FEATURE MATRIX-BASED MULTI-SCALE INTRINSIC MOTION SEGMENTATION

TRACKING DISMOUNTS WITH FEATURE MATRIX-BASED MULTI-SCALE INTRINSIC MOTION SEGMENTATION FRAMEWORK (FM-MIMS)

BY

PRAKHAR GARG, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2021) McMaster University
(Electrical and Computer Engineering) Hamilton, Ontario, Canada

TITLE:	Tracking Dismounts With Feature Matrix-Based Multi-scale Intrinsic Motion Segmentation Framework (FM-MIMS)
AUTHOR:	Prakhar Garg B.Eng. (Mechatronics Engineering & Management, co-op), McMaster University, Hamilton, Canada
SUPERVISOR:	Dr. Kirubarajan, Dr. Tharmarasa
NUMBER OF PAGES:	xviii, 88

Lay Abstract

The detection of dismounts has a variety of applications, such as tracking remotely for battlefield awareness or at checkpoints. They are however difficult to detect from aircraft because of their small featureless profile on Wide Angle Motion Imagery (WAMI). In such situations, conventional approaches of feature detection can prove ineffective. The Multi-scale Intrinsic Model Structure (MIMS) does an adequate job identifying dismounts in such situations. This thesis proposes a Feature Matrix Based Multi-scale Intrinsic Motion Segmentation (FM-MIMS) algorithm that would use partially identifiable features to make better-informed detections.

Abstract

Tracking dismounts using wide-area motion imagery (WAMI) from the air is a challenging problem. Such an algorithm has many applications on the battlefield, search and rescue, law enforcement, and more. This task is however nontrivial. Small target sizes and difficult to identify features make it extremely challenging to reliably detect people on the ground. Based on the Multi-scale Intrinsic Motion Structure framework (MIMS), this algorithm proposes an extension to the MIMS framework to allow simple feature identifiers to help improve the rate of successfully identifying a dismount. The Feature Matrix Based Multi-scale Intrinsic Motion Segmentation (FM-MIMS) does so by encoding size and motion features from a small target detector, and an optical flow detector. The features are used to drive the predictions made by a tensor voting algorithm that is then segmented to identify the target. Given the computationally intensive nature of tensor voting, this thesis proposes incremental changes to it as well as a layer of image prepossessing to allow for a robust method to detect dismounts. Feature extraction would allow the MIMS framework to extend its 4-9pixel target size to better detect larger targets in the 15-20pixel size range. At this scale dismounts are still very limited in identifiable features however, appendages such as arms and legs are visible. FM-MIMS intends to extract as much data as possible to increase the reliability of the detector.

To my parents

For always having faith in me

Acknowledgements

Thank you Dr. Kiruba for being my supervisor and guiding me through my masters. Without you, this thesis would not have reached completion.

Thank you Dr. Tharmarasa for teaching me everything I needed to know to get this thesis started.

Thank you Mike Bradford for your continued support and direction in shaping this thesis.

Thank you Cheryl Gies for having my back when I needed it the most.

Contents

List of Figures

List of Tables

Notation, Definitions, and Abbreviations

Notation

$\vec{A}$	A is a vector
$\in$	Element of
$\{\}$	a set

Definitions

Dismount A dismount is a human.

Low-observable targets

low-observable targets are targets for which the sensor responses have a value of SNR lower than 10dB.

Measurement

All observed quantities included

Target	Anything whose state is of interest to us. A person, object, or place selected as the aim of an attack.
Tensor	A mathematical object analogous to but more general than a vector, represented by an array of components that are functions of the coordinates of their space.
Track	A sequence of measurements that have been decided or hypothesized by the tracker to come from a single source.
Clutter	Non-persistent measurements which are not originated
Probability of detection	The ratio of detected aims to the number of all possible blips on the radar screen.
Probability of false alarm	Probability of false alarm or false alarm rate is the ratio of the number of false alarms to the number of non-hazardous objects.
Probability of missed detection	The ratio of targets not detected to the number of all possible blips on the radar screen. from a target. in a (possibly processed) report output from a sensor.

Abbreviations

AI	Artificial intelligence

DTI	Detection tracking and identification
FM-MIMS	Feature Matrix Based Multi-scale Intrinsic Motion
FMV	Full motion video
IFOV	Instantaneous Field Of View
IR	Infrared
MIMS	Multi-scale Intrinsic Motion Structure
ML	Machine Learning
PDE	Partial Differential Equation
PD	Probability of detection
PFA	Probability of false alarm
PMD	Probability of missed detection
SA	Situational Awareness
SNR	Signal to noise ratio
SWIR	Short wave Infrared
TBD	Track Before Detect
TBD	Track Before Detect
TV	Tensor Vote
VF	2D fundamental stick voting field

WAMI	Wide-Area Motion Imagery Structure

Chapter 1

Introduction

1.1 Motivation

Situational awareness is the perception of environmental elements and events concerning time and space. Greater situational awareness can drastically improve outcomes in a combat environment. As combat capabilities evolve, the need for gathering a greater volume of data from a distance grows. Even with modern electro-optic capabilities, it can be difficult to accurately identify dismounts and small targets from a distance. "Dismount tracking is the concept of tracking a person either by direct observation or indirectly by inference, such as determining where the person was when exiting direct view (i.e. in a car, building, or dwelling)." (Blasch *et al.*, 2012) This concept is important for security as tracking information can be used to establish intent and group behavioral patterns to better assess security risks in the environment.

Humanity can often spot subtle queues to identify a target. Detecting dismounts through code, however, is a difficult problem. The flexible nature of humans can allow them to be in various positions and cause variations in their observed shape. Humans also come in various shapes and sizes; some may not possess all the identifiable characteristics of an average person. For instance, some may be significantly larger than others, which may result in an image where portions of their body are difficult to observe and identify. Furthermore, a large variety of clothing and changes in the background add another dimension of complexity. Such an immense number of variations across multiple dimensions poses a non-trivial problem.

The following document will explore existing technologies and attempt to improve on them to allow for greater detection accuracy and reliability by depending on known environmental conditions. Furthermore, a more informed detection will be made by relying on a wider range of data.

The proposed FM-MIMS algorithm relies on a Multi-scale Intrinsic Motion Structures model proposed by Zhu *et al.* (2014) and extent it further. First, the input data will be prepossessed to extract relevant features. Next, the Feature matrices will be used in the tensor voting framework to find common relationships between the data. Tensor voting can be a computationally intensive task, depending on the size of data used. To relieve this problem the proposed algorithm includes some approximations such as a technique to find the optimal threshold values and, the use of Monte Carlo integration. Lastly, an optimal classifier for the algorithm is chosen.

1.2 The State Of The Art And Limitations

Various approaches have been used to identify and track dismounts in a scene. A diverse set of approaches have been studied and incrementally improved. Traditionally the focus has been on attempting to locate identifiable features and motion queues. Modern advancements in Machine Learning (ML) however, have attempted to use a more data-driven approach. As the need for gathering larger amounts of data increases, Wide-Angle Motion Imaging (WAMI) is often used. These allow for the surveillance of larger areas but, targets of interest are significantly smaller with less identifiable features. Some hybrid approaches designed for detecting low Signal to Noise Ratio (SNR) targets, low-observable targets, can be beneficial in such scenarios. These approaches are often ideal for specific scenarios in the domain of dismount tracking however none can be used across all scenarios.

The scale of the image, and the smaller size of the target, can make it difficult to detect targets. A different suite of Track Before Detect (TBD) and anomaly detection algorithms are often used. Conventionally for problems like this, a noise detection algorithm with Multiple Hypothesis Testing (MHT) or Track Before Detect (TBD) approach is often used. These techniques are often used in Short Wave Infrared (SWIR), electro-optic (EO), and radar tracking. These algorithms can be effective for extremely small targets. Such filters rely on parameters that define the size of the target or its variance from the background signal. In unknown environments, it can be difficult to predict such variables. These algorithms are also prone to higher numbers of false alarms and do not rely on any features to distinguish between potential targets and detections that are similar in appearance. The FM-MIMS algorithm (section 3) is designed for slightly larger targets with minimally identifiable features. The extra information provided by these features can help augment such as the one proposed by Zhu *et al.* (2014) and increase performance.

For smaller targets, ones that register as only a few pixels on the screen, there are very few features to recognize. In such scenarios traditional approaches of thresholding and pattern recognition yield higher levels of false alarms. “One of the shortcomings of the conventional tracking method is that when target echo level is low, also referred to as low-observable target, the thresholding process does not sufficiently separate target energy and energy from non-targets, also known as clutter. As a consequence, a higher probability of detection cannot be achieved without having to tolerate an increase in the probability of false detection.” (Park and Doherty, 2015) Low-observable targets are generally targets that have a size that ranges from a few pixels to the sub-pixel range. Depending on their size, MHT and TBD algorithms can be effective. Augmenting these algorithms with a motion model has proven effective in further reducing the probability of false tracks. In this section, we will explore the efficacy and shortcomings of such algorithms.

1.2.1 Feature Recognition

Traditional approaches often use feature recognition to detect identifiable body parts. “In general, a human descriptor is comprised of features organized in a structure. It is expected that the structure enables the description of human objects in various viewpoints and poses.” (Nguyen *et al.*, 2016) Descriptors can be based on edges, texture, shape, and motion queues. Algorithms will attempt to detect facial features and limbs to identify a dismount as a target of interest.

Feature recognition relies on descriptors to identify simple patterns. For instance, the following research identifies limbs by correlating their small size with an approximated oscillating motion. (Hersey *et al.*, 2008) (Narayanaswami *et al.*, 2012) Dismounts however do not constantly swing their arms and legs so more features would be required. Detecting a face can be a very reliable way to detect people. Some techniques can even compensate for partially occluded faces (Shanmugavadivu and Kumar, 2016). Facial recognition follows a similar technique but requires more complexity. Descriptors must first find smaller features such as eyes, nose, and mouth. The features and their relative positioning to each other could be used to recognize the face. These strategies can prove successful in domains where the data has a lot of variances. However, their heavy reliance on detecting key image features makes them prone to error when dismounts are further away "in certain images in which people or groups of people are far from the sensor (±4 meters), the technique used is not able to determine the existence of them." (Jiménez-Bravo *et al.*, 2020) These solutions are appropriate for detecting people on webcams and high-quality security cameras. They are not very effective for detection and tracking using WAMI.

These algorithms, although effective, have an aversion to noise at smaller scales. "The question remains whether a different algorithm specifically designed for the purpose would be able of de-noising these ultra low-resolution spectra and unveiling the cadence of human walking. These levels of resolution are more consistent with Wide Area Imagery, than close-range video." (Narayanaswami *et al.*, 2012) The accuracy of such algorithms is dependent on the detail available when gathering data. Higher levels of details and a greater number of identifiable descriptors are required for successful identification. These algorithms work well for larger targets where noise can be filtered out with relatively more ease.

The proposed FM-MIMS algorithm will attempt to use some of these concepts by generating feature matrices to identify simple features such as the spot size within the context of its background. This small target detector (section 3.3) will help reduce false detections.

1.2.2 Multiple Hypothesis Testing (MHT)

Multiple Hypothesis tracking is one of the earliest successful visual tracking algorithms. Proposed by Reid in 1979, the algorithm works by building a tree of potential track associations for each candidate target. As more data is acquired an informed decision can be made by calculating the likelihood of each track variation. The most likely tracks can then be selected and confirmed. This systemic approach to data association is ideally situated to exploit higher-order information. For instance, the likelihood calculations of a known target can be dependent on various motion and appearance models. In the dismount tracking domain, the velocity and size variations caused by swinging arms can be considered when detecting and identifying a target. MHT has been a popular approach in the radar and image tracking community. The growing popularity of MHT has resulted in many variations of the algorithm. Variations of the algorithm such as Multiple Hypothesis Testing with discriminatory appearance modeling(MHT-DAM) Kim *et al.* (2015), interacting multiple model/multiple hypothesis tracking (IMM/MHT) Blackman *et al.* (1999) and histogram probabilistic multi-hypothesis tracker (HPMHT) Davey *et al.* (2008) attempt to optimize a certain aspect based on specific requirements of the application at hand.

MHT algorithms can be very effective at identifying and tracking relevant targets. They can however be computationally expensive. The number of possible tracks increases exponentially relative to the number of detection in each frame. Evaluating each permutation can take a significant amount of time. “ In the early work on

MHT for visual tracking, target detectors were unreliable and motion models had limited utility, leading to high combinatoric growth of the search space and the need for efficient pruning methods" Kim *et al.* (2015) Efficient MHT algorithms depend on reliably pruning the tree of potential tracks. Without such pruning, MHT can quickly use a lot of memory over the course of a few frames. Furthermore, developing an accurate motion model can be a challenge. In the dismount tracking domain, it is difficult to model the overall motion of a target. Humans are capable of changing velocities and accelerations on a whim. To add to the model complexity, their range of motion is not constrained along any axis. Because of this MHT can be a useful tool in forming tracks of dismounts but ineffective in differentiating dismounts from other objects in the scene.

1.2.3 Track Before Detect (TBD)

Track before Detect (TBD) is another algorithm often used in radar and video tracking solutions. The fundamental approach is to integrate the signals of tentative targets over time. This is done when the signal from a target is too weak at a particular instance in time. The cumulative signal over a range of time could be strong enough to register as a detected target. TDB can be implemented in batch and recursive ways. Batch methods such as dynamic programming Boers and Driessen (2005) and maximum likelihood estimation Boers and Driessen (2001) can be computationally expensive. Recursive methods "performed well in simulated data, with each technique reliably detecting targets at 6dB, with occasional detections at 3dB." Rutten *et al.* (2005) As the SNR decreases, it becomes increasingly difficult to detect targets. "The potential risk of threshold method is that useful information may be thrown away and will lead to the loss of target, especially when the signal-to-noise ratio (SNR) is low." (Amrouche *et al.*, 2017)

1.3 Objectives And Scope

The objective is to explore a novel method of detecting dismounts on WAMI Equipment. A robust algorithm can be developed by applying lessons learned while exploring the state of the art in target detection algorithms. This can be achieved by extending the tensor voting (TV) algorithm to incorporate more data when forming associations. This Multi-scale Intrinsic Motion Structure (MIMS) model by Zhu *et al.* (2014) accomplished this. MIMS was able to detect targets that were 4-9pixels in size. It was able to do so while outperforming other well-known algorithms in its class.

This thesis will focus on the components used in the MIMS framework, particularly dismounts captured by airborne video, and attempt to improve its results. The extended algorithm, FM-MIMS, will aim to provide greater detection accuracy for larger targets than the MIMS algorithm was intended for. The adaptation will allow for tracks and detections that are more resilient to the variance in the size of targets that appear in the scene. This would reduce the need to switch between multiple tracking algorithms based on the distance of the sensor to the target.

Given that the focus of the algorithm is on dismounts captured by airborne video, the FM-MIMS algorithm will have to perform under the following conditions. The targets will register as a $15-30$ pixels target. Dismounts can often appear this large on WAMI video. The depression angle of the footage may vary from $15-90$ degrees below the horizon. This is an important parameter, and it would impact the variance in target size on any given frame. Closer targets would appear larger than those standing near the top of the frame. Many of the use cases of dismount detection are in poor or unpredictable lighting conditions. Therefore, the project aims to explore videos with varying lighting conditions and times of the day. This will account for varying shadow lengths and the appearance of the targets. The data used to train and test the algorithm will conform to the above parameters.

Success FM-MIMS algorithm will be determined if it has comparable or better performance relative to the MIMS framework while detecting targets that are $15 - 30$ pixels in size. An ideal detector would be able to do so with a Probability of Detection (PD) of $PD = 1$ and a Probability of False Alarm (PFA) of $PFA = 0$. This would mean that the algorithm can detect every single target in the scene and does not register any of the background noise or clutter scene as a detection.

1.4 The Contribution

This thesis will develop a new FM-MIMS framework based on the MIMS framework initially proposed by Zhu *et al.* (2014). FM-MIMS will utilize sensor metadata to predict the size of the dismount. Based on this data, FM-MIMS will add to the MIMS framework by appending pre-processing modules to detect features of interest in the scene. As airborne WAMI sensors are constantly experiencing motion jitters, this should allow the algorithm to be more resilient to changes in the observed scene. The metadata will heavily influence the output of the preprocessing modules to develop feature matrices that are resilient to gradual changes in the sensor's orientation and distance from the target. In addition to the preprocessors, FM-MIMS makes mathematical approximations to the small target detection module and the tensor voting algorithm to reduce the computation load of the framework. The approximations made in the small target detector omit background noise as this is filtered later in the algorithm. FM-MIMS also implements Monte Carlo integration as suggested by Guest (2009) to reduce the computation time required by multidimensional tensor voting. Essentially the contributions of this thesis, explored in section 3, aim to expand the capabilities of MIMS by developing a more reliable algorithm.

Chapter 2

Background

In this section, we will explore the individual components used in the framework and their theory.

2.1 Small Target Detection

Small target detection is a critical problem in many scenarios. When detecting dismounts using WAMI targets can often range from 2-30pixels in size depending on the distance and configuration of the sensor. Detecting small targets from a distance can pose a challenge for a few reasons. First and foremost relevant features such as texture and color become difficult, if not impossible, to distinguish. Conventionally small targets can be detected by the difference between the current frame with a mean filter. However, this often results in edges to objects, such as clouds, buildings, and shorelines, being falsely detected.

The method proposed by Xie *et al.* (2015) performs well in such scenarios as they "focus on removing target while preserving the high-frequency component in the background." (Xie *et al.*, 2015) This is done by modeling the image as

$$f = f_T + f_B + n \tag{2.1.1}$$

where f, f_T, f_B, and n are the image, target image, background image, and random noise, respectively. The noise is assumed to have a normal distribution with a variance of σ^2 and a mean of 0. Most images analyzed by the algorithm will be a combination of many surfaces and textures. Often there will be features of the land, water, and sky in the image. The algorithm should be able to sufficiently identify small targets in all these varying states. For this reason, a sliding window will be used on all the pixels in the image. The window will begin at one corner of the image and work its way to the opposite diagonal corner while having its center cover every single pixel in the image. The purpose of this method is to estimate f_B using an optimization-based filtering approach.

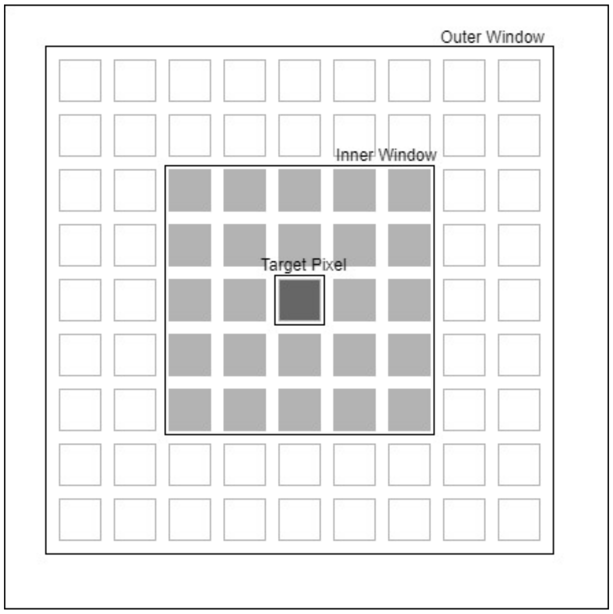


Figure 2.1: A visual representation of the moving window used for the sliding window. The white color represents pixels in the active area, light grey color represents pixels inside the inner window, and dark grey color is the center pixel.

