IMPROVED GEO-REFERENCING AND PRESCREENING FOR DETECTION OF BURIED EXPLOSIVE HAZARDS IN FORWARD-LOOKING INFRARED IMAGERY

A Dissertation presented to the Faculty of the Graduate School
University of Missouri

In Partial Fulfillment of the Requirements of the Degree
Doctor of Philosophy

by
KEVIN E. STONE

Dr. James M. Keller, Dissertation Supervisor

December 2014
The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled

IMPROVED GEO-REFERENCING AND PRESCREENING FOR DETECTION OF BURIED EXPLOSIVE HAZARDS IN FORWARD-LOOKING INFRARED IMAGERY

presented by Kevin Stone,

a candidate for the degree of doctor of philosophy,

and hereby certify that, in their opinion, it is worthy of acceptance.

__________________________________________
Professor James M. Keller

__________________________________________
Professor Dominic Ho

__________________________________________
Professor Alina Zare

__________________________________________
Professor Mihail Popescu
Thank you to my wife Elizabeth for waiting, and for pushing me to finish, to my mother and father for their unending love and support, and to my brother for many discussions about research and work that none of our friends found interesting.
ACKNOWLEDGEMENTS

Foremost, I would like to thank my advisor Dr. Jim Keller for many constructive and enlightening conversations, infinite patience and support, and putting up with all of the unanswered emails through the years. Taking your introduction to computational intelligence course during my final undergraduate semester is what sparked my interest in machine learning and computational intelligence, and is what started me on this path. More than once, I was unsure I would finish. Now that I have, I am very glad to have started.

I would also like to thank the other members of my committee: Dr. Dominic Ho, Dr. Mihail Popescu, and Dr. Alina Zare for their time, thoughts, conversations, and support during my time as a graduate student.

To all of the professors whose classes made it interesting: Dr. Ho, Dr. Han, Dr. DeSouza, Dr. He, Dr. Keller, Dr. Skubic, and Dr. Shang, and to all of the people I have worked with and have gotten to know in the lab, or outside of it, over the years (in no particular order): Pine, Stanton, Chris, David, Brian, Justin, Derek, Bob, Tim, and everyone I am forgetting, I say thank you.
# Table of Contents

**Acknowledgements** .................................................................................................................. ii

**List of Illustrations** ..................................................................................................................... v

**List of Tables** ............................................................................................................................... ix

**Abstract** ........................................................................................................................................ x

Chapter 1: Introduction ....................................................................................................................... 1

1.1 Motivation ...................................................................................................................................... 1

1.1.1 Geo-referencing ....................................................................................................................... 1

1.1.2 Prescreening .......................................................................................................................... 6

1.2 Contributions ............................................................................................................................... 11

1.3 List of Publications ..................................................................................................................... 12

Chapter 2: Background ...................................................................................................................... 13

2.1 Geo-referencing ........................................................................................................................... 13

2.1.1 Camera Model and Calibration ............................................................................................ 13

2.1.2 Image Pixel to World Transformation .................................................................................. 16

2.1.3 Flat Earth 1.0 ......................................................................................................................... 17

2.1.4 Structure from Motion .......................................................................................................... 22

2.1.4.1 Deficiency in Essential Matrix Refinement for Planar Scene ........................................ 28

2.2 FL-IR Prescreening ..................................................................................................................... 30

2.2.1 Size-Contrast Filter Ensemble ............................................................................................. 32

2.2.2 GMM ...................................................................................................................................... 35

2.2.3 MSER ..................................................................................................................................... 36

2.2.4 HOG features ....................................................................................................................... 37

2.2.5 Integral Image ....................................................................................................................... 38

2.2.6 Look-up Table Approach for Fast Prediction with Additive Kernels ................................. 39

2.2.7 Explicit Feature Map for Fast Training with Additive Kernels .......................................... 43

Chapter 3: Flat Earth 2.0 .................................................................................................................... 46

3.1 Model Assumptions and Theory ............................................................................................... 47

3.2 Implementation Details .............................................................................................................. 52

3.3 Proposed Fix for Essential Matrix Refinement Process of SfM Approach ............................. 59

3.4 Accuracy Comparison to SfM and Flat Earth 1.0 .................................................................... 64

3.5 Speed Comparison to SfM and Flat Earth 1.0 .......................................................................... 75
Chapter 4: MS-HOG Prescreener ................................................................. 79
4.1 MS-HOG features ............................................................................. 79
4.2 Boxcar Filtering Versus Integral Image for Fixed-Size HOG Cells .......... 85
4.3 Look-up Table Approximate Prediction for Explicit Feature Map .......... 86
4.4 MS-HOG Prescreener Algorithm Outline ......................................... 89
  4.4.1 Parameters .................................................................................. 89
  4.4.2 Training ....................................................................................... 94
  4.4.3 Operational Use ......................................................................... 95
4.5 Integral Image Versus Boxcar Filtered Image ..................................... 96
4.6 Datasets, Evaluation Methodology, and Algorithm Parameters .......... 97
4.7 Restricting Training to Targets with Visible Signature ....................... 101
4.8 Hard Negative Mining ..................................................................... 106
4.9 Additive Kernels ............................................................................ 111
  4.10 Comparison to Ensemble of Size-Contrast Filters Prescreening Algorithm ...... 119
Chapter 5: Conclusions ........................................................................ 127
  5.1 Summary ......................................................................................... 127
  5.2 Future Work ................................................................................... 128
Chapter 6: Appendix ............................................................................... 130
  6.1 Iterative Linear-LS Triangulation for Many Points at Once ............... 130
  6.2 Visualizing the MS-HOG SVM Classifier ...................................... 133
  6.3 Training a Secondary Classifier ..................................................... 136
  6.4 Estimation of Plane Normal Vector and Depth to Plane ................... 137
REFERENCES ....................................................................................... 140
VITA ......................................................................................................... 145
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Depiction of Flat Earth 1.0 model. The camera traverses a world that is a single large plane. It rotates only about the Z axis, and translates only in X and Y.</td>
</tr>
<tr>
<td>1.2</td>
<td>Example FL-IR target signatures. Top row: hard to detect targets due to low contrast and obscuration by tire tracks. Bottom row: easy to detect high contrast, unobscured targets.</td>
</tr>
<tr>
<td>2.1</td>
<td>Diagram of the pinhole camera image formation model. Each image point corresponds to the intersection of a line from a 3D point in the camera reference frame through the center of projection (C).</td>
</tr>
<tr>
<td>2.2</td>
<td>Top: World to image transformation error in pixels as the altitude offset between the INS and the world plane differs from the assumed offset of the Flat Earth 1.0 model for points 5, 10, 20, and 30 meters in front of the vehicle. Bottom: Image to world error for the same configuration in meters.</td>
</tr>
<tr>
<td>2.3</td>
<td>Top: World to image transformation error in pixels as the pitch angle between the INS and the camera differs from the assumed angle of the Flat Earth 1.0 model for points 5, 10, 20, and 30 meters in front of the vehicle. Note that the curves for 10, 20, and 30 meters are almost identical. Bottom: Image to world error for the same configuration in meters.</td>
</tr>
<tr>
<td>2.4</td>
<td>Epipolar geometry. Depiction of two cameras viewing the same point, P. P_L and P_R represent the projection of P into the image plane of the left and right camera respectively. E_L and E_R are the epipoles. O_L and O_R are the centers of projection of the left and right camera.</td>
</tr>
<tr>
<td>2.5</td>
<td>Top row: distance from camera estimates using position and pose as reported by INS for two different cameras. Left: Camera 1. Right: Camera 2. Blue indicates near and red indicates far. Depth maps were created by merging depth estimates from multiple frame pairs. Horizontal discontinuities where triangulation errors differ between frame pairs due to error in reported position are denoted by the black rectangular boxes. Bottom row: reconstruction using essential matrix refinement. Discontinuities eliminated.</td>
</tr>
<tr>
<td>2.6</td>
<td>Depiction of center sample and surrounding neighborhood used in 2D LMS GPR prescreening algorithm. The center value is predicted using a linear combination of the surround samples. A large error in the prediction is flagged as an anomaly.</td>
</tr>
<tr>
<td>2.7</td>
<td>Illustration of dual sliding window detector. Left: red window represents the inner window and the green window represents the outer window. Right: parameterization of the two windows.</td>
</tr>
</tbody>
</table>
2.8 Left: the sum of values in the red region can be computed by accessing only the four green blocks of the integral image. Sum = D-C-B+A. Right: extension of integral image to multiple channels, such as histogram features. ........................................ 39

3.1 Depiction of a forward looking camera observing a ground plane from multiple locations. The two resulting images of the ground plane are related by a homography (Equation 2.10). The homography is defined by $d$, the distance to the observed plane from Camera A, $n$, the normal vector to the plane, and $R$ and $t$, the rotation and translation from Camera A to Camera B. ........................................ 49

3.2 Flat Earth 2.0 theoretical registration error for various out-of-plane distances and distances from the camera assuming a depth to plane of three meters. ........................................ 51

3.3 Example images from infrared cameras A, B, C, and D in order from top to bottom. Left column: images from Lane A. Right column: images from Lane B. ............................... 62

3.4 Registration error for SfM approach with and without the proposed fix for handling planar scenes during essential matrix refinement. Plotted lines are average error. Bottom and top error bars correspond to 25th and 75th error percentiles. Top plot: run from Lane A. Bottom plot: run from Lane B. ........................................ 63

3.5 Registration error as a function of distance from the camera for Flat Earth 1.0 (FE1), SfM, and Flat Earth 2.0 (FE2 DLTPT and FE2 NVD2P) on Camera A. ..................... 67

3.6 Registration error as a function of distance from the camera for Flat Earth 1.0 (FE1), SfM, SfM with temporal post processing (SfM PP), and Flat Earth 2.0 (FE2 DLTPT and FE2 NVD2P) on Camera B. ................................................................. 68

3.7 Registration error as a function of distance from the camera for Flat Earth 1.0 (FE1), SfM, SfM with temporal post processing (SfM PP), and Flat Earth 2.0 (FE2 DLTPT and FE2 NVD2P) on Camera C. ................................................................. 69

3.8 Registration error as a function of distance from the camera for Flat Earth 1.0 (FE1), SfM, SfM with temporal post processing (SfM PP), and Flat Earth 2.0 (FE2 DLTPT and FE2 NVD2P) on Camera D. ................................................................. 70

3.9 SIFT keypoint locations superimposed on selected images from cameras A, B, C, and D, in order from top to bottom. Left: Lane A. Right: Lane B. ...................................................... 71

4.1 Two methods of multi-scale HOG feature extraction. Top row: image is resized while gradient operator size is fixed. Bottom row: image is fixed and gradient operators are enlarged. ...................................................... 81

4.2 Noisy infrared image patches. Top: patch from cooled long-wave infrared camera. Bottom: patches from cooled long-wave (left) and mid-wave (right) infrared cameras. Salt and pepper noise and dead pixels are common in cooled infrared cameras if the sensor temperature is not carefully controlled. ...................................................... 82
4.3 Image patch from mid-wave infrared camera. Impulse noise present. .......................... 83
4.4 Gradient magnitude at two orientations and three scales for image patch in Figure 4.3. Left: Gaussian pre-filtering. Right: Median pre-filtering. Orientations correspond to 0 and 90 degrees. ........................................................................ 84
4.5 Left: the sum of values in the red region can be computed by accessing only the green block of the boxcar filtered image. Right: extension to multiple channels, such as histogram features. ........................................................................ 86
4.6 Proposed dual cell-grid layout for MS-HOG prescreening algorithm. Visualized on image chips from three different infrared cameras. Grid layout remains the same across cameras, but the size of the cells is adjusted based on image resolution. ....... 90
4.7 The target signature in the left of the image is close to the image border causing part of the detection window to fall outside of the image. .................................................. 92
4.8 ROC curves for Camera B (cmB) when training with all targets (left) and training with only visible targets (right) on Lanes A, B, and C. .......................... 104
4.9 ROC curves for Camera E (cmE) when training with all targets (left) and training with only visible targets (right) on Lanes A, B, and C. .......................... 105
4.10 ROC curves for Camera B (cmB) with hard mining (left) and without hard mining (right) on Lanes A, B, and C. ................................................................. 109
4.11 ROC curves for Camera D (cmD) with hard mining (left) and without hard mining (right) on Lanes A, B, and C. ................................................................. 110
4.12 ROC curves for Camera B (cmB) using histogram intersection kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C. ................................................................. 112
4.13 ROC curves for Camera D (cmD) using histogram intersection kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C. ................................................................. 113
4.14 ROC curves for Camera E (cmE) using histogram intersection kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C. ................................................................. 114
4.15 ROC curves for Camera B (cmB) using chi-square kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C. ............. 115
4.16 ROC curves for Camera D (cmD) using chi-square kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C. ............. 116
4.17 ROC curves for Camera E (cmE) using chi-square kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C. .............. 117

4.18 ROC curves for Camera A (cmA) for proposed MS-HOG prescreener (left) and Ensemble of Size-Contrast Filters prescreener (right) on Lanes A, B, and C. ................. 120

4.19 ROC curves for Camera B (cmB) for proposed MS-HOG prescreener (left) and Ensemble of Size-Contrast Filters prescreener (right) on Lanes A, B, and C. ................. 121

4.20 ROC curves for Camera C (cmC) for proposed MS-HOG prescreener (left) and Ensemble of Size-Contrast Filters prescreener (right) on Lanes A, B, and C. ................. 122

4.21 ROC curves for Camera D (cmD) for proposed MS-HOG prescreener (left) and Ensemble of Size-Contrast Filters prescreener (right) on Lanes A, B, and C. ................. 123

4.22 ROC curves for Camera E (cmE) for proposed MS-HOG prescreener (left) and Ensemble of Size-Contrast Filters prescreener (right) on Lanes A, B, and C. ................. 124

4.23 Visualization of two trained linear kernel SVM classifiers. Top two images: cmA. Bottom two images: cmB. White indicates positive weights. Black indicates negative weights. ......................................................... 134
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Timing for explicit feature map method of Vedaldi and Zisserman using VLFeat [1] library implementation. Times are reported in seconds. <em>Mapping</em> indicates time to perform explicit feature mapping. <em>Dot Product</em> indicates time to calculate decision value by computing dot product of mapped vectors with the decision weight vector $w$. N indicates the number of examples. Input dimensionality was 2000. .......... 45</td>
</tr>
<tr>
<td>3.1</td>
<td>Detection performance on Cameras A, B, C, and D using a fixed detector and Structure from Motion (SfM), SfM with post-processing (SfM PP), Flat Earth 2.0 DLTPT (FE2 DLTPT), Flat Earth 2.0 NVD2P (FE2 NVD2P), and Flat Earth 1.0 (FE1) algorithms for geo-referencing. Numbers shown are NAUC @ 0.1 right hand side. Runs used for evaluation are the same used in Figure 3.5 through 3.8. .. 74</td>
</tr>
<tr>
<td>3.2</td>
<td>Seconds per frame required to process images from Cameras A, B, C, and D using Structure from Motion (SfM), Flat Earth 2.0 (FE2), and Flat Earth 1.0 (FE1) algorithms. Percentage indicates percentage of pixels in the image for which world location was calculated. ........................................ 77</td>
</tr>
<tr>
<td>3.3</td>
<td>Seconds per frame required for keypoint detection, feature extraction, and matching for Cameras A, B, C, and D. .......................................................... 78</td>
</tr>
<tr>
<td>4.1</td>
<td>Timing for explicit feature map prediction using mapping versus LUT with linear interpolation approach. Times are reported in seconds. Mapping indicates time to calculate decision value by performing explicit feature mapping and computing dot product with weight vector. LUT indicates time to calculate decision value by summing results of LUT for each dimension. N indicates the number of examples. ...... 89</td>
</tr>
<tr>
<td>4.2</td>
<td>Seconds per frame for boxcar and integral image (FILT) creation for Cameras A and B, as well as for MS-HOG feature extraction (EXTRACT) from the images. [SSX SSY] indicates horizontal and vertical pixel subsampling factors - a factor of 4 means 1 in every 4 pixels is processed. ........................................ 97</td>
</tr>
<tr>
<td>4.3</td>
<td>Camera type, resolution, distance ranges, large and small cell sizes, and median pre-filter sizes for Cameras A, B, C, D, and E used for prescreening evaluation. .......... 100</td>
</tr>
<tr>
<td>4.4</td>
<td>Results for additive kernel tests. Numbers shown are NAUC @ 0.1 right hand side for the vertically averaged curve from each lane. HistX and ChiX indicate histogram intersection kernel and chi-square kernel with X factor dimensionality enlargement. Lin indicates linear kernel. .......................................................... 118</td>
</tr>
<tr>
<td>4.5</td>
<td>Results for MS-HOG prescreener (MSHP) and ensemble of size-contrast filters prescreener (ESCP). Numbers shown are NAUC @ 0.1 right hand side for the vertically averaged curve from each lane. .......................................................... 125</td>
</tr>
</tbody>
</table>
ABSTRACT

A new method for geo-referencing - the act of assigning location in physical space - of infrared imagery collected from a single camera on a moving platform under the assumption of a piecewise planar world is proposed and evaluated. The purpose is to develop an algorithm with less computational complexity than an arbitrary structure from motion approach, but with better accuracy than a naïve flat earth model approach. The proposed algorithm is used in conjunction with a forward-looking buried explosive hazard detection system which consists of an infrared camera mounted on a moving platform equipped with an inertial navigation system. This system requires fast and accurate geo-referencing in order to convert alarms detected in the captured images to world coordinates.

