenterNet MobileNetV2 FPN 512×512	$6\,\mathrm{ms}$	23.4%
CenterNet ResNet50 V1 FPN 512×512	$27\,\mathrm{ms}$	31.2%
CenterNet ResNet50 V2 512×512	$27\,\mathrm{ms}$	29.5%
EfficientDet D2 768×768	$67\,\mathrm{ms}$	41.8%

Further, results from [65] show that, on a $512 \times 512\,\mathrm{px}$ input image, CenterNet with ResNet18 achieves 28.1% COCO mAP at $7\,\mathrm{ms}$ per image frame, which amounts to 142

FPS. On the same test, YOLOv3 with Darknet-53 achieved a 33% COCO AP (average precision) at 20 FPS.

4.2.1 Residual Neural Network (ResNet)

Having a deep network is desirable, as a deeper network can learn more features. However, as the network gets deeper, there may be instances where the accuracy saturates and then rapidly decreases. This is known as degradation [61]. Also, a deeper network leads to more parameters, which results in a more resource intensive model. As a solution to this problem, ResNet [23] introduced the residual block as shown in Fig. 4.2.

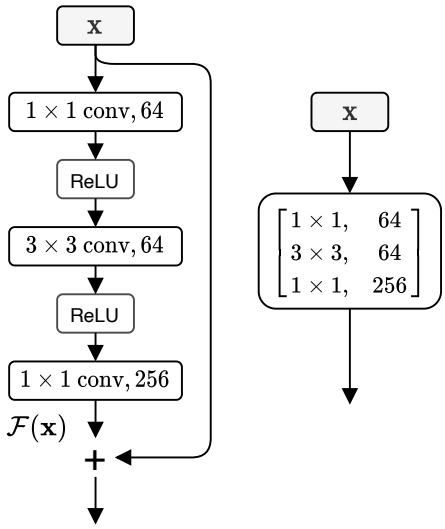


FIGURE 4.2: Residual blocks. The blocks are equivalent, where the figure to the right is a compressed version.

The residual block takes the output $\mathcal{F}(\mathbf{x})$ of one or more layers and combines it with a shortcut connection containing the value $\mathbf{x}$ which is feeding those layers. Since the module prevents degradation, the network's depth can increase, and the accuracy will improve over time. Results from [23] show that the effect of the residual connections increases proportionally with the number of layers.

ResNet uses **batch normalisation** [21] right after each convolution. Batch normalisation is a dynamic reparameterisation technique that was instigated by the challenges of training deep networks. The gradient (from gradient descent) specifies how each parameter of a layer should be updated if the other layers in the network remain unchanged. However, all layers are updated concurrently, and this might lead to unwanted outcomes since the gradient assumes that the functions in other layers stay unchanged. Batch normalisation effectively simplifies the process of synchronising the updates in multiple layers and can be introduced to the network's input and hidden layers.

ResNet with FPN achieves a $2-3\%$ increase in accuracy as compared to the original ResNet [36]. A ResNet with 50 layers is shown in Fig. 4.3.

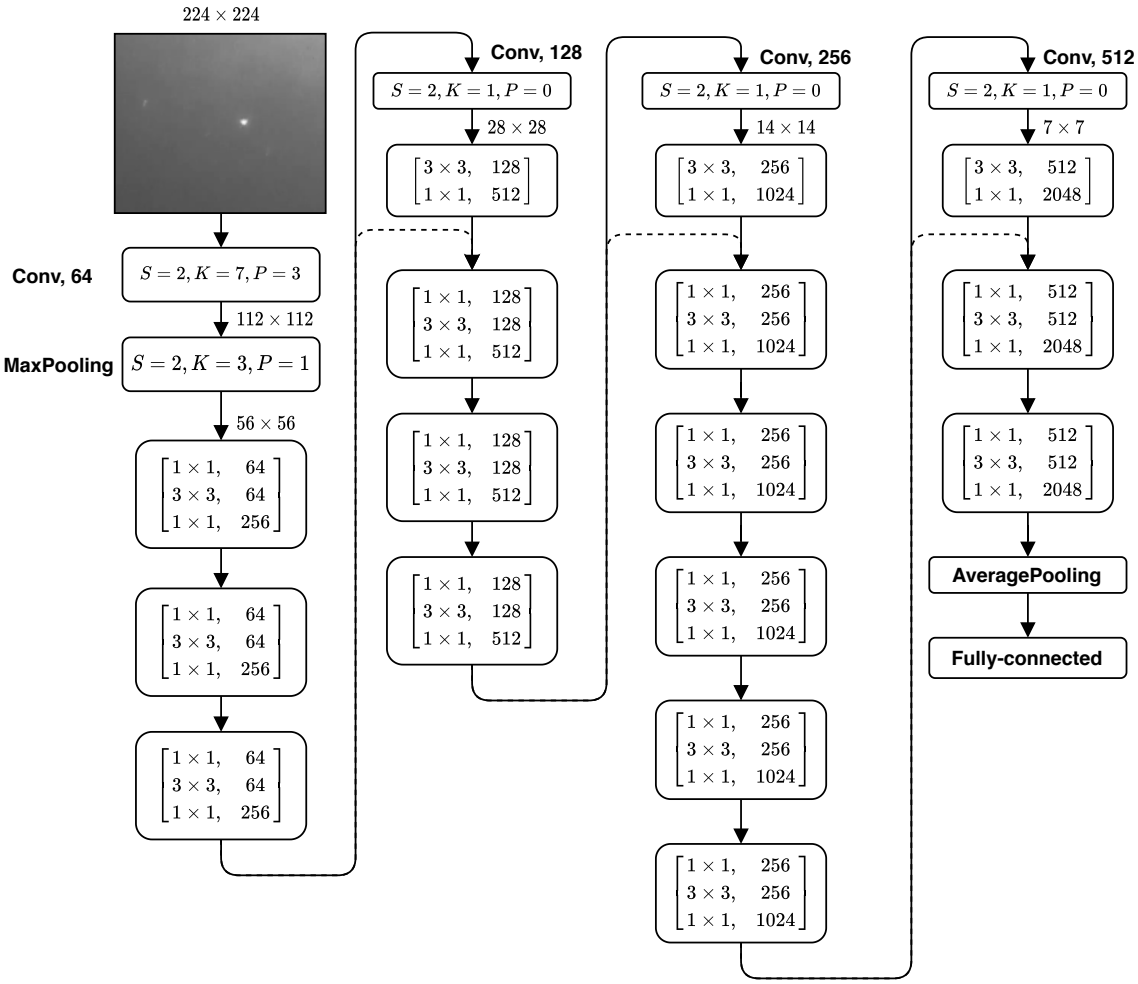


FIGURE 4.3: ResNet50 architecture.

4.2.2 Modified ResNet

The ResNet50 V1 FPN (512×512) is used as the object detector's feature extractor. The SIRST dataset contains images that are smaller than the original 512×512 px input size to the network. Thus, instead of upscaling the input images to 512×512 px, which would distort them, they are resized to 224×224 px. This is also performed for the original ResNet50.

Based on suggestions amassed from the literature review, the downsampling operations of the feature extractor is reduced to improve the detection of small objects. To achieve satisfactory results in terms of accuracy, the depth of the ResNet is maintained at 50 layers. The first convolutional layer of the original ResNet50 has a stride of 2, a kernel size of 7, and a padding of 3. The downsampling is reduced by changing the stride from 2 to 1 in the first convolutional layer, as shown in Listing 4.2.

```
# file: tensorflow/python/keras/applications/resnet.py
x = layers.Conv2D(64, 7, strides=1, # original strides=2
    use_bias=use_bias, name='conv1_conv')(x)
```

LISTING 4.2: Excerpt from a Python implementation of ResNet [53]: Reducing stride of the first convolutional layer.

The output shape after a spatial convolutional operation can be found by the following expression [16]:

$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1, \tag{4.1}$$

where i is the input square shape, k is the square kernel size, p is the padding, and s is the stride. The output shape of the first convolutional layer becomes:

$$o = \left\lfloor \frac{224 + 2(3) - 7}{1} \right\rfloor + 1 = 224, \text{ i.e., } 224 \times 224. \tag{4.2}$$

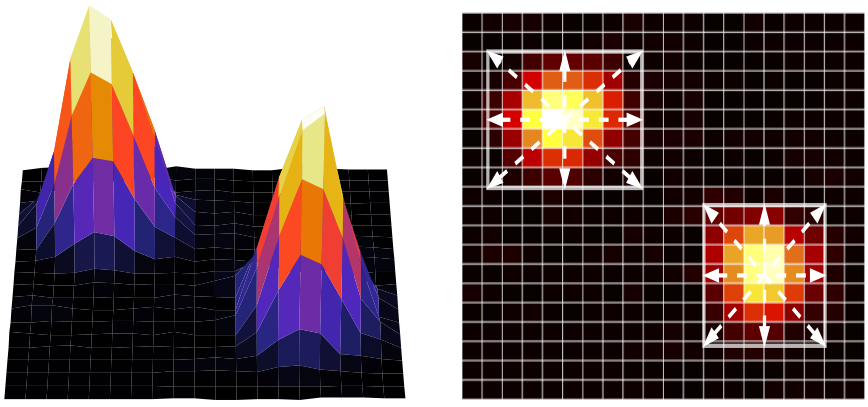
The shapes of the modified ResNet50 are listed in Table 4.3. Notice in Fig. 4.3, the output shape of the original ResNet's last convolutional layer "Conv, 512" has an output shape of 7×7.

TABLE 4.3: Shapes of modified ResNet.

Layer	Input shape	Output shape
Conv, 64	224×224	224×224
Max. pool	224×224	112×112
Conv, 128	112×112	56×56
Conv, 256	56×56	28×28
Conv, 512	28×28	14×14

4.2.3 CenterNet

CenterNet [65] is a keypoint-based object detector, which means that it represents an object as a single point in the centre of its bounding box. Other object properties, such as size and dimension, are obtained by moving from the centre location towards the bounding box's outline, as illustrated in Fig. 4.4(b).



(a) (b)

FIGURE 4.4: (a) Peaks in heatmap of two key-points. (b) Predicting object size from key-points.

First, an input image $\mathbf{D}$ of width W and height H is fed into a feature extractor (e.g. ResNet) in order to create a key-point heatmap $\hat{Y} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$, in which R is

the stride, and $\mathcal{C}$ is the number of classes. Peaks in the heatmap shown in Fig. 4.4(a) are mapped as object centre points $\hat{\mathcal{P}}$. The objects bounding box size is inferred from the centre points. Inference is performed in a single forward-pass, where the prediction $\hat{Y}_{x,y,c} = 1$ represents a detected key-point, and $\hat{Y}_{x,y,c} = 0$ corresponds to the background. CenterNet uses the classification loss function shown in Equation (4.3) termed penalty reduced logistic focal loss, with $\alpha = 2.0$ and $\beta = 4.0$.

$$L = \frac{-1}{N} \sum_{xyc} \begin{cases} \left(1 - \hat{Y}_{xyc}\right)^{\alpha} \log\left(\hat{Y}_{xyc}\right), & \text{if } Y_{xyc} = 1, \\ \left(1 - Y_{xyc}\right)^{\beta} \left(\hat{Y}_{xyc}\right)^{\alpha} \log\left(1 - \hat{Y}_{xyc}\right), & \text{otherwise,} \end{cases} \tag{4.3}$$

where N is the number of key-points in $\mathbf{D}$.
CenterNet does not employ sliding windows or non-maximum suppression. Non-maximum suppression is a post-processing method which retains only the proposed bounding boxes (also known as anchor boxes) with the most confident prediction scores, while suppressing adjacent and overlapping bounding boxes with lower scores.

4.3 Training the Data-driven Method

A prototype of the data-driven method was implemented using the Python programming language and the machine learning library TensorFlow. Python was selected as the preferred programming language as it is a simple language suitable for rapid software development, and has an extensive number of libraries, documentation, and examples readily available. **TensorFlow** is an open-source machine learning library developed by Google in C++, Python and CUDA, which allows users to create, train, and make use of neural network models. Several APIs are available, with the Python API being the most comprehensive and stable. In TensorFlow, a computational graph of edges and nodes is used to build a deep learning model [37]. The edges represent data in the form of tensors (i.e., multi-dimensional arrays), and the nodes perform operations on the tensors (e.g., multiplications). The *TensorFlow Object Detection API* [52] was utilised in this thesis to modify and train the CenterNet ResNet50 V1 FPN 512×512, as well as perform inference on images from the SIRST dataset.