M and N are the sizes of the inner and outer window respectively. The active area, marked in grey in figure 2.1, is the area between the inner and outer window. It is denoted as $\Omega = \{1, 2, ..., n\}$ where n is the total number of pixels in the active area and Ω is the set of pixels in the active area. The intensity of each pixel $i \in \Omega$ is denoted by x_i. The intensity of each background image pixel $i \in \Omega$ is denoted by y_i.

In the algorithm, Xie *et al.* (2015) uses a weighted quadratic model.

$$y_i = ax_i + b, i \in \Omega \tag{2.1.2}$$

The weighted quadratic model is

$$\min_{\begin{bmatrix} a & b \end{bmatrix}^T} \sum_{i \in \Omega} \left(y_i - \begin{bmatrix} x_i & 1 \end{bmatrix} * \begin{bmatrix} a \\ b \end{bmatrix} \right)^2 w_i + \epsilon a^2 \tag{2.1.3}$$

Here a and b are parameters in the local linear model. The regularization parameter ϵ is used to penalize large values of a. the weights w_i are used to dive various levels of importance to the individual pixels in the active area.

$$w_i = e^{-c*||x_0 - x_i||_2^2} \tag{2.1.4}$$

Where c is a constant and x_0 is the intensity of the center pixel. We can also denote

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ 0 \end{bmatrix} \tag{2.1.5}$$

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_i & \sqrt{\epsilon} \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix} \tag{2.1.6}$$

$$W = \begin{bmatrix} w_1 & 0 & \cdots & 0 & 0 \\ 0 & w_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_i & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{2.1.7}$$

This can be plugged into equation 2.1.3 and rearranged to get.

$$\begin{bmatrix} a \\ b \end{bmatrix} = (XWX^T)^{-1}XWY \tag{2.1.8}$$

If we set $x = y$ we can solve the system for a and b

$$a = \frac{\sigma^2(x)}{\sigma^2(x) + \frac{\epsilon}{\sum_{e\in\Omega} w_i}} \tag{2.1.9}$$

$$b = (1 - a) * E(x) \tag{2.1.10}$$

Where $\sigma^2(x)$ is the variance of the intensities of the pixels in the active area and, $E(x)$ is the expectation of the pixels in the active area. this can be used to estimate the background pixel's intensity.

$$y^* = ax_0 + b = ax_0 + (1 - a) * E(x) \tag{2.1.11}$$

This novel filter developed by Xie *et al.* (2015) can do a good job to estimate the background as long the M, N, and ϵ parameters are tuned to the right size. In section 3.3 this algorithm will be modified to better work with the dismount detection framework.

2.2 Tensor Voting

Tensor voting is a methodology that was initially developed to tackle vision problems in 2D. It has since been extended to address problems in N-D. This non-iterative process does not require any initial thresholding or guesswork. The scale is the only free parameter in the algorithm. This robust algorithm can handle multiple structures and extreme noise corruption.

"The goal is to extract geometric features such as regions, curves, surfaces, and the interactions between them." (Tang *et al.*, 2000) The algorithm has two main elements. Tensor calculus is used for data representation. Tensor voting is used for data communication. When using tensor voting, each data point is represented as a tensor. The tensors propagate their information, encoded in a tensor, to their neighbors. The location of each tensor collects the votes cast, analyzes them, and builds a saliency map of each feature type.

Each input token is first encoded in a second-order systematic tensor. Secondly, the tokens send their information to each other in a neighborhood where each token is a generic second-order symmetric tensor. This tensor encodes the confidence, curvature, and orientation information of the tensor. In the next stage, generic tensor tokens are used to propagate their information within their neighborhood. This leads to a dense tensor map that encodes the saliency at every point. The final tensor map is decomposed into its elementary components. each component represents the strength of its surface, curve, and junction features.

2.2.1 Tensor Encoding

Input tokens are first encoded as tensors. Tensors can be decomposed and represented as a summation of n-spheres. (Figure 2.2) In 1D the tensor is represented as a stick, in 2D a plate, in 3D a ball, and so on. These elements communicate with each other to help define the preferred orientation. When encoding the expectation of the data over time into a tensor, the input tensor $\begin{bmatrix} t_1, t_2, \cdots t_n \end{bmatrix}$ of length n would be represented as

$$T = \begin{bmatrix} t_1 \times t_1 & t_1 \times t_2 & \cdots & t_1 \times t_n \\ t_2 \times t_1 & t_2 \times t_2 & \cdots & t_2 \times t_n \\ \vdots & \vdots & \ddots & \vdots \\ t_n \times t_1 & t_n \times t_2 & \cdots & t_n \times t_n \end{bmatrix} \tag{2.2.1}$$

This second-order systematic tensor can be described by its associated eigensystem. where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$

$$T = \begin{bmatrix} \hat{e}_1 & \hat{e}_2 & \ldots & \hat{e}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \begin{bmatrix} \hat{e}_1^T \\ \hat{e}_2^T \\ \vdots \\ \hat{e}_n^T \end{bmatrix} \tag{2.2.2}$$

This eigensystem can be rearranged as

$$T = \sum_{i=1}^{n-1} s_n v_n + \lambda_n s_n \tag{2.2.3}$$

v_n is the normal vector represented by

$$v_n = \sum_{i=1}^{n} \hat{e}_i \hat{e}_i^T \tag{2.2.4}$$

where the eigenvectors represent the tensor's local coordinate system. s_n is the saliency represented by

$$s_n = \lambda_{n-1} - \lambda_n \tag{2.2.5}$$

These are the components that are used in the voting phase.

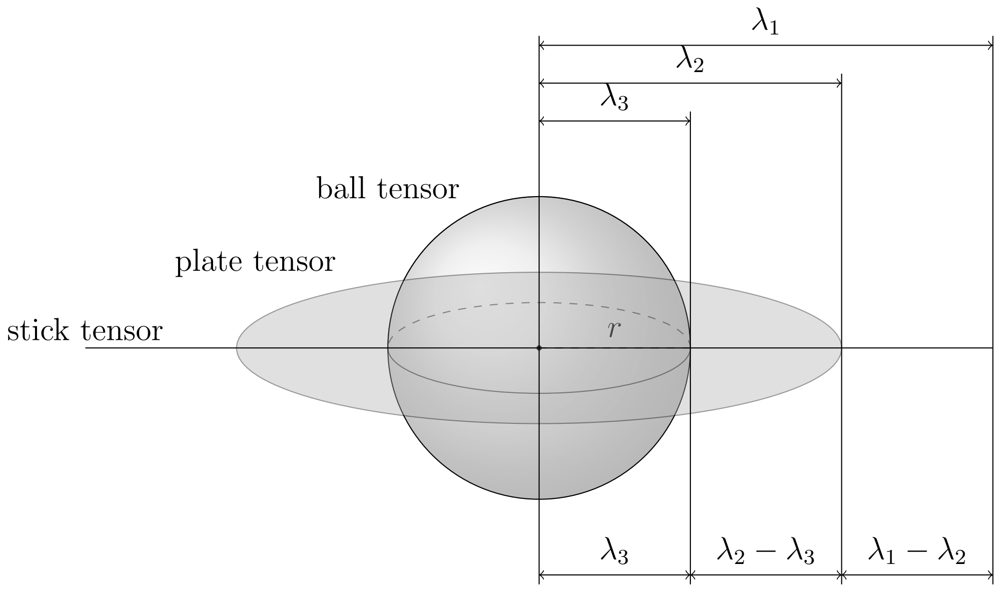


Figure 2.2: A tensor can be decomposed into its eigenvectors.

2.2.2 Token Refinement

Once all the tensors have been decomposed, they need to communicate their information to local tensors. "Each input token votes, or is made to align, with precomputed, discrete versions of the basis tensors in a convolution-like way, propagating preferred direction in a neighborhood. We call these precomputed basis tensors voting fields." (Tang *et al.*, 2001) Since each field is represented by an eigenvector orthogonal to the rest, they will all be computed separately and later reassembled.

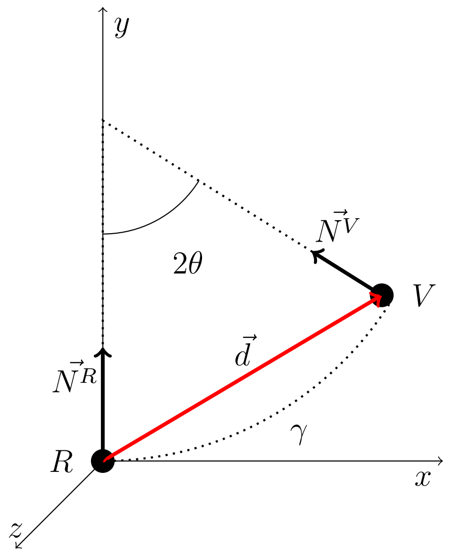


Figure 2.3: Design of the fundamental 3D stick tensor voting field

"Voting fields of any dimension are derivable from the 2D fundamental stick voting field." (Tong *et al.*, 2001) This will be referred to as the fundamental 2D stick voting field (VF) shown in Figure 2.3. The field is derived by asking what the most likely normal through V would be for a curve that passes through R and V with the normal $\vec{N^R}$. The most likely Communication shown by line γ in Figure 2.3 would be the osculating circle connecting points R and V as it would keep the curvature constant. A multidimensional VF can be formed by rotation of the 2D VF around $\vec{N^R}$ The magnitude of the VF is also subject to a decay function defined as

$$DF(\gamma,\phi,\sigma) = \begin{cases} e^{-\left(\frac{\gamma^2+c\phi^2}{\sigma^2}\right)}, & \theta \in \left[\frac{-\pi}{4}, \frac{\pi}{4}\right]. \\ 0, & otherwise. \end{cases} \tag{2.2.6}$$

where γ is the arc length of line RV, ϕ is the curvature, c is the scale factor and σ is the scale of analysis. In this equation, c is a scalar value that provides some control to how the curvature influences the equation. And σ is the only free parameter. This can be adjusted by the user to determine the size of the voting neighborhood.

c is defined as

$$c = \frac{(\sigma - 1)(-\log 0.1)}{\frac{\pi^2}{4}} \tag{2.2.7}$$

The stick vote is calculated by

$$V_{stick}(\vec{N^V}) = DF(\gamma, \phi, \sigma) \times \vec{N^V}\vec{N^V}^T \tag{2.2.8}$$

Multi-dimensional voting in a similar manner to the 2D stick field. They are done so by rotating the stick field around the appropriate axes as indicated in Table 2.1.

Dimension	Saliency	Normal Vectors	Tangent Vectors	Feature
0	λ_n	none	none	ball
1	$\lambda_{n-1} - \lambda_n$	$e_1, e_2, \ldots, e_{n-2}, e_{n-1}$	e_n	
2	$\lambda_{n-2} - \lambda_{n-1}$	$e_1, e_2, \ldots, e_{n-2}$	e_{n-1}, e_n	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
n-2	$\lambda_2 - \lambda_3$	e_1, e_2	$e_3, \ldots, e_{n-1}, e_n$	plate
n-1	$\lambda_1 - \lambda_2$	e_1	$e_2, e_3, \ldots, e_{n-1}, e_n$	stick

Table 2.1: Tangents and normals of multidimensional tensors

The vote is calculated as:

$$V = \int \cdots \int_0^{2\pi} R_\theta V_{Stick}(\vec{T^V}) R_\theta^T \, d\theta \tag{2.2.9}$$

where R_θ is the rotational matrix. The stick vote is first rotated with the rotation matrix R such that it is aligned with the e_1 vector of the stick vote at point P. The vote is integrated over a rotation around e_n to obtain the 1st dimension vote. Similarly, for the 2nd dimension, the stick vote is integrated over a rotation around e_n, followed by a second integration around e_{n-1}. The process continues until a vote is cast for every dimension.

2.2.3 Tensor Voting

Input site T^R will collect votes from all the voters i. All sites in the saliency tensor at each location T^R is the tensor sum of all the individual contributions of each voting token T^V. This vote consists of all the components computed in the previous section.

$$T_1^R = T_0^R + \sum_i T_{Ball}^{V_i} + \ldots + T_{Plate}^{V_i} + T_{stick}^{V_i} \tag{2.2.10}$$

2.2.4 Feature Extraction

To find meaningful features, T_i^R is decomposed back into its point, surface, curve, and ball features to be examined separately. "Surfaces and curves are extracted as the local maxima of surface and curve saliency respectively, while junctions can be extracted as the local maxima of junction saliency without any form of propagation. Since calculating votes for every location in the volume containing the data points is pointless and impractical, surface and curve extraction begins from seeds, location with the highest saliency, and voting is performed only towards the directions indicated by the surface normal and curve tangents." Tang *et al.* (2000) Features are extracted for each field by searching for the highest likelihood. In the ball field, we can search for the maxima. The strength of the other fields, however, also depends on the orientation. Therefore the gradient vector is defined as

$$\bar{g} = \nabla T_1^R = \begin{bmatrix} \frac{\delta T_1^R}{\delta t_1} \\ \frac{\delta T_1^R}{\delta t_2} \\ \vdots \\ \frac{\delta T_1^R}{\delta t_n - 1} \\ \frac{\delta T_1^R}{\delta t_n} \end{bmatrix} \tag{2.2.11}$$

$$q = \hat{n} \cdot \bar{g} \tag{2.2.12}$$

In equation 2.2.12, $\hat{n}$ is determined by the direction of the normal eigenvector.

2.3 AdaBoost

AdaBoost, or Adaptive boosting, is a boosting algorithm used as an Ensemble method in Machine Learning. "Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model." Lutins (2017) AdaBoost relies on several weak classifiers used sequentially to create a single strong classifier. It does so by using a single split from decision trees called decision stumps. Each stump is given a weight based on the difficulty to classify. Higher difficulties get a higher weight and lower difficulties get a lower weight. Weights are reassigned on every instance the algorithm is run based on incorrectly classified data.

A general learning framework is adopted where the learner receives samples randomly chosen from $X \times Y$ where X is the domain and Y is the set of possible labels. These samples are chosen randomly according to some unknown distribution $\mathscr{P}$ Assuming a training dataset of size N, the labeled instances are $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$.

The goal of AdaBoost is to find a Hypothesis h_f such that the outcomes are consistent with most of the samples where $h_f(x_i) = y_i$ where $i \in \mathbb{Z} : i \in [1, N]$. "Final hypothesis with low error relative to a given distribution D over the training examples." Freund and Schapire (1997) The distribution D is generally set to a uniform $(D(i) = 1/N)$ distribution but can be defined by the learner. It is unlike $\mathscr{P}$ which is determined by the nature of the data.

2.3.1 Background

Decision Trees

Both Adaboost and Random Forests are based on decision trees. A decision tree is a simple way to convert preexisting data into a means to predict and classify new data. Decision trees are used for less complex data over neural nets as the reasoning behind each decision can be observed and the logic easily visualized. The tree follows a flow chart-like structure. Where each node tests one particular attribute. Based on the outcome, the node can make a decision between one of two branches. As seen in Figure 2.4 each branch can have subsequent nodes to test multiple attributes.

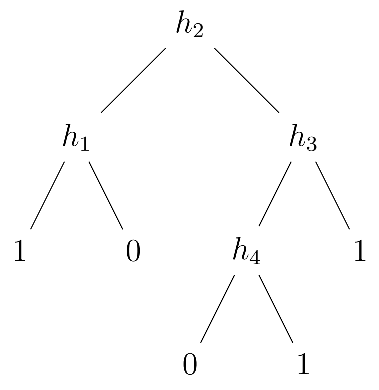


Figure 2.4: A sample decision tree

The tree structure can be made as long as necessary to ensure full coverage of the data. This is determined by the impurity of the data. Since all of the data is used to make a decision, it is considered a strong learner.

Stumps

AdaBoost relies on a similar concept of using multiple weak learners (nodes) together to create a strong classifier. Each decision node in this case is referred to as a Stump. Individual stumps only use one variable to make a decision.

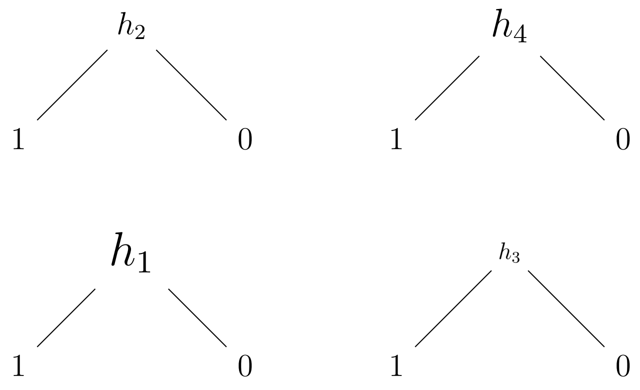


Figure 2.5: A sample forest of stumps

The process in section 2.3.2 will outline how different weights are assigned to each stump to achieve optimal outcomes

2.3.2 Adaptive Boosting Process

The algorithm is first initialized by setting the number of iterations T that AdaBoost will run for. The Weight factor w_i^t, where t is the current iteration, is initialized to w_i^0, which is commonly set to a uniform distribution.

$$w_i^0 = D(i) = \frac{1}{N} \tag{2.3.1}$$

After the first iteration, the weights change based on the performance of the initial stump. This is to guide how the next stump is created.

Picking A Classifier

The initial classifier must rely on the input that has the largest impact on the outcome. This is done by first normalizing w_i^t.

$$p^t = \frac{w^t}{\sum_{i=1}^{T} w_i^t} \tag{2.3.2}$$

Next a weak classifier (stump) $h_t : X \to [0,1]$ is used to classify based on each individual input. The function h_t is defined such that

$$h_t = \begin{cases} 1, & if[condition]. \\ 0, & otherwise. \end{cases} \tag{2.3.3}$$

To find the efficacy of each potential h_t, the impurity of the results must be computed. The original paper for Adaboost Freund and Schapire (1997) suggested an error equation however modern approaches use the Gini impurity. The Gini impurity is a number between 0 and 1 and is used to measure the likelihood of an incorrect classification.

$$G(h_t) = \sum_{i=1}^{N} P(h_t, i)(1 - P(h_t, i)) \tag{2.3.4}$$

where $P(h_t, i)$ is the probability of a certain classification i. The $h_t(x_i)$ with the lowest Gini impurity gets selected to be the first stump in the stump forest.

Evaluating Performance

Once a Stump has been chosen it is important to determine how much error is remaining in the classifier. This will help determine the weights for the dataset in the next iteration. The error is the sum of the weights of all the incorrectly classified data points.

$$E(h_t) = \sum_{i=1}^{N} w_{i,incorrect} \tag{2.3.5}$$

The total error will be 0 if a perfect stump and 1 for an imperfect stump. The output weight of the stump α_t can now be computed

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 - E(h_t)}{E(h_t)} \right) \tag{2.3.6}$$

In practice, a small error term is added to prevent divisions with 0.

Updating Weights

After a stump has been chosen, the weights for each data point in the set need to be reevaluated based on if they were correctly classified or not. The incorrectly classified data receive a higher weight to allow the next stump to do a more accurate job. Now we modify weights to make sure subsequent stumps correct for errors from the first stump.

$$w_i^t = \begin{cases} w_i^{t-1}e^{-\alpha_t}, & \text{correctly classified.} \\ w_i^{t-1}e^{\alpha_t}, & \text{incorrectly classified.} \end{cases} \tag{2.3.7}$$

The Final Classifier

After T iterations of this process, the final hypothesis can be outputted.