A new prescreening algorithm for detection of buried explosive hazards in infrared imagery is also proposed. The new algorithm uses a sliding window detector which extracts multi-scale histogram of oriented gradient features in a cell-structured fashion from dual grids centered inside the detection window. The feature vectors are classified using a SVM. This detector is compared to an existing prescreening algorithm that uses an ensemble of local RX anomaly detection filters trained via genetic algorithm. The proposed approach is shown to perform better across multiple cameras and data sets, and it does not require image preprocessing, such as contrast enhancement or denoising. Techniques for fast training and prediction of SVMs using additive kernels are combined. This allows the use of the non-linear histogram intersection and additive chi-square kernels with training time of a linear kernel SVM and prediction time independent of the number of support vectors.
Chapter 1: Introduction

1.1 Motivation

Buried explosive hazards (BEHs) pose a serious problem for both military and civilian organizations around the world [2]. While automatic detection technologies have advanced over the last two decades, current systems in the field, such as ground penetrating radar [3], [4] and electromagnetic induction metal detectors [5], [6], suffer from one major shortcoming: short standoff distance. This limitation has led to significant interest in using passive, forward-looking infrared (FL-IR) cameras for detection of buried explosive objects since they can provide increased standoff capability. However, transitioning from downward looking to forward looking technology presents a number of obstacles. Geo-referencing, the process of assigning location in physical space to the pixels in captured images, is one of these challenges. Another challenge is efficiently processing and exploiting the large amount of data captured by infrared cameras. Current detection algorithms for FL-IR rely on efficient, but simplistic, prescreening algorithms to quickly evaluate the image data and generate a set of candidate hit locations in world coordinates. These prescreening algorithms can most likely be improved by adopting more complicated, but still usable fast, classification techniques from the computer vision literature, such as those used for face recognition and pedestrian detection.

1.1.1 Geo-referencing

Accurate geo-referencing is important not only during system operation, but also during evaluation and testing of potential cameras and detection algorithms. A typical evaluation scenario consists of a vehicle with an onboard inertial navigation system (INS), generally
containing an inertial measurement unit (IMU) and differential global positioning system (DGPS), and computer equipment to collect images from one or more cameras. Evaluation is performed on common test lanes with emplaced ground truth target locations recorded in GPS or Universal Transverse Mercator (UTM) coordinates. Performance assessment is then conducted offline, and requires the ability to map ground truth locations into the captured imagery, as well as the ability to assign world locations to image pixels. Given accurate position information, available from the INS assuming it is positioned rigidly with respect to the camera(s), the first capability is easily achieved if the intrinsic parameters, such as focal length, aspect ratio, skew, distortion, etc…, and extrinsic parameters, rotation and translation from the INS to the camera reference frame, of the camera are known. The computational burden of this world to image mapping is negligible.

The second capability is much more difficult to implement. Given no other information sources, such as 3D point clouds from a laser scanner or depth camera, and making no assumptions about the terrain, structure from motion (SfM) is the only possible approach to the problem. Structure from motion, closely related to synthetic stereo and multiple baseline stereo, is a technique that attempts to estimate three-dimensional structure from sequences of images using correspondences and, optionally, position or motion information for the camera as it moves. For static scenes, structure from motion is very similar to a traditional stereo and multiple baseline stereo vision systems with the added complication that the rotation and translation between cameras is continuously changing. Unfortunately, for the predominantly forward motion case, coupled with small, but non-negligible, position uncertainty due to the INS, this difference adds a significant amount of computational expense. By making assumptions about the imaged terrain and the motion and field of view of the camera we can develop
models with much smaller computational footprints. However, these simplifications will lead to inaccuracy when our assumptions are violated.

Perhaps the most simplistic model possible, which we will refer to as Flat Earth 1.0, described in [7] and [8], is one in which the imaged terrain, in fact, the entire visible world, is considered a single plane and the camera is assumed to rotate around a single axis - the axis perpendicular to the world plane - along which it does not translate. This is depicted in Figure 1.1, where the red rectangular objects depict the camera and the blue plane represents the world. Such a model is suitable for a camera mounted in a fixed location on a vehicle traversing a perfectly flat road. The benefit of this model is that the transformation from camera, or vehicle, relative coordinates to image coordinates reduces to a fixed homography. In 3D, a homography is simply a perspective projection relating two views of the same planar surface, assuming a pinhole camera model. Thus, given the 2D position and orientation of the camera in the plane, one can immediately, and efficiently, map image pixels to world coordinates. The disadvantage of this model is that even the most benign test lanes are not perfectly flat. Bumps, pot holes, and ground curvature will all introduce inaccuracy, and this inaccuracy is amplified as distance from the camera increases. If the inaccuracy becomes too large, it will be unusable for accurate evaluation of cameras and algorithms.
FIGURE 1.1. Depiction of Flat Earth 1.0 model. The camera traverses a world that is a single large plane. It rotates only about the Z axis, and translates only in X and Y.

Both the Flat Earth 1.0 model and a full SfM approach have been tested for automatic FL-IR based explosive hazard detection [8], [9]. In [8], the drawbacks of the Flat Earth 1.0 model for large standoff distances were demonstrated. The SfM approach showed good performance, but at a very high computational cost. The goal of this dissertation is to develop and test a hybrid model, referred to as Flat Earth 2.0, that has less computational cost than the general SfM approach and better accuracy than the Flat Earth 1.0 approach. The only assumption that the Flat Earth 2.0 model makes is that what the camera sees in frames $N \pm T$, where $N$ is the current frame and $T$ is a positive integer, i.e. the frames used for SfM estimation for frame $N$, is one large plane. No assumptions about camera motion or assumptions about the orientation of the imaged plane with respect to the camera are made. The assumption of local planarity is viable for in-road buried explosive hazard detection using small FOV cameras, which are needed to achieve large standoff distances, and allows the correspondence search within frames $N \pm T$ to be reduced to homography transformations estimated from keypoint matches between frames. Thus, the heavy computational burdens of the SfM approach, such as essential matrix
estimation, essential matrix refinement (in the presence of position inaccuracy), rectification, and disparity search can all be eliminated.

Given the keypoint correspondences over the N±T frames, there are two possible approaches to calculating 3D position at each pixel. For the first approach, homographies are computed using the traditional direct linear transformation (DLT) process. Those homographies are then used to determine pixel coordinate correspondences between frames, after which the iterative linear-LS method of triangulation [10], which can be efficiently implemented for large number of points, can be used to calculate 3D position. Thus, if a dense depth map is required, all pixels can be triangulated, or, if world position is needed for only a few pixels in the frame, as would most likely be the case in actual operation, triangulation can be performed only for the pixels in question. Furthermore, since the linear-LS triangulation method directly solves for the 3D position that minimizes the image projection error, it directly handles the problem of position uncertainty.

For the second approach, homographies are computed in terms of the normal vector to the plane, depth to the plane, and the rotation and translation of the camera. More specifically, the rotation and translation of the camera are determined from the INS, and only the normal vector to the plane and depth to the plane are calculated from the keypoint matches. Depth for each pixel can then be calculated directly by using the normal vector and depth to the plane, without the need to solve a system of linear equations as required by linear-LS triangulation.

Violations of the local planarity assumption will result in incorrect pixel correspondences between images, since a homography will no longer be able to model the image to image transformation. The amount of inaccuracy depends on the out-of-plane discrepancy, i.e. the
distance of the 3D point from the plane along the plane normal vector, of the terrain, as well as the distance of the 3D point and the plane from the camera. For both of the approaches to 3D position calculation under the Flat Earth 2.0 model mentioned previously, this inaccuracy in pixel correspondence will cause errors in 3D position estimation. The mathematical form of this error is derived as part of the dissertation.

A deficiency of the SfM approach of [8] under the case of local planarity is also addressed. Specifically, the method of essential matrix refinement will produce garbage output in the case of a planar scene because that is a degenerate case for essential matrix estimation. Namely, the cross product of any vector with the homography matrix will yield a valid essential matrix. This problem can be remedied using two possible approaches. The first works by introducing a bias term to the essential matrix refinement optimization which keeps the camera positions closer to the locations reported by the INS when the residual error of a homography fit to the keypoint matches is low. The second approach simply switches from the full SfM method to the Flat Earth 2.0 method if the residual homography error is below some fixed threshold.

### 1.1.2 Prescreening

Detection of BEHs using long-wave and mid-wave infrared imagery relies on two distinct effects. The first effect is due to the buried objects themselves [11], [12]. Since the buried objects possess different thermal properties than the surrounding soil, their presence disrupts the normal heat transfer through the ground. This results in slight temperature variations at the soil surface above the buried objects compared to areas free of such objects. This temperature variation causes differences in the amount of emitted infrared radiation, which depends on temperature, and manifests as local differences, i.e. blobs, in the infrared images. Example
target signatures can be seen in Figure 1.2. This effect is highly dependent on the size, material makeup, and burial depth of the object, as well as heat differential below the surface. Without adequate heat differential, no heat transfer will take place and the object will not cause differences at the soil surface regardless of its properties. Diurnal temperature variation guarantees at least two times of day, during sunrise and sunset, when subsurface soil temperatures reach equilibrium, and this difference effect will not be present. Likewise, clouds can dampen the daytime and nighttime temperature extremes, acting as a low-pass filter, resulting in smaller subsurface temperature differentials and weaker target signatures on cloudy days. On the positive side, this effect does not depend on the length of time the object has been emplaced.

The second effect is caused by disturbing the soil when burying the object [12], [13]. When the hole containing the object is dug and refilled, soil from different depths can become mixed. This can cause the soil composition at the ground surface above the object to become sufficiently different from the surface soil surrounding the burial site that it leads to differences in emitted infrared radiation even when the soil above the object possesses the same temperature as the surrounding soil. This phenomenon is sometimes referred to as the disturbed earth effect or the restrahlen effect. While disturbed earth is not dependent on the properties of the buried object, the effect does diminish over time; usually in the span of a few days to a week.

The current set of prescreening algorithms described in the literature use traditional anomaly detection techniques, such as variants of the RX detector [14], which is popular in hyperspectral imaging, Gaussian mixture models [15]–[17], the top-hat transform [18], and
intensity threshold based segmentation [19]. Unfortunately, these types of detectors are sensitive to a wide variety of anomalies and are too simplistic to achieve the false alarm and detection rates required in an operational FL-IR system. Specifically, they produce unacceptably high false alarm rates in cluttered IR images. This is primarily due to the inability to be highly shape discriminative, independent of local contrast. This causes issues with weak target signatures, as well as target signatures embedded in tire tracks. Figure 1.2 presents several examples of target signatures that are difficult for these detection algorithms to pick up at reasonable false alarm rates.

![Figure 1.2. Example FL-IR target signatures.](image)

Top row: hard to detect targets due to low contrast and obscuration by tire tracks.
Bottom row: easy to detect high contrast, unobscured targets.

Drawing from the object and pedestrian detection literature, cell-structured HOG detection algorithms using linear SVMs and integral images have shown good discrimination ability while maintaining computational efficiency [20]–[22]. In [9], [23], HOG and Shearlet features showed good performance for BEH detection in patches extracted from FL-IR imagery at prescreener alarm locations. Thus, a sliding window, cell-structured detector is likely a good fit for FL-IR prescreening. In this work, a new prescreening algorithm using multi-scale HOG (MS-HOG) features, integral images, and a linear SVM is proposed with the aim of increasing detection rates and decreasing false alarm rates relative to the older ensemble of RX filters.
prescreening algorithm. The older algorithm is currently state-of-the-art with respect to BEH detection in infrared imagery, and operates by running an ensemble of localized RX filters, each trained using a genetic algorithm to reach a specific detection rate on the training data, and then fusing their results via weighted mean-shift. The cell structured patch classifier for the new prescreening algorithm will be trained using ground truth to image projections to generate positive examples and random sampling combined with hard negative mining, the process of using the current incarnation of the classifier to find misclassified examples in the training data, to generate negative examples. It is also shown that for a classifier using a single HOG cell size, i.e. the size of the localized rectangular patches in which features are extracted, boxcar filtering, equivalent to convolution with a square filter of all ones, can reduce the computational cost for dense feature sampling compared to using integral images.

It is not immediately clear that the training approach outlined above will work well since the resulting training data will be quite noisy. First, projections of the ground truth locations into the images is far from perfect. It is highly unlikely that the pixel locations obtained will fall directly in the middle of the visible target signatures in the images. Second, it is not possible to know whether a signature actually exists in the imagery for each target. Therefore, this approach will assuredly result in many positive training examples for which there is no visible signature in the image patch. Third, clutter in the lane can result in image signatures that closely resemble target signatures. Thus, there will be negative training examples which look like the positive examples the classifier should detect. Relying on human intervention to cleanse the training data is possible, and could lead to improved results. However, it would require significant time investment. In this dissertation, a simpler case is investigated. Specifically,
whether eliminating targets which lack a visible IR signature from the training data improves the resulting classifier.

Histogram features, such as the HOG features mentioned above and histograms of local binary patterns (LBP) [24], have become increasingly popular in the computer vision literature for their combination of simplicity and performance. Recent research has shown that classification results using histogram features can generally be improved by using non-linear kernels based on histogram similarity measures, such as the histogram intersection kernel and the chi-square kernel [25]. Unfortunately, use of non-linear kernels can dramatically increase the computational complexity of the classifier during both training and prediction. For the case of SVMs specifically, stochastic gradient descent training methods which rely on the use of the linear kernel are orders of magnitude faster than any training algorithm using a non-linear kernel [26], [27]. Furthermore, prediction when using non-linear kernels suffers from having to evaluate the kernel function for every support vector. This means that the computational complexity of prediction increases at least linearly in the number of support vectors when compared to prediction with the linear kernel. Therefore, even relatively small numbers of support vectors for large training sets, such as hundreds or a few thousand support vectors, results in a very large computational expense. These drawbacks have led to most time sensitive applications using linear classifiers such as SVM with linear kernel or least squares classification.

Fortunately, recent work has shown that prediction time of additive kernels can be made independent of the number of support vectors by using a look-up table approach [28]. Other work has shown that for many of these additive kernels explicit feature mapping into a feature space only three to five times larger than the original can adequately approximate the
high-dimensional feature space implied by the kernel [29]. This allows using fast linear training methods after mapping to the new feature space. Unfortunately, this explicit feature mapping can involve evaluating costly mathematical functions, which can make prediction much slower since each feature vector must be mapped to the new space before the linear classifier can be evaluated. In this dissertation, it is shown how the look-up table approximation method for fast prediction of additive kernels can be used with the explicit feature map approach to allow fast prediction independent of the complexity of the mapping itself. Results for the proposed prescreener using the linear kernel and additive non-linear kernels such as histogram intersection and additive chi-square are reported.

1.2 Contributions

The first contribution of this dissertation is to analyze, implement, and test a new model for geo-referencing FL-IR imagery captured from a single camera on a moving platform for use in buried explosive hazard detection. Current algorithms for geo-referencing under these conditions are either too slow or make unrealistically restrictive assumptions about the camera motion and terrain resulting in inaccuracy. The proposed method achieves a middle ground that relaxes the most restrictive motion and terrain assumptions of the faster method, but is still much faster than algorithms for the completely unrestricted case.

The second contribution of this dissertation is to implement and test a new sliding window based prescreening algorithm for buried explosive hazard detection in FL-IR imagery utilizing MS-HOG features efficiently computed using boxcar filtered images. The current state-of-the-art prescreening algorithm for buried explosive hazard detection in FL-IR relies on a simple dual sliding window anomaly detection algorithm, which is fast, but does not achieve
adequate detection and false alarm rates in difficult cases, such as when target signatures are obscured by tire tracks or have low contrast. The proposed algorithm shows much better performance under these kinds of difficult conditions.

The third contribution of this dissertation is to combine the look-up table approximation approach for fast prediction of kernel SVMs when using additive kernels with the explicit feature mapping approach that allows for fast SVM training when using additive kernels. This allows for both fast approximate training and prediction using additive non-linear kernels such as histogram intersection and additive chi-square. Results for the proposed prescreener using a SVM classifier with the linear kernel and the additive non-linear kernels are presented.

1.3 List of Publications

This dissertation builds on work described in the following publications:


Chapter 2: Background

2.1 Geo-referencing

Geo-referencing of imagery has a long history in the literature and has been used in mobile robotics [30], remote surveillance [31], and airborne mine detection [32] among other areas. Before the introduction of GPS and inertial measurement systems, geo-referencing of imagery, primarily aerial photography, was performed using ground control points and manual labor. With the introduction of high accuracy navigation systems based on GPS and inertial measurement units this process became more automated [33]. Many ground based applications use stereo cameras [34], [35], depth cameras [36], or laser rangefinders [37], [38] to determine the topography of their surroundings. However, most of these applications do not require high accuracy at range, and do not need to precisely relate points to another coordinate system such as UTM. The need for high precision geo-referencing of a single camera’s imagery at distances up to thirty meters collected from a continuously moving vehicle presents a difficult challenge. Another major difference between the requirements of our forward-looking system versus that of single camera aerial systems is that the primary direction of motion is along the optical axis of the camera, making accurate triangulation more difficult.

2.1.1 Camera Model, Image Formation, and Calibration

To assign physical coordinates to pixels in a captured image, the image formation process must be modeled mathematically. For cameras with normal fields of view the pinhole camera model is a good, widely used approximation [39]. It maps coordinates in the camera
frame of reference to coordinates on the image plane using a prospective transformation. The mathematical model is given by Equation 2.1, and is depicted in Figure 2.1.

\[
\begin{bmatrix}
I_u \\
I_v \\
1
\end{bmatrix} = \begin{bmatrix}
f & \tau & c_u & X_x \\
0 & \alpha f & c_v & X_y \\
0 & 0 & 1 & X_z
\end{bmatrix}
\]

FIGURE 2.1. Diagram of the pinhole camera image formation model. Each image point corresponds to the intersection of a line from a 3D point in the camera reference frame through the center of projection (C).

Here, \(I_u\) and \(I_v\) are the 2D coordinates in the image plane, \(f\) is the camera focal length, \(\tau\) is the skew parameter, which handles misalignment between the image plane axes in the sensor array, \(\alpha\) is the aspect ratio, which handles non-square pixels, \(X\) is the 3D location of the point in the camera reference frame, and \(c_u\) and \(c_v\) are the principal point offsets, which handle the principal point being offset from the center of the image plane. This more general model is usually not needed with modern cameras since they rarely suffer from manufacturing defects which lead to non-zero skew or significantly decentered images. By assuming zero skew, unit aspect ratio, principal point in the center of the image, and no distortion the model reduces to Equation 2.2, where the only necessary intrinsic camera parameter is the focal length.
Since the INS and camera will not usually share the same reference frame, we must first transform from world coordinates to the INS reference frame and then from the INS reference frame to the camera reference frame in order to determine the pixel coordinates of a world location. This formula is given in Equation 2.3.

\[
\begin{bmatrix}
I_u \\
I_v \\
1
\end{bmatrix} =
\begin{bmatrix}
f & 0 & 0 \\
0 & f & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X_x \\
X_y \\
X_z
\end{bmatrix}
\]

2.2

\[
\begin{bmatrix}
I_u \\
I_v \\
1
\end{bmatrix} =
\begin{bmatrix}
f & 0 & 0 \\
0 & f & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
R_{KC} & T_{KC} \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
R_{WI} \\
T_{WI}
\end{bmatrix}
\begin{bmatrix}
W_x \\
W_y \\
W_z
\end{bmatrix}
\]

2.3

Here, \( W \) is a three dimensional world coordinate, \( R_{WI} \) and \( T_{WI} \) are the 3x3 and 3x1 rotation and translation from world coordinates to the INS reference frame, and \( R_{IC} \) and \( T_{IC} \) are the 3x3 rotation and 3x1 translation from the INS reference frame to the camera reference frame. The camera and INS are assumed to be rigidly mounted on the vehicle, ideally as close together as possible, and therefore \( R_{IC} \) and \( T_{IC} \) do not change over time. Obviously, \( R_{WI} \) and \( T_{WI} \) are known because the INS knows its position in the world coordinate system, albeit with some slight inaccuracy. The world coordinate system is defined by: \( W_x = UTM\ Easting \) \( W_y = UTM\ Northing \) \( W_z = Altitude \).