Training CNNs relies a great deal on matrix multiplications, which can be performed in parallel. GPUs are well-suited for this type of computation, as their architecture allows for $100\times$ greater speed than CPUs at this task [37]. TensorFlow code is optimised for GPU execution using CUDA and cuDNN (CUDA Deep Neural Network library). Additionally, neural network models can be trained in parallel across GPU clusters using TensorFlow, thus providing greater amounts of available memory. This accelerates the training process and enables the creation of deeper networks.

Specifications of the machine used for training the data-driven methods are found in Table 4.4.

TABLE 4.4: Cloud GPU platform [38] specifications used for training data-driven methods.

Specifications	
OS	Ubuntu 18.04 LTS
CPU	Intel® Xeon® E5-2623 v4 @ 2.6 GHz
GPU	NVIDIA® Quadro® P5000 (16 GB)
RAM	30 GB
TensorFlow	2.1.0

4.3.1 Training Configuration

The data-driven models are trained from scratch. Unnecessary dataset augmentation methods are removed from the training pipeline (i.e., a file describing the training configuration). This includes methods for adjusting the hue, contrast, saturation, and brightness. Furthermore, the network's input shape was reduced from 512×512 px to 224×224 px as mentioned earlier.

The **learning rate** determines how fast the network learns. Goodfellow et al. [21] states that a high learning rate increases the training loss, while a low learning rate increases the risk of a slow training process, which potentially could get stuck at a high training loss. First, an initial learning rate, i.e., the warmup learning rate, is selected, then it is gradually increased. As shown in Listing 4.3, the default learning rate method for the chosen model is a cosine learning rate decay.

```
learning_rate {
    cosine_decay_learning_rate {
        learning_rate_base: 0.001
        total_steps: 250000
        warmup_learning_rate: 0.00025
        warmup_steps: 5000
    }
}
```

LISTING 4.3: Pipeline learning rate configuration for CenterNet ResNet50 V1 FPN.

When training a CNN, it is desirable to find and use hyperparameters that minimise the training loss as much as possible. But this is not feasible with constraints such as insufficient hardware, and limited time. Methods such as grid search and random grid search can be used to find the optimal hyperparameters [21]. Grid search is an inefficient method which tests every combination of hyperparameters and selects the best performing parameters. Whereas the random grid search randomly selects hyperparameters and picks the best performing parameters. However, some of these algorithms require hyperparameters of their own, which makes this a futile process. For the sake of simplicity, the original hyperparameters shown in Table 4.5 will be used, as these are commonly fine-tuned by the model's developers. These hyperparameters are used for both the modified and the original ResNet50-based methods.

TABLE 4.5: Pipeline values used for training the data-driven method.

Pipeline values	
Warmup learning rate	2.5×10^{-4}
Base learning rate	0.001
Batch size	64 / 32
Warmup steps	5000
Max. number of boxes	100

The **batch size** [21] is the number of training-samples from the dataset used in a single forward-pass. Typically, the batch size is less than the total number of training samples in the dataset. A large batch size consumes more memory and thus represents a constraint. The batch size is frequently a power of two, typically between 2^5 to 2^8. A batch size of 1 enables the network to generalise well to new data samples (i.e., test-samples). This requires a small learning rate, which lengthens the training process.

The data-driven method based on the original ResNet uses a batch size of 64. The modified ResNet uses a batch size of 32 since the GPU resources became exhausted, i.e., out of memory. The training is stopped when the loss is stagnating. The training of the modified ResNet50 was stopped when the total training loss reached approximately 0.3, requiring significantly more steps than the original ResNet50, which had a training loss of approximately 0.15, as illustrated in Fig. 4.5. If the training process is prolonged, the model is likely to overfit the training data. Both original and modified versions of ResNet50 have a size of $111.7\,\mathrm{MB}$.

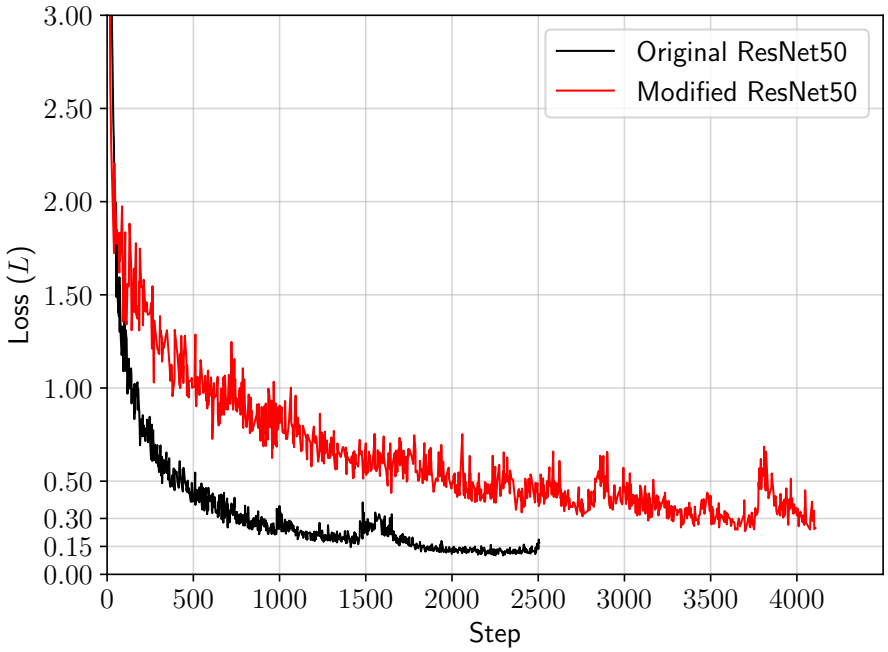


FIGURE 4.5: Training results of CenterNet with ResNet50-FPN. The original ResNet50 obtains a loss $L \approx 0.15$, and the modified ResNet50 obtains a $L \approx 0.3$.

4.4 Model-driven Approach

The proposed model-driven method shown in Fig. 4.6 is based on the IPI model [20] and RPCA-PCP via IALM [30].

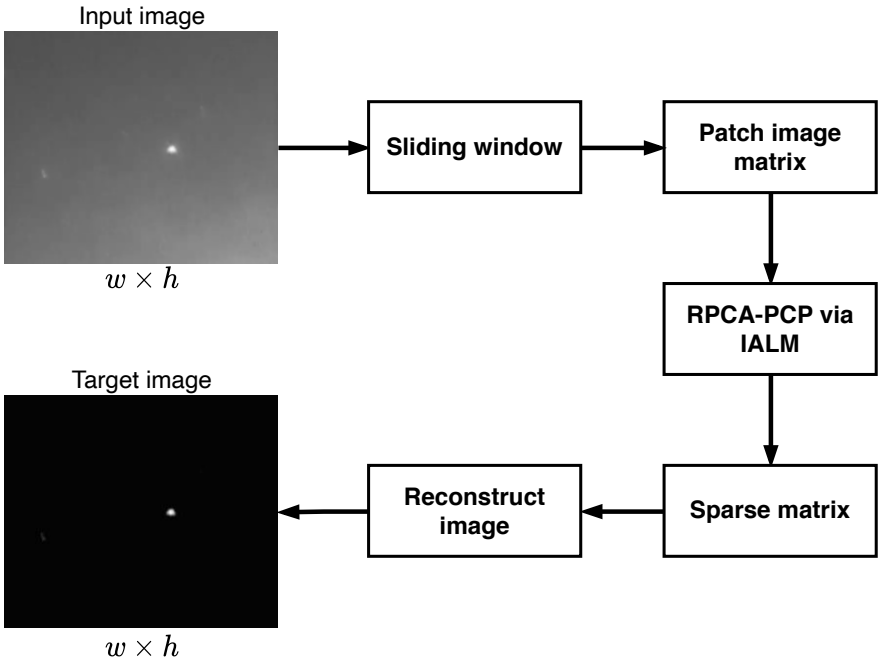


FIGURE 4.6: Proposed model-driven method based on the IPI model.

IALM is listed in Algorithm 2 and is at least $5\times$ faster than APG (Listed in Algorithm 1) and has a higher precision [3].

Algorithm 2 RPCA-PCP via IALM [30].

Require: IR patch-image matrix $\mathbf{D} \in \mathbb{R}^{m\times n}$, λ, compute $J(\mathbf{D})$.
1: $\mathbf{Y}_0 = \frac{\mathbf{D}}{J(\mathbf{D})}; \mathbf{S}_0 = 0; \mu_0 = \frac{1.25}{\|\mathbf{D}\|_2}$;
2: **while** not converged **do**
3: $(\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}) = \mathrm{svd}(\mathbf{D} - \mathbf{S}_k + \mu_k^{-1}\mathbf{Y}_k)$.
4: $\mathbf{L}_{k+1} = \mathbf{U}\mathcal{S}_{\mu_k^{-1}}[\mathbf{\Sigma}]\mathbf{V}^*$.
5: $\mathbf{S}_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}\left[\mathbf{D} - \mathbf{L}_{k+1} + \mu_k^{-1}\mathbf{Y}_k\right]$.
6: $\mathbf{Y}_{k+1} = \mathbf{Y}_k + \mu_k(\mathbf{D} - \mathbf{L}_{k+1} - \mathbf{S}_{k+1})$.
7: Update μ_k to μ_{k+1}.
8: $k \leftarrow k + 1$.
9: **end while**
10: **return** $\mathbf{S}_k$. ▷ Return the sparse matrix

In Algorithm 2, $J(\mathbf{D}) = \max\left(\|\mathbf{D}\|_2, \lambda^{-1}\|\mathbf{D}\|_\infty\right)$, and μ_{k+1} is given by:

$$\mu_{k+1} = \begin{cases} \rho\mu_k, & \text{if } \mu_k \frac{\|\mathbf{S}_{k+1}-\mathbf{S}_k\|_F}{\|\mathbf{D}\|_F} < \epsilon_2, \\ \mu_k, & \text{otherwise,} \end{cases} \tag{4.4}$$

where $\epsilon_2 = 10^{-5}$. However, this approach is slow. A faster approach [47] which yield equally satisfactory results is to let $\mu_{k+1} = \min(\rho\mu_k, \bar{\mu})$, where $\bar{\mu} = 10^7\mu_k$, and $\rho = 1.6$.

Further, the convergence criterion (also known as the stopping criterion) is defined as:

$$\frac{\|\mathbf{D} - \mathbf{L}_k - \mathbf{S}_k\|_F}{\|\mathbf{D}\|_F} < \epsilon_1, \tag{4.5}$$

where the tolerance $\epsilon_1 = 10^{-7}$ originally [30]. However, it was discovered that $\epsilon_1 = 10^{-2}$ gives decent results, at a much faster rate.

The original IPI model [20] post-processes the reconstructed target image with a median filter. Various post-processing methods are illustrated in Fig. 4.7. The best results are obtained using the median and 10th percentile filters. By comparing the surface plot of the median filter in Fig. 4.7(b) to the surface plot of the 10th percentile filter in Fig. 4.7(d), it is clear that the 10th percentile method reduces the intensity of the most prominent noise. The median filter, on the other hand, has a lower total amount of low-level noise. The mean filter shown in Fig. 4.7(e) performs similarly to the median filter, but the noise is more intense with the mean filter.