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{2.3.8}$$

Where $H(x)$ is the certainty of one particular outcome.

2.3.3 Overfitting

When working on training data sets it is important to account for overfitting. "In general, a hypothesis which is accurate on the training set might not be accurate on examples outside the training set; this problem is sometimes referred to as 'over-fitting.' Often, however, overfitting can be avoided by restricting the hypothesis to be simple" Freund and Schapire (1997) It is best to randomly select 2/3 of the data set to train the algorithm and the remainder of the 1/3 for testing. Over-fitting can further be avoided by using large sample sizes and keeping the value of T as low as possible.

2.4 Tracking with Multiscale Tensor Voting

This algorithm framework was developed by Zhu *et al.* (2014) The aim of the algorithm framework is, given a WAMI clip, to identify pedestrians.

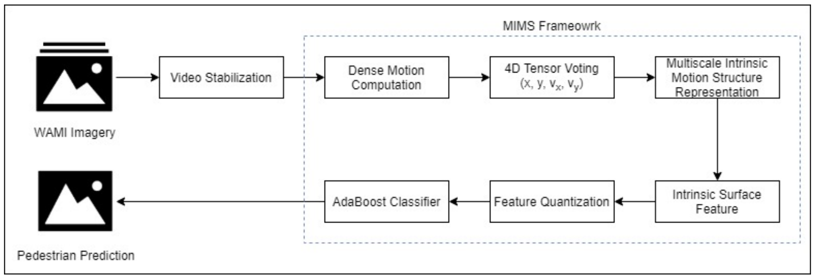


Figure 2.6: A sample decision tree

Before executing the algorithm it is important to stabilize the video. This is to compensate for the camera's motion. Once the image is stabilized, optical flow is used to view the movement over two consecutive frames.

The Intrinsic Motion Structure (IMS) is now defined as (d, s, σ) where d is the index of the maximum eigenvalue difference (dimensionality), s is the maximum eigenvalue difference (saliency) and, σ is the scale used. At each different value of σ we can expect the respective IMS values to change. The Multi-scale IMS (MIMS) is defined as multiple IMS values taken at regular scale intervals.

In order to find the IMS of each pixel, we must first initialize a 4D tensor (x, y, v_x, v_y) This information can be gathered by running optical flow over two consecutive frames. In 2D the point at each location is referred to as a pixel, in 4D the temporal shape of the motion is a fiber. In an image, multiple pixels moving in a similar pattern of motion would create multiple fibers. this is known as a fiber bundle. the properties of the bundle can vary at various scales. At a large enough scale, the bundle would appear as a single fiber.

2.4.1 Tensor Voting

Tensor voting is done using the method outlined in section 2.2. For this application, the input data, (x, y, v_x, v_y), is represented as the input tensor

$$T = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \begin{bmatrix} x & y & v_x & v_y \end{bmatrix} \tag{2.4.1}$$

After the tensor voting process, it can be decomposed into its eigenvalues λ_i and eigenvectors e_i where N I the dimensionality.

$$T = \sum_{i=1}^{N} \lambda_i e_i e_i^T = \sum_{i=1}^{N-1} \lambda_i - \lambda_{i+1} \sum_{k=1}^{i} e_k e_k^T + \lambda_N \sum_{i=i}^{N} e_i e_i^T \tag{2.4.2}$$

The motion dimensionality d and saliency s reveal the local motion structure of the object.

$$d = \arg\min_i (\lambda_i = \lambda_{i+1}) \tag{2.4.3}$$

$$s = \lambda_d = \lambda_{d+1} \tag{2.4.4}$$

A local motion structure would have a d dimensional normal space spanned by the eigenvectors $e_1, e_2, \ldots, e_d$. The $N - d$ dimensional tangent space is spanned by the eigenvectors $e_{d+1}, e_{d+2}, \ldots, e_{N-1}, e_N$

2.4.2 Multi-scale Representation

The only free parameter when computing the IMS is σ. When voting, this controls the size of the neighborhood. It is very difficult to judge the optimal σ to use in a situation. This can be a learned parameter that is trained with various relevant scenarios in mind. The approach is not ideal as it requires prior knowledge of the situation. Instead of this, a multi-scale approach is used.

$$\{(d_i, s_i, \sigma_i), i = 1, 2, \ldots, k\} \tag{2.4.5}$$

where d_i is the intrinsic dimensionality and $d_i \in \mathbb{N}^+, 1 \leq d_i < N$. σ_i is the saliency where $s_i \in \mathbb{R}^+$ at scale σ_i. Dismount detection is used as a binary classification problem.

Suppose S is normalized, between $[0, 1]$ and k bins are chosen to quantize the normalized saliency values as $\tilde{s}$. For binary dismount classification, the tuple X, y represents the sample. X is a residual pixel. y is the class label where $+!$ is a pedestrian and -1 is not a pedestrian. The value at a certain scale can be represented as

$$f(X, \sigma) = (d, \tilde{s}) \tag{2.4.6}$$

$f(X, \sigma)$ is used as a 2D indices of a 2D lookup table.

$$W_f^c = P(f(X, \sigma) \in bin_f, y = c), c = \pm 1 \tag{2.4.7}$$

where W_f^c encodes the 2D distribution of features amongst both the positive and negative training samples.

2.4.3 Boosting

According to section 2.3.3 AdaBoost relies on a weak classifier $h(x)$. In the context of MIMS it is classified as

$$h(X, \sigma) = \frac{1}{2} \ln \left(\frac{W_F^{+1} + \epsilon}{W_F^{-1} + \epsilon} \right) \tag{2.4.8}$$

where $f = f(X, \sigma)$, W_F^{+1} and W_F^{-1} is defined by equation 2.4.7.

$$B_f(u) = \begin{cases} 1 & u \in bin_f, f \in 2Dindices \\ 0 & otherwise \end{cases} \tag{2.4.9}$$

The weak classifier in equation 2.4.8 can be formulated as

$$h(X, \sigma) = \frac{1}{2} \ln \left(\frac{W_F^{+1} + \epsilon}{W_F^{-1} + \epsilon} \right) B_f(f(X, \sigma)) \tag{2.4.10}$$

As in section 2.3.3, the weak classifier 2.4.10 is used, similar to equation 2.3.8, to make a strong classifier $H(x)$.

$$H(x) = \sum_{i-1}^{T} \alpha_i h(X, \sigma_i) + b \tag{2.4.11}$$

where T is the number of weak classifiers used in $H(x)$. b is used as a threshold.

Chapter 3

Extending The Algorithm

This section will take an in-depth look at the modifications that have been made to MIMS in order to develop the proposed FM-MIMS algorithm. The contributions made here focus on making incremental changes at each stage of the framework to allow a more diverse set of features to be extracted from the scene while attempting to minimize the computational requirements of the algorithm.

The MIMS algorithm by Zhu *et al.* (2014) mentioned in section 2.4 has proven that "the MIMS feature outperforms the state of art appearance and motion-based features for pedestrian detection." Zhu *et al.* (2014). This section aims to extend the algorithm by adding a prediction module and identifying some features. This additional data would be used in tensors of higher-order to be used in the tensor voting algorithm. This model should allow for improved results when tracking a higher density of dismounts. Higher-order tensors allow the potential to include descriptors such as color and size in addition to the velocity descriptor used in MIMS. This would make the FM-MIMS algorithm more robust to noise and background clutter. Increasing the amount of data used proportionately increases the size of the tensors when tensor voting. This exponentially increases the amount of computation required when tensor voting. The following framework will address techniques to help reduce the computational requirements. This is done through some small target filtration techniques and optimizations in the algorithm.

3.1 The Framework

Similar to Figure 2.1 The framework for the proposed algorithm as shown in Figure 3.1. The new FM-MIMS framework relies on 3 key inputs, the WAMI video stream I, sensor metadata S, and the known properties of dismounts D.

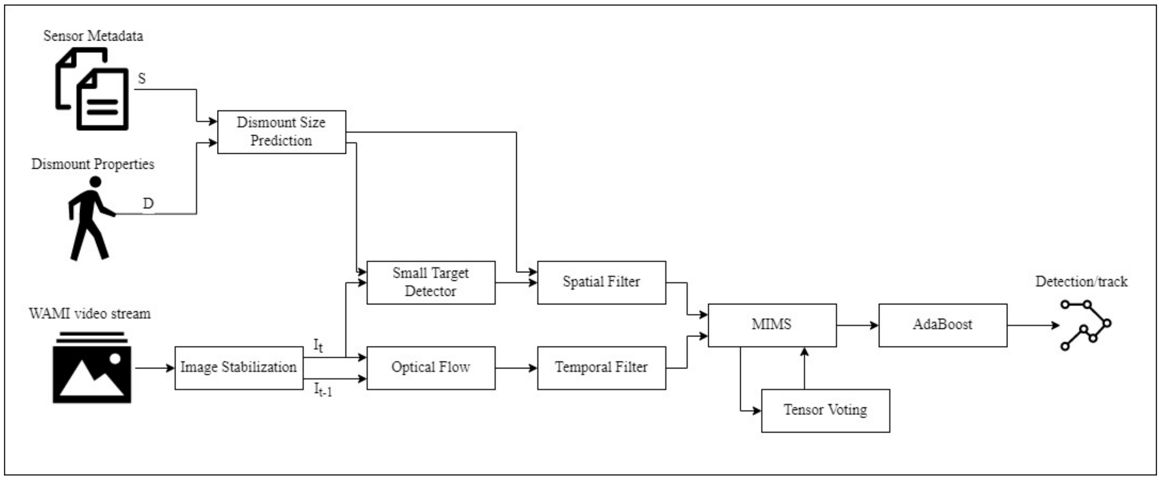


Figure 3.1: FM-MIMS framework

The FA-MIMS framework improves on MIMS by using known properties of the target such as the average size and known maximum velocity, along with sensor metadata to predict acceptable values of detections in the frame. These predictions drive the parameters used in the spatial and temporal filter to help disregard irrelevant data for tensor voting.

The framework begins with the dismount prediction module. This module uses the average minimum and maximum dimensions of the target as well as sensor metadata. The information is used to generate the minimum and maximum estimates of the size of the target as viewed by the sensor.

The small target detector uses a sliding window to develop a feature matrix. The matrix is an indicator of how likely the pixel is to be associated with a small target that meets the required parameters. The window size is determined by the parameters of the dismount size predictor. The maximum width and height determine the width and height of the outer window, respectively. The minimum width and height determine the width and height of the inner window, respectively.

In parallel to this, an image stabilization module is used to create a steadier video feed to be analyzed by the optical flow module. This is to account for the sensor, or the aircraft it is mounted to, moving.

The optical flow module, in turn, compares the current image in the video with a prior image to develop a second feature matrix. This matrix is a representation of the motion vector of each point.

The small target feature matrix and optical flow feature matrices are independently thresholded. Thresholding relies on a stochastic gradient descent approach to calculate the optimal threshold value for each matrix.

The thresholded matrices are used to compile individual tensors for each pixel, which are sent to the MIMS module. The MIMS module uses these tensors to run the tensor voting algorithm at multiple scales. The results from each scale are stored to be classified by AdaBoost.

Finally, the Boosting module uses the saliency and dimensionality values at various scales to individually identify if each pixel is a dismount.

3.2 Dismount Size Prediction

This module intends to approximate the plane of the ground below the sensor. The contribution of this module to the FM-MIMS model, proposed in this thesis, should allow feature detection parameters to be tuned more precisely. Although the ground may not be flat, it gives an approximation in 3d space of the location the dismount can be found. On this surface, the location of each pixel is mapped using the sensor's field for view and its resolution. A prism with the average dimensions of a dismount is placed at the location of each pixel. The coordinates of the prism are used to predict the approximate height and width of the image that would be received by the sensor. This approach will allow the algorithm to better discriminate between targets that are a reasonable size to be a dismount and those that are not.

The known properties of dismounts, D, is a composition of the largest width and height a dismount can be, w_{max}, and h_{max}, respectively. Similarly, the smallest dimensions, w_{min}, and h_{min}, of the dismounts were also used. These dimensions were used to approximate a rectangular prism of space the dismount could potentially occupy. These prisms paired with the sensor metadata are used to predict the size of a dismount at any given pixel.

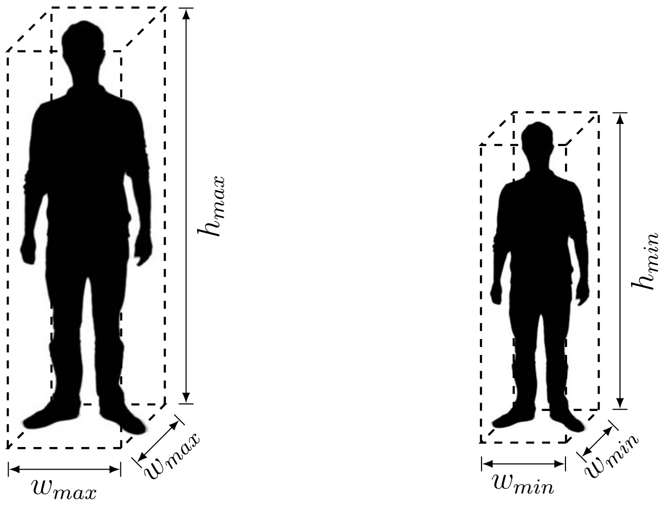


Figure 3.2: An approximation of the dimensions of a dismount

The metadata, S, is composed of the position and angle of the sensor in 3d space, $[x, y, z, \alpha, \beta]$, where x, y, and z are the position in cartesian space, α is the angle of depression from the horizon, and β is heading of the sensor. Furthermore, the sensor's horizontal and vertical field of view, FOV_h and FOV_v respectively, are required as well as the horizontal and vertical resolution of the sensor, r_h, and r_v. Using these sensor parameters, we can first calculate the instantaneous field of view (IFOV) of the sensor.

$$\begin{aligned} IFOV_h &= \frac{FOV_h}{r_h} \\ IFOV_v &= \frac{FOV_v}{r_v} \end{aligned} \tag{3.2.1}$$

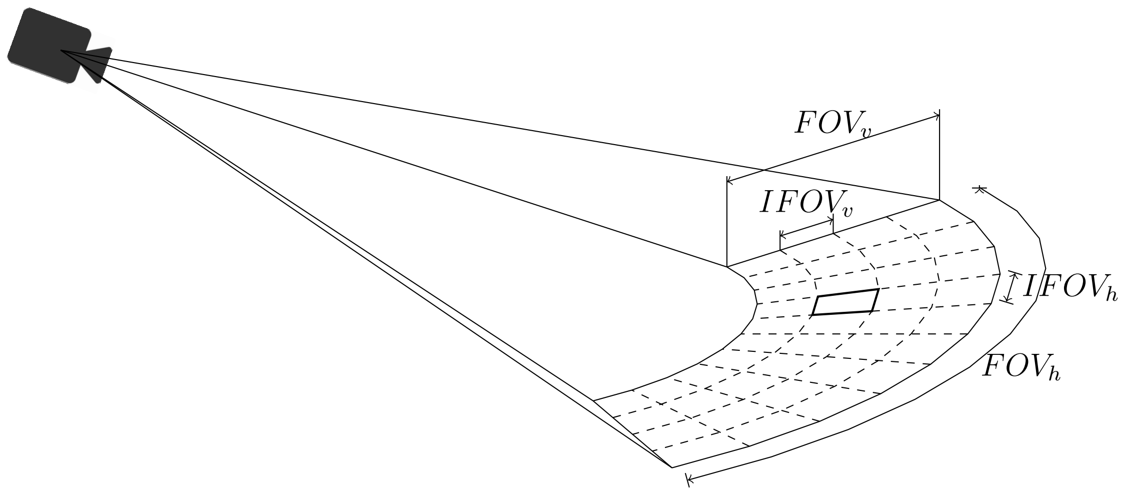


Figure 3.3: A visual representation of IFOV, FOV, and sensor resolution

Using Figure 3.3, The approximations of the dismounts from Figure 3.2 are then placed at the position of each pixel. The coordinates are used to determine the max size of the detection if the dismount was any given position. This is used to create a heat map of the maximum threshold. Similarly, the minimum dismount size is used to generate a map of the minimum threshold. These two maps will be used in section 3.3 to help drive the small target detector.

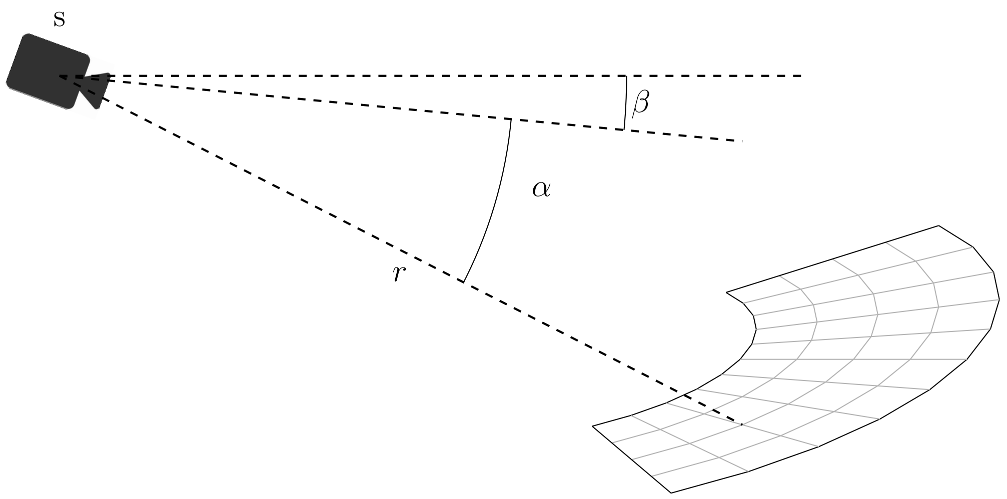


Figure 3.4: Spherical coordinates from the frame of reference of sensor S

In order to do this, spherical coordinates (r, α, β) are first used to identify the location of the projection of each pixel onto the ground plane. Where r is the radius, α is the depression angle and, β is the angle from the heading.

$$r = \frac{s_a}{-\sin(\alpha)} \tag{3.2.2}$$

$$p_{i,j} = (r, \alpha_i, \beta_j) \tag{3.2.3}$$

$$\alpha_{i,j} = i \times IFOV_v + \alpha, i \in Z, \frac{-r_v}{2} < i < \frac{r_v}{2} \tag{3.2.4}$$

$$\beta_{i,j} = i \times IFOV_h + \beta, i \in Z, \frac{-r_h}{2} < j < \frac{r_h}{2} \tag{3.2.5}$$

$$r_{i,j} = r \times \sin(\alpha_i) = \frac{s_a}{\sin(\alpha)} \times \sin(\alpha_i) \tag{3.2.6}$$

the location of each pixel $p_{i,j}$ is converted to cartesian coordinates to determine the boundary points d_i of an approximated dismount based on their w_{min} and h_{min}.

$$d_{x,k} = r_{i,j} \cos(\alpha_i) \cos(\beta_j) \pm \frac{w_{min}}{2} \tag{3.2.7}$$

$$d_{y,k} = r_{i,j} \cos(\alpha_i) \sin(\beta_j) \pm \frac{w_{min}}{2} \tag{3.2.8}$$

$$d_{z,k} = r_{i,j} \sin(\alpha_i) + \frac{h_{min}}{2} \pm \frac{h_{min}}{2} \tag{3.2.9}$$

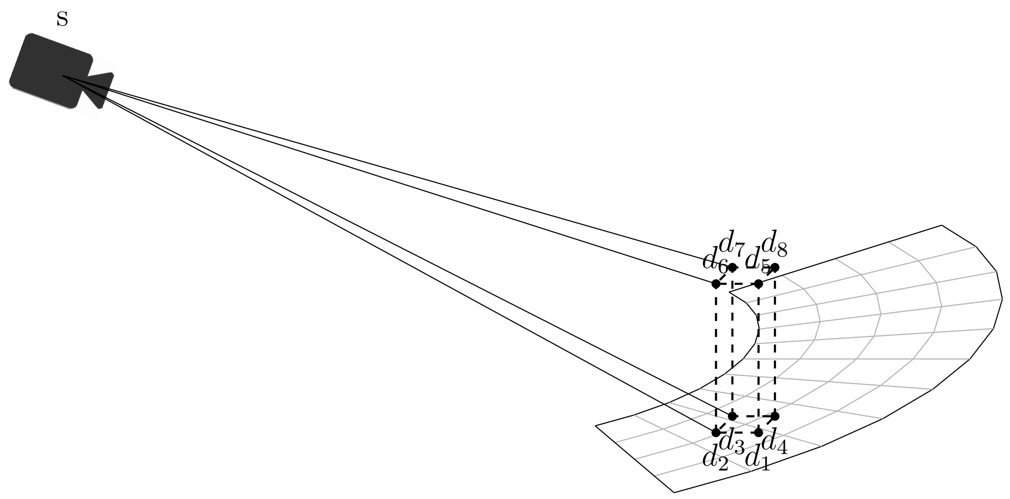


Figure 3.5: Dismount is placed at ground level in the line of sight of every pixel.