The simplifying assumptions for the camera model made above also help with the calibration process. Because the infrared camera is generally mounted high up on a vehicle and has a large effective focal length for zooming in down-track, traditional camera calibration techniques such as checkerboard patterns are difficult to use. We generally rely on placing a few
fiducial targets in front of the vehicle with known world locations. The vehicle then drives around capturing images of the fiducial targets. The targets are then manually located in the imagery, and the camera focal length and the rotation and translation from INS to camera reference frame are estimated by minimizing the projection error. Since the fiducial targets might lie in a plane, or very close to it, the parameter estimation process is guided using keypoint matches between frames as described in [8]. Specifically, the disagreement of point matches -- captured in frame sequences where the camera is undergoing simultaneous rotation and translation in multiple dimensions -- with the fundamental matrix is computed. Since 1.) the number of training examples from fiducials is relatively small, 2.) the rotation and translation between keypoint matches is limited due to field of view and depth from the camera, and 3.) the intrinsic and extrinsic parameters are being estimated simultaneously, using the full camera model (Equation 2.1) tends to badly overfit the data. This issue is significantly reduced by using the single parameter camera model (Equation 2.2).

### 2.1.2 Image Pixel to World transformation

Equation 2.3 describes how to map a world location into image coordinates, but it is the inverse problem, mapping image coordinates to world coordinates, that needs to be addressed for a target detection system to operate. Obviously, given only a single 2D image coordinate it is impossible to transform back to a 3D coordinate because the depth along the ray is not known. There are two possible ways to address this problem. The first is to assume a known depth for each pixel in the image. Under this assumption, a pixel coordinate can always be mapped to a coordinate in the camera reference frame. This is how the Flat Earth 1.0 model operates. The second solution is to find the same physical world location in other images captured by the
vehicle from different vantage points and use triangulation to compute the true depth of the point. This is how the structure from motion approach operates.

2.1.3 Flat Earth 1.0

The simplifying assumptions of the Flat Earth 1.0 model are two-fold. First, the world is parallel to the $W_{xy}$ plane, i.e. $W_z$ is some fixed value, $A$, that never changes for all points in the world. Second, the INS and camera rotate only about the $W_z$ axis and are a fixed offset, $B$, along $W_z$ from the world plane. This second assumption comes from the proposition that if the camera and INS are rigidly mounted on the vehicle, and the vehicle travels only on the $W_{xy}$ plane, then there will never be rotations about the $W_x$ and $W_y$ axes. Under these rather drastic assumptions, there are a number of simplifications that can be made to Equation 2.3, as shown in Equation 2.4.

\[
\begin{bmatrix}
I_u \\
I_v \\
1
\end{bmatrix} =
\begin{bmatrix}
f & 0 & 0 \\
0 & f & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
R_1 & R_2 & R_3 & T_1 \\
R_4 & R_5 & R_6 & T_2 \\
R_7 & R_8 & R_9 & T_3
\end{bmatrix}
\begin{bmatrix}
\cos \theta & -\sin \theta & 0 & T_{WX} \\
\sin \theta & \cos \theta & 0 & T_{WY} \\
0 & 0 & 1 & -(A + B) \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
W_x \\
W_y \\
A \\
1
\end{bmatrix}
\]

Equation 2.4

Here, the rotation and translation matrices $R_{WI}$ and $T_{WI}$ have been expanded to their individual elements. Furthermore, the $R_{WI}$ transformation has been written explicitly as only Z-axis rotation. $A$ is the fixed altitude value for $W_z$, and $B$ is the height of the INS above the world plane. This reduces to Equation 2.5.

\[
\begin{bmatrix}
I_u \\
I_v \\
1
\end{bmatrix} =
\begin{bmatrix}
f & 0 & 0 \\
0 & f & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
R_1 & R_2 & R_3 & T_1 \\
R_4 & R_5 & R_6 & T_2 \\
R_7 & R_8 & R_9 & T_3
\end{bmatrix}
\begin{bmatrix}
\text{INS}_x \\
\text{INS}_y \\
-B \\
1
\end{bmatrix}
\]

Equation 2.5
From Equation 2.5, it can be seen that the actual value of A has no relevance. Only the offset in the Z dimension between the INS and points in the world plane matters. Thus, we can set A to whatever value we wish without changing the result. In fact, we don’t need to measure altitude, pitch, or roll for this model. Equation 2.5 can be reduced further still by combining constants, as shown in Equation 2.6.

\[
\begin{bmatrix}
I_u \\
I_v \\
1
\end{bmatrix} =
\begin{bmatrix}
f & 0 & 0 \\
0 & f & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
R_1 & R_2 & T_1 \\
R_4 & R_5 & T_2 \\
R_7 & R_8 & T_3
\end{bmatrix}
\begin{bmatrix}
\text{INS}_x \\
\text{INS}_y \\
1
\end{bmatrix}
\]

Equation 2.6

Now the real benefit of the Flat Earth 1.0 model can be seen. Specifically, instead of three unknown world coordinates there are only two, and the transformation, a 3x3 perspective transformation, is invertible. The disadvantage is that if the INS, and therefore the camera and vehicle, undergoes a roll or pitch rotation, or the INS translates in \(W_z\), altering the value of B, the model will be incorrect. In the next few pages the effect of these errors on the accuracy of the image to world transformation will be quantified.

The easiest error to analyze is that of a changing offset, B, between the world plane and the INS in the altitude dimension. A common phenomena that could cause this is if the terrain in front of the vehicle gradually slopes upward or downward due to a hill. For this experiment, as well as those that follow, the simulated camera is positioned three meters above the world plane with a downward look angle of 11.3 degrees, which would center the camera on the ground at 15m away, and resolution of 640x480 pixels. Figure 2.2 shows the error in accuracy of the Flat Earth 1.0 model for points that are five, ten, twenty, and thirty meters directly in front of the vehicle, as the true offset in the altitude dimension differs from -0.4m to 0.4m from the
assumed offset from the world plane. These calculations assume perfect, sub-pixel accuracy, i.e. pixel coordinates resulting from projection of world coordinates into the image are not rounded to integer values, of the point in the image plane. Errors are shown for both the world to image projection (left) and the image to world projection (right). The image to world projection error is signed. Positive values indicate further from the vehicle, while negative values indicate closer.

**FIGURE 2.2.** Top: World to image transformation error in pixels as the altitude offset between the INS and the world plane differs from the assumed offset of the Flat Earth 1.0 model for points 5, 10, 20, and 30 meters in front of the vehicle. Bottom: Image to world error for the same configuration in meters.
Due to the effects of perspective projection, as a points’ distance from the camera increases, the amount of change, in terms of pixel coordinate distance, in its projection into the image decreases. That is to say, as the point gets farther from the camera it tends to project to the same position in the image plane. Thus, points that are far from the camera have small image localization error, but, at the same time, accuracy in assigning world position to the point decreases because that image pixel corresponds to a large area on the ground. Thus, at the same offset in Figure 2.2, the pixel error decreases and the world error increases as the distance from the camera increases, just as one would expect. In fact, if world accuracy of less than 0.5m is required then at five meters from the camera differences up to 0.3m, which corresponds to a slope angle of 3.4% or a percentage grade of 6%, can be tolerated. Percentage grade is the ratio of rise over run. A 6% grade is, in fact, rather steep, and is the limit for interstate highways in the United States. However, at 30m only differences up to 0.05m, a slope angle of 0.1% or percentage grade of only 0.17%, which is hardly perceptible, can be tolerated. Thus, due to the perspective nature of the image formation process the accuracy of the Flat Earth 1.0 model degrades much more rapidly as distance from the camera increases when the underlying assumptions are violated. This can be seen when the pitch, i.e. rotation from front to back, of the camera changes as well. Figure 2.3 shows the image and world error as the pitch of the camera is altered from the assumed pose. This could be caused by bumps or washes in the road.

At five meters, pitch changes of one degree have little effect on the image to world accuracy. Image accuracy suffers, but this is because the image resolution on the ground, in terms of pixels per square meter, is quite large. However, at thirty meters, pitch changes of only 0.2 degrees result in geo-referencing errors of roughly one meter.
Based on these results, it is clear that while the Flat Earth 1.0 model is viable for five meter standoff and accuracies of 50 centimeters, it is completely unusable at thirty meters if the same level of accuracy is required. These observations are what prompted us to develop a structure from motion approach as described in [8] when the army moved from short (4-6m) to longer (10-30m) standoff distance data captures.

FIGURE 2.3. Top: World to image transformation error in pixels as the pitch angle between the INS and the camera differs from the assumed angle of the Flat Earth 1.0 model for points 5, 10, 20, and 30 meters in front of the vehicle. Note that the curves for 10, 20, and 30 meters are almost identical.
Bottom: Image to world error for the same configuration in meters.
2.1.4 Structure from Motion

Generally speaking, structure from motion refers to recovering the 3D structure of a scene from a sequence of 2D images. In the literature, this process has also been referred to as multiple baseline stereo and synthetic stereo, which share many similarities. At their heart, all of these techniques rely on the fundamentals of traditional stereo vision, i.e. epipolar geometry and the essential and fundamental matrices.

Epipolar geometry describes the geometric relationship between 3D points and their 2D image projections when two pinhole cameras view the same 3D scene. This is depicted in Figure 2.4, where \( P \) is a 3D point, \( P_L \) and \( P_R \) are the corresponding 2D image coordinates of that point in the left and right cameras, \( O_L \) and \( O_R \) are the centers of projection of the two cameras, and \( E_L \) and \( E_R \) are the epipoles. The centers of projection, \( O_L \) and \( O_R \), define which rays passing through their respective image plane are captured. That is, only rays which pass through the camera’s center of projection are captured. The epipoles are the points in each image plane where the line connecting the centers of projection of the two cameras intersects each image plane. In other words, these are the points where the other camera’s center of projection projects onto the current camera’s image plane. It is possible for one or both epipoles to lie at infinity, for instance, if both cameras are facing directly forward. In that case, the line connecting their centers of projection will never intersect either image plane. The main observation of epipolar geometry, in relation to stereo vision and triangulation, is that if the projection of point \( P \) onto one of the image planes is known, and the relative positioning of the cameras with respect to one another is known, then the line on the other camera’s image plane on which the projection of point \( P \) must lie can be determined. This is easy to see by inspection of Figure 2.4, where if \( P_R \)
is known, the line \( E_R - P_R \) is known, and the line \( E_L - P_L \) on which \( P_L \) must lie is known. Such a line is referred to as an epipolar line. This expresses the observation that a ray which projects to a single pixel in one camera, is seen as a line in the other camera.

![Figure 2.4. Epipolar geometry. Depiction of two cameras viewing the same point, \( P \). \( P_L \) and \( P_R \) represent the projection of \( P \) into the image plane of the left and right camera respectively. \( E_L \) and \( E_R \) are the epipoles. \( O_L \) and \( O_R \) are the centers of projection of the left and right camera.](image)

The relationship between corresponding points in the two image planes is captured by the essential matrix, which is defined in Equations 2.7 and 2.8. Here, \( I_L \) and \( I_R \) are normalized homogenous coordinates in the left and right images respectively, and \( R \) and \( t \) are the rotation and translation from the left camera’s frame of reference to the right camera’s frame of reference. \([t]_s\) denotes the matrix representation of the cross product. That is, the 3x3 skew-symmetric matrix where \([t]_s\) multiplied by a vector is equal to the cross product of \( t \) and that vector.

\[
(I_R)^T E(I_L) = 0 \quad 2.7
\]

\[
E = R[t]_s \quad 2.8
\]

\[
F = K^{-T} E K^{-1} \quad 2.9
\]
The essential matrix is just a mathematical expression of the idea that the lines P-O_L and P-O_R, along which the point P projects into the image planes, form a plane with O_L-O_R. This plane is sometimes referred to as the *epipolar plane*. In Equations 2.7 and 2.8 we see that \( I_R \) is rotated into the left camera’s frame of reference, and then the dot product between that result and the cross product of \( t \) and \( I_L \) is computed. The cross product of \( t \) -- which is really the line segment O_L-O_R from Figure 2.4 -- and \( I_L \) -- which is O_L-P_L in Figure 2.4, will give the normal vector to the O_L-O_R-P plane. If P-O_R lies in the plane, as it should, then the dot product of it with the normal vector to the plane should be zero, which is exactly the relationship the essential matrix encodes. A key point is that the magnitude of the translation vector does not matter here, only the direction. Thus, the essential matrix is unique only up to a scale factor, resulting in five degrees of freedom, three for rotation and two for translation. The fundamental matrix, given in Equation 2.9, is a generalization of the essential matrix which incorporates the intrinsic parameters of the camera, K. This means the fundamental matrix relates actual pixel coordinates, not normalized coordinates as the essential matrix does.

The traditional approach to stereo vision has three steps. The first step is rectification. Rectification transforms the two images such that corresponding epipolar lines lie along the same image row, i.e. so that they become collinear. Therefore, after rectification, the epipolar line for a pixel in row y of one image is the set of pixels in row y of the other image. Rectification isn’t strictly necessary, but it significantly speeds up the second step, correspondence search between images, by restricting it to one dimension. The standard rectification method, known as planar rectification [40], uses linear transformations to transform the two images into a common image plane. However, this approach will fail, i.e. produce unusually large, unbounded,
or badly warped images, when the separation between the two cameras has a large forward component. Since the main component of camera motion in a forward-looking system is in the forward direction, planar rectification is unusable. Polar rectification [41], as used in [8], can work in the case of forward motion, but is significantly slower. Polar rectification requires the essential matrix.

As mentioned, the second step in a traditional stereo vision approach is correspondence search between images. Correspondence search is well studied in the literature [42]–[44] with methods of widely varying complexity. As in [8], the semi-global block matching algorithm from the open source computer vision library OpenCV [45] is used for all experiments involving SfM.

The final step is triangulation. Triangulation is also well studied in the literature. For brevity, a long discussion is avoided here and the reader is referred to our previous work [8] and the excellent works of [10] and [46]. In brief, if two points satisfy the essential matrix equation then all triangulation methods will yield the same result. Therefore, the point correspondences are first minimally altered using Kanatani’s Optimal Correction algorithm [46] so that they agree with the fundamental matrix. Triangulation is then performed using the basic mid-point method. Kanatani’s algorithm alters the coordinates of a pair of pixels such that they agree with the fundamental matrix, while at the same time minimizing the total displacement (in square pixels). It does this using a gradient approach. A much more complicated algorithm that finds the same solution is that of Hartley and Sturm [10]. Hartley and Sturm’s method is considered the gold standard for triangulation in the literature.

The generalization from traditional, two camera, fixed baseline, stereo vision to multiple baseline stereo is relatively straightforward. The only difference is that there are more than two
cameras. The two main approaches for dealing with this are 1.) treat each set of image pairs independently and merge the results or 2.) perform N-view triangulation. Multiple baseline stereo is itself very similar to synthetic stereo, the only difference being that in synthetic stereo there is a single camera moving through space, where two images captured by the camera from multiple positions are used to simulate a two camera stereo setup. The challenges in synthetic stereo are 1.) the baselines are continuously changing as the camera moves through space and 2.) how to get accurate position information for the cameras. Likewise, structure from motion can be seen as the combination of synthetic stereo and multiple baseline stereo. Structure from motion can be performed with or without camera position information, as well as with or without knowledge of the intrinsic parameters of the camera. In the case of no motion information and no intrinsic parameter information, the camera poses and projection matrices must all be estimated from point correspondences between images. This is a very difficult task, and, at best, the 3D scene structure can only be estimated up to an unknown scale factor. In other words, it is impossible to know whether the camera is far away and the scene elements are large or the camera is close and the scene elements are small. With knowledge of the camera positioning and its intrinsic parameters, the task becomes much easier.

One major difficulty in our forward looking setup is that the position information is not exact. There are errors in both position and pose, and naively using the position and pose information reported by the INS to construct the essential matrices will lead to slight errors. These errors will differ between each pair of views. That is, if the same point P is observed in three frames: T, T+1, and T+2, and the reported INS position information is exact, then all pairwise essential matrices will agree with the observed point correspondences. Furthermore, the depth maps computed from each frame pairing using triangulation will be in agreement.
However, if the position information is not exact the essential matrices will be incorrect. The triangulation process will then adjust the observed point matches in each pair to agree with an incorrect essential matrix. This will lead to correlated triangulation errors across the image, and, due to the random nature of the position error and the resulting errors in the essential matrices, that correlated triangulation error will differ for each image pair. Therefore, when the depth maps are merged these slight differences will not only cause inaccuracy in the depth field, but in the case that a section of the image is observed by only a single image pair there will be visible discontinuities as shown in Figure 2.5.

In [8], this problem was handled by performing what is referred to as \textit{essential matrix refinement} between each pair of views, and jointly between the whole set of views. This process involved minimizing the error between scale-invariant feature transform (SIFT) [47] keypoint correspondences as measured by the required displacement computed using Kanatani’s Optimal Correction algorithm using the implied essential matrices. An initial guess for the essential matrices was computed using the position and pose reported by the INS. These position and pose estimates were then refined using a constrained local search algorithm that attempted to minimize the correspondence error while restricting the position to be within a cube of 20cm around the position reported by the INS. While this method works well, its main drawback is computational complexity.
FIGURE 2.5. Top row: distance from camera estimates using position and pose as reported by INS for two different cameras. Left: Camera 1. Right: Camera 2. Blue indicates near and red indicates far. Depth maps were created by merging depth estimates from multiple frame pairs. Horizontal discontinuities where triangulation errors differ between frame pairs due to error in reported position are denoted by the black rectangular boxes. Bottom row: reconstruction using essential matrix refinement. Discontinuities eliminated.