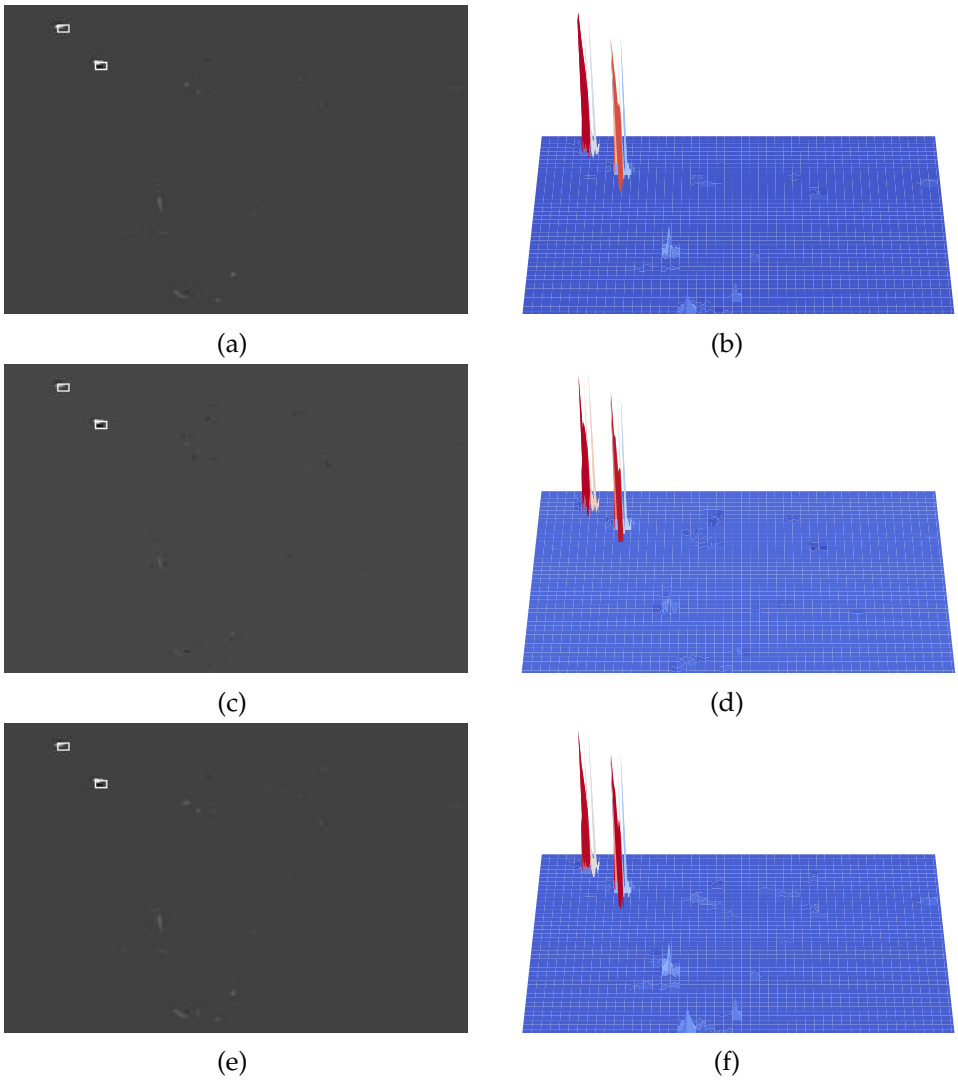


FIGURE 4.7: Different post-processing methods of target-patch. The white rectangles are ground truth bounding boxes. (a) – (b) obtained using median filter. (c) – (d) obtained using 10th percentile filter. (e) – (f) obtained using mean filter.

The model-driven methods will utilise the median filter. Further, pixels with values greater than a predefined threshold in the resulting post-processed target image are then considered to be IR small targets. For instance, pixels with a value greater than 150 in Fig. 4.8 can be considered part of an IR small target.

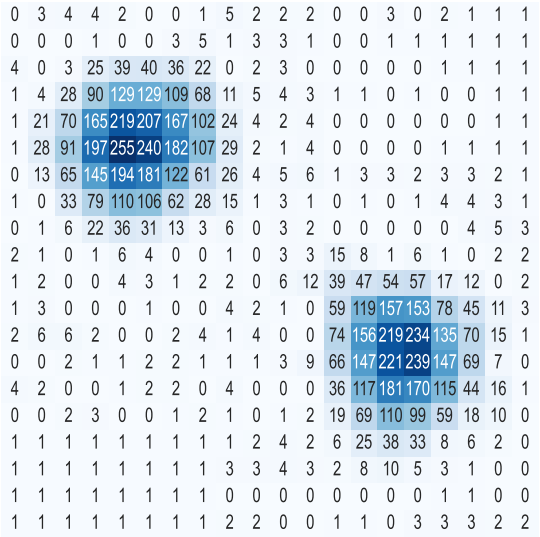


FIGURE 4.8: Pixel values of IR small target after being processed by the model-driven method.

4.5 Evaluation Metrics

This section describes the evaluation process for the proposed methods. For evaluation of the methods, the modified SIRST dataset mentioned in Section 4.1 is used.

In the context of search and rescue, it is more undesirable to miss an actual target than to make a false prediction. As a result, the evaluation metrics cannot be as straightforward as determining the accuracy. That is, missing a target should have a greater impact on the final score than incorrectly detecting an object that is not a target. For the **model-driven** method, the sparse matrix is the prediction. The centre of a predicted target will be the location of pixels with a value higher than a certain threshold. This method does not use the rectangular values of the ground truth bounding box GT_{bbx}, but a circle with a small radius. The centre GT_{bbx_c} coordinates are converted from a rectangle to a circle by computing the centre (x_c, y_c) of each GT_{bbx}:

$$GT_{bbx_c} = (x_c, y_c) = \left(\frac{x_{max} - x_{min}}{2}, \frac{y_{max} - y_{min}}{2} \right), \tag{4.6}$$

where x_{max}, x_{min}, y_{max}, and y_{min} represent the corners of a bounding box. The accuracy of the model-driven methods is measured by checking if the centre of the GT_{bbx_c} intersects with the centre of the predicted target $\hat{y}_c$.

For object detection methods similar to the **data-driven method** it is common to calculate the overlap between the ground truth bounding box and the predicted bounding box $\hat{y}_{bbx}$ using intersection over union (IoU), as shown in Equation (4.7). However, IoU

does not work well for the proposed methods, as the ground truth boxes are significantly smaller than the predicted boxes. When a predicted box is located in the same area as a ground truth box, the predicted box generally has a larger size, and this yields a poor IoU.

$$\text{IoU} = \frac{|\hat{y}_{bbx} \cap GT_{bbx}|}{|\hat{y}_{bbx} \cup GT_{bbx}|} \tag{4.7}$$

A more suitable metric proposed by Dai et al. [13] for data-driven and model-driven methods is the **normalised IoU**:

$$\text{nIoU} = \frac{TP}{T + P - TP}, \tag{4.8}$$

where P is the number of true positive and false negative predictions, i.e., $P = TP + FN$, and T is the number of ground truth samples in the training data.

Additionally, the following basic outcomes, true positive (TP), false positive (FP), false negative (FN), and true negative (TN) are recorded after testing each method. A prediction is classified as true positive if the method correctly predicts the existence of an object. To deem a prediction to be true positive, the predicted location must be within proximity of the ground truth location. This includes the situations where the predicted bounding box and the ground truth bounding box are proper subsets of one another, i.e., $GT_{bbx} \subset \hat{y}_{bbx} \vee \hat{y}_{bbx} \subset GT_{bbx}$.

A false positive prediction occurs when the method incorrectly predicts the existence of an object. This includes the prediction of an object in a location which is not within a proximity of the ground truth bounding box. Next, a prediction is classified as false negative when the method incorrectly predicts that an object does not exist. Lastly, a true negative prediction is when the method correctly predicts that a target does not exist. The above-mentioned conditions can be used to calculate the evaluation metrics listed below:

- **Precision**, also known as positive predictive value (PPV) [35] measures the percentage of correct detections. Namely, a measure of how many of the predicted targets correspond to ground truth targets. The precision is given by the following equation:

$$PPV = \frac{TP}{TP + FP}. \tag{4.9}$$

- **Recall**, also known as true positive rate (TPR) [35], is a measure of how many true positives the system truly detected. In other words, the number of ground truth targets detected. The recall is given by the following equation:

$$TPR = \frac{TP}{TP + FN}. \tag{4.10}$$

- The **false positive rate** (FPR), also known as false alarm rate [35] is given by the following equation:

$$FPR = \frac{FP}{FP + TN}. \tag{4.11}$$

A lower FPR score implies better performance.

Another metric is the **F-score**, e.g., the F_1 score [35], which is the harmonic mean of precision and recall:

$$F_1 = 2\frac{PPV \times TPR}{PPV + TPR}; \tag{4.12}$$

however, because F_1 equally weights the importance of recall and precision, it is preferable to use F_β [43] to give recall a $\beta\times$ greater impact than precision:

$$F_\beta = \frac{(1+\beta^2)(PPV)(TPR)}{\beta^2 PPV + TPR}, \tag{4.13}$$

for the proposed system, it is preferable to select $\beta = 2$, as the recall value is more critical for evaluating the proposed methods.

Another applicable metric is the **Matthews correlation coefficient** (MCC), which returns a value between -1 and 1:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}. \tag{4.14}$$

The modified SIRST dataset is unbalanced, as it contains 268 positive samples and 210 negative samples. Chicco and Jurman [8] recommend using the MCC rather than the F_1 or accuracy (ACC) when evaluating predictions from a binary classifier, since ACC and F_1 can produce inaccurate results when applied to unbalanced datasets. ACC is defined as the following:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4.15}$$

MCC can resolve this issue by assimilating the imbalance. MCC assigns a high score to a classifier that performs well on all four conditions (i.e., TP, FN, TN, and FP). A score of 1 indicates that all predictions are correct, -1 indicates that all predictions are incorrect, and a score of 0 means that the classifier behaves as if the predictions were randomly produced.

The proposed methods should have a low amount of false negative predictions and a high amount of true positive predictions. According to the above-mentioned metrics, this equates to a high recall, a high F_β, and a high MCC. Additionally, the methods should have a high nIoU score, as this is the metric used to measure performance on the SIRST dataset.

The processing time, also known as **inference time**, is the time a method spends on making a prediction from an unseen image. The inference time required for a single image can be converted into frames per second (FPS). For example, if a method processes a single image in $100\,\mathrm{ms}$, this translates to $1/100$ frames per ms, or 100 FPS.

4.6 Testing

The modified SIRST dataset was used for experimental evaluation. The dataset has 210 negative images, and 214 positive images containing 268 IR small targets. The model- and data-driven approaches were compared to the best performing methods from [13]

and [12] in terms of nIoU on the SIRST dataset, which are the infrared patch-image (IPI) model [20], attentional local contrast network (ALCNet) [13], and asymmetric contextual modulation with U-Net (ACM-U-Net) [12]. All testing of the methods proposed in this thesis were conducted on the machine specified in Table 4.6.

TABLE 4.6: Hardware test specifications used for testing data-driven and model-driven methods.

Specifications	
OS	macOS Big Sur 11.2.3
CPU	Intel® Core i5 @ 2.7 GHz
GPU	Intel® Iris Graphics 6100 (1536 MB)
RAM	8 GB (1867 MHz DDR3)
TensorFlow	2.4.1

The **model-driven methods** are evaluated by adjusting the parameters shown in Table 4.7. The tolerance ϵ_1 is required by the stopping criterion. If the value of the stopping criterion is below ϵ_1 the solution of RPCA-PCP via IALM has converged. The *iteration* parameter is used to forcibly stop IALM if it has not converged. Further, adjacent pixels with a value above the *threshold* produce an IR small target. A high threshold removes false positives, but it could also exclude true positive predictions.

TABLE 4.7: Model-driven methods and their parameters.

Abbreviation	**Tolerance** (ϵ_1)	**Iterations**	**Stride**	**Patch size**	**Threshold**
MD-v1	0.1	500	20	80	150
MD-v2	0.01	1000	20	80	150

According to [20], a patch size of 80×80 and a sliding step or stride of 14 in the sliding window produces acceptable results. However, a stride of 20 was selected for MD-v1 and MD-v2 as this decreases the required computational time. If the patch size exceeds 80×80, performance degrades. When the IR small targets are small, a smaller stride is recommended.

The score threshold S_{th} is the only adjustable parameter when evaluating the **data-driven methods**. The score threshold discards predictions which have a confidence score less than S_{th}. Table 4.8 contains abbreviations used for the various data-driven methods.

TABLE 4.8: Abbreviations for data-driven methods.

Abbreviation	**Feature extractor**	**Score threshold** (S_{th})
DD-v1-03	Original ResNet50	0.3
DD-v1-05	Original ResNet50	0.5
DD-v2-03	Modified ResNet50	0.3
DD-v2-05	Modified ResNet50	0.5

Chapter 5

Results and Discussion

This chapter summarises the results and discusses the significance of the results in light of the research proposal.

5.1 Test Results

All results from evaluating the data-driven and model-driven methods on the modified SIRST dataset are shown in Table 5.1.

TABLE 5.1: Results from evaluating the data-driven and model-driven methods.

	MD-v1	**MD-v2**	**DD-v2-03**	**DD-v2-05**	**DD-v1-03**	**DD-v1-05**
Recall	0.610	0.711	0.885	0.722	**0.904**	0.800
FPR	0.386	0.733	0.038	**0.010**	0.086	0.038
Precision	0.585	0.345	0.966	**0.990**	0.926	0.963
MCC	0.224	-0.024	**0.842**	0.720	0.817	0.760
F_β	0.604	0.586	0.900	0.763	**0.908**	0.828
nIoU	0.397	0.328	0.772	0.561	**0.774**	0.650

The best performing data-driven and model-driven methods in terms of nIoU are DD-v1-03 and MD-v1, respectively. As shown in Table 5.2, these two are compared with the best performing methods from [13] and [12].