The positions of d_k are converted to polar coordinates with the sensor, s, at the origin. Where k is the number of coordinates used to approximate the dimensions of the dismount. In this case $1 \leq k \leq 8$, the coordinates could be represented as:

$$d_{\alpha,k} = \sin^{-1}\left(\frac{d_{z,k} - s_z}{r_{i,j}}\right) \tag{3.2.10}$$

$$d_{\beta,k} = \cos^{-1}\left(\frac{d_{z,k} - s_z}{r_{i,j}cos(d_{\alpha,k})}\right) \tag{3.2.11}$$

The maximum difference between angles is used to determine how large the dismount would appear if it were to stand at the position of that pixel.

$$dismount\_max\_width = max\left(\frac{d_{\beta,k} - d_{\beta,k'}}{IFOV_v}\right) \tag{3.2.12}$$

$$dismount\_max\_height = max\left(\frac{d_{\alpha,k} - d_{\alpha,k'}}{IFOV_v}\right) \tag{3.2.13}$$

The minimum size predictions of the target are done in a similar manner using w_{min} and h_{min} in place of w_{max} and h_{max}. This results in 4 values for each pixel, $dismount\_max\_height$, $dismount\_max\_width$, $dismount\_min\_height$, $dismount\_min\_width$ that drive the small target detector described in section 3.3.

3.3 Dynamic Small Target Detector

The small target detector used is similar to the one mentioned in section 2.1. In order to simplify equation 2.1.1, the random noise in the background is ignored. This proposed relaxation of the cost function helps eliminate the need to compute the weights w_i. Furthermore, the regularization parameter *epsilon* is set to 1 to eliminate another term. The above changes are designed to simplify the filter and reduce the computation requirements.

$$f = f_T + f_B + n \tag{3.3.1}$$

Where f, f_T, f_B, and n are the image, target image, and background image, respectively. The moving window around the active area was initially assumed to be a fixed-sized square. Given the predictions computed in section 3.2, the dimensions of the inner and outer windows vary based on the size of the prediction.

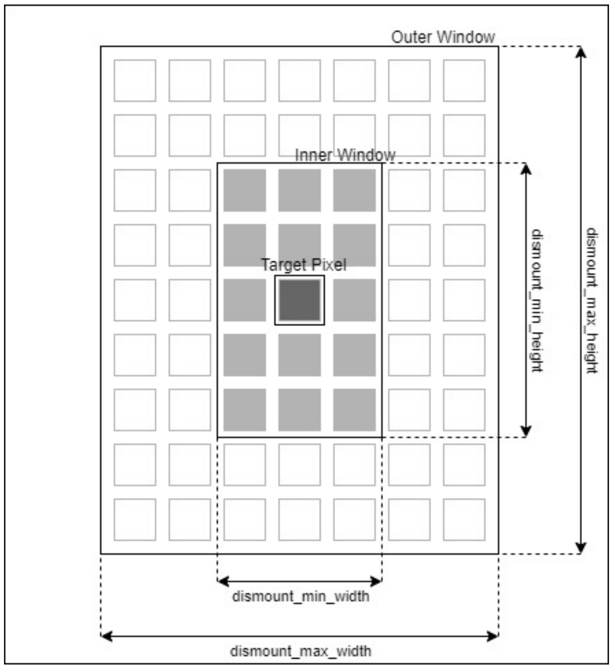


Figure 3.6: A visual representation of the updated sliding window. The white color represents pixels in the active area, the light grey color represents pixels inside the inner window, dark grey color is the center pixel.

The active area, marked in grey, is denoted as $\Omega = \{1, 2, ..., n\}$ where n is the total number of pixels in the active area. The intensity of each pixel $i \in \Omega$ is denoted by x_i. The intensity of each background image pixel $i \in \Omega$ is denoted by y_i. The background, b, and an arbitrary scale a can now be modeled as.

$$\begin{aligned} y_i &= a * x_i - b \\ &= \begin{bmatrix} x_i & 1 \end{bmatrix} * \begin{bmatrix} a \\ b \end{bmatrix} \end{aligned} \tag{3.3.2}$$

Using this we get the following model

$$\min_{\begin{bmatrix} a & b \end{bmatrix}^T} (y_i - \begin{bmatrix} x_i & 1 \end{bmatrix} * \begin{bmatrix} a \\ b \end{bmatrix})^2 \tag{3.3.3}$$

The term a^2 is added to the cost function, to penalize very high values of a.

$$\min_{\begin{bmatrix} a & b \end{bmatrix}^T} (y_i - \begin{bmatrix} x_i & 1 \end{bmatrix} * \begin{bmatrix} a \\ b \end{bmatrix})^2 + a^2 \tag{3.3.4}$$

Let i be the total number of pixels contained between the inner and outer window. Let x_i and y_i equal the intensity of the image pixel and the intensity of the target pixel respectively. These can be used to create the following matrices.

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_i \\ 0 \end{pmatrix} \tag{3.3.5}$$

$$X = \begin{pmatrix} x_1 & x_2 & x_3 & \cdots & x_i & 1 \\ 1 & 1 & 1 & \cdots & 1 & 0 \end{pmatrix} \tag{3.3.6}$$

Unless a target-less image of the current image is available, y_i is not available. This makes it impossible to solve the equation. To overcome this, we estimate that the pixels in the outer window are the same as the background ($y_i = x_i$). This changes matrix Y as follows:

$$Y = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ 0 \end{pmatrix} \tag{3.3.7}$$

Variables a and b can now easily be computed. The computed values can be plugged into the following equation where y_0 and x_0 are the pixel intensity values of the center pixel output and input respectively of the sliding window. This will create an optimized image detecting small targets.

$$y_0 = (x_0 - \begin{bmatrix} x_0 & 1 \end{bmatrix} * \begin{bmatrix} a \\ b \end{bmatrix})^2 \tag{3.3.8}$$

The output image with the pixel values y_0, is then thresholded to find detections. Section 3.5 explores how the optimal threshold values can be learned using Stochastic Gradient Descent.

3.4 Temporal Detector Using Optical Flow

In the MMIS framework, Zhu *et al.* (2014) compared the use of numerous dense optical flow algorithms. Zhu *et al.* (2014) Specifically tested HOG, Laplacian of Gaussian (LoG), Frame-2-Frame FLow, Intrinsic Dimensionality (ID), Structure Saliency (SS), and Defusion Maps. Their testing showed that the best results came from the Superpixel LoG MMIS algorithm.

FM-MIMS proposes the use of a simple optical flow algorithm to estimate the flow of the video using the current input image I_t and the prior image I_{t-1}. Flow is estimated using the Horn-Schunck method a/l Kanawathi *et al.* (2009). This is the first method used for optical flow estimation. "It has introduced a new framework whereby it is regarded as a solution for issues in reducing the pixel intensity in a certain image." (Ramli *et al.*, 2016). The FM-MIMS algorithm used the Horn-Schunck method for its robustness and the global constraint for smoothness. This should allow for smooth motion profiles that work well with tensor voting.

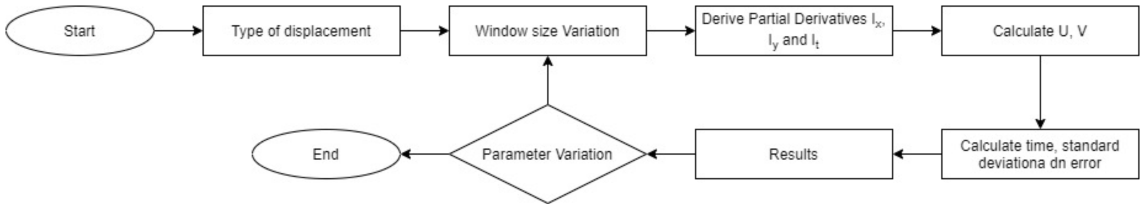


Figure 3.7: The Horn Schunck Algorithm

This global method introduces a global constraint of smoothness over the computed flow field. The energy function used is

$$E(w) = \iint_{\omega} \left((f_x u + f_y v + f_t)^2 + \alpha \left(|\nabla u|^2 + |\nabla v|^2 \right) \right) dx\, dy \qquad (3.4.1)$$

where f_x and f_y are partial derivatives of f with respect to x and y respectively and $|\nabla u| = \sqrt{u_x^2 + u_y^2}$. u, v are the optical flow vectors defined as dx/dt and dx/dt respectively. The equation is based on three assumptions. The first is assuming that all the features in the image consistently have the same grey value. Secondly that the computed flow field has to be consistently smooth. Thirdly it is assumed that neighboring points have almost the same velocity. The optical flow varies smoothly. This corresponds to the isotropic regularizer Er defined as

$$Er(u,v) = \frac{1}{2}\left(|\nabla u|^2 + |\nabla v|^2\right) \tag{3.4.2}$$

This can help for the following partial differential equations (PDE)

$$\alpha \Delta u - I_x(I_x u + I_y v + I_t) = 0 \tag{3.4.3}$$

$$\alpha \Delta u - I_y(I_x u + I_y v + I_t) = 0 \tag{3.4.4}$$

The Horn-Schunck method uses the block Gauss-Seidel relaxation

$$u^{k+1} = u^{-k} - I_x \frac{I_x u^{-k} + I_y u^{-k} I_t}{\alpha + I_x + I_y} \tag{3.4.5}$$

$$v^{k+1} = v^{-k} - I_y \frac{I_x u^{-k} + I_y u^{-k} I_t}{\alpha + I_x + I_y} \tag{3.4.6}$$

where k, $\bar{u}$ and, $\bar{v}$ are the iteration counter, an average of neighboring points to u and an average of neighboring points to v respectively. I_x, I_y, I_t is defined as

$$I_x = \frac{\begin{array}{r}(P(0)_{x+1,y} + P(1)_{x+1,y} + P(0)_{x+1,y+1} + P(1)_{x+1,y+1}) - \\ (P(0)_{x,y} + P(1)_{x,y} + P(0)_{x,y+1} + P(1)_{x,y+1})\end{array}}{8} \tag{3.4.7}$$

$$I_y = \frac{\begin{array}{r}(P(0)_{x,y+1} + P(1)_{x,y+1} + P(0)_{x+1,y+1} + P(1)_{x+1,y+1}) - \\ (P(0)_{x,y} + P(1)_{x,y} + P(0)_{x+1,y} + P(1)_{x+1,y})\end{array}}{8} \tag{3.4.8}$$

$$I_t = (P(1)_{x,y} + P(1)_{x+1,y} + P(1)_{x,y+1} + P(1)_{x,y+1}) - \frac{(P(0)_{x,y} + P(0)_{x+1,y} + P(0)_{x,y} + P(0)_{x,y})}{8} \tag{3.4.9}$$

u^{-k} and v^{-k} can be determined by using

$$u^{-k} = \frac{u_{x-1,y} + u_{x,y-1} + u_{x+1,y} + u_{y+1}}{2} + \frac{u_{x-1,y-1} + u_{x-1,y+1} + u_{x+1,y-1} + u_{x+1,y+1}}{4} \tag{3.4.10}$$

$$v^{-k} = \frac{v_{x-1,y} + v_{x,y-1} + v_{x+1,y} + v_{y+1}}{2} + \frac{v_{x-1,y-1} + v_{x-1,y+1} + v_{x+1,y-1} + v_{x+1,y+1}}{4} \tag{3.4.11}$$

In the above equations, the number of iterations k and the assumed global smoothness are used to optimize the optical flow estimation. “The number of iteration is adjusted to gain knowledge on how much loop is needed for simulation. Meanwhile the smoothing coefficient acts as lubricant, where in every iteration, it will smooth out to produce the best motion of optical flow.” (a/l Kanawathi *et al.*, 2009)

3.5 Threshold Optimisation

The outputs from both the small target detector (section 3.3) and the temporal detector (section 3.4) are thresholded to eliminate lower intensity values. This addition to the FM-MIMS attempts to find the optimal threshold for either output is one that maximizes the number of detections and minimizes the number of false alarms. This is done by using stochastic gradient descent.

The function for PFA and PD curve given a threshold can be represented by the cumulative distribution function (CDF) $f(x)$.

$$\hat{y} = w_1 + w_2 f(x - w_1) \tag{3.5.1}$$

Where $\hat{y}$ is the expected value at the threshold x and w_i are the different weights $1 \leq i \leq 3$. The root mean squared error (RMSE) is used as the cost function

$$error = \sqrt{\frac{1}{N}\sum^{N}(\hat{y} - y)} \tag{3.5.2}$$

Once the CDF and PD functions are mapped using gradient descent the threshold is can be calculated by using the global minima of their sum.

$$threshold = \min(CDF + PD) \tag{3.5.3}$$

This process is separately conducted to find the threshold for the output of the small target detector(section 3.3) as well as the temporal detector (section 3.4).

3.6 Multidimensional Tensor Voting

The process for multidimensional tensor voting is outlined in section 2.2. The drawback of this algorithm is that as the size of the tensor grows, the algorithm requires exponentially more computing power. This becomes a concern as more descriptors are used for each pixel of data. This section addresses the concern in the MIMS algorithm by using a Monte Carlo Integration in place of the integrals used in equation 2.2.9. The single summation of randomly oriented vectors proposed by equation 3.6.5 attempts to eliminate the multi integral calculation in the FM-MIMS. The integration can become exponentially taxing as the dimensions of the tensors used in tensor voting increase.

Similar to section 2.2 the input data's expectation over time $\begin{bmatrix} t_1, t_2, \cdots t_n \end{bmatrix}$ in encoded into the tensor

$$T = \begin{bmatrix} t_1 \times t_1 & t_1 \times t_2 & \cdots & t_1 \times t_n \\ t_2 \times t_1 & t_2 \times t_2 & \cdots & t_2 \times t_n \\ \vdots & \vdots & \ddots & \vdots \\ t_n \times t_1 & t_n \times t_2 & \cdots & t_n \times t_n \end{bmatrix} \tag{3.6.1}$$

This can be decomposed into the eigensystem where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$

$$T = \begin{bmatrix} \hat{e}_1 & \hat{e}_2 & \dots & \hat{e}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \begin{bmatrix} \hat{e}_1^T \\ \hat{e}_2^T \\ \vdots \\ \hat{e}_n^T \end{bmatrix} \tag{3.6.2}$$

$$T = \sum_{i=1}^{n-1} s_n v_n + \lambda_n s_n$$

where the normal vector is $v_n = \sum_{i=1}^{n} \hat{e}_i \hat{e}_i^T$ and $s_n = \lambda_{n-1} - \lambda_n$. The voting field and decay function for the tensors are the same as defined in section 2.2.

$$DF(\gamma, \phi, \sigma) = \begin{cases} e^{(\frac{\gamma^2 + c\phi^2}{\sigma^2})}, & \theta \in [\frac{-\pi}{4}, \frac{\pi}{4}]. \\ 0, & otherwise. \end{cases} \tag{3.6.3}$$

Where $c = \frac{(\sigma-1)(-\log 0.1)}{\pi^2/4}$. The stick vote is then calculated by

$$V_{stick}(\vec{N^V}) = DF(\gamma, \phi, \sigma) \times \vec{N^V} \vec{N^V}^T \tag{3.6.4}$$

The conventional multidimensional voting process is where the computational complexity increases. This is due to the compounded integrals. Equation 2.2.9 can be replaced with a Monte-Carlo integration approach, where we assume the probability of each point $p(x)$ has a uniform distribution.

$$V = \frac{1}{N}\sum_{i=1}^{N} R_i V_{Stick}(\vec{T^V}) R_i^T \tag{3.6.5}$$

Where R_i is a part of N randomly selected rotation matrices along the planes defined by table 2.1. Once this is complete the tensor voting can continue as outlined in section 2.2 and 2.2.4.

$$T_1^R = T_0^R + \sum_i T_{Ball}^{V_i} + \ldots + T_{Plate}^{V_i} + T_{stick}^{V_i} \tag{3.6.6}$$

3.7 MMIS

The MIMS algorithm defined in section 2.4 is used. The free parameters σ is varied to acquire multiple saliency, $s_i \in \mathbb{R}^+$, and dimensionality, $d_i \in \mathbb{N}^+, 1 \leq d_i < N$, values.

$$\{(d_i, s_i, \sigma_i), i = 1, 2, \ldots, k\} \tag{3.7.1}$$

The value at a certain scale can be represented as $f(X, \sigma) = (d, \tilde{s})$ where S is normalized, between $[0, 1]$ and k bins are chosen. $f(X, \sigma)$ is used as a 2D indices of a 2D lookup table.

$$W_f^c = P(f(X, \sigma) \in bin_f, y = c), c = \pm 1 \tag{3.7.2}$$

where W_f^c encodes the 2D distribution of features amongst both the positive and negative training samples.

3.8 AdaBoost

Similar to the method proposed by Zhu *et al.* (2014), the algorithm initialized by setting the number of iterations T that AdaBoost will run and the weight factor w_i^t, where t is the current iteration, is initialized to w_i^0 with a uniform distribution.

$$w_i^0 = D(i) = \frac{1}{N} \tag{3.8.1}$$

The first classifier is chosen based on which input has the largest impact on the outcome. w_i^t is normalized.

$$p^t = \frac{w^t}{\sum_{i=1}^{T} w_i^t} \tag{3.8.2}$$

Next a weak classifier (stump) $h_t : X \to [0, 1]$ is used to classify based on each individual input. The function $h_t(x)$ is defined such that

$$h_t(X, \sigma) = \frac{1}{2} \ln \left(\frac{W_F^{+1} + \epsilon}{W_F^{-1} + \epsilon} \right) \tag{3.8.3}$$

where $f = f(X, \sigma)$, W_F^{+1} and W_F^{-1} is defined by equation 3.7.2.