2.1.4 Deficiency in Essential Matrix Refinement for Planar Scene

The essential matrix refinement algorithm used in [8] has one major deficiency which became very apparent when it was used on very narrow field of imagery. It will not work if the keypoint correspondences are coplanar. That is, if all of the world locations corresponding to the keypoint matches lie in the same world plane. The reason for this is that a planar scene is a degenerate case for essential matrix estimation. Namely, the cross product of any vector with
the homography matrix will yield a valid essential matrix. Recall Equations 2.7 and 2.8, and
Equation 2.10 below for a homography relating normalized image coordinates between two
images of the same plane in space.

\[
(I_R)^T E(I_L) = 0 \tag{2.7}
\]

\[
E = R[t] \tag{2.8}
\]

\[
H = R - \frac{m^T}{d} \quad I_1 = H * I_2 \quad I_1 = R^* I_2 - t^* \left( \frac{n^T I_2}{d} \right) \tag{2.10}
\]

If we take the cross product of a random vector, \( Q \), with \( H \), and plug the result into
the essential matrix equation we get the relation in Equation 2.11.

\[
(I_R)^T [Q]_x H(I_L) = (I_R)^T [Q]_x (I_L) = 0 \tag{2.11}
\]

Which clearly holds, since \( H(I_L) \) will be equal to \( I_R \), up to some scale factor which won’t
affect the direction of the vector, and the cross product of \( I_R \) with a random vector will give a
vector orthogonal to \( I_R \). Therefore, the dot product of \( I_R \) with that result will be zero. Thus, if
all point correspondences are related by the same homography, there is no unique rotation and
translation direction. This causes the essential matrix refinement to produce garbage estimates.

Two possible fixes for the above issue are 1.) detect coplanarity of the point
correspondences and use a fallback method that does not require the essential matrix, or 2.)
bias the optimization toward the position and pose reported by the INS when the point
correspondences become closer to coplanar. The second method will be investigated as part of
this dissertation.
2.2 FL-IR Prescreening

Prescreening algorithms have a long history in the explosive hazard detection literature [48], [49]. This is a result of the need to process large amounts of data in real-time, and to enable more sophisticated classification algorithms to focus on the anomalous areas. For downward looking GPR, a popular prescreening algorithm uses an adaptive 2D least mean squares (LMS) approach where weights are applied to a set of local background samples to predict the response at the center location. If the error between the predicted response and the observed response is large, an anomaly is flagged. As the vehicle moves down the lane, the weights are updated after each evaluation by applying a gradient descent update term based on the squared error between the predicted and observed responses. This model is depicted in Figure 2.6 (reproduced from [49]) and Equation 2.12.

In Figure 2.6, the clear square is the center sample which is to be predicted from the surrounding samples indicated by the black circles. Each surround point is given a weight, and the prediction is simply the dot product of the surround sample values with the weight vector. This is shown in Equation 2.12, where \( \mathbf{w}_i \) is the current weight vector, \( \mathbf{u} \) is the vector of surrounding samples, \( \mathbf{e} \) is the prediction error, \( \eta \) is the learning rate, and \( \mathbf{w}_{i+1} \) is the updated weight vector. This 2D algorithm is generalized to 3D by quantizing the depth dimension into a fixed number of bins, applying the 2D algorithm to each of the depth samples inside a bin, and then summing the squared prediction error over all depth samples. This creates one anomaly value per depth bin.
FIGURE 2.6. Depiction of center sample and surrounding neighborhood used in 2D LMS GPR prescreening algorithm. The center value is predicted using a linear combination of the surround samples. A large error in the prediction is flagged as an anomaly.

\[
y = w_i^T \ast u
\]
\[
e = d - y
\]
\[
w_{i+1} = w_i - \eta \ast e \ast u
\]

While this 2D LMS approach works well for downward looking GPR data, it is not easily applicable to FL-IR imagery which is more complex and not well modeled by linear weights of surrounding pixel values. Such a model implies a slowly varying, linear relationship between the center sample and the surround, which is most definitely not the case for FL-IR imagery. FL-IR prescreeners in the literature have relied on the traditional localized RX (Reed-Xiaoli), dual sliding window, anomaly detection algorithm [9], [50], referred to as a size-contrast filter, as well as methods based on maximum stable extremal region (MSER) segmentation [17] and Gaussian mixture models [16]. In performance tests, the size-contrast filter algorithm has shown the best performance.
A major issue for all FL-IR prescreeners that operate directly on the incoming images from the IR camera is that the same world location can, and usually will, be seen in multiple images. Multiple images can mean as many as 50 images or more depending on frame rate and vehicle speed. Thus, simply running an image space algorithm and projecting that algorithm’s detected anomalies on each frame to world coordinates will result in many, slightly different, hit locations for each object. A FL-IR prescreening algorithm must have an efficient method for dealing with this redundancy, such as clustering or quantization.

2.2.1 Size-Contrast Filter Ensemble

In [51], a prescreening algorithm for FL-IR was developed that uses a technique very similar to the localized RX anomaly detection algorithm. In this algorithm, two windows, an inner window and a surrounding outer window are slid across the image. At each position the mean and variance of the pixels in the inner and outer windows are computed. An anomaly score, $D_M$, is then calculated using Equation 2.13 by treating the inner window mean as a sample from the single variable normal distribution defined by the mean and variance of the outer window. Figure 2.7 shows a depiction of the algorithm.

$$D_M = \sqrt{\frac{(\mu_I - \mu_O)^2}{\sigma_O^2}}$$  \hspace{1cm} 2.13

Equation 2.13 is simply the Mahalanobis distance between the point defined by the inner window, and the distribution defined by the outer window. In [51], the Mahalanobis distance is replaced by the Bhattacharyya distance, which can be seen as treating the inner window as a distribution itself, instead of as a point, and measuring the distance between the two
distributions. Furthermore, [51] uses an ensemble of these dual sliding window detectors, each trained to achieve a different detection rate at the minimum possible false alarm rate on the training set.

![Illustration of dual sliding window detector. Left: red window represents the inner window and the green window represents the outer window. Right: parameterization of the two windows.](image)

Reference [51] addresses the problem of multiple looks, as well as the problem of non-maximal suppression of the anomaly detector, by first performing weighted mean-shift clustering of the hits from each frame. This is followed by weighted mean-shift clustering of the combined set of hits from all frames. Finally, the hits from each individual anomaly detector are combined using weighted mean-shift clustering.

This algorithm works well when the targets are clearly defined blobs, such as those shown in the bottom row of Figure 1.2. However, when the targets are embedded in tire tracks, or the contrast is low compared to the image noise, as seen in top row of Figure 1.2, this method does not perform as well. This drawback is what has motivated the development of a new prescreening method in this dissertation. Several improvements are possible to this algorithm which could help to address the deficiencies mentioned. One is the inclusion of a guard band. A guard band acts as a separation region between the inner and outer window, and
would help deal with irregular target shapes and target size variation. Another improvement is discontinuity preserving image denoising, such as median filtering, prior to processing to help deal with the noise issue.

Similar RX detection algorithms were developed in [50]. The main idea behind this work was to address the dependence of target shape on distance from the vehicle and its negative impact on detection performance. Two ideas were presented to address the problem. The first involved slicing the original images horizontally to create multiple views of the lane, each corresponding to a specific distance from the vehicle. The second involved varying the size of the inner and outer RX windows as they are slid across the image so that the size of the detector in world coordinates remains the same. These two ideas are presented as separate detection algorithms. For the first algorithm, the imagery is segmented into multiple horizontal slices corresponding to specific distance ranges from the vehicle. These slices are then used to build multiple plane-views of the lane on which detection algorithms can be run. The intuition is that by restricting each plane-view to imagery at a fixed distance from the vehicle the size of the targets will not vary due to distance. Detections on each plane-view can then be merged or aggregated. Unfortunately, the plane-view framework, while showing promise, is computationally burdensome. The second method, referred to as FLRX, based on modifying the size of the RX windows, requires less overhead and can work within the constraints of integral image calculation of the RX responses. It showed improved results as well. One unanswered question is whether simply restricting the range of the detector to a small interval, using geo-referencing without resorting to the plane-view framework, would achieve the same results. If multiple ranges are indeed necessary, a separate detector could be trained for each interval.
2.2.2 GMM

Gaussian Mixture Models are traditionally used for background subtraction in video taken with static cameras. However, in [16], a GMM was used for BEH detection in FL-IR imagery. GMM background subtraction works by modeling the distribution of pixel values at each discrete location in the image with a GMM of $K$ components. When a new image is processed, the probability of each observed value, $x_i$, given the GMM at its image location, $\lambda_i$, $\lambda = \{\pi_r, \mu_r, \Sigma_r\} ; r = 1, \ldots, K$, is computed using Equation 2.14.

$$p(x_i | \lambda_i) = \sum_{r=1}^{K} \pi_r N(x_i | \mu_r, \Sigma_r) \quad \text{s.t.} \quad \sum_{r=1}^{K} \pi_r = 1 \quad \pi_r \geq 0$$  \hspace{1cm} 2.14

Here, $N(x_i | \mu_r, \Sigma_r)$, $r = 1, \ldots, K$, are the Gaussian distributions with mean $\mu$ and covariance matrix $\Sigma$. $\pi_r$ are the mixture weights, and $i$ indexes the image location. Pixels with probabilities below a certain threshold are considered foreground, while pixels with probabilities above the threshold are considered background. If a binary decision is not required, the probabilities can be interpreted as a background confidence. In order to deal with dynamic backgrounds, the GMM at each pixel location is updated over time.

Applying GMM background subtraction to images from a moving camera presents unique challenges. Specifically, when the camera moves the GMM models may no longer be valid. If the camera quickly transitions from viewing a road to viewing a house, the entire image will be detected as foreground until the models have time to adapt to the change. Thus, there is a fragile interplay between the GMM update speed and the movement speed of the camera. FL-IR presents other issues with road effects such as tire tracks, wash areas, and berms. If the
vehicle drives approximately straight down the road, the down-track effects, such as tire tracks and the berm, should align from one frame to another, but this will never be perfect. In [16], the number of GMM components and the thresholds were set taking these types of effects into account. This method worked ok for data with clear, distinct, blob-like target signatures. However, the false alarm rate was rather high. The main drawback of this GMM approach is that a per-pixel GMM has no way to take the shape of objects into account. It is literally looking at each pixel intensity value individually to decide whether an anomaly is present. Thus, low contrast target signatures that have well defined shape will be hard to detect. This could be improved by modeling gradients instead of pixel values, and by modeling distributions within blocks instead of individual pixel locations.

2.2.3 MSER

In [17], the maximally stable extremal region (MSER) algorithm [52] was used as a prescreener for FL-IR imagery. MSERs are regions, connected sets of pixels, that are either all darker or all brighter than their surroundings, and that are stable across a range of image intensity thresholds. Stable means that the change in area of the connected component is small relative to the change in binarization threshold. Such regions are a good description of the blob-like target signatures found in FL-IR imagery. The MSER algorithm has three main drawbacks when used as a detection algorithm. The first, like GMM, is that it does not explicitly model shape. However, post processing of the detected MSERs could prune regions based on their size, shape, and orientation. Second, the returned MSERs can overlap or be nested inside one another. When this occurs it can be difficult to tell which region to use. Third, and most significantly, target signatures embedded in tire tracks would not be detected as MSERs because
the image intensity variations would cause breaks in the regions. This is also a problem with the less well defined target signatures that lack clear borders. Furthermore, as was the case with GMM, the MSER algorithm is highly dependent on contrast for detection. Low contrast targets will be very difficult to detect.

### 2.2.4 HOG features

Histogram of oriented gradients (HOG) is a feature descriptor that models the occurrence of different gradient orientations in localized areas of an image. It is one of the most popular features in the computer vision field, and has been used for pedestrian detection [20], face recognition [53], object detection [54], and image classification [55]. The popularity of HOG features arises from their simplicity and ease of computation, as well as their invariance to brightness, contrast (with normalization), and local deformation.

The first step in the computation of HOG descriptors is to compute the horizontal and vertical gradient at each pixel in the image. Usually this is done using the basic central differences operators shown in Equation 2.15, where \( g^H_{x,y} \) is the horizontal gradient at location \((x, y)\) in image \( I \), and \( g^V_{x,y} \) is the vertical gradient. However, more complex methods may be used as will be discussed in Chapter 4. Next, the gradient orientation, \( O_{x,y} \), and magnitude, \( M_{x,y} \), are computed at each pixel location using the horizontal and vertical gradients as shown in Equation 2.16. Finally, a histogram is constructed for rectangular patches of pixels of size MxN, referred to as cells, by using the orientation to index the histogram bins and the magnitude for accumulation. The final descriptor is formed by concatenating histograms from nearby cells, usually based on a grid layout, such as 3x3 or 5x5 with some predefined pixel
spacing between cells. Lastly, the feature vector may be L1 or L2 normalized to provide contrast invariance.

\[ g^H_{x,y} = I(x+1, y) - I(x-1, y) \quad g^V_{x,y} = I(x, y+1) - I(x, y-1) \]

\[ M_{x,y} = \sqrt{(g^H_{x,y})^2 + (g^V_{x,y})^2} \quad O_{x,y} = \text{atan2}(g^V_{x,y}, g^H_{x,y}) \]

HOG descriptors can be highly shape selective, as well as contrast invariant. This makes them an ideal candidate for FL-IR prescreening. However, a dense sampling of HOG descriptors over an image can be computationally intensive to compute if done so in a naïve way. The use of integral images can significantly speed up the computation.

2.2.5 Integral Image

The integral image concept was first introduced in the well-known work of Viola and Jones [56]. For each location in an image, the integral image representation contains the sum of all pixels values above and to the left. This representation is useful if the sum of values inside many rectangular patches is required. Specifically, the sum inside any rectangular patch can be computed using only four accesses into the integral image representation, as shown in Figure 2.8. Viola and Jones used the integral image to quickly compute the difference between the means of rectangular blocks. However, the concept is easily extended to histogram features by computing one integral image per histogram bin. To compute the histogram for a block, the sums from the individual integral images can simply be concatenated together. For a dense sampling of large blocks over the image, the integral image representation is many times faster than the naïve approach.
2.2.6 Look-up Table Approach for Fast Prediction with Additive Kernels

The well-known kernel trick allows linear classifiers which rely only on dot products between vectors to be easily extended to non-linear classification. The “trick” is simply to replace all dot product calculations by a kernel calculation defined by the chosen kernel function $k(x, y)$. This turns out to be equivalent to the original linear classifier operating in an implicit feature space, $\varphi$, where the dot product between two transformed vectors is equal to the kernel function of those two vectors in the original feature space, i.e. $k(x, y) = \text{dot}(x_\varphi, y_\varphi)$. The dimensionality of this implicit feature space, $\varphi$, can be much larger than that of the original space, and can even be infinite in the case of the radial basis function kernel. The chosen kernel function, $k$, must be symmetric and semi-positive definite, i.e. it must satisfy Mercer’s theorem [57].

The kernel trick is what allows the hugely popular soft-margin SVM to be easily extended to non-linear classification. Recall the soft-margin SVM primal formulation in Equation
2.17, where \( w \) is the weight vector, i.e. decision boundary, to be learned, \( x_i \) are the training vectors, \( C \) controls the margin/classification accuracy tradeoff, \( \xi_i \) are the slack variables, \( b \) is the bias, and \( y_i \) are the labels.

\[
\begin{align*}
\min_{w,\xi,b} & \left( 0.5 \|w\|^2 + C \sum_{i=1}^{N} \xi_i \right) \\
\text{s.t.} & \quad y_i (\langle w, x_i \rangle + b) \geq 1 - \xi_i \\
& \quad C > 0 \quad \xi_i > 0
\end{align*}
\]

2.18

\[
\max_{\alpha} \left( \sum_{i=1}^{N} \alpha_i - 0.5 \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \right) \quad \text{s.t.} \quad 0 \leq \alpha_i \leq C \quad \sum_i \alpha_i y_i = 0
\]

\[
w = \sum_i \alpha_i y_i x_i \\
h(x) = \langle w, x \rangle + b
\]

2.19

\[
h(x) = \sum_{i=1}^{M} \alpha_i y_i k(x, x_i) + b
\]

2.20

In the primal form, the \( x_i \) appear only in the dot product with \( w \). However, the dot product here cannot simply be replaced with a kernel function because the regularization term \( 0.5 \|w\|^2 \) requires \( w \) in explicit form, i.e. in \( \phi \). However, the dual formulation, given in Equation 2.18, where \( \alpha_i \) are the Lagrange multipliers, does not require \( w \) explicitly, and the dot product could be replaced with our desired kernel function. Equation 2.19 shows how to compute \( w \) explicitly from the non-zero Lagrange multipliers and their corresponding vectors, and the resulting decision function \( h(x) \). However, if a kernel function, \( k \), is used in Equation 2.18 then there can be no closed form \( w \) without mapping to the possibly infinite dimensional feature space \( \phi \). Instead, the decision function form in Equation 2.20 can be used, where only the non-zero Lagrange multipliers and the corresponding vectors, referred to as support vectors, are needed.
As can be seen from Equation 2.20, evaluation of the decision function requires $M$ kernel evaluations and has complexity $O(MF)$, where $M$ is the number of support vectors and $F$ is the dimensionality of the input space. This is compared to the $O(F)$ complexity of evaluating $h(x)$ in 2.19 if no kernel function is used. Thus, the downside to non-linear SVMs using the kernel trick, in regards to a system in the field, is the potentially higher cost of classification if the number of support vectors is large or the kernel function is computationally intensive to calculate.