TABLE 5.2: normalised IoU (nIoU) of the best performing data- and model-driven methods, and the best performing methods from [13] and [12].

	MD-v1	**IPI**	**ALCNet**	**ACM-U-Net**	**DD-v1-03**
nIoU	0.397	0.607	0.728	0.731	**0.774**

The preliminary findings from Table 5.2 suggest that DD-v1-03 (nIoU $= 0.774$) outperforms ACM-U-Net (nIoU $= 0.731$) and ALCNet (nIoU $= 0.728$). This is perhaps not surprising as ACM-U-Net [12] uses a ResNet with 26 layers, and ALCNet [13] uses a ResNet with 20 layers, thus achieving a lower nIoU than DD-v1-03 and DD-v2-03 (nIoU $= 0.772$). However, there are systematic differences in the setup used for testing the methods. And it is possible that the proposed data-driven methods achieve a higher nIoU as they are trained on the augmented SIRST dataset which contains more samples. It is therefore misleading to state that DD-v1-03 outperforms the methods in Table 5.2.

TABLE 5.3: Average time (in seconds) used for processing a single image.

	MD-v1	**MD-v2**	**DD-v2-03**	**DD-v2-05**	**DD-v1-03**	**DD-v1-05**
Avg. time [s]	4.98	5.04	0.85	0.8	**0.2**	**0.2**

The average FPS of the proposed methods is depicted in Fig. 5.1. The FPS was calculated using the average processing time required for a single image, which is listed in Table 5.3. The DD-v1 methods are clearly the fastest methods with an average speed above 4 FPS. The FPS of the methods from Table 5.2 were not included because they were tested on different hardware, and their results are not comparable to the methods proposed in this thesis.

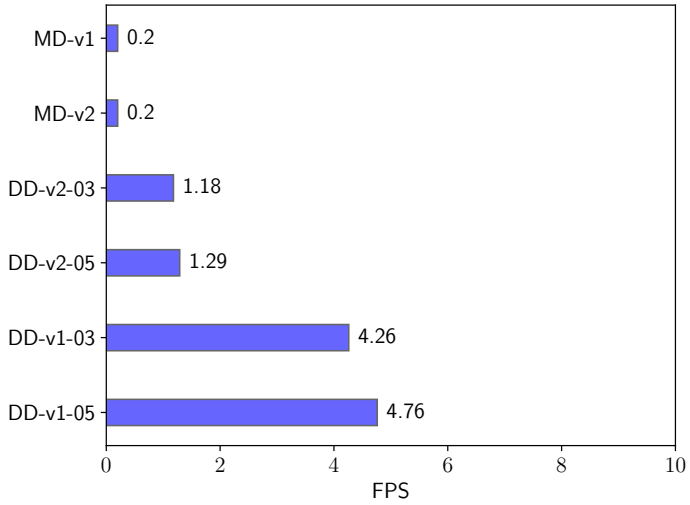
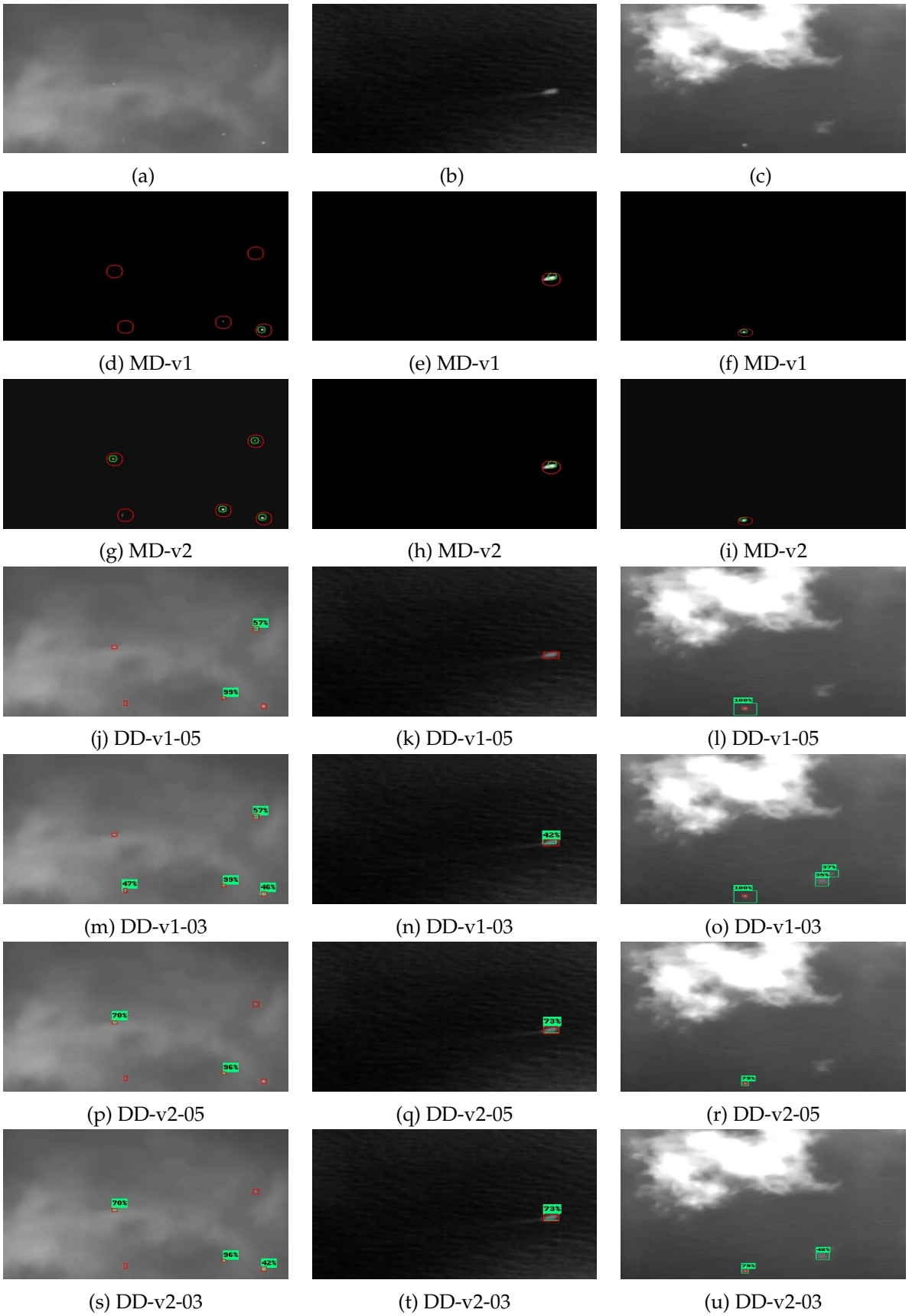


FIGURE 5.1: Final test results showing the average frames per second (FPS) of the proposed methods.

A set of predictions performed by the proposed methods are shown in Fig. 5.2. Further, Fig. 5.2(k) demonstrates that DD-v1-05 is the only method to miss the clearly visible target in Fig. 5.2(b). None of the methods are able to detect all five IR small targets in Fig. 5.2(a). However, four IR small targets were detected by MD-v2 and DD-v1-03, as shown in Fig. 5.2(g) and 5.2(m), respectively.



(a) (b) (c)

(d) MD-v1 (e) MD-v1 (f) MD-v1

(g) MD-v2 (h) MD-v2 (i) MD-v2

(j) DD-v1-05 (k) DD-v1-05 (l) DD-v1-05

(m) DD-v1-03 (n) DD-v1-03 (o) DD-v1-03

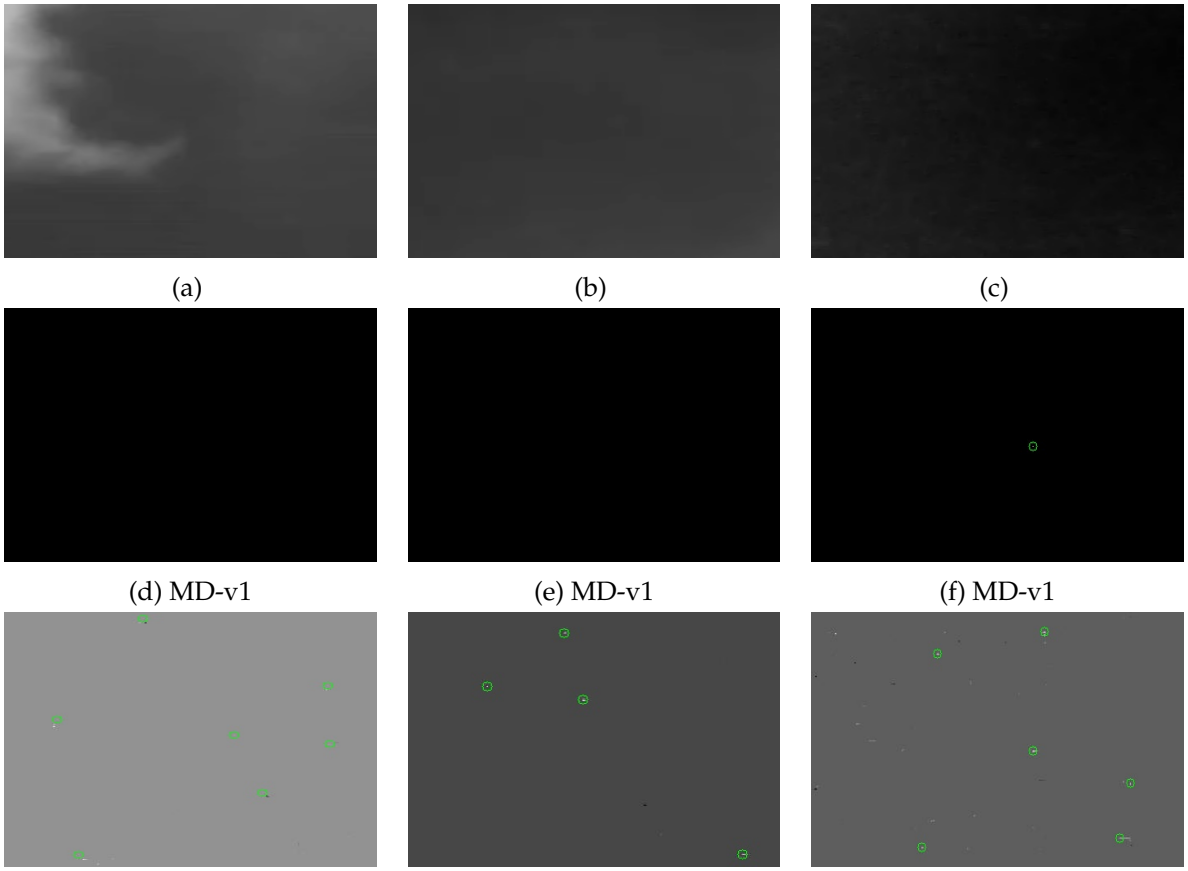
(p) DD-v2-05 (q) DD-v2-05 (r) DD-v2-05

(s) DD-v2-03 (t) DD-v2-03 (u) DD-v2-03

FIGURE 5.2: Predictions performed by proposed methods. Red boxes represent the ground truth targets. Green boxes represent predictions. (a) – (c) raw images from the dataset. (d) – (f) obtained using MD-v1. (g) – (i) obtained using MD-v2. (j) – (l) obtained using DD-v1-05. (m) – (o) obtained using DD-v1-03. (p) – (r) obtained using DD-v2-05. (s) – (u) obtained using DD-v2-03.

5.1.1 Analysis of the Model-driven Methods

As expected, and demonstrated in Table 5.1, the model-driven methods are inaccurate when compared to the data-driven methods. MD-v1 ($F_\beta = 0.604$, $MCC = 0.224$, precision $= 0.585$) outperforms MD-v2 ($F_\beta = 0.586$, $MCC = -0.024$, precision $= 0.345$) in terms of F_β, MCC and precision. In comparison to MD-v1, MD-v2 performs a meticulous decomposition, which may account for the low precision value, i.e., the large amount of false positive predictions. Compared to a single false positive prediction by MD-v1 as shown in Fig. 5.3(f), MD-v2 makes numerous false positive predictions in all images, as shown in Fig. 5.3(g), 5.3(h) and 5.3(i). The original images of Fig. 5.3 do not contain any IR small targets.