The efficiency of each potential h_t is computed using the Gini impurity number

$$G(h_t) = \sum_{i=1}^{N} P(h_t, i)(1 - P(h_t, i)) \tag{3.8.4}$$

where $P(h_t, i)$ is the probability of a certain classification i. The $h_t(x_i)$ with the lowest Gini impurity gets selected to be the first stump in the stump forest. The error is the sum of the weights of all the incorrectly classified data points.

$$E(h_t) = \sum_{i=1}^{N} w_{i,incorrect} \tag{3.8.5}$$

The output weight of the stump α_t can now be computed

$$\alpha_t = \frac{1}{2} \log \left(\frac{1 - E(h_t)}{E(h_t)} \right) \tag{3.8.6}$$

$$w_i^t = \begin{cases} w_i^{t-1} e^{-\alpha_t}, & \text{correctly classified.} \\ w_i^{t-1} e^{\alpha_t}, & \text{incorrectly classified.} \end{cases} \tag{3.8.7}$$

After T iterations of this process, the final hypothesis can be outputted.

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{3.8.8}$$

Where $H(x)$ is the certainty of one particular outcome.

3.9 Summary

This section discussed in detail the contributions of new components and modifications to the MMIS framework that are needed to develop the FM-MIMS framework. A new Dismount size predictor was developed to influence metadata-dependent tuning parameters for the addition of the small target. The Small target detector suggests using a more relaxed optimization method relative to the original detector developed by Xie *et al.* (2015). The temporal detector in the MIMS framework was never defined. The FM-MIMS algorithm proposes the use of the Horn-Schunck method to observe the flow of pixel intensity through the frame. The final contribution to the FM-MIMS is a modification to the original tensor voting algorithm. Here the integration step was replaced with a Monte Carlo integration to reduce computation time and the mathematical complexity of working with large vectors. These modifications to the original MIMS framework allow for FM-MIMS to be a theoretically more reliant dismount detection algorithm.

Chapter 4

Testing, Results, and Discussion

Since the FM-MIMS algorithm was designed as an improvement to the MMIS Detection algorithm, the newly developed algorithm was tested against the original MMIS detection. In MIMS, Zhu *et al.* (2014) established that their technique was able to outperform HOG, Laplacian of Gaussian (LoG), Frame-2-Frame Flow, Intrinsic Dimensionality (ID), Structure Saliency (SS), and Diffusion Mapping. Thus if the FM-MIMS algorithm can outperform MMIS detection, we can assume it would also outperform HOG, Laplacian of Gaussian (LoG), Frame-2-Frame Flow, Intrinsic Dimensionality (ID), Structure Saliency (SS), and Diffusion Mapping.

The following results are based on data collected from a WFOV camera shooting video from an aircraft. The camera has a resolution of 1920×1080 pixels and is shooting at a high frame rate. 100 frames of the Video footage were used to train all data-driven models and a subsequent 25 frames were used for testing.

4.1 Prediction

The prediction module was relatively simple as its purpose was to execute simple trigonometric functions on each pixel of the image. The following predictions were made using flight optometry. The approximate altitude of the camera was 3000m with a depression angle of approximately -30 degrees. The dimensions of an average dismount were determined by MedicineNet (Ratini (2019)). The maximum and minimum height of the dismounts used were 1.8288m and 1.9144m respectively. The maximum and minimum width of the dismounts used were 0.6096 and 0.3048 respectively.

4.1.1 Height Predictions

Figure 4.1: A visual representation of the maximum height (in pixels) predictions generated. The values ranged from 27.9299(black) to 29.0655(white)

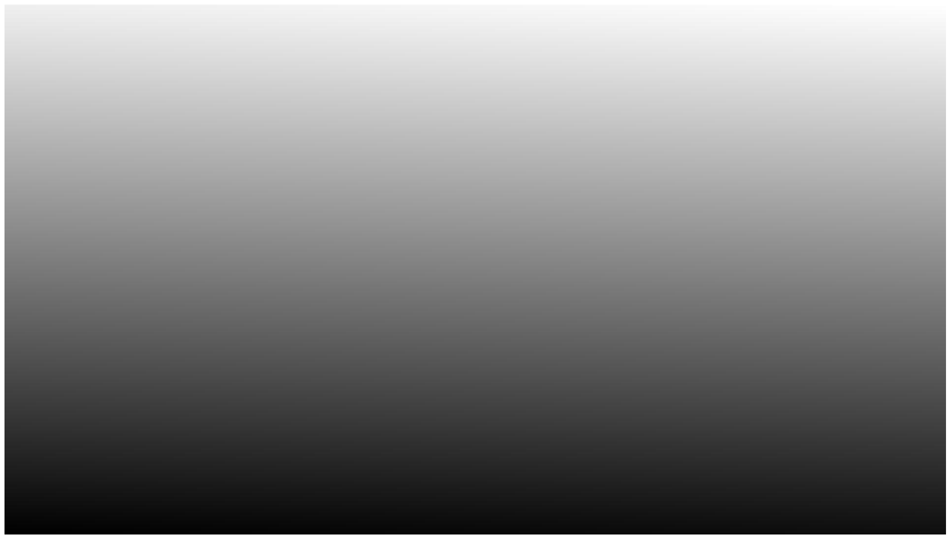


Figure 4.2: A visual representation of the minimum height (in pixels) predictions generated. The values ranged from 27.1559(black) to 28.1186(white)

4.1.2 Width Predictions

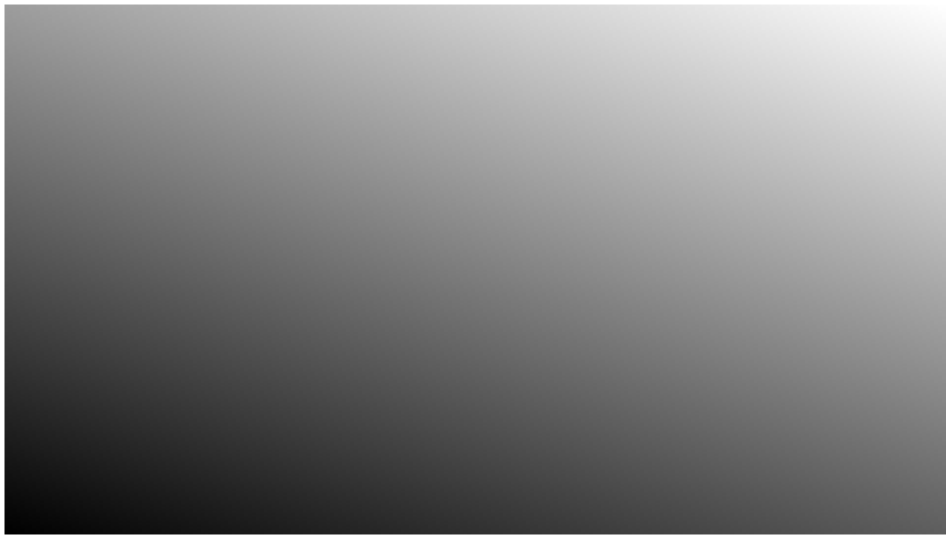


Figure 4.3: A visual representation of the maximum width (in pixels) predictions generated. The values ranged from 10.0912(black) to 10.8617(white)

Figure 4.4: A visual representation of the minimum width (in pixels) predictions generated. The values ranged from 5.0456(black) to 5.4308(white)

4.1.3 Future Improvements

Given the above dimensions, there is very little variance in the dimensions of a dismount across the frame. Although knowing the dimensions is important, computing them at each pixel was an inefficient approach. The variance for the sizes may increase at sensor depression angles closer to the horizon (-5 to -15 degrees) but as that is out of the scope optimizations can be made by calculating one prediction at the center of the frame.

4.2 Small Target Detection

The small target detector was very effective in picking out spots of the sizes defined in section 4.1. Some examples of accurately detected dismounts can be seen in Figure 4.6. The drawback of this algorithm is that given the scale of the objects in the environment it also picks out every spot that is approximately the same size (see Figure 4.7). The focus of the FM-MIMS algorithm at this stage was to maintain all of the detections and filter out the clutter in subsequent modules.

In figure 4.5, the probability of detection is 1, and the probability of false alarms is 0.0382.

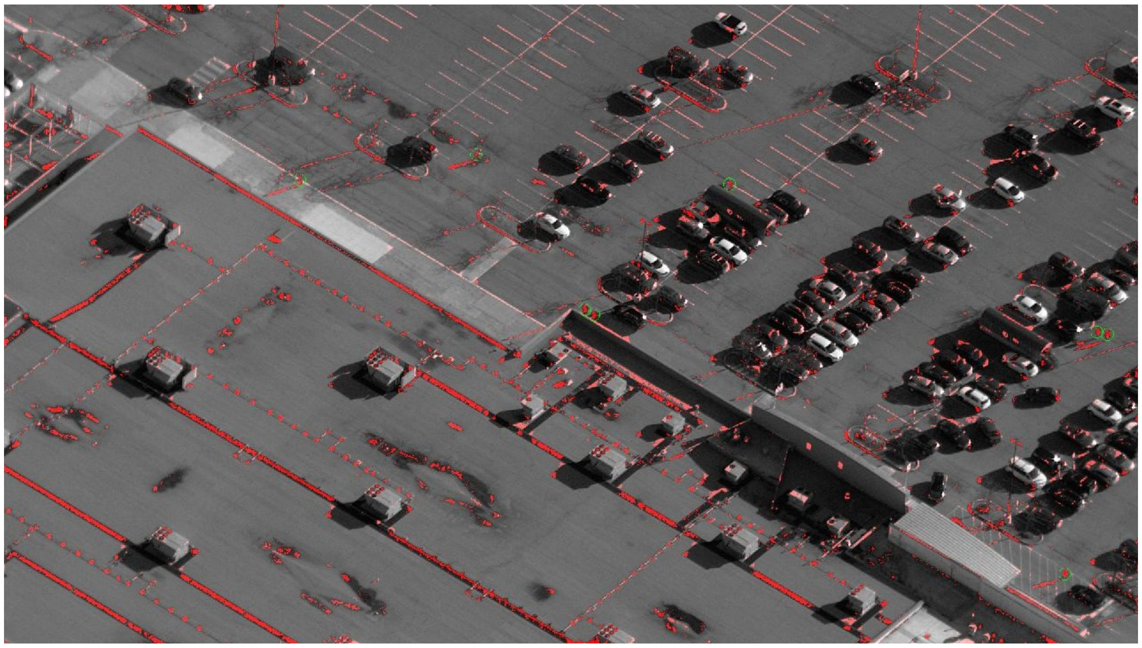
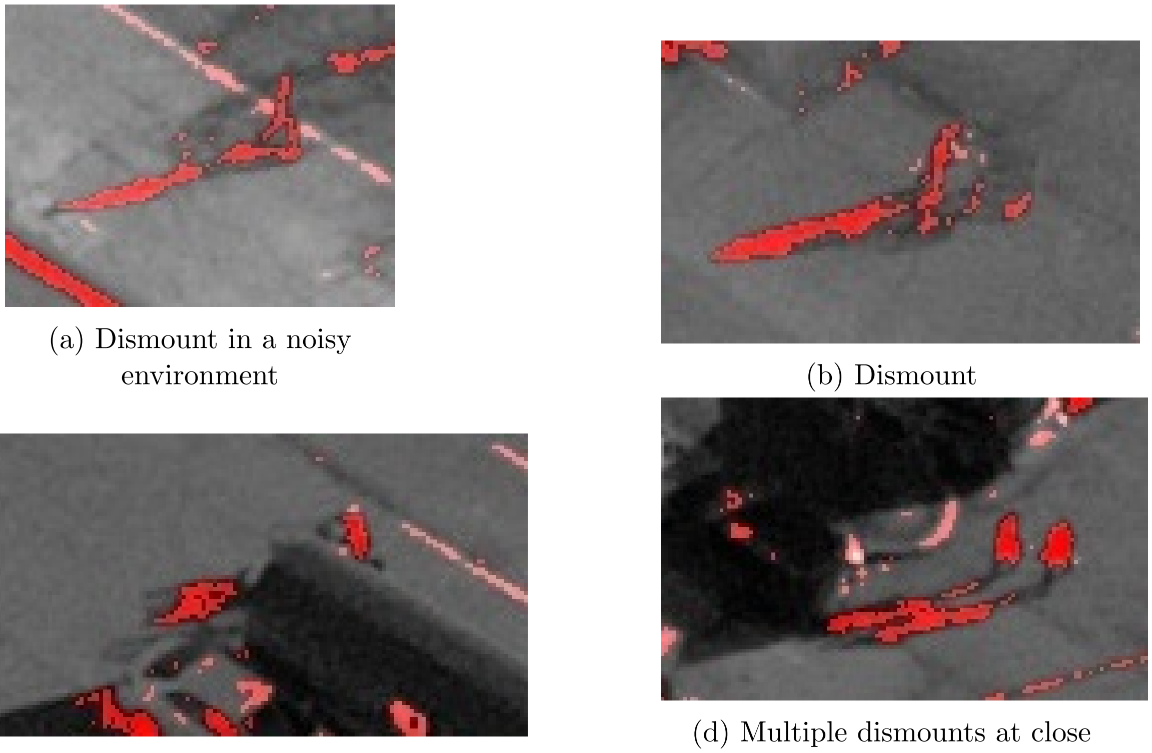


Figure 4.5: A visual representation of the small target detector. The detections identified by the algorithm are highlighted in red. The location of all accurately detected dismounts is marked with a green circle.


(a) Dismount in a noisy environment

(b) Dismount

(c) Occluded dismount

(d) Multiple dismounts at close proximity

Figure 4.6: Examples of dismounts found in the scene

Figure 4.6 shows the various types of dismounts that can be found in the scene. The algorithm can accurately identify and highlight them. This included targets in noisy environments, partially occluded targets, and multiple nearby targets.

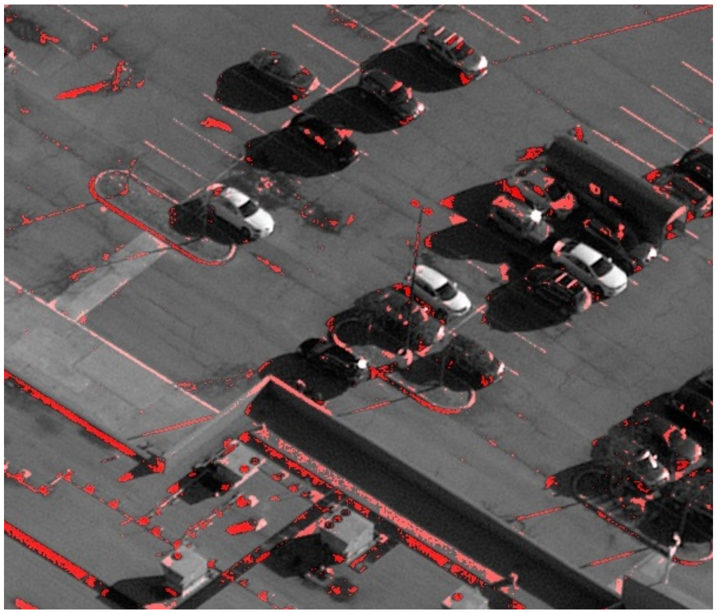


Figure 4.7: Example of car windows, lamp posts, and other environmental clutter detected.

This filter on its own is not adequate to satisfy the requirements. Figure 4.7 highlights the many instances of random objects that are also highlighted. It is important to note here that a lot of the noise is caused due to straight edges where the texture or intensity of the pixel rapidly changes. The clutter is also caused by static objects such as parked cars, lamp posts, and objects on the roof. This source of clutter is addressed by the temporal filter.

4.3 Temporal Detection

The temporal filter uses optical flow to identify moving objects in the frame. In this particular scene, the dismounts are not the only moving objects. There are also cars, shadows and because of the constant change in perspective, the edges of the building. This filter, however, is not ideal for detecting dismounts especially those that may be stationary. The algorithm relies on them being in constant motion and so does an adequate job with those that are walking.

In figure 4.8, the probability of detection is 0.6250, and the probability of false alarms is 0.0144.

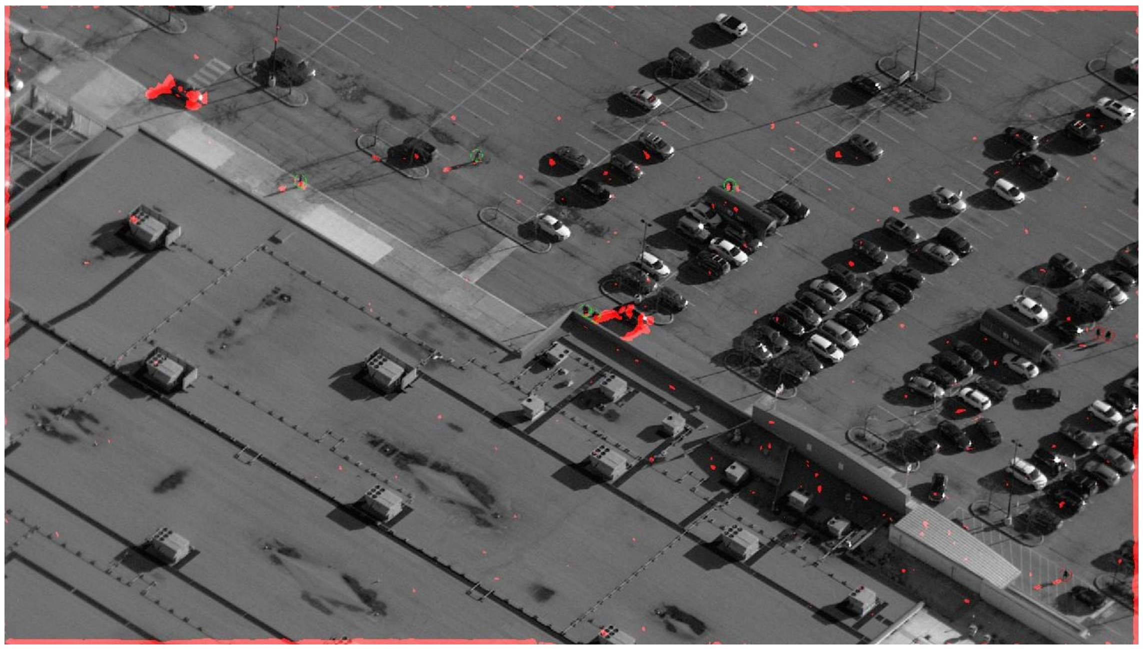
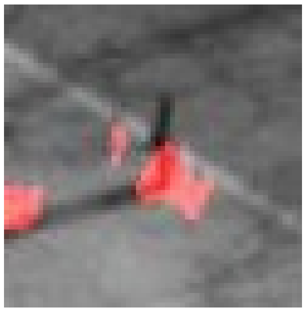
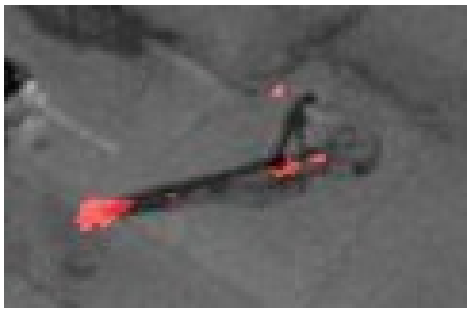
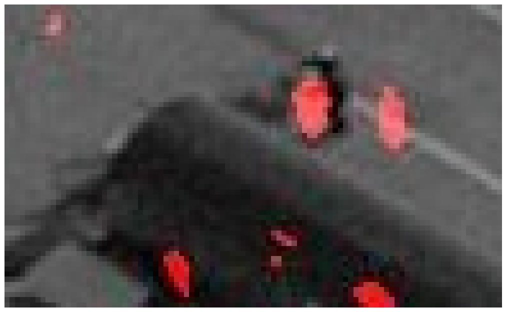


Figure 4.8: A visual representation of the optical flow detector. The detections identified by the algorithm are highlighted in red. The location of all accurately detected dismounts is marked with a green circle.