There are many possible kernel functions one could use, but functions tailored to the problem generally perform best. For computer vision applications, histogram features such as HOG are extremely popular, and kernels based on common histogram similarity measures work well. Two such kernels are the histogram intersection kernel and the additive chi-square kernel given in Equations 2.21 and 2.22 respectively. Here, $x_1$ and $x_2$ are feature vectors, and $f$ denotes the feature dimension.

\[
k(x_1, x_2) = \sum_{f=1}^{F} \min(x_1^f, x_2^f)
\]  \hspace{1cm} 2.21

\[
k(x_1, x_2) = \sum_{f=1}^{F} \frac{2x_1^f x_2^f}{x_1^f + x_2^f}
\]  \hspace{1cm} 2.22

In [25], it was shown that $h(x)$ using the histogram intersection kernel (Equation 2.21) can be computed exactly in $O(\log(M)F)$, by observing that each feature dimension can be treated independently and the results from each dimension summed. All kernels with this property are referred to as additive kernels here. In each dimension, $f$, if the corresponding feature values from all support vectors, $x_i^f$, $i = 1, 2, \ldots, M$ are sorted and stored in a list.
\( \theta = \{x_1^f, x_2^f, x_3^f, \ldots \} \), then the result for the intersection kernel will consist of two sums. The first sum comes from all support vectors where \( x_i^f \) is less than or equal to \( x_i^f \), the feature value of vector \( x \), the vector to be classified, in dimension \( f \). For brevity, \( x_i^f \) is referred to as \( s \). This sum is given in Equation 2.23, where \( r \) is the index of the first element in \( \theta \) that is \( \geq s \).

The second sum is for all support vectors where \( x_i^f \) is greater than \( s \), given in Equation 2.24.

By pre-computing these two sums, excluding the multiplication by \( s \) in 2.24, for all values in \( \theta \), we need only perform a binary search to find the two sums, multiply the second sum by \( x_i^f \), and add the two values to compute the contribution of each feature dimensions to \( h(x) \).

\[
\sum_{i=1}^{r-1} \alpha_i y_i x_i^f \quad 2.23
\]
\[
s \sum_{i=r}^{M} \alpha_i y_i \quad 2.24
\]

While [23] is an improvement in prediction speed, it is still dependent on the number of support vectors and only applies to the intersection kernel. However, it points the way toward a more general method for approximate solutions for additive kernels that was described in [28]. Specifically, using a lookup table (LUT) for each dimension. Since the final decision function is the summation of the results from each dimension, we can simply build a lookup table for each dimension. If the step between entries in the table is fixed, lookups can be performed in constant time, making the overall complexity for calculating \( h(x) \), \( O(F) \). For a better approximation, linear or polynomial interpolation between LUT entries can be used. To build the LUTs, the minimum and maximum value in the training data for each feature dimension is found, and a predefined number of steps, the LUT size, are computed between these extremes.
For each of these values the decision function value for the corresponding dimension is calculated using the exact formulation of \( h(x) \) and stored in the LUT.

### 2.2.7 Explicit Feature Map for Fast Training with Additive Kernels

The preceding section showed how prediction in linear time could be achieved using additive kernels. However, training is still problematic for large datasets. Large scale SVM solvers generally require linear kernels [26], [27], as linear kernel SVMs can be trained much more efficiently. This has led to several attempts at creating approximate feature maps. Remember, using a kernel function is equivalent to running the linear algorithm in a feature space, \( \varphi \), in which the dot product between two vectors is equivalent to the evaluating the kernel function in the original space. Mapping to \( \varphi \) is not generally possible since it typically has very high dimensionality. However, we could attempt to find a mapping to a space, \( \varphi' \), which has higher dimensionality than the original feature space, but much lower dimensionality than \( \varphi \), with the goal of approximating \( k(x, y) = \text{dot}(x, y) \), i.e. \( k(x, y) \approx \text{dot}(x, y) \). Perhaps the most well-known method for this is using Nystrom’s approximation [58], [59], which exploits the fact that the kernel is linear in \( \varphi \), by selecting an n-dimensional subspace and projecting the data onto the subspace. The subspace is determined from the training data, and can be selected greedily to span as much of the data variability as possible or randomly. This is similar to Kernel PCA [60].

In [29], Vedaldi and Zisserman derive closed form, data independent feature maps with user variable dimensionality for common additive kernels such as histogram intersection, additive chi-square, and Jensen-Shannon. Their approximations show good performance with dimensionalities only 3 to 5 times larger than the original feature space. By explicitly mapping
training vectors to the new feature space, efficient linear SVM solvers can be utilized greatly speeding up training for large problems. During prediction, vectors are mapped to the new space and the dot product with the explicit form of $w$ is calculated. The one downside to this method is that the mapping itself adds computational complexity during prediction, and the overhead can be too much for use in a system that needs to classify tens of thousands of vectors per second.

As a test, the MATLAB implementation of Vedaldi in the VLFeat [1] library for performing the explicit mapping is used to map various numbers of 2000 dimensional vectors. For 500, 2000, and 8000 vectors of dimensionality 2000, Table 2.1 shows the time in seconds to perform the mapping for the histogram intersection kernel with new feature space dimensionalities of 3, 5, 7, and 9 times that of the original. The time to compute the dot product of those vectors with a random $w$ is also reported. As can be seen, the mapping itself can take more than 50 times as long as computing the dot products. In Chapter 4, we show how to combine the LUT prediction approach with the explicit feature mapping to achieve prediction in the same $O(F)$ time seen in Section 2.2.6.
TABLE 2.1.
Timing for explicit feature map method of Vedaldi and Zisserman using VLFeat [1] library implementation. Times are reported in seconds. *Mapping* indicates time to perform explicit feature mapping. *Dot Product* indicates time to calculate decision value by computing dot product of mapped vectors with the decision weight vector $\mathbf{w}$. $N$ indicates the number of examples. Input dimensionality was 2000.

<table>
<thead>
<tr>
<th>Dimensionality Increase</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N=500</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mapping</td>
<td>0.179</td>
<td>0.219</td>
<td>0.256</td>
<td>0.302</td>
</tr>
<tr>
<td>Dot Product</td>
<td>0.003</td>
<td>0.005</td>
<td>0.007</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td>N=2000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mapping</td>
<td>0.720</td>
<td>0.860</td>
<td>1.011</td>
<td>1.188</td>
</tr>
<tr>
<td>Dot Product</td>
<td>0.012</td>
<td>0.020</td>
<td>0.026</td>
<td>0.032</td>
</tr>
<tr>
<td></td>
<td>N=8000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mapping</td>
<td>2.830</td>
<td>3.642</td>
<td>4.397</td>
<td>5.176</td>
</tr>
<tr>
<td>Dot Product</td>
<td>0.041</td>
<td>0.093</td>
<td>0.112</td>
<td>0.121</td>
</tr>
</tbody>
</table>
Chapter 3: Flat Earth 2.0

In this chapter, the foundations of the Flat Earth 2.0 (FE2) model, the proposed geo-referencing algorithm that has fewer restrictions on camera motion and world topography than the Flat Earth 1.0 (FE1) model and less computational complexity than the full SfM approach, are developed. This new model is then compared to FE1 and SfM in terms of speed and accuracy. The proposed fix for essential matrix refinement in the SfM approach when handling planar scenes is also evaluated.

The key insight behind the FE2 model is that if the camera captures images of the same plane from multiple locations, then the pixel coordinate correspondences between any two of those images are defined by a homography. In such cases, rectification and disparity search prior to triangulation become unnecessary because the pixel correspondences can be determined from the homography. More than that, the homographies relating the separate views are defined by the normal vector to the plane and distance to the plane, as well as the rotation and translation between camera locations and the camera intrinsic parameters. With knowledge of the plane normal and depth to the plane, the depth for any pixel, in any of the images, can be calculated directly without the need for triangulation. However, this requires the decomposed homographies. That is, the rotation, translation, plane normal, depth to plane, and camera intrinsic parameters must be known. If the homographies are estimated directly, i.e. using the popular direct linear transformation (DLT) technique, then homography decomposition methods such as [61], [62] could be applied. However, these methods have limited accuracy, and can only recover translation up to an unknown scale factor.
Instead of using the plane information from the homographies to directly calculate depth, another possible approach is to use the homographies to determine pixel correspondences. Traditional triangulation, coupled with rotation and translation calculated from the INS, could then be used to determine world location. In fact, if a triangulation method which does not require the essential matrix was used, the essential matrix would not be required by such a method. That would eliminate the need for the expensive essential matrix refinement step of the SfM approach. Furthermore, if a triangulation method in which the number of unknowns remains constant regardless of the number of views was used, each pair of views would no longer need to be independently processed. Fortunately, iterative linear-LS triangulation fulfills all of these criteria, and it can be efficiently implemented for large numbers of points.

3.1 Model Assumptions and Theory

The one and only assumption made by the FE2 model is that what the camera sees in frames $N \pm T$, where $N$ is the current frame and $T$ is the range of frames used for depth estimation of pixels in frame $N$, is a plane. Thus, the world is required to be piecewise, or locally, planar. This restriction is necessary because otherwise the pixel coordinate correspondences could not be modeled by homographies. No restrictions are placed on camera movement or the orientation or position of the viewed plane with respect to the camera or INS.

As established in previous chapters, no road is perfectly planar. Therefore, an important question is how will this model cope with deviations from the ideal? Recall Equation 2.10, which describes a homography, $H$, relating two views of the same plane from different locations.
\[ H = R - \frac{m^T}{d} \]
\[ I_1 = H^* I_2 \]
\[ I_1 = R^* I_2 - t^* \left( \frac{n^T I_2}{d} \right) \]

Equation 2.10

\[ H \] depends on the rotation, \( R \), and translation, \( t \), between camera locations, as well as the normal vector to the plane, \( n \), and its depth from the camera \( d \). This is depicted for the case of a forward looking camera in Figure 3.1. An intuitive way to understand this equation is that \( I_1 \) and \( I_2 \) are normalized pixel coordinates in homogenous form, i.e. they are rays of unknown depth emanating from the camera’s center of projection. If the true depth of \( I_2 \) was known, then \( I_2 \) could be multiplied by that depth and then rotated and translated to the new frame of reference at \( I_1 \). This would work for any point regardless of the scene structure, but in the general case the true depth of \( I_2 \) is unknown. This does not matter for rotation since it is scale invariant; however, translation is not scale invariant. In the planar case, the magnitude and sign of all rays projected onto the plane normal, \( n \), will be the same. Furthermore, at the true scale that result is equal to \(-d\). This knowledge can be used to appropriately scale the translation, which is exactly what the factor \( \left( \frac{n^T I_2}{d} \right) \) is doing. If \( I_2 \) is at the true scale, this factor will be equal to \( \left( \frac{-d}{d} \right) = -1 \), and Equation 2.10 reduces to \( I_1 = R^* I_2 + t \), as expected.

Now, think about what happens for world locations which do not lie in the plane for which the homography is defined. The first term, \( R^* I_2 \), will not cause problems. This is why if there is no translation of the camera the scene need not be planar. The second term, \( t^* \left( \frac{n^T I_2}{d} \right) \), will be larger or smaller than the correct translation depending on how far out of
the plane, in the direction of the normal vector, the point is. In other words, how much smaller or larger \( n^T I_2 \) is than its expected value \(-d\). This will cause the computed pixel coordinate of the world location to be incorrect. The vector connecting the true pixel coordinate of the world location in \( I_1 \) to the pixel coordinate predicted by the homography is called the parallax vector [63].

![Diagram](image.png)

**FIGURE 3.1.** Depiction of a forward looking camera observing a ground plane from multiple locations. The two resulting images of the ground plane are related by a homography (Equation 2.10). The homography is defined by \( d \), the distance to the observed plane from Camera A, \( n \), the normal vector to the plane, and \( R \) and \( t \), the rotation and translation from Camera A to Camera B.

Knowing that pixel coordinate correspondences determined from the homographies will be incorrect for pixels which observe world locations that do not lie in the plane for which the homographies are defined, the question that must be answered is how will depth estimation for such world locations be affected? To answer this question, a few basics must be covered. First, given a homography relating pixel coordinates in image \( Q_1 \) to pixels coordinates in image \( Q_2 \), as shown in Equation 3.1, where \( K \) is the intrinsic parameter matrix of the camera, the depth for a pixel in \( Q_1 \) can be calculated using Equation 3.2, assuming that the observed world location lies in the plane defined by \( n \) and \( d \). Furthermore, the 3D world location, in terms of camera relative coordinates when image \( Q_1 \) was captured, is given by Equation 3.3. To calculate the error in world location estimation, the estimated location from Equation 3.3 for the pixel
coordinate to which the world location projects must be subtracted from the true world location. This is shown in Equation 3.4, where $A$ represents the world location, and $\|\|$ is the L2 norm. This expression can be further simplified if $n^T A$ is rewritten in terms of its expected value $-d$ and a deviation $\varepsilon$, as shown in Equation 3.5. Using this notation, $\varepsilon$ represents the distance from the plane to location $A$ along the normal vector $n$. In other words, $\varepsilon$ is the out-of-plane violation. The resulting expression shows that the world location estimation error is equal to $\alpha \| A \|$, where $\alpha = -\left( \frac{\varepsilon}{d + \varepsilon} \right)$. From inspection, it is clear that if $\varepsilon = 0$, implying that $A$ lies in the plane, then $\alpha = 0$, and there will be no error in the estimation. If $\varepsilon$ is non-zero, the error depends on the ratio $\frac{\varepsilon}{d + \varepsilon}$, as well as the L2 norm of $A$. Since $A$ is defined in terms of camera relative coordinates, the L2 norm of $A$ is simply the distance of $A$ from the camera.

This expression, $\alpha \| A \|$, for the estimation error makes intuitive sense. The error is 0 if the point lies in the plane, and if the point does not lie in the plane, the error is proportional to the distance from the camera, as well as to the ratio of out-of-plane violation distance to distance of the point from the camera along the plane normal.

In our case, the typical distance to the plane, which is simply the height of the camera above the ground, is roughly three meters. Figure 3.2 shows the error in world location estimation for various combinations of $\| A \|$ and $\varepsilon$ assuming the distance to the plane is three meters. At 5 meters from the camera, a 10 inch out-of-plane distance leads to a 0.5 meter error. At 10 meters from the camera, a 6 inch out-of-plane distance leads to a 0.5 meter error. At 15
meters from the camera, a 4 inch out-of-plane distance leads to a 0.5 meter error. At 30 meters from the camera, a 2 inch out-of-plane distance leads to a 0.5 meter error.

\[
\begin{bmatrix}
  u \\
  v \\
  1
\end{bmatrix}_{Q^2} = H
\begin{bmatrix}
  u \\
  v \\
  1
\end{bmatrix}_{Q^1} = K \left( R - \frac{m^T}{d} \right) K^{-1}
\begin{bmatrix}
  u \\
  v \\
  1
\end{bmatrix}_{Q^1}
\]

\[\text{depth} = \frac{-d}{n^T K^{-1} P}\]

\[(K^{-1} P)^{* \text{depth}} = (K^{-1} P) \left(\frac{-d}{n^T K^{-1} P}\right)\]

\[A = \begin{bmatrix}
  X \\
  Y \\
  Z
\end{bmatrix}_w \quad \left\| K^{-1} KA \left(\frac{-d}{n^T K^{-1} KA}\right) - A \right\| = \left\| A \left(\frac{-d}{n^T A}\right) - A \right\| = \left\| A \left(\frac{-d}{n^T A}\right) - 1 \right\|
\]

\[\left\| A \left(\frac{-d}{n^T \alpha - \varepsilon}\right) - 1 \right\| = \alpha \| A \| \quad \alpha = \left(\frac{d}{d + \varepsilon}\right) - 1 = \left(\frac{\varepsilon}{d + \varepsilon}\right)
\]

FIGURE 3.2. Flat Earth 2.0 theoretical registration error for various out-of-plane distances and distances from the camera assuming a depth to plane of three meters.
3.2 Implementation Details

As mentioned at the beginning of Chapter 3, there are two possible ways to implement FE2. The first is to compute the frame to frame homographies directly using the DLT method, i.e. not worrying about the decomposed form, and then use them to determine pixel coordinate correspondences. Iterative linear-LS triangulation can then be used to calculate the world location of each pixel. The second way is to compute the homographies in decomposed form, i.e. estimating the normal vector and plane depth while using the rotation and translation calculated from the INS as shown in Section 4 of the Appendix, and then compute the depth for each pixel using the parameters of the plane as shown in Equation 3.2.

For the first case, it is best to calculate homographies between successive pairs of frames in the video stream, and then use the fact that the transformations are linear to calculate homographies between non-successive frames. That is, it is better to calculate \( H_{N}^{N+2} \), the homography between frames N and N+2, using \( H_{N+1}^{N+2} \) than to calculate it from keypoint matches between frames N and N+2. The reason is that all popular keypoint descriptors, such as SIFT and SURF, have no invariance to perspective transformation. By using successive frame pairs, differences due to perspective transformation are minimized.

For the second case, accuracy of plane normal and depth to plane estimation depends on the size of the translation between camera viewpoints. Larger translation will yield better accuracy, but only if the accuracy of keypoint matching and localization remains fixed. However, differences due to perspective projection increase as translation increases, which degrades keypoint matching accuracy. Thus, there is a dilemma between needing larger translation for
good plane normal and depth to plane estimation, but needing smaller translation for accurate keypoint matching and localization.

While these two algorithms may seem similar, they are actually quite different. In the first case, the homographies are being used more as a convenient, and compact, mathematical parameterization of the pixel coordinate correspondences between images than as a true description of scene structure. Because the rotation, translation, and camera intrinsic parameter matrix are not fixed, there is no guaranteed relation between the homography computed by DLT and the true geometry of the scene. In fact, the term homography only loosely applies. This method is really just using a perspective projection to mathematically model the pixel correspondences. In the case that the scene truly is planar, this transformation is indeed a homography. Once the pixel correspondences are determined from the perspective transformation, a traditional triangulation algorithm, iterative linear-LS, is used to determine world location. As along as the pixel correspondences are accurate, the triangulation process will yield the correct result.

In the second case, the rotation, translation, and camera intrinsic parameter matrix are fixed to the best guess available. Fixing these parameters constrains the perspective projection transformation to the structure of a homography, and to the true 3D structure of the scene. This yields a plane normal vector and depth to plane which can be used for depth calculation at each pixel. The advantage of this case over the first is that no triangulation algorithm is necessary, which means less computational complexity. The disadvantage is that there is less tolerance for non-planar scenes or inaccuracy in the rotation and translation. In such cases, the first method has all eight degrees of freedom of the perspective transformation with which to model the
pixel correspondences. Whereas, the second method is constrained to three degrees of freedom (two for the normal vector and one for depth).

We now present algorithm outlines for both approaches for use in an operational system that captures images as a platform moves. Both approaches share the computationally intensive tasks of keypoint finding, keypoint descriptor extraction, and descriptor matching. Depending on the selected keypoint algorithm and descriptor the time required for this step can vary substantially. All tests performed here use the SIFT [47] implementation of David Lowe for keypoint finding and descriptor extraction as it has performed best in terms of resulting accuracy. However, it is also one of the slowest methods.