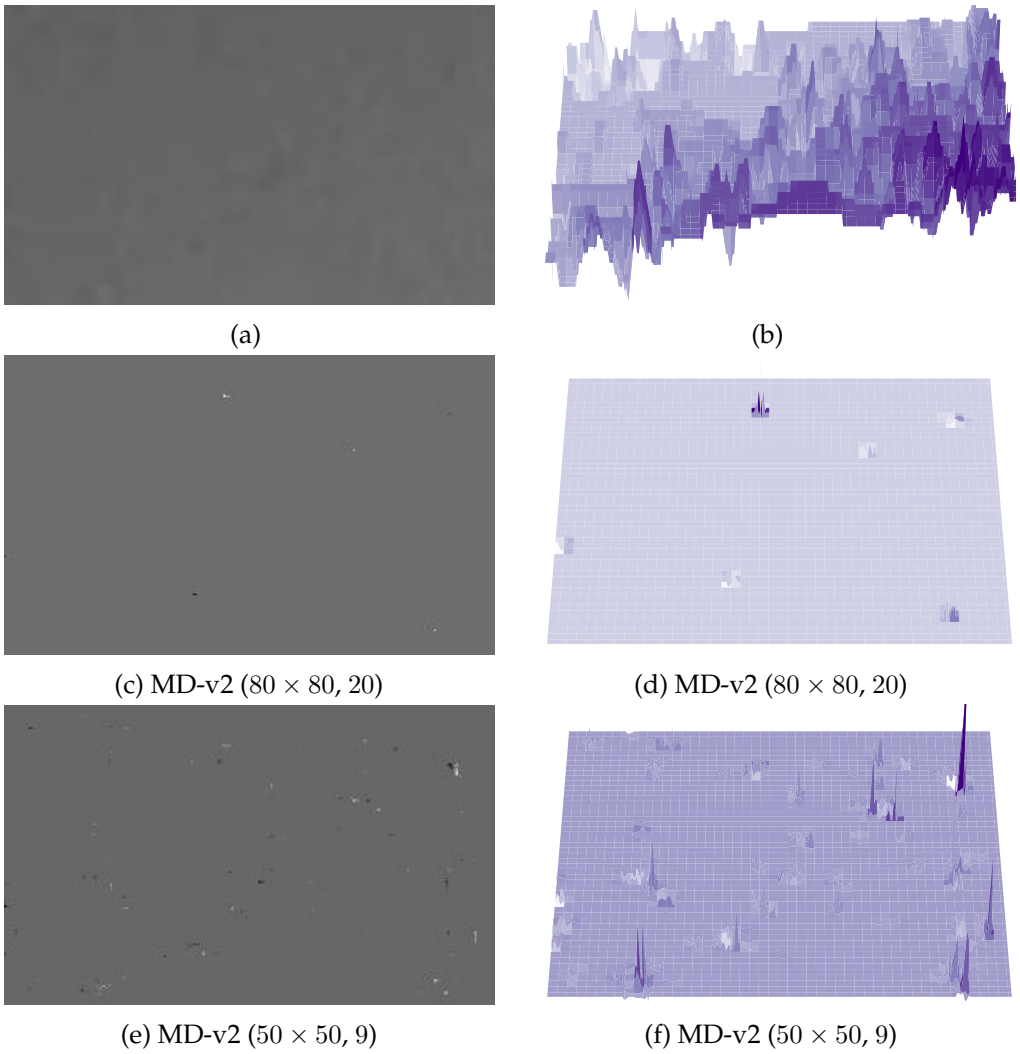
(a) (b) (c)

(d) MD-v1 (e) MD-v1 (f) MD-v1

(g) MD-v2 (h) MD-v2 (i) MD-v2

FIGURE 5.3: Predictions made by model-driven methods showing false positive predictions on true negative images. (a) – (c) negative images randomly selected from the dataset. (d) – (f) obtained using MD-v1. (g) – (i) obtained using MD-v2.

Using a patch size of 50 and a stride equal to 9 results in a significantly low precision, i.e., many false positive predictions, as illustrated in Fig. 5.4(f). The correct prediction for the original image shown in Fig. 5.4(a) should be true negative. Both the original MD-v2 (patch size $= 80$, stride $= 20$) shown in Fig. 5.4(c), and the MD-v2 with a patch size of 50 and stride of 9 shown in Fig. 5.4(e) make several false positive predictions. However, MD-v2 with a reduced patch size and stride extracts too much noise from the image. In addition, a low patch size and stride results in a longer processing time.



(a) (b)

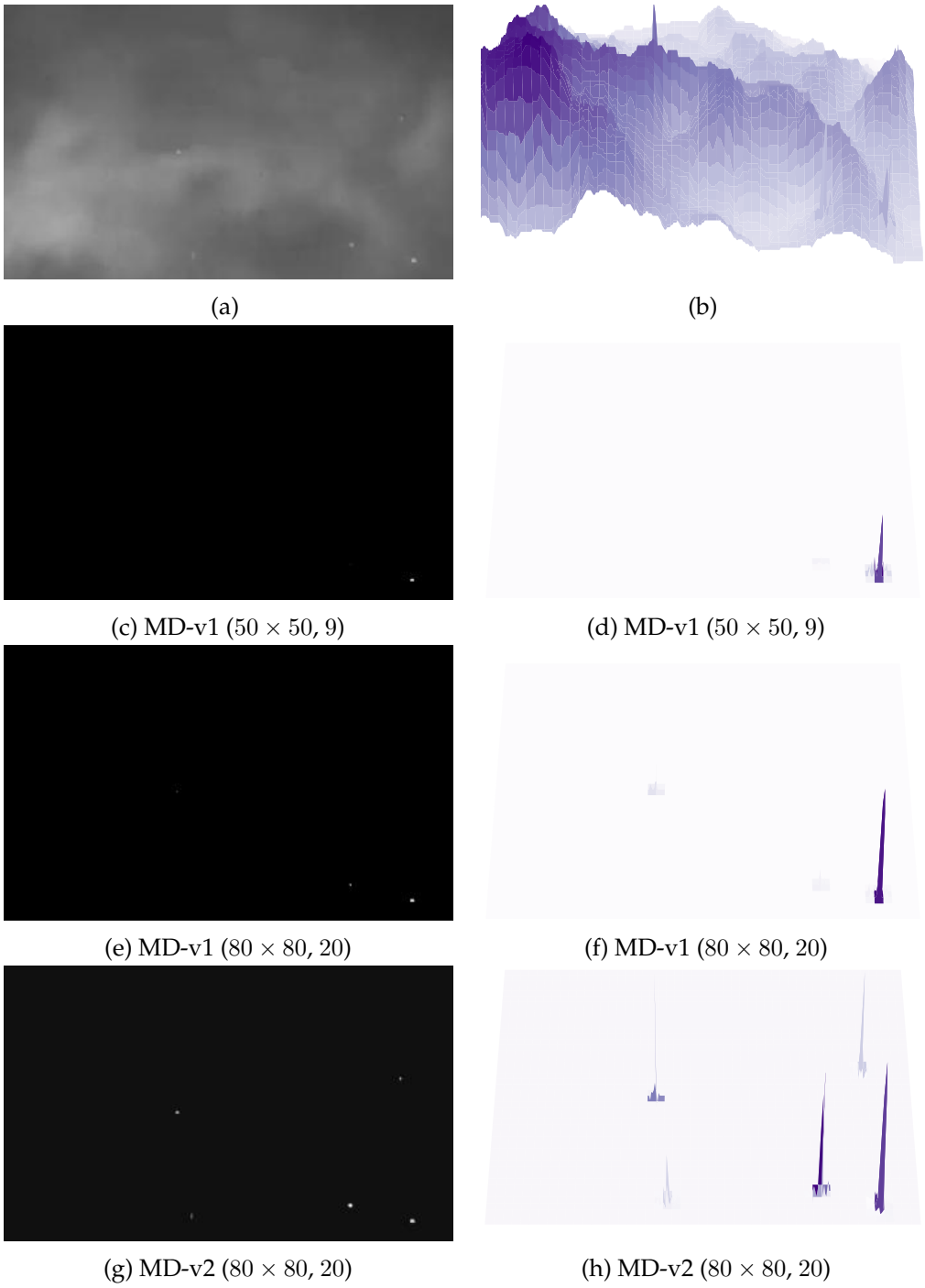
(c) MD-v2 (80×80, 20) (d) MD-v2 (80×80, 20)

(e) MD-v2 (50×50, 9) (f) MD-v2 (50×50, 9)

FIGURE 5.4: RPCA-PCP via IALM on negative images. Peaks in the surface plots correspond to brighter pixels. (a) – (b) negative image from the dataset. (c) – (d) obtained using original MD-v2. (e) – (f) obtained using MD-v2 with a patch size $= 50 \times 50$, and a stride $= 9$.

As previously stated, the original SIRST dataset is devoid of negative samples. The false positive rate of MD-v1 ($FPR = 0.386$) and MD-v2 ($FPR = 0.733$) indicate that the model-driven methods generate a large amount of false positive predictions on negative images. This could be the reason MD-v1 (nIoU $= 0.397$) and MD-v2 (nIoU $= 0.328$) achieve a lower nIoU than the original IPI model (nIoU $= 0.607$), which has not been tested on negative samples.

To conclude, the MD-v1 and MD-v2 cannot compete with the data-driven methods in terms of accuracy. In addition, MD-v1 and MD-v2 have an excessive computational time, with an average speed of 0.2 FPS. The lengthy computation time is primarily caused by the sliding window and singular value decomposition. Additional parameter adjustments, for example adjusting the tolerance and patch-size could have been investigated, but the lengthy computational time rendered the effort inefficient. The model-driven methods are however effective at identifying targets in complex environments, as shown in Fig. 5.5. Fig. 5.5(g) demonstrates how adept MD-v2 is at removing noise and extracting the five IR small targets.



(a) (b)

(c) MD-v1 (50×50, 9) (d) MD-v1 (50×50, 9)

(e) MD-v1 (80×80, 20) (f) MD-v1 (80×80, 20)

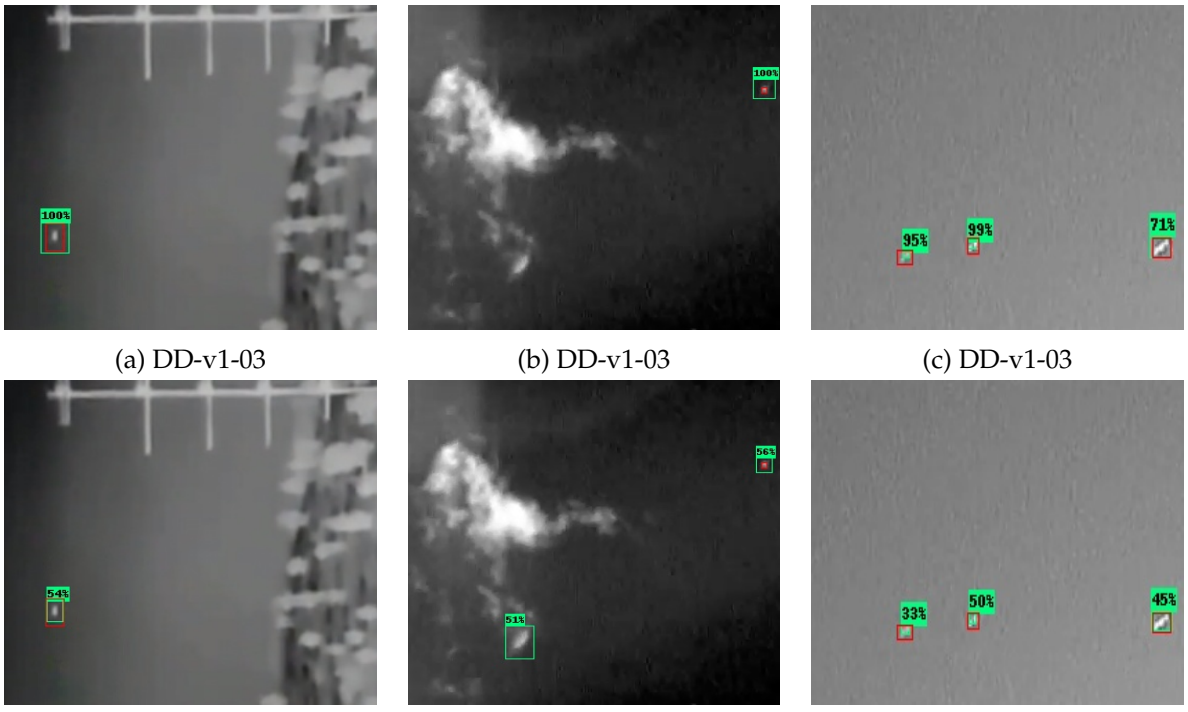
(g) MD-v2 (80×80, 20) (h) MD-v2 (80×80, 20)

FIGURE 5.5: Different model-driven methods. The peaks in the surface plots represent brighter pixel values. (a) – (b) positive image from the dataset. (c) – (d) obtained using MD-v1 (50×50, stride $= 9$). (e) – (f) obtained using MD-v1. (g) – (h) obtained using MD-v2.