(a) Dismount in a noisy environment


(b) Dismount


(c) Occluded dismount

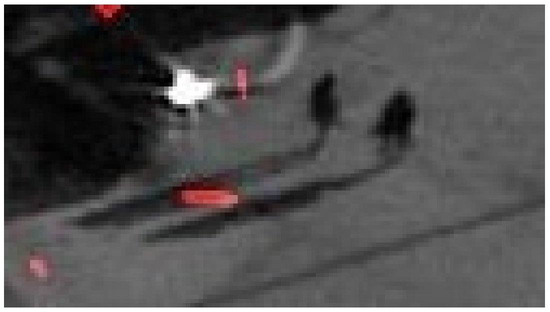

(d) Multiple dismounts at close proximity

Figure 4.9: Examples of dismounts found in the scene

Figure 4.9 shows the same dismounts as Figure 4.6. The areas registering as detection are different from those indicated by the spatial filter. This algorithm takes advantage of the arms and legs that swing back and forth when the dismounts walk. In Figure 4.9d, both people do not appear to be moving at all and are therefore not detected.

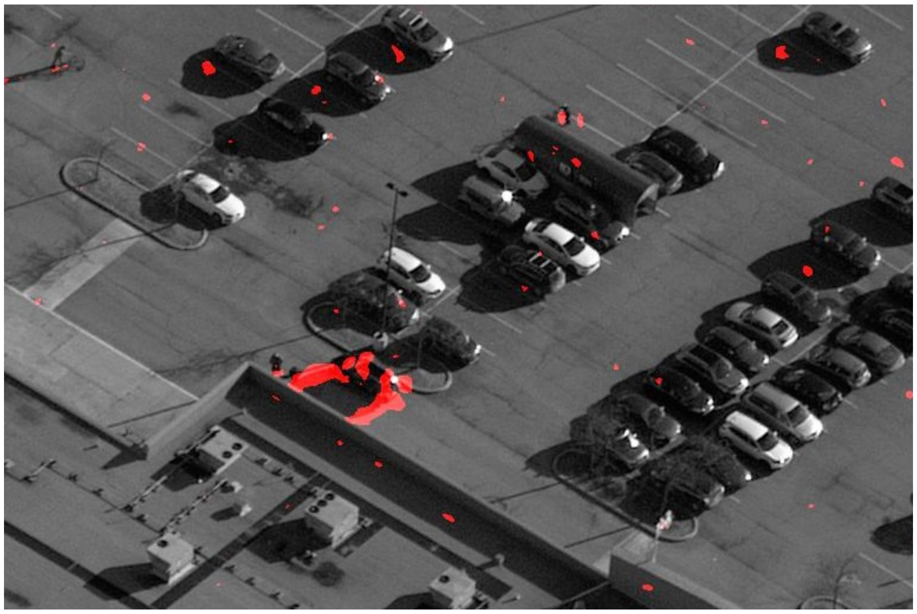


Figure 4.10: Example of car windows, lamp posts, and other environmental clutter detected.

An important observation to note is that this algorithm yields a 60% lower probability of false alarm relative to the small target detector. The false alarms are seen in Figure 4.10 are largely around the moving car and along with the shadows of parked cars.

4.3.1 Next Steps

Due to the time and computation resources available, the efficacy of the Horn-Schunck method relative to the methods (HOG, Laplacian of Gaussian (LoG), Frame-2-Frame Flow, Intrinsic Dimensionality (ID), Structure Saliency (SS) and Defusion Maps) used by Zhu *et al.* (2014) could not be tested. Time permitting all the methods would be tested and compared over 100 frames of data to identify which was able to yield the highest probability of detection while minimizing the probability of false alarms.

A key factor in the delays caused by running the Horn-Schunck method was the execution time of the algorithm. This could be drastically reduced by using an implementation of the algorithm that uses the GPU better. Furthermore, when testing against other dense optical flow algorithms, the average processing time per frame should also be compared.

4.4 Thresholding

This simple Linear regression-based threshold estimation problem proved very effective in estimating the appropriate threshold values for the small target and, temporal detector. By using stochastic gradient descent to model the probability of detection and the probability of false alarm curve, thresholds were easy to compute and could potentially be updated as tracks were confirmed.

In the tests conducted stochastic gradient descent was used to calculate the thresholds based on 100 frames of training data. the computed thresholds were then used in the FM-MIMS algorithm.

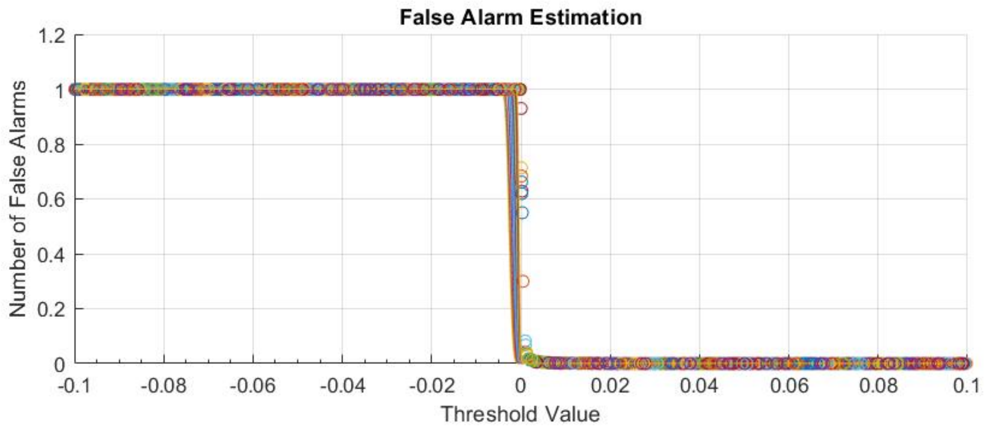


Figure 4.11: Fitting the false alarm curve

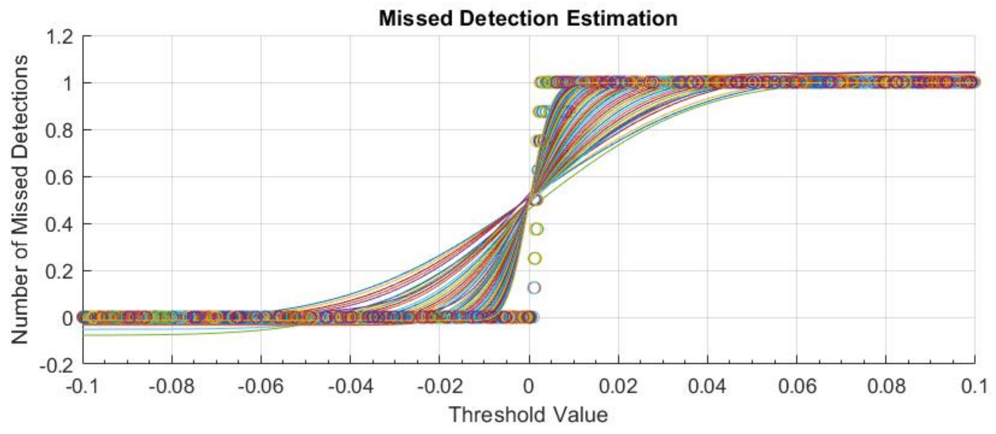


Figure 4.12: Fitting the probability of missed detection curve

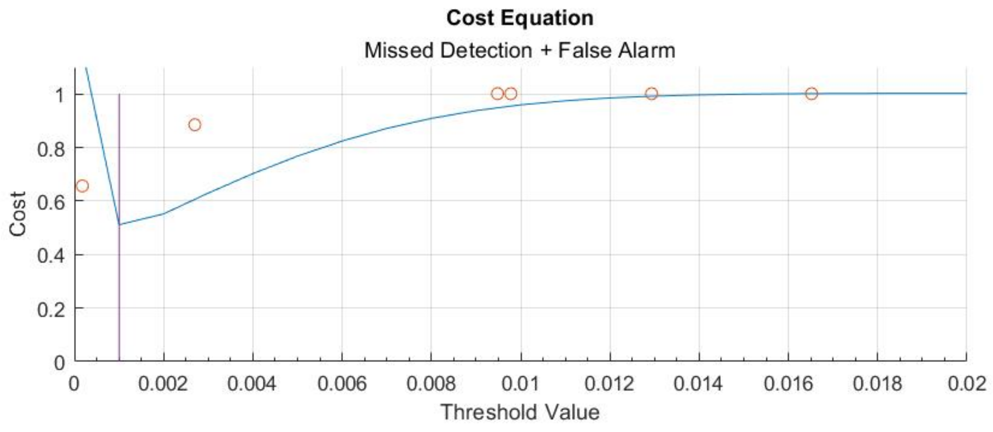


Figure 4.13: Combining the probability of false alarm curve and probability of missed detection curve to find the minimal cost. The minima of the cost equation is marked by the vertical purple line at threshold=0.001

4.4.1 Future Improvements

The next step for this module would be to allow confirmed tracks to influence the validation process. This feedback look would allow the threshold to adapt to changing environmental conditions. The greatest risk with this method of unsupervised learning is that the model learns to set the threshold to 0 to allow for the maximum number of defections. A bias would have to be added to equation 3.5.3 to prevent the algorithm from drifting towards unreasonably low, or high, values.

4.5 Tensor Voting

The tensor voting module was the most complex part of this framework. It required a significant amount of vector integration and trigonometry for each pixel in the frame. The motivation for all prior modules and modifications to the tensor voting algorithm was to relax the amount of computation required by the algorithm.

4.5.1 Development Constraints

While developing the multidimensional tensor voting in MatLab, we encounter difficulties with executing the algorithm in a meaningful amount of time. Execution times for Multidimensional Tensor voting were as high as 40 seconds/frame. Where the size of the frame was 200×200 pixels and a tensor of length 3 was used. This caused significant delays in modeling and testing the FM-MIMS algorithm. If this algorithm was used with MIMS, the same computation would have to be performed at least 10 times at varying scales. This resulted in an execution time of approximately 400 seconds/frame. Not only does this substantially increase the execution time but it also blocks almost all of the resources on the machine running it. Given the number of vectors and summations required, 100% of the computer's processors and memory would be utilized. The tensor voting algorithm as proposed was too demanding on computer resources.

4.5.2 Relaxation Of Requirements

In order to simplify the task, a smaller 2-dimensional tensor voting algorithm was used where the input image was the product of a small target detector (section 3.3) and the temporal detector (section 3.4). The lower dimensionality of the tensor reduced the computation time to approximately 20 seconds/frame for a 1920×1080 pixels frame. This simplification stresses how the computational load to process the algorithm exponentially grows relative to the size of the tensor.

4.5.3 Results

Below are sample outputs of the tensor voting algorithm run on the same frame with a scale of 10.

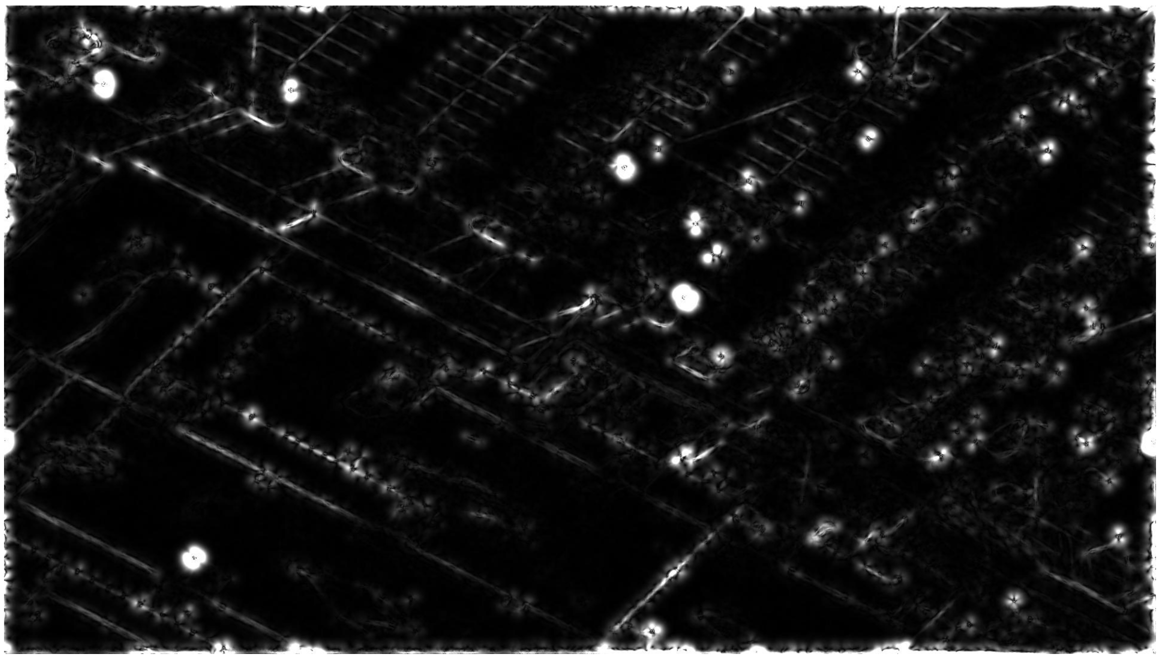


Figure 4.14: Saliency of FM-MIMS algorithm

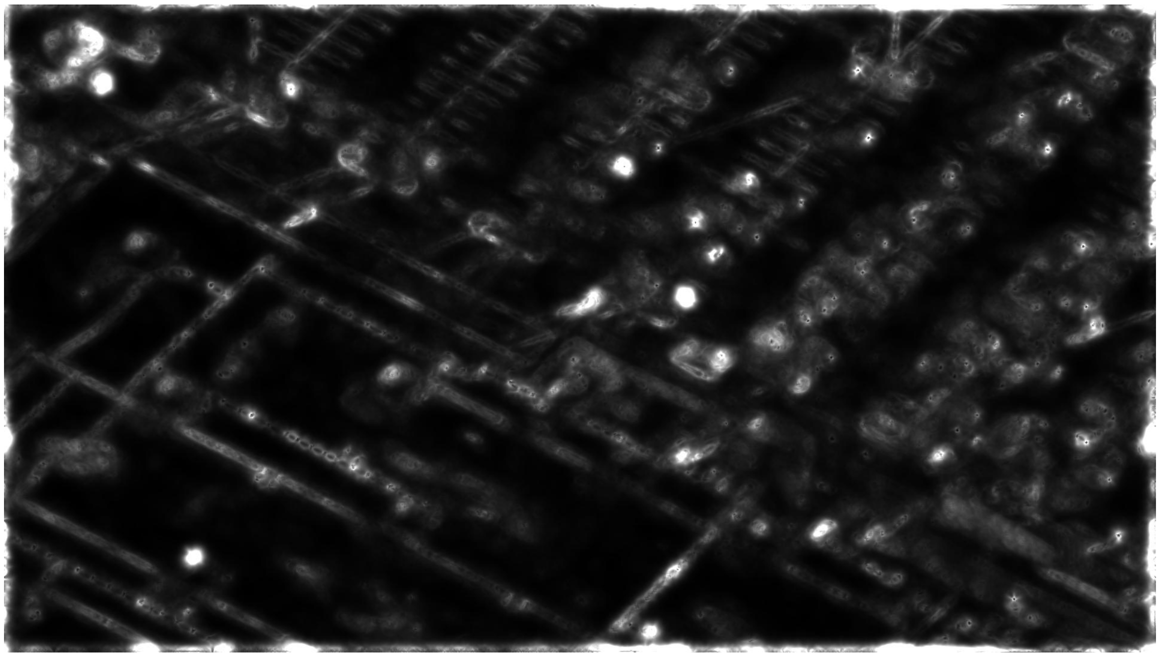


Figure 4.15: Ballness of FM-MIMS algorithm

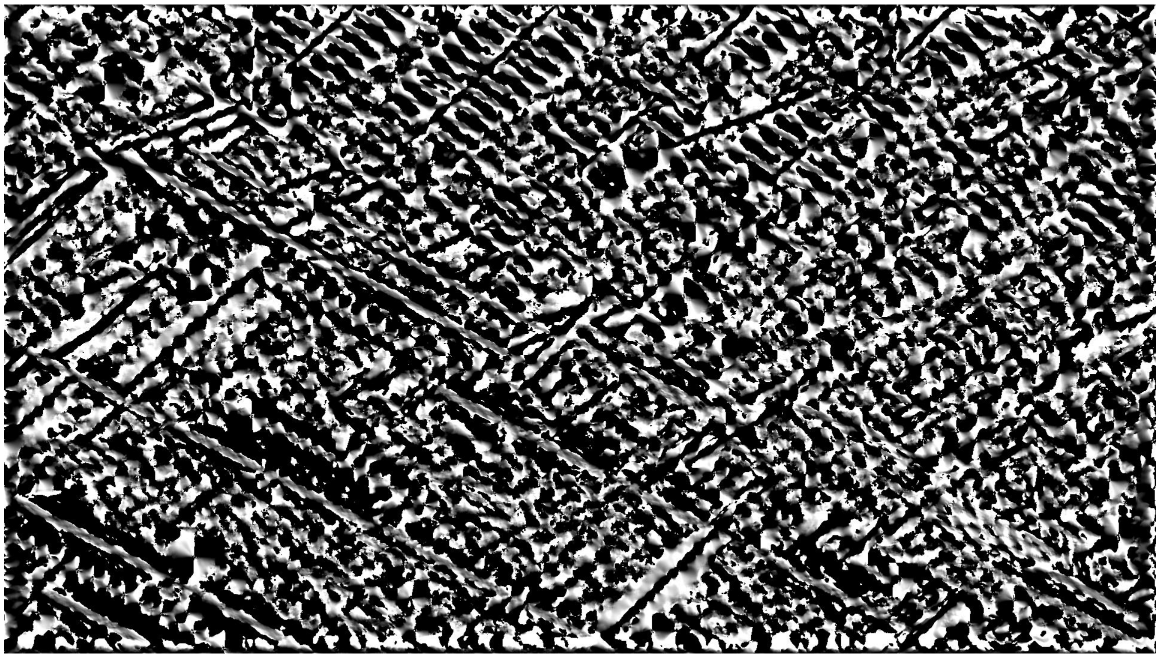


Figure 4.16: Orientation of FM-MIMS algorithm

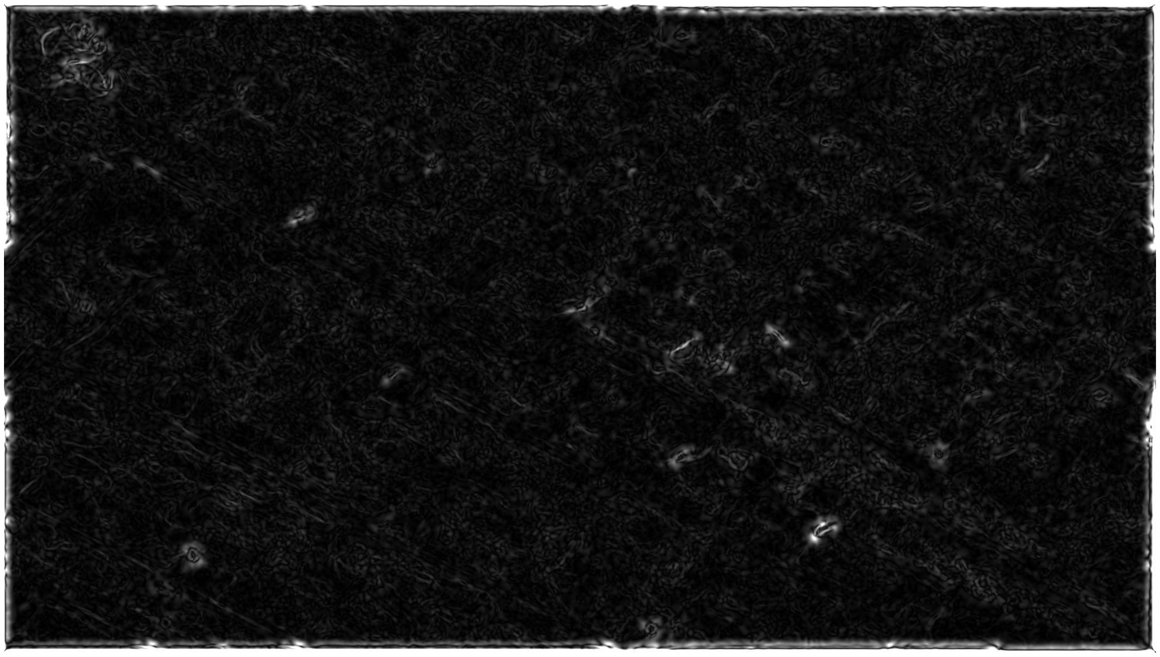


Figure 4.17: Saliency of the MIMS algorithm by Zhu *et al.* (2014)