**Flat Earth 2.0: Outline of Homography Only Approach**

1. Compute keypoint locations and descriptors for each newly captured frame. Save.

2. Collect pose and position information from INS for each newly captured frame. Save.

3. Let $I_N$ indicate the current frame, and $I_Q$ indicate the set of frames surrounding $I_N$, $I_{N±T}$, that can be used for triangulation.

   a. Select frames from $I_Q$ based on the following requirements:

      i. At least H1 meters of movement to ensure a large enough baseline.

      ii. No more than H2 meters of movement to ensure enough image overlap.

      iii. At least H3 keypoint matches to $I_N$.

   b. For each frame in $I_Q$, compute homography in decomposed form to $I_N$ using saved keypoints, rotation and translation between views computed from saved INS pose and position information, and camera intrinsic parameter matrix.
4. Form final plane estimate by robustly combining plane normal and depth to plane estimates from all homographies computed in 3b.

5. Use Equation 3.3 to solve for 3D point in camera reference frame at selected pixel locations in $I_N$.

**Flat Earth 2.0: Outline of Homography + Triangulation Approach**

1. Compute keypoint locations and descriptors for each newly captured frame. Save.

2. Collect pose and position information from INS for each newly captured frame. Save.

3. For each newly captured frame, compute homography (perspective projection transformation) to previous frame using DLT method and saved keypoints. Save.

4. Let $I_N$ indicate the current frame, and $I_Q$ indicate the set of frames surrounding $I_N$, $I_{N\pm T}$, that can be used for triangulation.

   a. Select frames from $I_Q$ based on the following requirements:

      i. At least H1 meters of movement to ensure a large enough baseline.

      ii. No more than H2 meters of movement to ensure enough image overlap.

5. Transform image locations in $I_N$ to all selected frames in $I_Q$ using previously computed homographies.

   a. For dense reconstruction transform all pixel locations in $I_N$.

   b. If interested in only a few pixels, such as those marked as anomalies by a prescreener, transform only corresponding subset.

6. Use iterative linear-LS triangulation method to solve for 3D point in camera reference frame at selected pixel locations in $I_N$. 
The homography + triangulation approach has a second computationally expensive part, triangulation. Iterative linear-LS triangulation solves for the 3D world coordinate which minimizes the projection error, as shown in Equation 3.6. Here, \([X \ Y \ Z \ 1]\) is a world location in homogenous coordinates which we want to solve for, \([U \ V \ 1]\) is the corresponding projection onto the image plane for one view, and the 3x4 matrix is the projection matrix for that view. The projection matrix encapsulates the world to INS transformation, the INS to camera transformation, and the camera projection matrix. Therefore, it will be different for each view. Below, the two equations arising for the single view are derived, from which the generalization to multiple views is obvious.

\[
\begin{bmatrix}
U \\
V \\
1
\end{bmatrix} = \begin{bmatrix}
A & B & C & D \\
E & F & G & H \\
I & J & K & L
\end{bmatrix} \begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]

Thus, \(U = \frac{AX + BY + CZ + D}{IX + JY + KZ + L}\)

\(V = \frac{EX + FY + GZ + H}{IX + JY + KZ + L}\). Hence,

\(U(IX + JY + KZ + L) = AX + BY + CZ + D\)

\(V(IX + JY + KZ + L) = EX + FY + GZ + H\), and so

\(X(UI - A) + Y(UJ - B) + Z(UK - C) = D - UL\)

\(X(VI - E) + Y(VJ - F) + Z(VK - G) = H - VL\)
From Equation 3.6 it is seen that two equations arise from a single view of the world location. By using two or more views the three unknowns, \([X \ Y \ Z]\), can be solved for using least squares. The formulation is given in Equation 3.7. Here, \(N\) indicates the view from which the terms arise.

\[
Ax = b
\]

\[
A = \begin{bmatrix}
U_N I_N - A_N & U_N J_N - B_N & U_N K_N - C_N \\
V_N I_N - E_N & V_N J_N - F_N & V_N K_N - G_N \\
\vdots & \vdots & \vdots \\
D_N - U_N L_N & H_N - V_N L_N & \vdots
\end{bmatrix}
\]

As seen, each view of the world location adds two equations, but the number of unknowns is always three. Thus, this method requires solving a three parameter least squares problem with up to \(4T\) equations, given a frame radius of \(T\). This can be efficiently solved simultaneously for many pixels at once, even with varying numbers of equations per pixel, using the Moore-Penrose pseudo-inverse and matrix-matrix and matrix-vector operations. MATLAB code for this is provided in Section 1 of the Appendix.

In the iterative scheme, the least squares problem is continuously reweighted so that the minimization more closely minimizes the true projective error instead of the scaled version given by Equation 3.6 [10]. In Equation 3.6, the error we wish to minimize is in \(U\) and \(V\), but we are really minimizing \(U\) and \(V\) multiplied by the denominator of the left hand side of each equation. The weights are computed using Equation 3.8, which is just one divided by the current estimate of the denominator term, as given by the previous iterations’ solution. This term is the
same for both equations arising from the same view. The iterative process stops when the change in computed location from one iteration to the next is below some threshold. For most pixels, convergence occurs after only a few iterations. To avoid unnecessary computations, the iterative process is only applied to non-converged pixels at each further step. \( \| \) indicates absolute value. Under ideal circumstances, the absolute value operation should not be required since the denominator is expected to be positive; however, certain conditions can cause the denominator to become negative. Taking the absolute value keeps the calculation well-behaved in such cases, and does not alter the solution since the weight term is multiplied to both sides of the equation. Furthermore, the notion of negative weights is ill-defined with respect to a least squares problem.

\[
W = \begin{bmatrix}
\frac{1}{|I_N X_N + J_N Y_N + K_N Z_N + L_N|} \\
\frac{1}{|I_N X_N + J_N Y_N + K_N Z_N + L_N|} \\
\vdots
\end{bmatrix} \tag{3.8}
\]

In an operational system, a dense pixel to world map would probably not be needed. Only those locations corresponding to an anomaly detected by a prescreening algorithm would need to have their world location calculated. Thus, the cost of triangulation would be much lower than for dense output. In that case, the main hurdle to real-time implementation is the keypoint finding, extraction, and matching.
3.3 Proposed Fix for Essential Matrix Refinement Process of SfM Approach

Here, the proposed fix for the essential matrix refinement (EMR) process of the SfM approach when dealing with a planar scene is evaluated. As discussed in Section 2.1.4.1, the problem occurs because a planar scene is a degenerate case for essential matrix estimation. Namely, the cross product of any random vector with the homography matrix relating the two views of the plane will yield a valid essential matrix. Thus, when the optimization process used by the EMR step attempts to adjust the camera positions so that the keypoint correspondences agree with the resulting fundamental matrix there is no unique solution.

It is important to remember the purpose of the EMR step. When finding keypoint matches between views, a large number of correct matches will not agree with the essential matrix calculated using the location and pose reported by the INS. This is due to small, unavoidable errors inherent to GPS. Unfortunately, for many steps of the stereo processing chain a large set of keypoint matches that agree with the essential matrix is required. There are two ways to address the dilemma. The first is to alter the keypoint matches so that they agree with the essential matrix. This could be done using the optimal correction algorithm. The second option is to alter the essential matrix, which means adjusting the supposed location and pose of the cameras, so that it agrees with the keypoint matches. Because a majority of the keypoint matches are expected to be well-localized, and because we know for sure that there is error in the INS reported positions, the second option is selected. However, adjusting the camera locations assumes that there is a single correct relative positioning of the cameras which can be found using the information in the keypoint matches. In the case of a planar scene, this is not true. Thus, in the absence of information, the best bet is to rely on the INS since the mean error
in reported location and pose should be zero. Therefore, as the scene tends toward planar, the EMR optimization process should be restricted to a smaller area around the original values reported by the INS.

Formally, the proposed fix is to introduce a penalty term into the optimization process causing it to favor positions closer to the position reported by the INS when the mean residual error of a homography fit to the keypoint correspondences approaches zero. This approach is used because if the keypoints are perfectly coplanar, the residual error will be zero. The term

$$
\tau = \text{dis3D}(P_{\text{INS}}, P_{\text{OPT}}) \ast \min(\frac{\text{HE}_-\text{THRESH}}{\text{HE}}, 1.0),
$$

is added to the objective function value during minimization. Here, \text{dis3D} is a function which returns a single value quantifying the difference between two 3D locations, in terms of both position and pose. \( P_{\text{INS}} \) is the position and pose reported by the INS. \( P_{\text{OPT}} \) is a new position and pose being scored by the optimization process. \( \text{HE} \) is the mean residual error, in pixels, of the homography fit to the keypoint correspondences, and \( \text{HE}_-\text{THRESH} \) is the chosen threshold. Thus, when the error is small, the full penalty term is added, and, when the error is large, the penalty term goes to zero.

Another dilemma arises since there is no intuitive distance measure for considering both difference in location and pose simultaneously. This is addressed using the common strategy of performing a weighted average between rotation distance and translation distance in order to summarize the two values in a single number. For rotation distance, the formula

$$
D_r = 2 \arccos\left(\langle q_{\text{INS}}, q_{\text{OPT}} \rangle\right)
$$

is used, where \( q_{\text{INS}} \) is the quaternion representation of the INS reported pose, \( q_{\text{OPT}} \) is the quaternion representation of the new pose being evaluated.
indicates dot product, and \( | | \) indicates absolute value. \( D_R \) is effectively the angle of rotation between the two poses. For the translation distance, \( D_T \), Euclidean distance is used. The combined distance is computed as \( D_R \times 0.3 + D_T \times 0.1 \).

For evaluation, imagery from a MWIR camera, referred to as Camera A, is used. Example images from Camera A are shown in the top row of Figure 3.3. This camera has a large effective focal length and an extremely narrow field of view. It captures images of a twelve meter down-track by three meter cross-track section of the road. Distance from the camera ranges from eleven meters to twenty-three meters. Such a small section of the road is almost perfectly planar. Therefore, SfM without the proposed fix should be susceptible to the issue on this data.

Two runs of data, captured on different lanes on different days, are used for scoring. These lanes are comparable to improved roads, and are therefore quite smooth. Known ground truth locations are projected into the imagery using the intrinsic and extrinsic parameters of the camera. For each observation of a ground truth target, the triangulated world location for the pixel coordinate to which that target projects in the image is compared to the true world location. Figure 3.4 shows the triangulation error as a function of distance from the vehicle for the SfM approach with and without the proposed fix for planar scenes. The solid lines correspond to average error. The bottom and top error bars correspond to the 25\(^{th}\) and 75\(^{th}\) error percentiles. These plots were created by vertically averaging the error versus distance plots of each ground truth location in the lane.

As can be seen in Figure 3.4, the error is significantly reduced after introducing the penalty term into the optimization. In the 11-16 meter range, the error is reduced by up to 50%. At further distances the error is reduced by as much as 80%.
FIGURE 3.3. Example images from infrared cameras A, B, C, and D in order from top to bottom. Left column: images from Lane A. Right column: images from Lane B.
FIGURE 3.4. Registration error for SfM approach with and without the proposed fix for handling planar scenes during essential matrix refinement. Plotted lines are average error. Bottom and top error bars correspond to 25th and 75th error percentiles. Top plot: run from Lane A. Bottom plot: run from Lane B.
3.4 Accuracy Comparison to SfM and Flat Earth 1.0

The accuracy of the FE2 model is compared to that of SfM, with the fix for EMR presented in the previous section included, as well as to FE1. Multiple runs from multiple cameras and data captures are used for evaluation. Specifically, two runs, captured on different lanes and on different days, are used from each of the four cameras whose images are presented in Figure 3.3. For brevity, the four cameras: A, B, C, and D are referred to as cmA, cmB, cmC, and cmD. Of the four cameras, cmA best meets the assumptions of the FE2 model. It has a very narrow field of view and sees only a small section of the road. Testing on the other three cameras is performed to gain insight into how FE2 performs under less than ideal scene conditions. Specifically, how it compares to FE1 under such conditions. None of the four cameras meets the assumptions of the FE1 model. Therefore, SfM is expected to have smaller registration error than FE1 in all cases. The same scoring methodology used in Section 3.3 is used here.

Both the homography only and homography + triangulation approaches for 3D position estimation under the FE2 model are tested. The homography only approach is referred to as NVD2P, which stands for normal vector depth to plane, and the homography + triangulation approach is referred to as DLTPT, which stands for direct linear transformation plus triangulation. If 1.) the observed scene is planar, 2.) the rotation and translation, \( R \) and \( t \), between camera locations, as calculated from the INS, are perfectly accurate, 3.) the keypoint matches are perfectly localized, and 4.) \( K \), the intrinsic parameter matrix of the camera, is spot on, then both approaches should yield the same result. Obviously, such a perfect scenario is impossible, and there will be differences between the two approaches when they are applied to real-world data. The cause of these differences lies in how \( R \), \( t \), and \( K \) are used.
In the NVD2P approach, fixed values of $R$, $t$, and $K$ are used during homography computation. With $R$, $t$, and $K$ fixed, only the plane normal and depth to plane, $n$ and $d$, must be computed based on the keypoint matches. Once these two variables are determined, depth at each pixel can be computed using $K$, $n$, and $d$. Unfortunately, if there is noise in the INS reported positions, then $R$ and $t$ will also be noisy. In fact, since $R$ and $t$ represent the difference between two positions, their noise variance will be twice the noise variance in the INS position measurements. That is assuming the INS noise is zero-mean, normally distributed, and independent. Furthermore, since the NVD2P approach uses frame $I_N$, the current frame for which geo-referencing is being performed, as the source frame for each of the pairwise homographies, any error in the INS position for frame $I_N$ will be present across all normal vector and depth to plane estimates. This means that the estimates will be correlated. Therefore, the geo-referencing error will not approach zero as the number of estimates being averaged goes to infinity.

On the other hand, in the DLTPT approach the homographies are computed directly from the keypoint matches. This allows the perspective projection to take on whatever values best match the keypoint correspondences. The INS reported positions and the predetermined intrinsic parameter matrix $K$ are only used during triangulation. No differencing between INS positions is required. Furthermore, instead of solving for depth for each pixel in frame $I_N$, DLTPT solves for the 3D position which minimizes the projection error into all views. If errors in the INS position are uncorrelated, this approach will drive the geo-referencing error to zero as the number of views used goes to infinity. Therefore, DLTPT should have a clear advantage over NVD2P in the presence of noise in the INS position information.
DLTPT should also have an advantage when the observed scene is not planar. In that case, NVD2P has only three degrees of freedom, one for depth and two for the normal vector, with which to fit the keypoint matches. That is, it must select the plane which best fits the keypoint matches under the fixed $K$, $R$, and $t$ values. Depth for each pixel in $I_N$ is then computed assuming all points lie in this selected plane. DLTPT, on the other hand, can use the full eight degrees of freedom of the perspective transformation to best fit the keypoint correspondences. As long as the resulting pixel correspondences are accurate, the triangulation process will yield the correct 3D point for each pixel.

The results for cmA are shown in Figure 3.5. As expected, FE2 performs well. Both NVD2P and DLTPT result in significantly smaller registration error than FE1 as distance from the camera increases. Both approaches approximately equal the registration error of SfM. Since cmA’s imagery fulfills the piecewise planar requirement - not perfectly, but very close - FE2 and SfM would be expected to perform equally well. This is indeed observed, and acts as a basic check on the implementations.

The results for cmB are shown in Figure 3.6. cmB observes a much larger down-track range than cmA. The top of the image is set at the horizon, and the bottom of the image observes a piece of hardware mounted on the vehicle. That bottom section of the image is excluded from processing and evaluation. Scoring is performed in the range 8-25 meters from the camera. Over this range, FE2 outperforms FE1 on both runs. Performance on Lane A is slightly worse than on Lane B. This is most likely because Lane A has more trees and bushes close to the edge of the lane, which results in keypoint matches that are not coplanar. This can be seen in Figure 3.9, which shows SIFT keypoints superimposed on images from each camera.
**FIGURE 3.5.** Registration error as a function of distance from the camera for Flat Earth 1.0 (FE1), SfM, and Flat Earth 2.0 (FE2 DLTPT and FE2 NVD2P) on Camera A. Plotted lines are average error. Bottom and top error bars correspond to 25\(^{th}\) and 75\(^{th}\) error percentiles.

Top plot: run from Lane A. Bottom plot: run from Lane B.
FIGURE 3.6. Registration error as a function of distance from the camera for Flat Earth 1.0 (FE1), SfM, SfM with temporal post-process (SfM PP), and Flat Earth 2.0 (FE2 DLTPT and FE2 NVD2P) on Camera B. Plotted lines are average error. Bottom and top error bars correspond to 25th and 75th error percentiles. Top plot: run from Lane A. Bottom plot: run from Lane B.
FIGURE 3.7. Registration error as a function of distance from the camera for Flat Earth 1.0 (FE1), SfM, SfM with temporal post-process (SfM PP), and Flat Earth 2.0 (FE2 DLTPT and FE2 NVD2P) on Camera C. Plotted lines are average error. Bottom and top error bars correspond to 25th and 75th error percentiles. Top plot: run from Lane A. Bottom plot: run from Lane B.
FIGURE 3.8. Registration error as a function of distance from the camera for Flat Earth 1.0 (FE1), SFM, SFM with temporal post-process (SFM PP), and Flat Earth 2.0 (FE2 DLTPT and FE2 NVD2P) on Camera D. Plotted lines are average error. Bottom and top error bars correspond to 25th and 75th error percentiles. Top plot: run from Lane A. Bottom plot: run from Lane B.
FIGURE 3.9. SIFT keypoint locations superimposed on selected images from cameras A, B,C, and D, in order from top to bottom. Left: Lane A. Right: Lane B.
The results for cmC are presented in Figure 3.7. Like cmB, cmC sees to the horizon. It also observes roughly the same area cross-track. The main differences between cmB and cmC are that the nearest distance for cmC is much greater, 18 meters versus 8 meters, and that the INS accuracy is better for cmB. The INS accuracy difference is due to a small time delay in sampling from the INS at the instant a frame is grabbed that occurred during the 2012 data collection. Because the distance from the vehicle is 18 meters and greater, FE1 would not be expected to perform well on cmC. For both Lane A and Lane B the 25\textsuperscript{th} percentile error is greater than one meter at all distances for FE1. As in the cmB case, FE2 outperforms FE1 on both runs. For FE2, the DLTPT approach shows superior accuracy compared to NVD2P. Since this setup suffers from both position inaccuracy and a non-planar scene, DLTPT would be expected to perform better. SfM and SfM with temporal post-processing (SfM PP) score slightly better than FE2 DLTPT. The temporal post-processing step smoothes the SfM depth estimates in time by blending results from neighboring frames. It is described in [8]. As can be seen in Figure 3.9, there are a significant number of keypoints detected on the berm and surrounding vegetation which will throw off the homography estimation. This especially hurts NVD2P, which accounts for its inferior performance.