5.1.2 Analysis of the Data-driven Methods

As illustrated in Table 5.1, all data-driven methods have a high precision and a low false positive rate, with the best scores going to DD-v2-05 (precision $= 0.990$, $FPR = 0.010$). A S_{th} of 0.3 results in a higher recall, MCC, F_β and nIoU. Note that a S_{th} less than 0.3 will introduce additional false positive predictions. The S_{th} equal to 0.5 discards some of the false predictions, however, it reduces not only the false positive predictions, but also true positive predictions. Further, a high S_{th} increases the amount of false negative predictions. This is not desirable, as the system should aim at detecting all potential targets.

DD-v1-03 ($F_\beta = 0.908$, $MCC = 0.817$, recall $= 0.904$) and DD-v2-03 ($F_\beta = 0.900$, $MCC = 0.842$, recall $= 0.885$) are the most accurate methods. As illustrated in Fig. 5.6(a), 5.6(d), 5.6(c) and 5.6(f), DD-v1-03 is more confident than DD-v2-03. Fig. 5.6(b) further demonstrates the performance of DD-v1-03 to that of DD-v2-03 shown in Fig. 5.6(e), where DD-v2-03 misidentifies a cloud as a target.

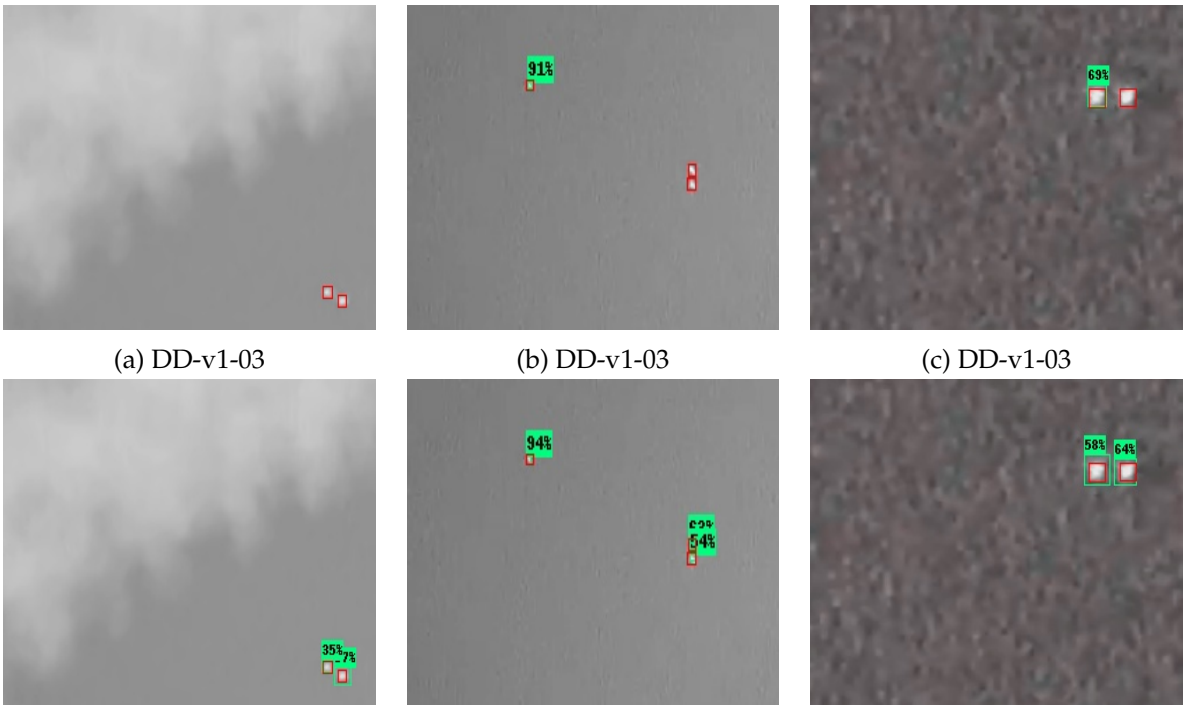


(a) DD-v1-03 (b) DD-v1-03 (c) DD-v1-03

(d) DD-v2-03 (e) DD-v2-03 (f) DD-v2-03

FIGURE 5.6: Performance of DD-v1-03 and DD-v2-03. (a) – (c) obtained using DD-v1-03. (d) – (f) obtained using DD-v2-03.

DD-v2-03 has a marginally lower F_β than DD-v1-03, but as illustrated in Fig. 5.7, DD-v2-03 clearly performs better than DD-v1-03 on the selected images. The targets vary in size from small, as shown in Fig. 5.7(b), to medium, as shown in Fig. 5.7(c). Also, Fig. 5.6(c), in which DD-v1-03 is able to detect all targets, depicts a scene similar to those shown in Fig. 5.7(a) and 5.7(b). However, the reasons for why DD-v2-03 outperforms DD-v1-03 in these images remain unclear.



(a) DD-v1-03 (b) DD-v1-03 (c) DD-v1-03

(d) DD-v2-03 (e) DD-v2-03 (f) DD-v2-03

FIGURE 5.7: Performance of DD-v1-03 and DD-v2-03. (a) – (c) obtained using DD-v1-03. (d) – (f) obtained using DD-v2-03.

Fig. 5.8 summarise the similarities between DD-v1-03 and DD-v2-03. DD-v2-03 has the highest MCC score, DD-v1-03 has the highest nIoU, recall, F_β, and it is faster. When considering the average time required to process a single image, it is clear that DD-v1-03 is the most appropriate approach for IR small target detection.

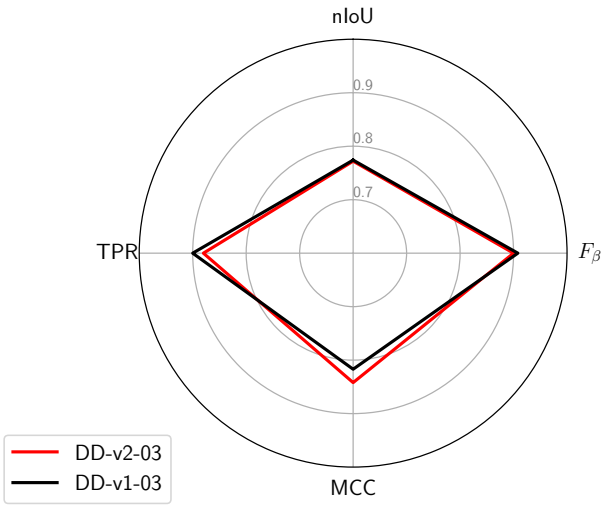


FIGURE 5.8: Comparison of DD-v1-03 and DD-v2-03 in terms of recall, MCC, nIoU, and F_β.

There is no statistically significant difference between the modified and original ResNet50 in terms of performance. This might be caused by the fact that the modified ResNet50 was not downsampled sufficiently, or alternatively, the original ResNet50

already had suitable feature map sizes. Further reduction of the downsampling operations results in a slower system. As shown in Table 5.3, the modified ResNet models (i.e., DD-v2-03 and DD-v2-05) run slower due to the increased parameter count caused by the reduced downsampling.

Further, the modified ResNet was trained with a batch size of 32, which is likely why the training process is slower than the training of the original ResNet. A low batch size should result in a model that generalises well to previously unobserved data. However, there are no significant differences between using a batch size of 64 or 32 in terms of accuracy.

DD-v1-03 operates at approximately 4 FPS. However, this is not comparable to running object detection on a continuous video stream. That is, the system might be able to process four image frames per second on average, but not achieve the same results on a live video stream as the CPU usage will become exceedingly high due to the need for decoding and processing of the incoming video in addition to running inference on the individual frames of the video stream. The FPS would be higher if the methods were deployed on a GPU-equipped machine.

The high accuracy of the data-driven methods could be the result of using the CenterNet meta-architecture with the ResNet feature extractor. CenterNet appears to perform well at the task of IR small target detection as it extracts peaks from keypoint heatmaps generated by ResNet. A surface plot of a feature map from the modified ResNet50 is shown in Fig. 5.9. The peak representing the IR small target is clearly visible.

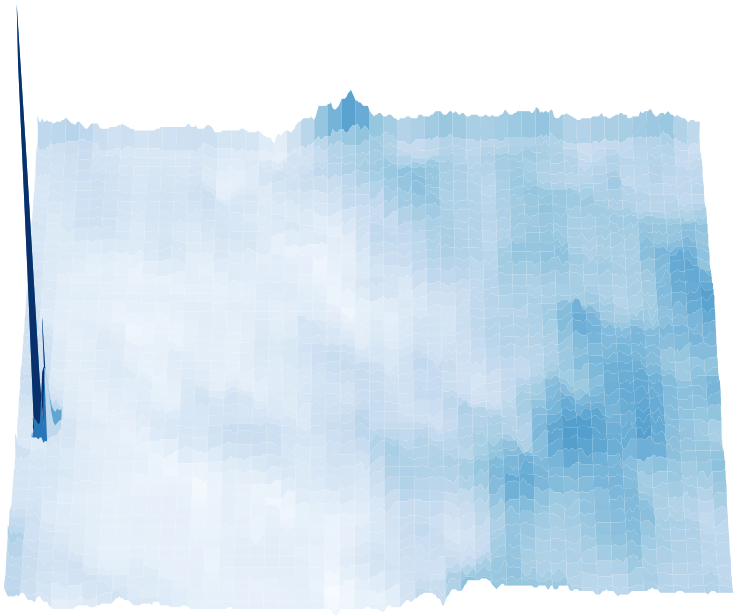


FIGURE 5.9: Surface plot of feature map from the modified ResNet50. The peak represents an IR small target.

5.2 General Discussion of the Proposed System

What should the system do when a target or multiple targets are detected? Further, how can the location of a target be determined? These questions have to be answered. Henceforth, it is assumed the location of a given target is known. The location of the target could be dispatched to the search and rescue team.

It is not recommended for the UAV to approach the predicted target, as this consumes a significant amount of battery power. The proposed system is likely best suited for the search of IR small targets at great heights above ground level.

Another challenge is to define how the system should behave in response to previously predicted targets. The system could register specific GPS positions. For example, the system notifies the ground crew that a target might be located at position $P(x, y, z)$. Then, the system ignores predicted targets that fall within a certain radius of P. Also, the system should ignore heat signatures coming from the ground crew.

5.2.1 Determining Size of Heat Signatures

The heat signature of, e.g., a boat could be $10\times$ larger than a human. It is therefore desired to translate the pixel size of a heat signature to its real-world size. If an object has a height $h_{obj_{\mathbf{D}}}$ in an image ($\mathbf{D}$) with the width and height ($w_{\mathbf{D}} \times h_{\mathbf{D}}$) as shown in Fig. 5.10(a). And the image was captured from a distance $d_{obj_{\mathbf{W}}}$ orthogonal to the object as shown in Fig. 5.10(b). Then, in order to determine the object's height ($h_{obj_{\mathbf{W}}}$) in a real-world frame ($\mathbf{W}$), the following relationship [19] is obtained:

$$h_{obj_{\mathbf{W}}} = \frac{d_{obj_{\mathbf{W}}} \times \left(\frac{h_s \times h_{obj_{\mathbf{D}}}}{h_{\mathbf{D}}}\right)}{F_L}, \tag{5.1}$$

where F_L is the camera's focal length, and h_s is the camera's sensor height. Both parameters are measured in millimetres. However, the size might not be correctly determined from this equation with different angles between the UAV and the heat signature. As the size of the object could become warped. In addition, the IR small target might belong to a small part of the actual object, i.e., the rest of the target is hidden behind a layer which blocks thermal radiation.