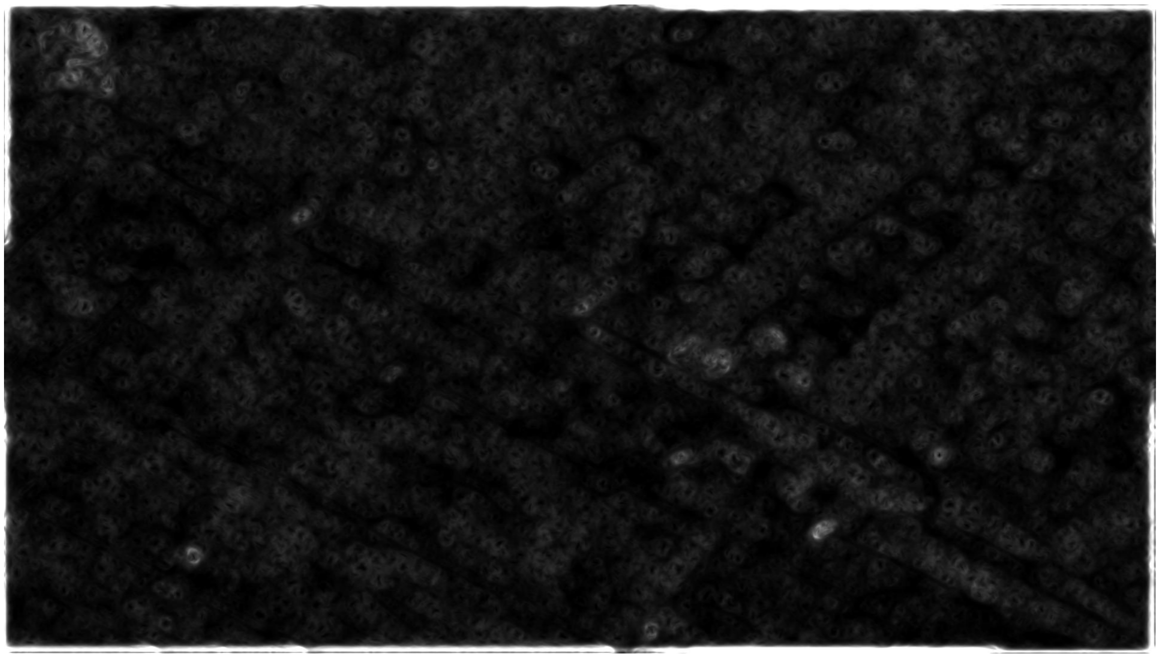


Figure 4.18: Ballness of the MIMS algorithm by Zhu *et al.* (2014)

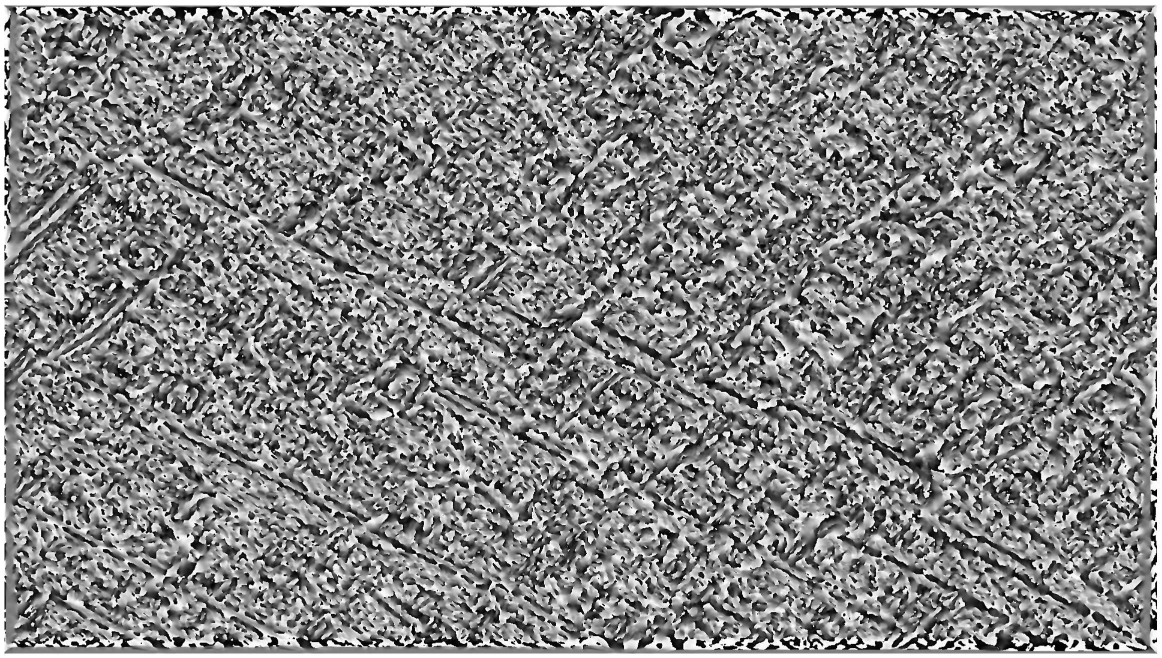


Figure 4.19: Orientation of the MIMS algorithm by Zhu *et al.* (2014)

4.5.4 Discussion

The results from this process, shown in the above figures, show definite correlations being formed between the regions of interest in the image. The Saliency map can recognize targets that are spaced along a line. This makes it easier to identify detections created by curbs, building edges, and even regularly spaced clutter such as the vents on the roof of the building. The ball votes, seen in Figure 4.15 and Figure 4.18, are even more interesting as they identify the outliers. These are regions of the image that do not conform to any of the patterns found in the background clutter. Given that the visual footprint or motion ques of dismounts are not consistent with their background, this is the property that would identify potential regions with dismounts.

Furthermore, the features appear more well defined in the FM-MIMS algorithm, when comparing the saliency and ballness outputs from the MIMS and FM-MIMS algorithm. This shows that the initial feature detection functions help improve the outcome of pixel data association.

4.5.5 Next Steps

The best way to run this algorithm would be to leverage as much of a computer's GPU as possible. The next steps would require the code to be reformulated to optimally use all the GPU cores to compute the appropriate rotation matrices and Monte Carlo is integration required. Running the voting processes in parallel over all the cores available in the GPU would drastically reduce execution time.

The Monte Carlo integration version of tensor voting should also be compared against the original algorithm to test for the accuracy of output values and the impact on execution time. In an ideal case, the tests should be able to observe that as the number of samples used in equation 3.6.5 grows, the values of each pixel converge at the values computed by the original tensor voting algorithm (Tang *et al.* (2000)).

4.6 MMIS

The core value of the MIMS framework was provided by executing the tensor voting algorithm at different scale values and observing how the resulting saliency and dimensionality of the output evolves. Given that the tensor voting algorithm was forced to be simplified (see section 4.5.1) the dimensionality will consistently remain 1. Instead, the ballness values were compared as they were an adequate indicator of noise in the scene.

Below is a direct comparison of 5 samples of saliency(), dimensionality(), and ballness original algorithm

4.6.1 Original MIMS algorithm

Dismounts

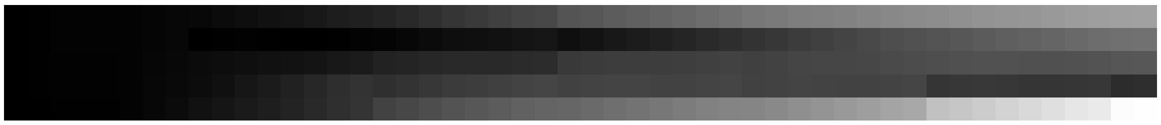


Figure 4.20: MIMS structure of the saliency of 5 random pixels(top to bottom) from scale 1(left) to scale 50(right) with dismounts

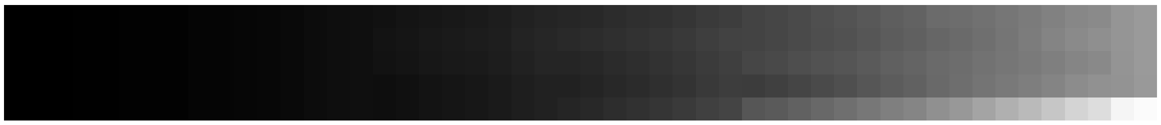


Figure 4.21: MIMS structure of the ballness of 5 random pixels(top to bottom) from scale 1(left) to scale 50(right) with dismounts

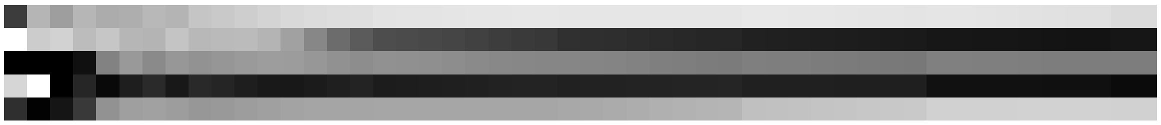


Figure 4.22: MIMS structure of the orientation of 5 random(top to bottom) pixels from scale 1(left) to scale 50(right) with dismounts

Non-Dismounts

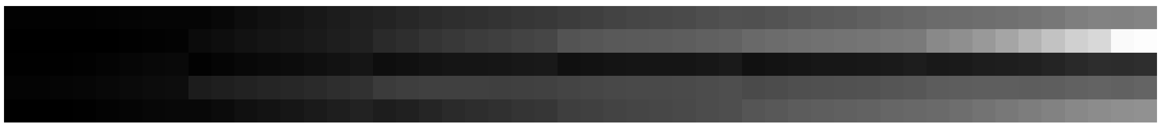


Figure 4.23: MIMS structure of the saliency of 5 random pixels(top to bottom) from scale 1(left) to scale 50(right) without dismounts

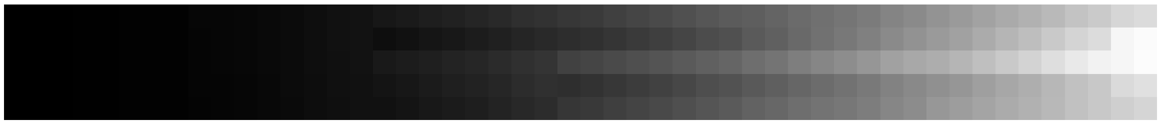


Figure 4.24: MIMS structure of the ballness of 5 random pixels(top to bottom) from scale 1(left) to scale 50(right) without dismounts

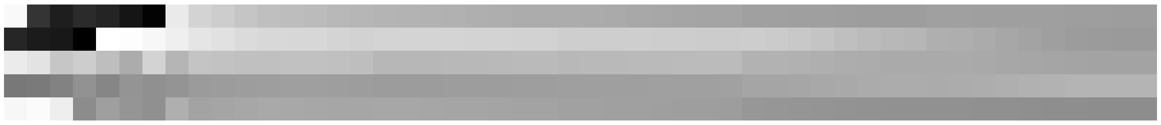


Figure 4.25: MIMS structure of the orientation of 5 random pixels(top to bottom) from scale 1(left) to scale 50(right) without dismounts

4.6.2 FM-MIMS Algorithm

Dismounts

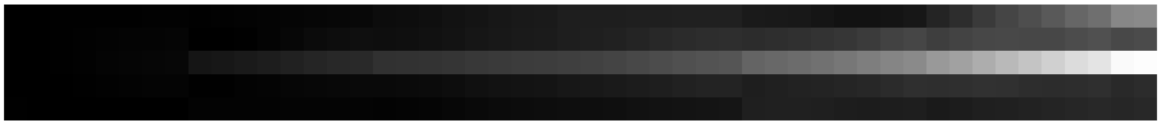


Figure 4.26: FM-MIMS structure of the saliency of 5 random pixels(top to bottom) from scale 1(left) to scale 50(right) with dismounts

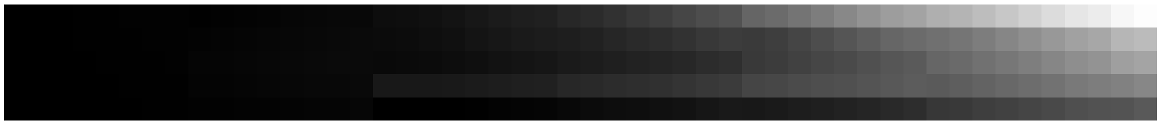


Figure 4.27: FM-MIMS structure of the ballness of 5 random pixels(top to bottom) from scale 1(left) to scale 50(right) with dismounts

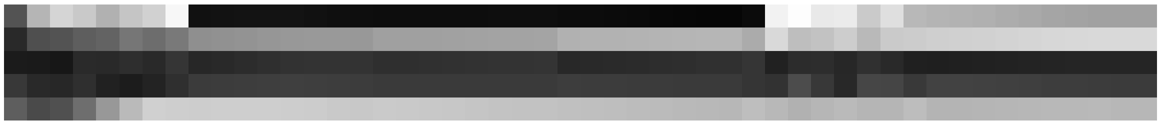


Figure 4.28: FM-MIMS structure of the orientation of 5 random(top to bottom) pixels from scale 1(left) to scale 50(right) with dismounts

Non-Dismounts

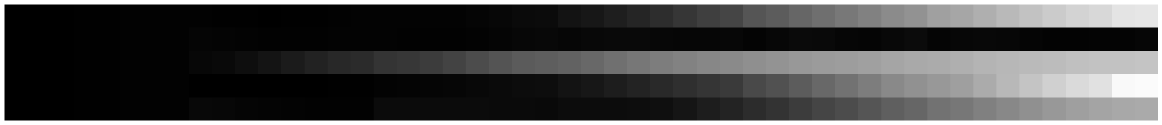


Figure 4.29: FM-MIMS structure of the saliency of 5 random pixels(top to bottom) from scale 1(left) to scale 50(right) without dismounts

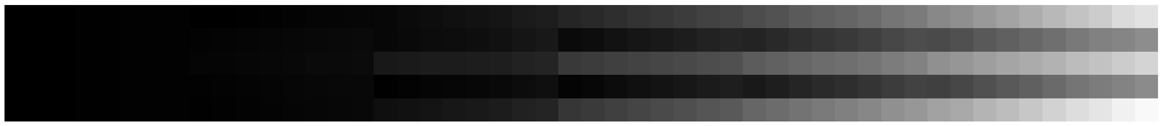


Figure 4.30: FM-MIMS structure of the ballness of 5 random pixels(top to bottom) from scale 1(left) to scale 50(right) without dismounts

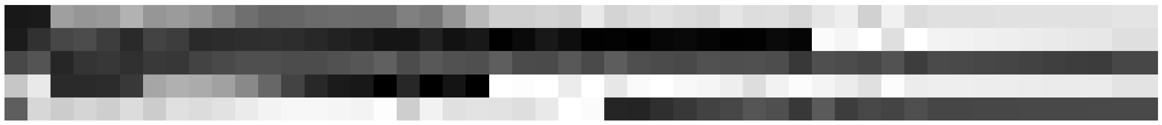


Figure 4.31: FM-MIMS structure of the orientation of 5 random pixels(top to bottom) from scale 1(left) to scale 50(right) without dismounts

4.7 Discussion

In order to compare the MIMS structures, the same pixels were compared at the same scale across the various algorithms as well as across the 3 properties (saliency, ballness and, orientation). To the naked eye, each gradient profile appears very similar to one another. There is no method to verify whether or not the gradient profiles provide any valuable information.

4.8 Comparing Classifiers

In the MIMS algorithm developed by Zhu *et al.* (2014), AdaBoost was exclusively used as the classifier. This section of testing attempted to identify if AdaBoost was the best classifier for the job. Three algorithms were considered, AdaBoost, Random Forest, KNN, and RUSBoosted Trees.

4.8.1 Test Setup

Using 1600 samples where 800 samples were of randomly selected ground truth, and 800 of randomly selected pixels. The same frames and pixels were used to train all 4 classifiers. Only the MIMS model computed in the previous section was used as predictors. While training the classifiers, all classifications were given an equal cost of 1.

4.8.2 Discussion

In both cases, the random forest outperforms the 4 other classification filters. We can also see that on average the Classifiers for the revised filtering algorithm outperform those by the original MMIS algorithm, in terms of accuracy. The accuracy of each filter still leaves a lot to be desired as the uncertainty is extremely high. Furthermore, the rates in table 4.1 are not consistent with the results shown in Zhu *et al.* (2014). This could be due to the style of dense optical flow used (section 4.3.1), the approximations in the tensor voting algorithm (section 4.5.2), or the change in parameters used for MIMS (section 4.7). The many modifications required to allow the code to execute prevents the classifiers from producing any results comparable to Zhu *et al.*

(2014).

4.8.3 Classifier Performance

Algorithm	Accuracy(Validation)	Execution time per frame (s)
AdaBoost	58.5%	12.759
Random Forest	60.8%	13.329
RUSBoosted Trees	57.1%	8.4031
KNN	56.8%	1182.7

Table 4.1: Classifier results for MIMS algorithm designed by Zhu *et al.* (2014)

Algorithm	Accuracy(Validation)	Execution time per frame (s)
AdaBoost	60.8%	10.388
Random Forest	64.1%	12.026
RUSBoosted Trees	53.0%	7.7030
KNN	56.9%	1180.4

Table 4.2: Classifier results for the FM-MIMS algorithm

4.8.4 Choosing a Classifier

Due to performance issues, KNN and RUSBoost were not tested any further. Both models consistently had the worst accuracy from the set. Furthermore, KNN required almost 100$\times$ more computation time relative to the other algorithms tested. Not only was it slower, it consistently scored the lowest accuracy rate.