Finally, the results for cmD are presented in Figure 3.8. cmD has the largest field of view of the four cameras, and it sees the largest cross-track area. Roughly the top third of the image is sky. cmD’s imagery is also much less sharp than that of the other cameras which makes precise keypoint localization, as well as stereo correspondence search, difficult. FE1 performs reasonably well within the 10-15 meter range. However, its performance begins to suffer after 15 meters. On the Lane A run, FE2 NVD2P is bested by FE1 in the near range, 10 to 15 meters, but does slightly better beyond 15 meters. FE2 DLTPT shows significantly better performance.
than FE2 NVD2P, equaling SfM and SfM PP on the Lane A run. Again, as can be seen in Figure 3.9, there are hardly any keypoints directly on the road surface. Most of the keypoints are located on the berm and surrounding vegetation. This is a nearly impossible scenario for NVD2P. On the Lane B run, FE2 NVD2P does very badly, while FE2 DLTPT is equal to FE1 in the near range and better than FE1 beyond 15 meters. SfM and SfM PP show the best performance at all distances. Again, the superiority of FE2 DLTPT over FE2 NVD2P is not surprising given the position inaccuracy, non-planar scene, and distribution of the keypoint matches in the image.

An obvious thought is that the accuracy of FE2 DLTPT for cmB, cmC, and cmD could be improved by restricting the keypoint correspondences that are used to estimate the frame to frame homographies to lie in specific parts of the image, i.e. within the detection region of interest where detection is performed. This would eliminate keypoint matches between objects in the distance, as well as matches on the road surface that are far away. That is, matches that are past the point where the road can accurately described as a plane. Ideally, one would identify the road surface within the detection region of interest and use only keypoint matches from that area. However, that would require the world location of the keypoints, which isn’t known until after the calculations are performed. An iterative process that alternates between triangulation and refinement of the set of keypoint matches used could address this issue.

Accuracy of the geo-referencing methods was also compared by running a previously trained detector on each of the runs and looking at the difference in detection performance while varying the geo-referencing method. The detector used was the MS-HOG prescreener described in Chapter 4. That algorithm detects targets using a sliding window approach, and is trained in a supervised fashion using image windows of ground truth and non-ground truth
locations. Detections in the individual images are clustered using mean-shift in world coordinates with a uniform kernel of radius 50cm. The mean-shift peaks from each of the individual images are then combined using a second round of mean-shift clustering, again with a uniform kernel of radius 50cm. Since the radius for mean-shift clustering is fixed, errors in geo-referencing have a pronounced effect. The NAUC @ 0.1 right hand side for each of the camera, run, and geo-referencing algorithm combinations is shown in Table 3.1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Camera A</th>
<th>Camera B</th>
<th>Camera C</th>
<th>Camera D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lane A</td>
<td>Lane B</td>
<td>Lane A</td>
<td>Lane B</td>
</tr>
<tr>
<td>SfM PP</td>
<td>0.986</td>
<td>0.825</td>
<td>0.963</td>
<td>0.852</td>
</tr>
<tr>
<td>SfM</td>
<td>N/A</td>
<td>N/A</td>
<td>0.957</td>
<td>0.851</td>
</tr>
<tr>
<td>FE2 DLTPT</td>
<td>0.986</td>
<td>0.823</td>
<td>0.962</td>
<td>0.851</td>
</tr>
<tr>
<td>FE2 NVD2P</td>
<td>0.986</td>
<td>0.822</td>
<td>0.964</td>
<td>0.851</td>
</tr>
<tr>
<td>FE1</td>
<td>0.914</td>
<td>0.597</td>
<td>0.914</td>
<td>0.774</td>
</tr>
</tbody>
</table>

Negative correlation between detection performance and geo-referencing error, as shown in Figures 3.5 through 3.8, is quite clear from inspection of Table 3.1. In all cases, detection performance using FE2 DLTPT is better than using FE1. FE2 DLTPT and FE2 NVD2P are approximately equal on cmA and cmB, but FE2 DLTPT performs much better on cmC and cmD. SfM does significantly better than FE2 DLTPT on only one of the runs from cmC and one of the runs from cmD. Since this test only measures detection performance with a 50cm halo for scoring, it is forgiving for small amounts of geo-referencing error. This allows the FE2 approaches to perform similarly to SfM on cameras besides cmA.
3.5 Speed Comparison to SfM and Flat Earth 1.0

Having evaluated the accuracy of FE2, it is clear that in certain cases it can match the more complicated SfM approach. Furthermore, FE2 outperforms FE1 in almost every case. However, FE2 will never yield significantly better accuracy than SfM. Thus, why use it? The answer is computational complexity.

While SfM with temporal post-processing is consistently the most accurate approach, it is very computationally intensive. Another drawback is that there is no way to apply it to only a few pixels in the image. The EMR step depends on processing the entire image, and the semi-global block matching algorithm for stereo correspondence requires processing the entire rectified image. Luckily, in the FE2 DLTPT algorithm a majority of the processing scales linearly with the number of pixels processed, and that processing can easily be restricted to any random subset of pixels in the image. The ability to restrict processing to a small number of pixels, and thereby reduce the computational expense, is a huge benefit because in actual operation, world location is only needed for a small percentage of pixels. Specifically, those pixels that trigger the detector.

Table 3.2 shows the average number of seconds per frame required by each algorithm - SfM, FE2 DLTPT, and FE1 - to process imagery from each of the four cameras. These numbers do not include the time for keypoint detection and feature extraction. The time for keypoint matching is included in SfM, but not in FE2 DLTPT. This difference is insignificant when comparing the two for dense depth estimation, but is addressed later in the case of sparse depth estimation. For FE2 DLTPT and FE1, seconds per frame is also reported for the cases of processing only 20%, 5%, and 1% of pixel locations in the image. For cmA, cmB, and cmD tests
were run on an Intel Core i7-3930K CPU, and for cmC tests were run on an Intel Xeon X5650 CPU. All algorithms were implemented in MATLAB. Several parts of the SfM code were written in C and assembly. This code was accessed via a MATLAB Executable (MEX-file). No explicit multithreading was used during these tests. However, MATLAB’s internal multithreading was not disabled.

For the 100% case, FE2 DLTPT ranges from 3.5 times faster than SfM to 15.8 times faster. Surprisingly, the largest boost is seen with cmA. Based purely on the number of pixels to process, cmA would be expected to show the smallest speedup. This is because all of cmA’s image must be processed, i.e. all pixel locations are expected to be a reasonable distance from the camera. Whereas, portions of the image for cmB, cmC, and cmD - for example the sky in cmD - can safely be ignored since the world locations for those sections will always be very far from the camera. However, even though more pixels are processed for cmA, the iterative linear-LS triangulation algorithm, which is the most computationally expensive part of FE2 DLTPT, converges more quickly. Specifically, after the first refinement step there are roughly 70% fewer pixels that have failed to converge compared to the other cameras.

For an operational system to travel 10kph and process a new frame every 25cm of forward advance, a minimum frame rate of 11 fps is required. That translates to 0.09 seconds per frame. If 25cm is relaxed to 50cm or 100cm, the time requirements become 0.18 and 0.36 seconds per frame respectively. Even with multithreading, neither SfM nor FE2 DLTPT are capable of meeting even the laxest time requirement if 100% of the pixels must be processed. However, if only 1% of the pixels must be processed, FE2 DLTPT is well within the time range needed.
TABLE 3.2

Seconds per frame required to process images from Cameras A, B, C, and D using Structure from Motion (SfM), Flat Earth 2.0 DLTPT (FE2), and Flat Earth 1.0 (FE1) algorithms. Percentage indicates percentage of pixels in the image for which world location was calculated.

<table>
<thead>
<tr>
<th>Camera</th>
<th>Percentage of Pixels Processed Per Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>SfM</td>
</tr>
<tr>
<td>A</td>
<td>42.9</td>
</tr>
<tr>
<td>B</td>
<td>16.1</td>
</tr>
<tr>
<td>C</td>
<td>34.3</td>
</tr>
<tr>
<td>D</td>
<td>16.3</td>
</tr>
</tbody>
</table>

Even though FE2 DLTPT can get below the strictest time requirement, 0.09 seconds per frame, that neglects the time taken for keypoint detection, feature extraction, and matching. In actual operation, i.e. processing <1% of pixel locations, these other tasks will account for most of the computational burden. Table 3.3 shows the seconds per frame required for keypoint detection, feature extraction, and matching using SIFT and SURF keypoints. Two SIFT implementations were tested. The first is that of David Lowe [64]. The second is the SIFT implementation in the VLFeat library [1]. The SURF implementation used is that of OpenCV, accessed through MATLAB using the detectSURFFeatures and extractFeatures functions of the Computer Vision Toolbox. Clearly, SIFT is not a plausible option for real-time operation, at least not without introducing multithreading and/or image downsampling. SURF is much faster, and it could be viable for a real-time system when coupled with multithreading. Obviously, the time for keypoint matching varies significantly based on how many keypoints are detected in each image. For example, cmD’s imagery, which is blurry and sees a lot of the sky, results in fewer keypoints than the imagery of cmA. This leads to the order of magnitude difference in time required for keypoint matching. The keypoint matching step scales as O(NM), where N and M are the number of keypoints in the two images being matched. Both SIFT and SURF have
thresholds which can be used to control the number of keypoints found. By dynamically adjusting these thresholds during operation the computational complexity of keypoint matching could be controlled.

**TABLE 3.3**
Seconds per frame required for keypoint detection, feature extraction, and matching for Cameras A, B, C, and D.

<table>
<thead>
<tr>
<th>Camera</th>
<th>Keypoint Detection and Feature Extraction</th>
<th>Keypoint Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SIFT - Lowe</td>
<td>SIFT - VLFeat</td>
</tr>
<tr>
<td>A</td>
<td>2.89</td>
<td>1.00</td>
</tr>
<tr>
<td>B</td>
<td>1.80</td>
<td>0.95</td>
</tr>
<tr>
<td>C</td>
<td>3.60</td>
<td>1.60</td>
</tr>
<tr>
<td>D</td>
<td>0.93</td>
<td>1.01</td>
</tr>
</tbody>
</table>
Chapter 4: MS-HOG Prescreener

This chapter outlines the proposed prescreening algorithm to be implemented and tested as part of the dissertation. This includes the multi-scale HOG features, as well as the LUT approximation for use with the explicit feature map for achieving linear training and testing time requirements for SVMs with additive kernels. Detection results are presented for multiple cameras from multiple datasets. Results are compared to the ensemble of size-contrast filters prescreening algorithm mentioned in Section 2.2.1.

4.1 MS-HOG features

For features, a multi-scale, noise tolerant extension of the traditional HOG features described in Section 2.2.4 is used. In many object detection applications the detector is of fixed size and image patches of varying sizes are extracted from the image and resized to match the size of the detector in order to handle detection at multiple scales. In such scenarios, having the features capture multi-scale information is not critical because scale variations are handled by resizing the image patches. In other words, the detector should only detect objects of the correct class, at the correct size, in the patches it is presented. Our problem cannot be handled in this way for two reasons. The first is that we lack bounding boxes for target signatures in the training data. The second is that we need to detect target signatures of varying shapes and sizes, and this variation is caused by two factors. The variation in the size of the BEHs themselves, as well differences in the strength of their signatures in the IR imagery. Multi-scale HOG features are used to address this issue.
Extension of HOG features to multiple scales can be performed in two ways. The first method is to keep the feature calculation fixed and resize the image. The image is progressively downsampled, usually with a box filter, to form a so called *image pyramid*. HOG features are computed on each successive image of the image pyramid with the smallest image corresponding to the largest HOG scale. Features from the different scales are then concatenated together. This is the most popular approach for multi-scale HOG feature extraction, and has been used in popular methods such as the deformable parts model [54].

The second method is to fix the image and enlarge the gradient operators. Simply enlarging the central differences operators from Equation 2.15 will not yield satisfactory results because while the distance between taps is enlarged, the pixel area on the image is not. To compensate for this, the image is filtered using a Gaussian filter of increasing standard deviation at each increasing scale. This approach is similar to what would be obtained using the first method if the resizing was performed by smoothing with a Gaussian filter and then decimating. This approach is used in the second stage feature extraction of our previous work [9]. The authors in [65] propose a similar scheme they refer to as “Scale Space HOG”, but they do not enlarge the gradient operators at each scale, somewhat defeating the purpose. The authors of [66] use binomial filtering, a fast approximation of Gaussian filtering, as a means of controlling feature scale, but they do not mention whether the corresponding feature computations are enlarged to match. Figure 4.1 depicts both methods. Various other approaches to multi-scale HOG are proposed in [67]. For this work, the pre-filtering approach with varying gradient operator size is used because it allows to precisely control both aspects of the calculation: 1.) the size of the image regions and 2.) the distance between them. It also keeps the image size the same for all scales, which simplifies feature extraction from the multi-channel integral image.
Two major changes are made to the multi-scale HOG features of [9]. The first is aimed at noise tolerance. Infrared imagery, depending on the camera type and quality, can be quite noisy. This is particularly problematic if a sharp, high resolution image is desired. This is challenging for cooled infrared cameras, where the sensor needs to be kept at a carefully controlled temperature. Figure 4.2 shows images from two cooled, long-wave infrared cameras and one cooled, mid-wave infrared camera. As can be seen, noise is clearly an issue. Salt and pepper, or impulse, noise, where individual pixels have extreme values, is common. Problems with the sensor or temperature control can cause issues for cooled infrared cameras, such as the horizontal lines seen in the bottom row of Figure 4.2. In some cases, the high resolution of these cameras, in terms of pixels per square meter on the ground, simply brings out unwanted high frequency detail.
FIGURE 4.2. Noisy infrared image patches.
Top: patch from cooled long-wave infrared camera.
Bottom: patches from cooled long-wave (left) and mid-wave (right) infrared cameras.
Salt and pepper noise and dead pixels are common in cooled infrared cameras if
the sensor temperature is not carefully controlled.

Traditional HOG features computed using the central difference operators are not in any
way noise tolerant. While Gaussian filtering prior to gradient computation can help with
Gaussian noise, it does not help with the impulse like noise seen in the images in Figure 4.2. This
type of noise is common in infrared imagery. Therefore, we propose replacing the Gaussian
filtering step of the multi-scale HOG approach with median filtering since it is effective at
removing impulse noise. Figure 4.4 shows the gradient magnitude for the image patch in Figure 4.3 computed at two orientations and three scales using Gaussian and median pre-filtering. For the Gaussian case, the horizontal and vertical central difference operators are enlarged to a length of $length = \text{round}(3\sigma)^2 + 1$, where $\sigma$ is the standard deviation of the Gaussian blur in the corresponding direction. The standard deviation is allowed to differ in the horizontal and vertical directions. For the median case, the horizontal and vertical operators are enlarged to a length of $length\_h = N + 2$ and $length\_v = M + 2$, for median filtering using a filter of size $M \times N$, respectively. From Figure 4.4, it is clear that median pre-filtering is not as affected by impulse noise, especially at the smaller scales. The target signature is more emphasized in the median case as well.

![FIGURE 4.3. Image patch from mid-wave infrared camera. Impulse noise present.](image)
FIGURE 4.4. Gradient magnitude at two orientations and three scales for image patch in Figure 4.3. Left: Gaussian pre-filtering. Right: Median pre-filtering. Orientations: 0 and 90 degrees.
The second major change to HOG computation is allowing the strength of pre-filtering to differ in the horizontal and vertical directions. The idea behind this is to better focus on the scales and orientations of the target signatures, and to try to remove undesirable effects from the imagery. One such effect is that of tread patterns in tire-tracks. As can be seen in the top row, second image from the left of Figure 1.2, the tire tread can create small horizontal stripes, similar to a BEH signature. Due to the individual pixel based aggregation of HOG features inside cells, it is difficult to distinguish these from the more horizontally elongated target signatures. By horizontally elongating the pre-filter kernels, these artifacts can be reduced. Elongating the pre-filter kernels in the horizontal dimension also helps to adjust for the much higher resolution, in terms of pixels per square meter on the ground, in the cross-track direction, as compared to the down-track, and adds further tolerance to impulse noise.

4.2 Boxcar Filtering Versus Integral Image for Fixed-Size Cells

In Section 2.2.5 it was shown how use of the integral image representation can significantly speed up extraction of sums from rectangular blocks in the image. The sum inside any rectangular region, allowing for varying width and height, can be computed using just four accesses into the integral image. This can be improved further if the rectangular regions are all the same size, as is the case in many applications. Suppose we are only interested in rectangular blocks of size $M \times N$. If the image was filtered with a 2D boxcar filter of size $M \times N$, i.e. a convolution filter of size $M \times N$ with all taps equal to 1, we could find the sum inside any such rectangular block on the image in a single access, as shown in Figure 4.5. This is a significant improvement over the integral image representation. Furthermore, boxcar filtering can be efficiently implemented with computational complexity of
\(O(W(M + 2H) + H(N + 2W)) = O(4WH + WM + HN)\) for an image of size \(H \times W\). For common sizes of \(M \times N\), this isn’t much worse than the \(O(2WH)\) computational complexity of the integral image. In most applications involving dense feature extraction the calculation of either will be negligible compared to that of the extraction process itself.

As an aside, a boxcar filter, or box car function, is zero everywhere except for a single interval where it is equal to a constant. The name boxcar originated because such a 1D function resembles a boxcar, a type of railroad car, when plotted.

**FIGURE 4.5.** Left: the sum of values in the red region can be computed by accessing only the green block of the boxcar filtered image. Right: extension to multiple channels, such as histogram features.