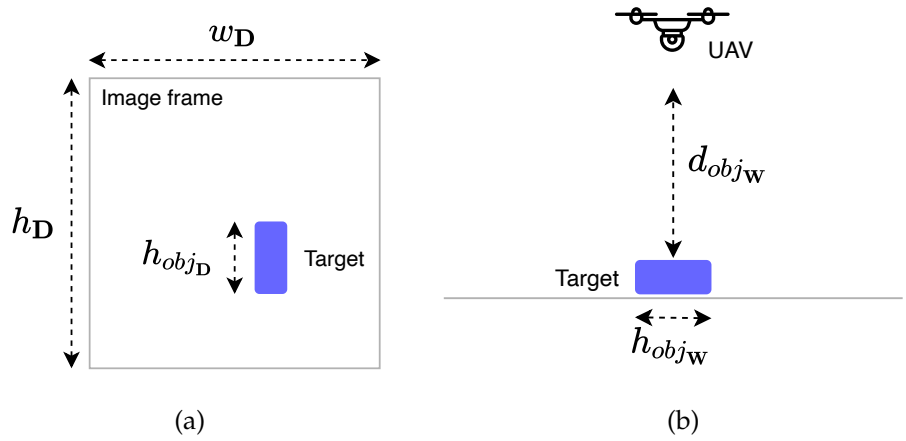


FIGURE 5.10: Computing the real-world size of a heat signature from an image. (a) Image frame. (b) World frame.

5.2.2 Computation Offloading

CNNs are great object detectors. However, their computational cost results in slow inference on single-board computers and other computationally constrained embedded devices. Due to obvious physical and power constraints, the system proposed in this thesis will have limited on-board computational power. To accelerate intensive tasks (e.g., deep learning inference), cloud computing can be used. This is referred to as computation offloading [26]. This requires the transfer of images or video from the UAV to the edge, and then to the cloud, as illustrated in Fig. 5.11. Then, the required computations are carried out in the cloud, and the results are relayed back to the UAV. This is a difficult task that likely results in unacceptable latency [7], i.e., the time interval between transferring data from the UAV to the cloud and receiving a response will be lengthy. Another issue arises if the number of UAVs connected to the edge increases, forcing the UAVs to share bandwidth, thus reducing the upload speed of data to the edge. However, the UAVs could be more selective about the type of data they upload. As an example, assume a UAV which only uploads images after entering a designated search area. Further, if the UAV remains in the same area, it will only upload a new image every other second. This could aid in the reduction of bandwidth usage and energy consumption.

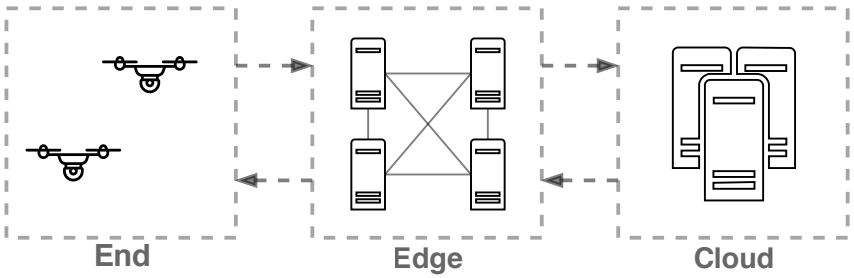


FIGURE 5.11: Simplified edge and cloud computing.

Edge computing could be used to address several of these issues. With edge computing, a collection of computing devices brings the capability to solve computationally intensive tasks closer to the UAVs, thereby reducing latency [7]. However, the computers located at the edge may be insufficient to perform real-time (e.g., more than 20 FPS) inference due to restricted memory and processing power. Devices with limited computational resources can benefit from specially designed CNN models that prioritise speed over accuracy. Some of the methods for optimising and compressing CNN models listed by Chen and Ran [7] are:

- **Parameter quantisation**, which converts floating-point numbers to integers with a small bit width, thereby avoiding slow computations. This increases the speed of each computation but decreases the precision and accuracy of the model.
- **Parameter pruning**, which eliminates superfluous parameters (e.g., parameters with values close to 0).
- And **knowledge distillation**, which is a technique where a small network imitates a powerful one. This is accomplished by training a small network on the powerful network's predictions.

Further, hardware such as GPUs, FPGAs and ASICs can help accelerate the deep learning inference compared to using CPUs. It is recommended [60] to employ an FPGA (field-programmable gate array) rather than a GPU when running deep learning inference at the edge. Compared to GPUs and CPUs, FPGAs are more energy efficient and *easier* to reconfigure [7]. However, deep learning libraries for FPGAs are not as prevalent or optimised as they are for GPUs. Moreover, it is possible to optimise the computational time by implementing parts of the software in C or C++ instead of Python.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis investigates the detection of IR small targets. The use of autonomous UAVs, IR imaging, and object detection to assist search and rescue missions has tremendous potential. A literature review was conducted with a focus on IR imaging, search and rescue with UAVs, general object detection, deep learning-based object detection, and IR small target detection. The literature review led to the discovery of two distinct methodologies, a model-driven approach based on low-rank and sparse matrix decomposition which employs RPCA-PCP via IALM, and a deep learning-based data-driven approach using CenterNet with ResNet. The primary limitation of this thesis was the absence of a substantial dataset. Test results indicate that the proposed system is effective at detecting IR small targets. As expected, the data-driven approach outperformed the model-driven approach. Although accurate, the data-driven methods are slow. Training the data-driven methods on additional IR small target samples will further improve their accuracy.

The IR small target detection system's detailed mechanism remains incomplete, as much work remains to be done. A special emphasis should be placed on the validation of the proposed system in real-world scenarios. The results of this thesis establish unequivocally that CNN-based object detection methods are highly accurate at IR small target detection. Conclusively, the proposed system has the potential to make a significant impact by assisting search and rescue missions.

6.2 Future Work

Several areas are worth investigating further. The proposed system should be thoroughly tested on images and video captured by a long-wave IR imaging camera at various heights above ground level. Further, research on local processing vs. edge processing and cloud computing should be conducted. When the system requires an instantaneous response, local processing is preferred, allowing the search to be performed in remote areas where cellular network coverage is limited. Edge computing and computation offloading is preferred when the UAV can connect to a high-speed network. Further, one should perform an investigation into how the system should operate if the UAV connection is lost.

Other critical areas to investigate include determining how to obtain the location of the detected IR small target and deciding how to proceed when a target or multiple targets are detected. The UAV's height above ground level must be measured as this

distance is required for the IR small target size computation in Equation (5.1). Possibly, a radar altimeter or a laser rangefinder can measure this distance.

Meta-architectures and feature extractors will continue to improve in performance. Adding a colour imaging camera and external lighting to the system could aid in target classification. The detection of humans and objects associated with human activity at a lower altitude will further help in narrowing down the location of a missing person. In addition, methods for target tracking should be investigated as they could further improve the system's ability to localise missing victims and allow the system to focus on a single target if needed.

Search and rescue with airborne optical sectioning is an intriguing approach to detecting victims hidden beneath a forest canopy. Targets on the ground are nearly invisible in some cases because their heat signatures are heavily obscured by trees. In these extremely complex scenarios, simply thresholding the heat signature will not help detect the IR small target. Schedl et al. [44] propose a synthetic aperture imaging technique to solve this problem. Images from colour and thermal imaging cameras recorded above a forest structure are registered and integrated to remove obstructions such as trees and other foliage. This method improves the visibility of heat signatures coming from hidden targets by combining images captured from a variety of different viewpoints. The resulting images gives a clear view of the ground hidden beneath the treetops.

A valuable addition to the proposed system would be a method that enables the detection of targets covered by a radiant barrier, i.e., a material which blocks thermal radiation. This could perhaps be a sub-system based on ground-penetrating radar.

Appendix A

Inexact Augmented Lagrange Multiplier (IALM)

The following Python implementation of the RPCA-PCP via IALM method is based on [30] and [47].

```
"""
    Inexact augmented Lagrange multiplier (IALM)
"""

import numpy as np
from numpy import linalg
from md_utils import shrinking


def jay_func(y_mat, lambd):
    """
    implements
        J(D) = max(norm_{2}(D), lambda^(-1)*norm_{inf}(D))
    """
    return max(linalg.norm(y_mat, 2), np.dot(np.reciprocal(lambd),
    linalg.norm(y_mat, np.inf)))


def rpca_ialm(data_mat, lmbda, max_iter, tol):
    """
    Required input:
        D       - (m x n) data matrix
        lambda  - weight of sparse error

    Adjustable parameters:
        tol         - tolerance for stopping criterion (DEFAULT=1e-2)
        max_iter    - maximum number of iterations (DEFAULT=1000)

    Return:
        s_hat - estimate of S
    """

    d_norm  = linalg.norm(data_mat)
    l_k     = np.zeros(data_mat.shape)
    s_k     = np.zeros(data_mat.shape)
    y_k     = data_mat/jay_func(data_mat, lmbda)
    mu_k    = 1.25/linalg.norm(data_mat, 2)
    mu_bar  = mu_k*1e7
    rho     = 1.6

```

```
    # Solving RPCA-PCP via IALM
    converged = k = 0
    while converged == 0:
        U, sigm, v = linalg.svd(data_mat-s_k+np.reciprocal(mu_k)*y_k,
                                full_matrices=False) # economy SVD
        sigm  = np.diag(sigm)
        l_kp1 = np.dot(U, shrinking(sigm, np.reciprocal(mu_k)))
        l_kp1 = np.dot(l_kp1, v)
        shr   = data_mat-l_kp1+np.dot(np.reciprocal(mu_k), y_k)
        s_kp1 = shrinking(shr, lmbda*np.reciprocal(mu_k))
        mu_k  = min(mu_k*rho, mu_bar)
        k     = k+1
        l_k = l_kp1
        s_k = s_kp1

        stop_criterion = linalg.norm(data_mat-l_k-s_k, 'fro')/d_norm
        if (converged == 0 and k >= max_iter) or stop_criterion < tol:
            converged = 1
    s_hat = s_k
    return s_hat
```

Appendix B

Accelerated Proximal Gradient (APG)

The following Python implementation of RPCA-PCP via accelerated proximal gradient is based on [20] and [30].

```
"""
    Accelerated Proximal Gradient (APG)
"""

from math import sqrt
import numpy as np
from numpy import linalg
from md_utils import shrinking


def rpca_apg(data_mat, lmbda, max_iter, tol):
    """
    Required input:
        D       - (m x n) data matrix
        lambda  - weight of sparse error

    Adjustable parameters:
        tol         - tolerance for stopping criterion (DEFAULT=1e-6)
        max_iter    - maximum number of iterations (DEFAULT=1000)

    Return:
        s_hat - estimate of S
    """
    U_i, sigm_i, v_i = linalg.svd(data_mat, full_matrices=False)
    l_k = l_m1 = np.zeros(data_mat.shape)
    s_k = s_m1 = np.zeros(data_mat.shape)
    t_k = t_m1 = 1

    mu_k    = sigm_i[1]
    mu_bar  = 0.05*sigm_i[3]
    eta     = 0.99

    # Solving RPCA-PCP via APG
    converged = k = 0
    while converged == 0:
        y_k_l = l_k + ((t_m1 - 1)/t_k)*(l_k - l_m1)
        y_k_s = s_k + ((t_m1 - 1)/t_k)*(s_k - s_m1)
        g_k_l = y_k_l - (1/2) * (y_k_l + y_k_s - data_mat)
        U, sigm, v = linalg.svd(g_k_l, full_matrices=False)
        sigm  = np.diag(sigm)
        l_kp1 = np.dot(U, shrinking(sigm, mu_k/2))
        l_kp1 = np.dot(l_kp1, v)
        g_k_s = y_k_s-(1/2)*(y_k_l + y_k_s - data_mat)
```

```
44          g_k_s = np.squeeze(np.asarray(g_k_s))
45          s_kp1 = shrinking(g_k_s, lmbda*mu_k/2)
46          t_kp1 = 0.5*(1 + sqrt(1 + 4*t_k*t_k))
47          temp  = l_kp1+s_kp1-y_k_l-y_k_s
48          s_kp1_l = 2*(y_k_l-l_kp1)+temp
49          s_kp1_s = 2*(y_k_s-s_kp1)+temp
50          mu_k = max(mu_k*eta, mu_bar)
51          k = k+1
52          t_m1 = t_k
53          t_k  = t_kp1
54          l_m1 = l_k
55          s_m1 = s_k
56          l_k  = l_kp1
57          s_k  = s_kp1
58          stopping_criterion = linalg.norm([s_kp1_l, s_kp1_s])/(2*max(1,
    linalg.norm([l_kp1, s_kp1])))
59          if  (stopping_criterion <= tol) or \
60              (converged == 0 and k >= max_iter):
61              converged = 1
62      s_hat = s_k
63      return s_hat
```

Bibliography

[1] R.D. Arnold, H. Yamaguchi, and T. Tanaka. "Search and rescue with autonomous flying robots through behavior-based cooperative intelligence". In: *Int J Humanitarian Action* 3.18 (2018). DOI: 10.1186/s41018-018-0045-4.