As AdaBoost and Random Forest had very similar performances, both were tested in the complete MIMS system.

4.9 Overall Performance

Once all the components were in place the efficacy of the detection algorithm was tested by running the MIMS algorithm by Zhu *et al.* (2014) and the FM-MIMS algorithm on the same set of 100 test frames. The results can be found in Table 4.3. It is interesting to note that the Spatio-temporal filter (*spatial* $\times$ *temporal*) outperformed all the other algorithms. This filter by itself did not use any machine learning except for a threshold value trained using linear regression.

Detector	Probability of Detection	Probability of Missed Detection	Probability of False Alarm
MMIS Algorithm			
Temporal	70.79%	29.21%	2.79%
AdaBoost	96.37%	3.62%	37.54%
RandomForest	94.13%	5.87%	32.15%
Temporal×AdaBoost	50.96%	49.04%	0.41%
Temporal×RandomForest	49.50%	50.50%	0.38%
FM-MIMS Algorithm			
Spatial	100%	0%	3.78%
Temporal	70.79%	29.21%	2.79%
Spatiotemporal	99.00%	1.00%	0.56%
AdaBoost	98.33%	1.67%	41.39%
RandomForest	99.62%	0.37%	35.68%
Spatiotemporal×AdaBoost	76.92%	23.08%	0.15%
Spatiotemporal×RandomForest	91.17%	8.83%	0.20%

Table 4.3: Average results of running the detectors discussed

A key constraint of the FM-MIS framework was the execution time. Table 4.4 displays the average execution time of each module as well as the time required to execute the framework over a single frame of data. The FM-MIMS execution time was less than half that of MIMS. Although the proposed algorithm is quicker, a processing time of 145.003 seconds/frame is still very impractical to run in real-time. here it is

Important to note that the MIMS module takes the longest time to execute. This module however is composed of the tensor voting algorithm being executed multiple times at various scale values. Because of this, the greatest delay in FM-MIMS and MIMS is caused by tensor voting. The next slowest module in FM-MIMS is the small target detector.

Module Name	Avg. Execution time (seconds/frame)
MMIS Modules	
Background Stabilization	7.038
Optical Flow	2.440
MIMS	318.006
- Tensor Voting	20.6579
Boosting - AdaBoost	12.759
Total	340.901
FM-MIMS Algorithm	
Background Stabilization	7.010
Dismount Size Prediction	14.562
Small Target Detector	27.253
Optical Flow	1.210
Spatial Filter	1.394
Temporal Filter	1.394
MIMS	81.792
- Tensor Voting	2.796
Boosting - AdaBoost	10.388
Total	145.0030

Table 4.4: Average execution times of each module

4.9.1 Algorithm Performance Over Time

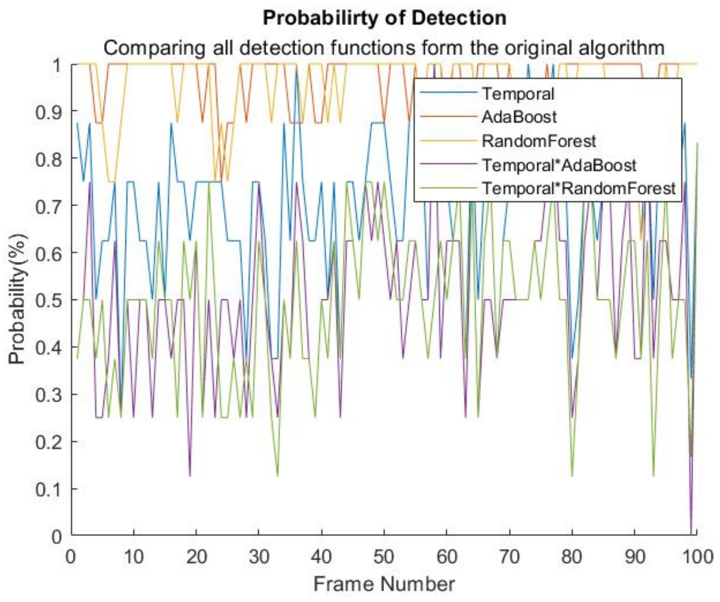


Figure 4.32: Probability of detection of the MIMS algorithm over 100 frames

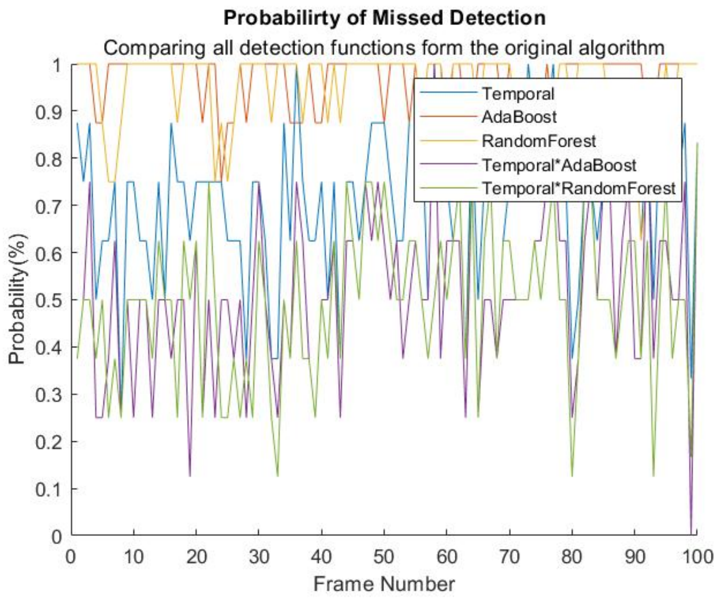


Figure 4.33: Probability of missed detections of the MIMS algorithm over 100 frames

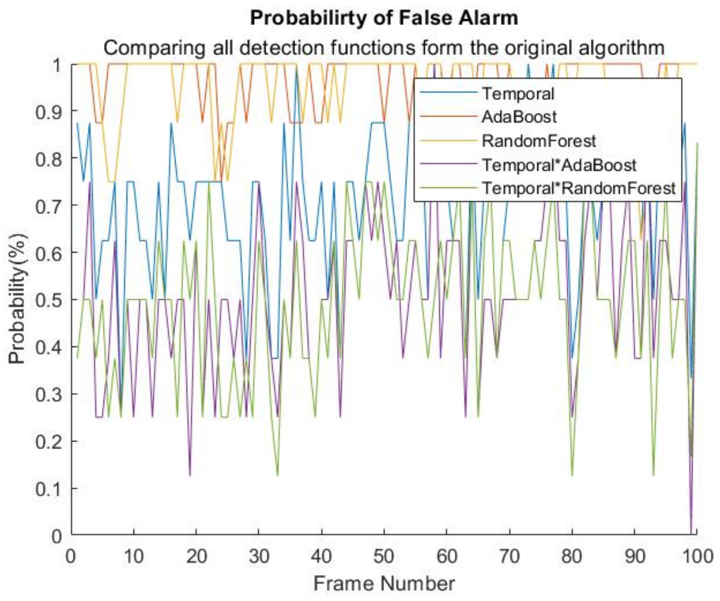


Figure 4.34: Probability of false alarm of the MIMS algorithm over 100 frames

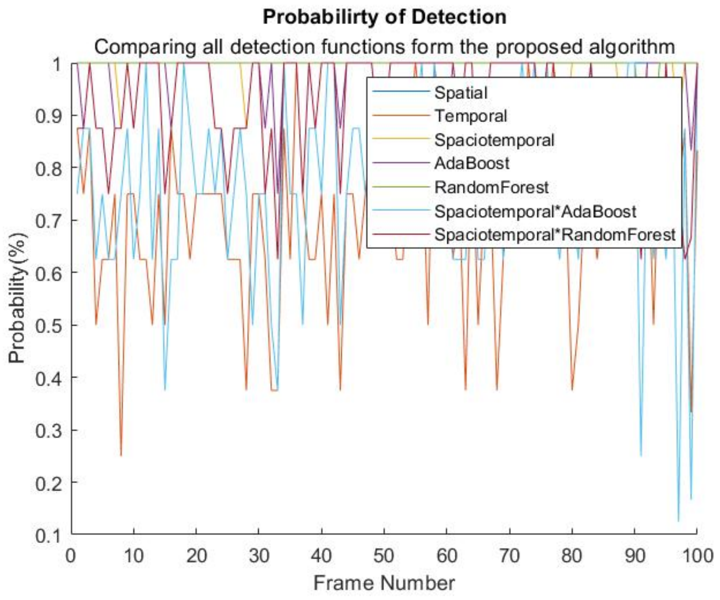


Figure 4.35: Probability of detection of the FM-MIMS algorithm over 100 frames

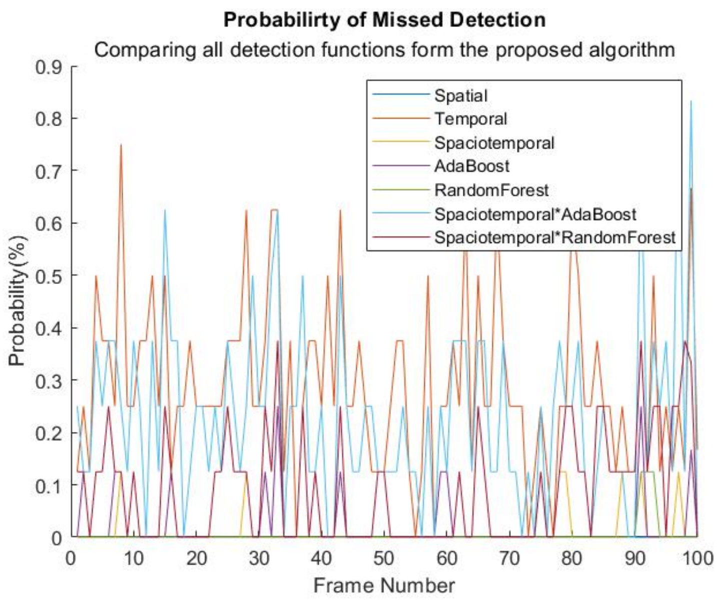


Figure 4.36: Probability of missed detections of the FM-MIMS algorithm over 100 frames

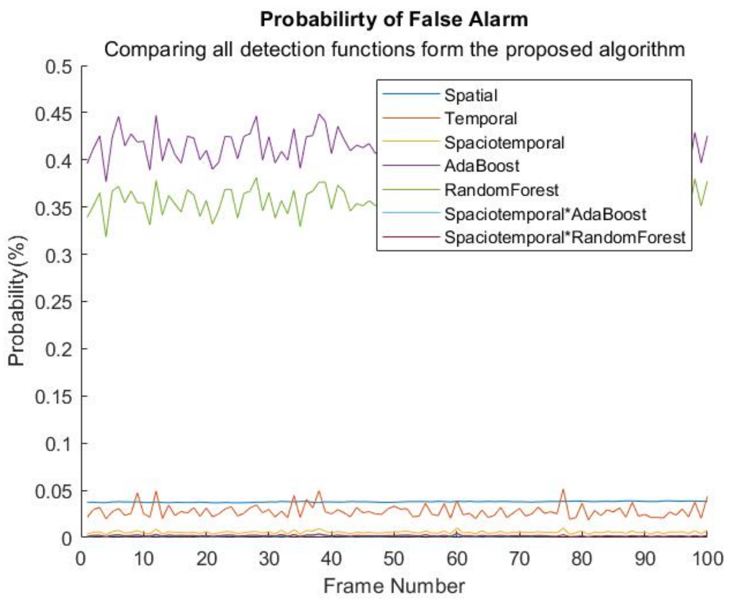


Figure 4.37: Probability of false alarms of the FM-MIMS algorithm over 100 frames

Chapter 5

Conclusion

5.1 Dismount Detection

Dismount detection is a complex problem. Historically it has been difficult to detect extremely small targets with limited features and a high amount of background clutter. Some detection algorithms depend solely on picking out identifiable features. Others track changes in the scene and attempt to match the motion of pixels to known motion models for dismounts. Both approaches require prior knowledge of the scene and the dynamics of the dismount. Zhu *et al.* (2014)'s Multi-scale Intrinsic Motion model proposed a novel method to detect dismounts with high accuracy relative to other models. The Model although effective, was only designed to detect dismounts in the $4 - 9$ pixel range.

5.2 Extending MIMS To FM-MIMS

Zhu *et al.* (2014)'s model was originally designed to detect dismounts in the $4 - 9$ pixels range. The FM-MIMS model attempts to extend this algorithm to work with detections that range from $15 - 30$ pixels. FM-MIMS extends the MIMS algorithm by adding the influence of sensor metadata to the detection algorithm. This additional information is used to form predictions and better tune the feature detection algorithms used. While MIMS relies solely on the positions and motion queues, FM-MIMS adds to this by introducing the small target detection feature matrix. The matrix allows for a better understanding of whether the observed pixels meet the criteria derived from sensor metadata. The added data could assist the tensor voting algorithm make more informed data associations potentially leading to fewer missed detections.

Given that FM-MIMS adds complexity to the MIMS algorithm, it increases the computational burden of detecting dismounts. To address this, FM-MIMS suggests some relaxation techniques and approximations to reduce the number of calculations. As observed in section 4 this was not sufficient (see section 5.3). Further relaxations were made to the framework to allow FM-MIMS to be viable. This was done by relying on a single feature matrix derived from a hybrid of the small target detector and optical flow detector. Even though the results from Zhu *et al.* (2014) could not be replicated, the TM-MIMS framework showed incremental improvements at the unit level relative to MIMS. The contributions of the FM-MIMS algorithm require a great deal more work but show promise.

5.3 Limitations Of FM-MIMS

The Extended MIMS model was limited by the computation power available. Increasing the tensor size to accommodate for additional features, exponentially increased the computational requirements of the algorithm. The proposed changes to the Tensor voting integration equation were not sufficient to account for it. The tensor voting algorithm may be effective in identifying features, but it would be difficult to execute in real-time. The Framework would be best suited for post-processing. Given the limitations imposed by the tensor voting algorithm. Many approximations were made preventing a thorough test of the extended MIMS algorithm.

5.4 Required Improvements

As discussed in previous sections, the FM-MIMS algorithm is far from perfect. The system is not capable of identifying detections in real-time, it still does not rely on a wide array of identity matrices and lastly, it requires a lot of computing resources to be executed. First and foremost, the code must be refactored such that it optimally takes advantage of the computer's GPU. Without proper GPU integration, FM-MIMS can take several minutes to process a frame. Secondly, the tensor voting algorithm should be extended as proposed in section 3.6. This will allow the algorithm to compare different feature matrices and improve the quality of data association. By doing so, the aim is to produce more distinguishable features (see section 4.6) in the Multi-scale Intrinsic Motion Structure (MIMS). Lastly, the training and testing of the models need to be conducted on a more diverse data set. A wider range of scenes with varying viewing angles, lighting conditions, background clutter, and density of dismounts would help prevent overfitting the models used.

Bibliography

a/l Kanawathi, J., Mokri, S. S., Ibrahim, N., Hussain, A., and Mustafa, M. M. (2009). Motion detection using horn schunck algorithm and implementation. In *2009 International Conference on Electrical Engineering and Informatics*, volume 01, pages 83–87.

Amrouche, N., Khenchaf, A., and Berkani, D. (2017). Multiple target tracking using track before detect algorithm. In *2017 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, pages 692–695.

Blackman, S., Dempster, R., Busch, M., and Popoli, R. (1999). Imm/mht solution to radar benchmark tracking problem. *IEEE Transactions on Aerospace and Electronic Systems*, **35**(2), 730–738.

Blasch, E., Ling, H., Wu, Y., Seetharaman, G., Talbert, M., Bai, L., and Chen, G. (2012). Dismount tracking and identification from electro-optical imagery. In O. Mendoza-Schrock, M. M. Rizki, and T. V. Rovito, editors, *Evolutionary and Bio-Inspired Computation: Theory and Applications VI*, volume 8402, pages 129 – 138. International Society for Optics and Photonics, SPIE.

Boers, Y. and Driessen, J. (2001). Particle filter based detection for tracking. In

Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148), volume 6, pages 4393–4397 vol.6.

Boers, Y. and Driessen, J. (2005). Multitarget particle filter track before detect application. *Radar, Sonar and Navigation, IEE Proceedings -*, **151**, 351 – 357.

Davey, S., Rutten, M., and Cheung, B. (2008). A comparison of detection performance for several track-before-detect algorithms. In *2008 11th International Conference on Information Fusion*, pages 1–8.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**(1), 119–139.

Guest, I. (2009). *Digital video moving object segmentation using tensor voting: A non-causal, accurate approach*. Ph.D. thesis, University of Cape Town.

Hersey, R. K., Melvin, W. L., and Culpepper, E. (2008). Dismount modeling and detection from small aperture moving radar platforms. In *2008 IEEE Radar Conference*, pages 1–6.

Jiménez-Bravo, D. M., Mutombo, P. M., Braem, B., and Marquez-Barja, J. M. (2020). Applying faster r-cnn in extremely low-resolution thermal images for people detection. In *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–4.

Kim, C., Li, F., Ciptadi, A., and Rehg, J. M. (2015). Multiple hypothesis tracking revisited. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Lutins, E. (2017). Ensemble methods in machine learning: What are they and why use them?

Narayanaswami, R., Tyurina, A., Diel, D., Mehra, R. K., and Chinn, J. M. (2012). Investigation of kinematic features for dismount detection and tracking. In O. E. Drummond and R. D. Teichgraeber, editors, *Signal and Data Processing of Small Targets 2012*, volume 8393, pages 24 – 37. International Society for Optics and Photonics, SPIE.

Nguyen, D. T., Li, W., and Ogunbona, P. O. (2016). Human detection from images and videos: A survey. *Pattern Recognition*, **51**, 148–175.

Park, J. D. and Doherty, J. F. (2015). Track detection of low observable targets using a motion model. *IEEE Access*, **3**, 1408–1415.

Ramli, S., Othman, N., Zainudin, N., Wahab, N. S. A., Hasbullah, N. A., Zizi, T. T., Azizi, F. M., and Ibrahim, N. (2016). Comparison between horn-schunck and brox techniques for motion detection in thermal video. In *2016 International Conference on Information and Communication Technology (ICICTM)*, pages 27–30.

Ratini, M. (2019). Average height for men worldwide (stats inside).

Rutten, M., Ristic, B., and Gordon, N. (2005). A comparison of particle filters for recursive track-before-detect. In *2005 7th International Conference on Information Fusion*, volume 1, pages 7 pp.–.

Shanmugavadivu, P. and Kumar, A. (2016). Rapid face detection and annotation with loosely face geometry. In *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, pages 594–597.

Tang, C.-K., Lee, M.-S., and Medioni, G. (2000). *Tensor Voting*, pages 215–237. ResearchGate.

Tang, C.-K., Medioni, G., and Lee, M.-S. (2001). N-dimensional tensor voting and application to epipolar geometry estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **23**(8), 829–844.

Tong, W.-S., Tang, C.-K., and Medioni, G. (2001). First order tensor voting, and application to 3-d scale analysis. In *Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–175.

Xie, K., Fu, K., Zhou, T., Yang, J., Wu, Q., and He, X. (2015). Small target detection using an optimization-based filter. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1583–1587.

Zhu, J., Javed, O., Liu, J., Yu, Q., Cheng, H., and Sawhney, H. (2014). Pedestrian detection in low-resolution imagery by learning multi-scale intrinsic motion structures (mims). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.