[2] Mesay Bejiga et al. "A Convolutional Neural Network Approach for Assisting Avalanche Search and Rescue Operations with UAV Imagery". In: *Remote Sensing* 9 (2017), p. 100. DOI: 10.3390/rs9020100.

[3] Thierry Bouwmans et al. "Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset". In: *Computer Science Review* 23 (Feb. 2017), 1–71. ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2016.11.001.

[4] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. DOI: 10.1017/9781108380690.

[5] Emmanuel J. Candès et al. *Robust Principal Component Analysis?* 2009. arXiv: 0912.3599.

[6] G. Chen et al. "A Survey of the Four Pillars for Small Object Detection: Multiscale Representation, Contextual Information, Super-Resolution, and Region Proposal". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2020), pp. 1–18. ISSN: 2168-2232. DOI: 10.1109/TSMC.2020.3005231.

[7] Jiasi Chen and Xukan Ran. "Deep Learning With Edge Computing: A Review". In: *Proceedings of the IEEE* 107.8 (2019), pp. 1655–1674. DOI: 10.1109/JPROC.2019.2921977.

[8] Davide Chicco and Giuseppe Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genomics* 21.1 (2020), p. 6. ISSN: 1471-2164. DOI: 10.1186/s12864-019-6413-7.

[9] *COCO - Common Objects in Context*. Feb. 2021. URL: https://cocodataset.org/.

[10] The SciPy community. *numpy.linalg.svd*. Jan. 2021. URL: https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html.

[11] Yimian Dai. *YimianDai/sirst*. 2021. URL: https://github.com/YimianDai/sirst.

[12] Yimian Dai et al. "Asymmetric Contextual Modulation for Infrared Small Target Detection". In: *IEEE Winter Conference on Applications of Computer Vision, WACV 2021*. 2021.

[13] Yimian Dai et al. *Attentional Local Contrast Networks for Infrared Small Target Detection*. 2020. arXiv: 2012.08573 [cs.CV].

[14] Navneet Dalal and Bill Triggs. "Histograms of Oriented Gradients for Human Detection". In: *International Conference on Computer Vision & Pattern Recognition (CVPR '05)*. Vol. 1. IEEE Computer Society, June 2005, pp. 886–893. DOI: 10.1109/CVPR.2005.177.

[15] DJI. *DJI Counts More Than 500 People Rescued By Drones Around The World*. Dec. 2020. URL: https://www.dji.com/newsroom/news/dji-counts-more-than-500-people-rescued-by-drones-around-the-world.

[16] Vincent Dumoulin and Francesco Visin. *A guide to convolution arithmetic for deep learning*. 2018. arXiv: 1603.07285 [stat.ML].

[17] Houzhang Fang et al. "Infrared Small Target Detection with Total Variation and Reweighted ℓ_1 Regularization". In: *Mathematical Problems in Engineering* 2020 (Jan. 2020), pp. 1–19. DOI: 10.1155/2020/1529704.

[18] Jan Frantzen. *Savnet eldre kvinne funnet av dronepilot: Nå redder droner liv i Norge*. Apr. 2020. URL: https://www.uasnorway.no/savnet-funnet-av-dronepilot-na-redder-droner-liv-ogsa-i-norge.

[19] Wayne Fulton. *Calculate Distance or Size of an Object in a photo image*. URL: https://www.scantips.com/lights/subjectdistance.html.

[20] Chenqiang Gao et al. "Infrared Patch-Image Model for Small Target Detection in a Single Image". In: *Image Processing, IEEE Transactions on* 22.12 (2013), pp. 4996–5009.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[22] Austen Groener, Gary Chern, and Mark Pritt. "A Comparison of Deep Learning Object Detection Models for Satellite Imagery". In: *2019 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)* (Oct. 2019). DOI: 10.1109/aipr47015.2019.9174593.

[23] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[24] P. Iob et al. "Avalanche Rescue with Autonomous Drones". In: *2020 IEEE 7th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. 2020, pp. 319–324. DOI: 10.1109/MetroAeroSpace48742.2020.9160116.

[25] Reinhard Klette. *Concise computer vision: An Introduction into Theory and Algorithms*. Springer-Verlag London, 2014. DOI: 10.1007/978-1-4471-6320-6.

[26] Anis Koubaa et al. "DeepBrain: Experimental Evaluation of Cloud-Based Computation Offloading and Edge Computing in the Internet-of-Drones for Deep Learning Applications". In: *Sensors* 20.18 (2020). ISSN: 1424-8220. DOI: 10.3390/s20185240.

[27] C. Kyrkou and T. Theocharides. "EmergencyNet: Efficient Aerial Image Classification for Drone-Based Emergency Monitoring Using Atrous Convolutional Feature Fusion". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13 (2020), pp. 1687–1699. DOI: 10.1109/JSTARS.2020.2969809.

[28] Tsung-Yi Lin et al. *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV].

[29] Tzuta Lin. *tzutalin/labelImg*. 2021. URL: https://github.com/tzutalin/labelImg.

[30] Zhouchen Lin, Minming Chen, and Yi Ma. *The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices*. 2010. arXiv: 1009.5055.

[31] Hong-Kang Liu, Lei Zhang, and Hua Huang. "Small Target Detection in Infrared Videos Based on Spatio-Temporal Tensor Model". In: *IEEE Transactions on Geoscience and Remote Sensing* 58.12 (2020), pp. 8689–8700. DOI: 10.1109/TGRS.2020.2989825.

[32] Wei Liu et al. *SSD: Single Shot MultiBox Detector*. 2015. arXiv: 1512.02325.

[33] D. G. Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, pp. 1150–1157. DOI: 10.1109/ICCV.1999.790410.

[34] Eleftherios Lygouras et al. "Unsupervised Human Detection with an Embedded Vision System on a Fully Autonomous UAV for Search and Rescue Operations". In: *Sensors* 19.16 (2019). ISSN: 1424-8220. DOI: 10.3390/s19163542.

[35] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[36] Nhat-Duy Nguyen et al. "An Evaluation of Deep Learning Methods for Small Object Detection". In: *Journal of Electrical and Computer Engineering* 2020 (Apr. 2020), pp. 1–18. DOI: 10.1155/2020/3189691.

[37] Bo Pang, Erik Nijkamp, and Ying Nian Wu. "Deep Learning With TensorFlow: A Review". In: *Journal of Educational and Behavioral Statistics* 45.2 (2020), pp. 227–248. DOI: 10.3102/1076998619872761.

[38] Paperspace. *Instance Types*. 2021. URL: https://docs.paperspace.com/gradient/instances/instance-types.

[39] Maria Gaia Pensieri, Mauro Garau, and Pier Matteo Barone. "Drones as an Integral Part of Remote Sensing Technologies to Help Missing People". In: *Drones* 4.2 (2020). ISSN: 2504-446X. DOI: 10.3390/drones4020015.

[40] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[41] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[42] Stamatios Samaras et al. "Deep Learning on Multi Sensor Data for Counter UAV Applications—A Systematic Review". In: *Sensors* 19.22 (2019). ISSN: 1424-8220. DOI: 10.3390/s19224837.

[43] Yutaka Sasaki. *The truth of the F-measure*. Oct. 2007. URL: https://www.researchgate.net/publication/268185911_The_truth_of_the_F-measure.

[44] David C. Schedl, Indrajit Kurmi, and Oliver Bimber. "Search and rescue with airborne optical sectioning". In: *Nature Machine Intelligence* 2.12 (Nov. 2020), 783–790. ISSN: 2522-5839. DOI: 10.1038/s42256-020-00261-3.

[45] Caroline Silva, Thierry Bouwmans, and Carl Frélicot. "An eXtended Center-Symmetric Local Binary Pattern for Background Modeling and Subtraction in Videos". In: *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISAPP 2015*. Berlin, Germany, Mar. 2015. DOI: 10.5220/0005266303950402.

[46] Mario Silvagni et al. "Multipurpose UAV for search and rescue operations in mountain avalanche events". In: *Geomatics, Natural Hazards and Risk* 8.1 (2017), pp. 18–33. DOI: 10.1080/19475705.2016.1238852.

[47] Qiang Siwei. *qiangsiwei/tensor_tools/algorithms/rpca/IALM/inexact_alm_rpca.m*. Dec. 2015. URL: https://github.com/qiangsiwei/tensor_tools/blob/master/algorithms/rpca/IALM/inexact_alm_rpca.m.

[48] Jingxuan Sun et al. "A Camera-Based Target Detection and Positioning UAV System for Search and Rescue (SAR) Purposes". In: *Sensors* 16.11 (2016). ISSN: 1424-8220. DOI: 10.3390/s16111778.

[49] TensorFlow. *TensorFlow 2 Detection Model Zoo*. Sept. 2020. URL: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md.

[50] TensorFlow. *TensorFlow Core v2.4.1 API for Python - Module: tf.keras.activations*. Mar. 2021. URL: https://www.tensorflow.org/api_docs/python/tf/keras/activations.

[51] TensorFlow. *TensorFlow Core v2.4.1 API for Python - tf.keras.layers.Dense*. Mar. 2021. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense.

[52] TensorFlow. *TensorFlow Object Detection API*. May 2021. URL: https://github.com/tensorflow/models/tree/master/research/object_detection.

[53] Tensorflow. *tensorflow/resnet.py*. Mar. 2021. URL: https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/keras/applications/resnet.py.

[54] TensorFlow. *TFRecord and tf.train.Example - TensorFlow Core*. Apr. 2021. URL: https://www.tensorflow.org/tutorials/load_data/tfrecord.

[55] Yulun Tian et al. "Search and rescue under the forest canopy using multiple UAVs". In: *The International journal of robotics research* 39.10-11 (2020), pp. 1201–1221. ISSN: 0278-3649.

[56] Paul Viola and Michael Jones. *Rapid object detection using a boosted cascade of simple features*. 2001.

[57] M. Vollmer and K.P. Möllmann. *Infrared Thermal Imaging: Fundamentals, Research and Applications*. Wiley, Feb. 2018. ISBN: 978-3-527-41351-5.

[58] Huan Wang, Manshu Shi, and Hong Li. "Infrared dim and small target detection based on two-stage U-skip context aggregation network with a missed-detection-and-false-alarm combination loss". In: *Multimedia Tools and Applications* 79.47 (2020), pp. 35383–35404. ISSN: 1573-7721. DOI: 10.1007/s11042-019-7643-z.

[59] K. Wang et al. "Detection of Infrared Small Targets Using Feature Fusion Convolutional Network". In: *IEEE Access* 7 (2019), pp. 146081–146092. DOI: 10.1109/ACCESS.2019.2944661.

[60] Xiaofei Wang et al. "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey". In: *IEEE Communications Surveys & Tutorials* 22.2 (2020), 869–904. ISSN: 2373-745X. DOI: 10.1109/comst.2020.2970550.

[61] Youzi Xiao et al. "A review of object detection based on deep learning". In: *Multimedia Tools and Applications* 79.33 (Sept. 2020), pp. 23729–23791. ISSN: 1573-7721. DOI: 10.1007/s11042-020-08976-6.

[62] Yunyang Xiong et al. *MobileDets: Searching for Object Detection Architectures for Mobile Accelerators*. 2020. arXiv: 2004.14525 [cs.CV].

[63] Huaizhong Zhang et al. "A novel infrared video surveillance system using deep learning based techniques". In: *Multimedia tools and applications* 77.20 (2018), pp. 26657–26676. ISSN: 1573-7721.

[64] Zhong-Qiu Zhao et al. *Object Detection with Deep Learning: A Review*. 2019. arXiv: 1807.05511 [cs.CV].

[65] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. *Objects as Points*. 2019. arXiv: 1904.07850 [cs.CV].

[66] Haidi Zhu et al. *A Review of Video Object Detection: Datasets, Metrics and Methods*. 2020. DOI: 10.3390/app10217834.

[67] Zhengxia Zou et al. *Object Detection in 20 Years: A Survey*. 2019. arXiv: 1905.05055.