### 4.3 Look-up Table Approximate Prediction for Explicit Feature Map

In Section 2.2.6 it was shown how a separate LUT can be used for each feature dimension when using an additive kernel; thereby allowing predictions in time independent of the number of support vectors. However, this technique did nothing to improve training time. In Section 2.2.7, it was shown how explicit feature maps, with dimensionality only three to five times larger than that of the original input space, could be used to approximate additive kernels.
This allowed the use of efficient, linear SVM solvers. The drawback to that technique was that the computational complexity of the mapping becomes a bottleneck during prediction. Here, it is shown how the same LUT technique of Section 2.2.6 can be applied to achieve prediction time independent of both the feature map complexity and resulting dimensionality.

The key facts are that the explicit feature maps of [29] are data independent, i.e. the same input always produces the same output, and that they operate independently on each dimension of the input, i.e. if two dimensions of the input share the same value then their higher dimensional representation after mapping will be the same. These two facts allow us to collapse both the mapping process and the multiplication by the weight vector in the enlarged feature space down to a single LUT per input dimension. Let use denote the mapping \( \psi(x) \to x' \), which maps \( x \in \mathbb{R}^D \) to \( x' \in \mathbb{R}^{D(2^{\beta+1})} \), where \( \beta \) is a positive integer greater than zero. Working with the case of \( D = 1 \), the mapping will result in a vector of dimensionality of \( 2\beta + 1 \), which will, of course, be the dimensionality of the weight vector learned by the SVM.

Equation 4.1 shows how the SVM decision value will be computed for the case of \( \beta = 1 \).

\[
\psi([x]) \to \begin{bmatrix} x'_1 & x'_2 & x'_3 \\ w_1 & w_2 & w_3 \end{bmatrix} = h(x) \tag{4.1}
\]

Thus, all that is needed to build the LUT is to find the minimum and maximum values of \( x \) in the training data, compute the intermediate values based on the desired LUT size, map each of those values using \( \psi \) to the higher dimensional space \( x' \), and then evaluate Equation 4.1, storing the result in the LUT. The generalization to larger \( \beta \) is straightforward, and the generalization to \( D > 1 \) is simply to perform the above process separately for each input dimension. That is, for each input dimension, find the minimum and maximum values in the
training set, compute the intermediate values for the LUT entries, map each value using the function $\psi$, extract the appropriate weights for that input dimension from $w$, and then evaluate Equation 4.1 to populate the LUT.

Speed comparisons were performed between the proposed LUT method with linear interpolation refinement versus explicitly mapping each vector and evaluating the dot product with $w$. These results are presented in Table 4.1. The original feature dimensionality is 1000. Dimensionality enlargement factors of 3, 5, 7, and 9 times using 1000, 10000, and 30000 vectors were evaluated. As expected, the prediction time for the LUT approach is independent of the mapping dimensionality enlargement factor. Relative to the time of the Mapping approach, there is little difference between LUT sizes of 100 and 800. As the number of vectors to process increases, the LUT approach increases its speed advantage. This is due to reduced memory footprint and better CPU cache usage. The speed advantage for the LUT approach varies from roughly 6 times faster for the 1000 vector factor of three dimensionality increase case to almost 50 times faster for the 30000 vector factor of nine dimensionality increase case. In a real-time system this speed difference is very important. Even with this speed improvement, the LUT approach is 6 to 7 times slower than prediction using a linear kernel, i.e. a weight vector the same dimensionality as the input. In terms of accuracy, for the 100 length LUT, max relative error compared to the mapping approach was on the order of 0.1%, and for the 800 length LUT max relative error was on the order of 0.01%.
### TABLE 4.1
Timing for explicit feature map prediction using mapping versus LUT with linear interpolation approach. Times are reported in seconds. Mapping indicates time to calculate decision value by performing explicit feature mapping and computing dot product with weight vector. LUT indicates time to calculate decision value by summing results of LUT for each dimension.

N indicates the number of examples.

<table>
<thead>
<tr>
<th>Dim. Increase</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT Size</td>
<td>100</td>
<td>800</td>
<td>100</td>
<td>800</td>
</tr>
<tr>
<td>N=1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mapping</td>
<td>0.182</td>
<td>0.181</td>
<td>0.221</td>
<td>0.221</td>
</tr>
<tr>
<td>LUT</td>
<td>0.035</td>
<td>0.038</td>
<td>0.032</td>
<td>0.036</td>
</tr>
<tr>
<td>N=10000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mapping</td>
<td>1.784</td>
<td>1.181</td>
<td>2.194</td>
<td>2.191</td>
</tr>
<tr>
<td>LUT</td>
<td>0.084</td>
<td>0.108</td>
<td>0.082</td>
<td>0.107</td>
</tr>
<tr>
<td>N=30000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mapping</td>
<td>5.397</td>
<td>5.422</td>
<td>6.594</td>
<td>6.593</td>
</tr>
<tr>
<td>LUT</td>
<td>0.184</td>
<td>0.196</td>
<td>0.187</td>
<td>0.182</td>
</tr>
</tbody>
</table>

#### 4.4 MS-HOG Prescreener Algorithm Outline

This section outlines the steps, details, and parameters of the proposed prescreening algorithm. This includes the generic algorithm parameters, as well as training specific parameters. Details of the training process are also explained. Values used for individual parameters during experimentation that are not reported in this section are reported in the results sections – 4.5 through 4.10.

##### 4.4.1 Parameters

One of the more important parameters for the prescreener is the size, shape, and spacing of the cells in the cell-grid used for aggregation of the per-pixel MS-HOG features. On one hand, the cells should be small enough to capture details of the object shape. However, they must also be large enough to handle the expected variation in target signature size and
shape. The layout of the cell-grid is important as well. If the grid is too large, irrelevant background information will be included. At the same time, some local background information is needed to provide context and aid discrimination. Because of these contrasting needs, it is unlikely that a single cell size and cell-grid layout will be optimal. This observation has been confirmed in algorithms for pedestrian detection where utilizing cells of varying shapes and sizes, as compared to a grid of uniform cells, improves performance [66], [68]. However, this fact must be balanced with the need to keep the number of unique cell sizes to a minimum so that boxcar filtering is efficiently utilized. We therefore choose to use dual cell-grids. One grid with larger cells for capturing courser detail of the target and local background, and one grid with smaller cells for capturing finer details of the object shape. Figure 4.6 shows proposed cell sizes and grid layouts for three infrared cameras. The larger grid is 5x5 while the smaller grid is 4x8. Both are centered within the detection window. While the same cell-grid layout can be used across cameras, the cell sizes need to be adjusted for each camera based on image resolution. A major benefit of this is that a trained detector can be adapted from one camera’s imagery to another given only the intrinsic and extrinsic camera parameters.

![Image](image.png)

FIGURE 4.6. Proposed dual cell-grid layout for MS-HOG prescreening algorithm. Visualized on image chips from three different infrared cameras. Grid layout remains the same across cameras, but the size of the cells is adjusted based on image resolution.
Another key parameter is the pixel spacing between classifier evaluations on the image. By running the detector only every U pixels horizontally and V pixels vertically, the computational expense can be reduced by a factor of UV. However, this subsampling may sacrifice detection quality. Since the MS-HOG features are pooled inside cells, the size of the cells, which controls the sensitivity to translation, determines the maximum subsampling size. A distance of 1/2 of the smallest cell width and height is a good upper bound on the subsampling factors. To further reduce computational complexity, the temporal dimension can be subsampled as well. That is, only run the detector every 2\textsuperscript{nd} or 3\textsuperscript{rd} frame. The maximum temporal sub-sampling factor depends on the vehicle speed and frame rate of the camera.

Image border handling is also problematic. As can be seen in Figure 4.7, if a target is close to the edge of the image it may be impossible to center the detection window on it without spilling over the image boundary. In the literature, there have been various solutions to this problem. The simplest is to not consider positions in the image where any part of the detection window would fall outside of the image. This solution is problematic since targets that are never observed far enough inside the image will be missed even though they may be visible in the imagery. Another simple solution is to pad the image, using either pixel extension or mirroring, thereby allowing features to be computed for cells outside the image boundary. Of course, since the resulting features are only approximations, the classifier prediction quality will deteriorate as more of the detection window falls outside the original image. A more complicated border handling technique is to introduce an indicator variable for each cell in the detection window into the feature vector [69]. These indicator variables are also referred to as latent variables. The indicator variables are set to 1 for cells with any part falling outside the image boundary and to 0 otherwise. For cells with indicator variable set to 1, the other feature
values are set to 0. After testing these various techniques, we decided to use a compromise between the first method and second method. The image is padded using mirroring to allow for up to 20% of the detection window to fall outside of the image. Image positions where more than 20% of the detection window would fall outside the original image are not considered.

FIGURE 4.7. The target signature in the left of the image (bottom) is close to the image border causing part of the detection window (top) to fall outside of the image.

The MS-HOG feature parameters: number of orientation bins, number of scales, size of median pre-filter for each scale, and feature vector normalization method must also be set. 12 orientation bins and 3 scales are used. Since the image contrast can vary significantly from one run to another, the MS-HOG histograms must be normalized to achieve contrast invariance. For
this, L2 normalization is used. Specifically, the feature vector resulting from concatenation of the histograms from each of the individual cells in a cell-grid, \( x_{ij} \), where \( i \) indexes the cell and \( j \) indexes the MS-HOG feature, is normalized using two methods. The first method normalizes the entire feature vector to unit length using Equation 4.2. The second method normalizes each MS-HOG feature to unit length separately using Equation 4.3. The two resulting vectors are then concatenated to form the final feature vector for the cell-grid. The feature vectors from each cell-grid are then concatenated to form the final feature vector for the detection window.

\[
x'_{ij} = \frac{x_{ij}}{\sqrt{\sum_i \sum_j x_{ij}^2}} \quad 4.2
\]

\[
x'_{ij} = \frac{x_{ij}}{\sqrt{\sum_j x_{ij}^2}} \quad 4.3
\]

Lastly, parameters for training, confidence thresholding, and spatial and temporal hit clustering are required. Training parameters include the number of randomly selected negative examples for initial training, the number of hard negative mining iterations, and the SVM parameters, such as \( C \) and the kernel function. A detection threshold on the signed distance to the SVM decision boundary, which is interpreted as the hit confidence, is needed to limit the number of output hit locations. This threshold also provides an intuitive relationship between the confidence threshold and target signature visibility in the imagery. All pixel locations in the image generating a confidence value above the detection threshold are projected into world coordinates where weighted mean-shift, using the uniform kernel, is used for peak detection. The weights for weighted mean-shift are computed as \( \exp(d) \), where \( d \) is the hit confidence,
i.e. the signed distance to the SVM decision boundary. The peaks from individual frames are then merged into a single set, and another step of weighted mean-shift is used to identify the final hit locations for the run. For the second weighted mean-shift step, the weights are computed as the sum of the weights for all points within the mean-shift radius of the peak during the first weighted mean-shift step. The radius for mean-shift is set to 50cm. It is not learned from the data.

4.4.2 Training

As mentioned previously, positive examples for training are collected by projecting UTM ground truth coordinates for each lane into the imagery. Only projections of the ground truth objects when they are within the minimum and maximum distance limits from the camera are used. A limited distance range is utilized for two reasons. The first reason is that it limits the variation in object size, and focuses the detector on the section of the image with the highest resolution on targets. The second reason is that it speeds up processing by restricting the area on the image that must be processed. The initial set of negative examples is collected by projecting randomly sampled UTM locations from within the lane boundaries, that are at least 1.25 meters from all ground truth locations, into the images. Again, observing the distance range restrictions. The SVM classifier is trained using the combined set of positive and negative examples. Further rounds of hard negative mining, where the current incarnation of the classifier is evaluated on the training data to identify false positives and add them to the training set, are used for refinement.
4.4.3 Operational Use

Here, the processing steps for using a trained model in an operational system that captures and processes images as a vehicle moves are outlined. For a causal system, the temporal mean-shift step must be performed using only per-frame peaks from frames that have already been captured. To accommodate this requirement, the lane is split into down-track slices of equal size (Q meters of forward vehicle advance), and temporal clustering is performed for each slice once enough frames have been captured and processed to cover the area of the slice along with some overlap (Z meters) into the surrounding slices. The operational steps are given below.

**MS-HOG Prescreener: Operational System Outline**

1. For each newly captured image
   
   a. Compute MS-HOG feature channels

   b. Perform box-car filtering of each MS-HOG feature channel for each cell size

   c. Extract cell-structured features, as specified by the cell-grid layouts, every U pixels horizontally and V pixels vertically from boxcar filtered feature images
      
      i. Ignore pixels with distance from camera outside the distance limits

   d. Compute confidence values using SVM model and extracted feature vectors

   e. Project pixels with confidence values above detection threshold into world coordinates

   f. Perform weighted mean-shift clustering and store peak locations

2. For each world detection region (Q meters of forward advance)
   
   a. Perform weighted mean-shift clustering of per-frame peak locations using all peaks inside of, or within Z meters of, the detection region

   b. Return peaks as final detection locations
4.5 Integral Image Versus Boxcar Filtered Image

This section investigates whether using a boxcar filtered image is faster than using an integral image for extraction of the dense MS-HOG features. The time in seconds per frame for calculation of the integral and boxcar images (FILT) and extraction of the MS-HOG features from the them (EXTRACT) is reported in Table 4.2 for Cameras A and B. These cameras are the same as Cameras A and B mentioned in Chapter 3. For each camera, different amounts of horizontal (SSX) and vertical (SSY) subsampling are tested. A subsampling factor of 2 means 1 out of every 2 pixels is skipped. A subsampling factor of 4 means 3 out of every 4 pixels are skipped. Both cameras have a resolution of 640 by 480.

Surprisingly, there is little difference in the time required to construct the integral image versus the boxcar image. Even though the integral image has lower theoretical computational complexity, the overhead involved in accessing the image in memory and storing the result seems to dominate. Thus, the two operations take roughly the same amount of time. However, since there are two different cell sizes being used, two boxcar images must be created. Whereas, only one integral image must be created. Therefore, the FILT time for boxcar in Table 4.2 is roughly double the time for integral image.

For extraction of features once the images are created, the boxcar image has a definite advantage in terms of speed. For all subsampling factors for both cameras there is a consistent 1.4x speed increase. For small subsampling factors, this makes a noticeable difference in the total time (FILT + EXTRACT). For larger subsampling factors, the FILT time becomes a larger component in the total time, and the overall speed difference is not as large. For example, for Camera A with no subsampling, the total FILT+EXTRACT operation takes 3.73 seconds using the
integral image and 2.73 seconds using the boxcar image. That is a speedup of 1.37x. Whereas, for [4 2] subsampling, the total operation takes 0.57 seconds using the integral image and 0.47 seconds using the boxcar image. The speedup is only 1.21x. Thus, there is some benefit to using the boxcar image, but it is not great.

**TABLE 4.2**

Seconds per frame for boxcar and integral image (FILT) creation for Cameras A and B, as well as for MS-HOG feature extraction (EXTRACT) from the images. [SSX SSY] indicates horizontal and vertical pixel subsampling factors - a factor of 4 means 1 in every 4 pixels is processed.

<table>
<thead>
<tr>
<th>Camera [SSX SSY]</th>
<th>Boxcar FILT</th>
<th>Boxcar EXTRACT</th>
<th>Integral Image FILT</th>
<th>Integral Image EXTRACT</th>
<th>Speedup EXTRACT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A [1 1]</td>
<td>0.132</td>
<td>2.60</td>
<td>0.067</td>
<td>3.66</td>
<td>1.41x</td>
</tr>
<tr>
<td>A [2 1]</td>
<td>0.132</td>
<td>1.35</td>
<td>0.067</td>
<td>1.86</td>
<td>1.38x</td>
</tr>
<tr>
<td>A [4 2]</td>
<td>0.132</td>
<td>0.34</td>
<td>0.067</td>
<td>0.50</td>
<td>1.47x</td>
</tr>
<tr>
<td>B [1 1]</td>
<td>0.106</td>
<td>2.41</td>
<td>0.053</td>
<td>3.53</td>
<td>1.46x</td>
</tr>
<tr>
<td>B [2 1]</td>
<td>0.106</td>
<td>1.20</td>
<td>0.053</td>
<td>1.77</td>
<td>1.48x</td>
</tr>
<tr>
<td>B [4 2]</td>
<td>0.106</td>
<td>0.31</td>
<td>0.053</td>
<td>0.47</td>
<td>1.51x</td>
</tr>
</tbody>
</table>

**4.6 Datasets, Evaluation Methodology, and Algorithm Parameters**

This section discusses the datasets used for evaluation of the proposed MS-HOG prescreening algorithm. It also discusses the evaluation methodology, and further details some of the algorithm parameters.

Data from five cameras, collected as part of two separate data collections, is used for evaluation. Both data collections occurred at the same arid U.S. Army test site. The first data collection took place in May of 2012. The second collection took place in December of 2013. Two cameras were used during the first data collection. Both are used for evaluation. These two cameras are referred to as Camera C and Camera D. These are the same Camera C and Camera D mentioned in Chapter 3. Camera C and Camera D are both LWIR cameras.
Four cameras were used during the second data collection, and three of those are used for evaluation. These three cameras are referred to as Camera A, Camera B, and Camera E. Cameras A and B are the same ones mentioned in Chapter 3. Camera E is a LWIR camera with roughly the same intrinsic and extrinsic parameters as Camera A, which is MWIR. For brevity, the cameras will be abbreviated cmA, cmB, cmC, cmD, and cmE.

Runs were captured over the same three lanes during both collections. A run consists of the imagery collected as the vehicle drives from one end of a lane to the other. The three lanes are referred to as Lanes A, B, and C. Lanes A and B are the same lanes described in Chapter 3. Lane C is another flat lane, but with a different target distribution in terms of depth and metal content. Of the three lanes, A is the easiest and C is the most difficult in terms of target detection using infrared cameras. For brevity, the lanes will be abbreviated LA, LB, and LC. LA has 44 emplaced targets. LB has 50 emplaced targets. LC has 79 emplaced targets. All targets are buried in-road at depths between one and six inches. Each of the lanes covers approximately 4000 square meters. For collection one, 25 runs – 9 LA, 7 LB, 9 LC – are used for evaluation. The runs are evenly split in terms of direction of travel on each lane, and they span a six day period. Time of day ranges from just after sunrise to mid-afternoon. For collection two, 24 runs – 8 LA, 8 LB, 8 LC – are used for evaluation. Again, the runs are evenly split in terms of direction of travel on each lane. All times of day, including nighttime, are included. The runs span a seven day period.

Leave one lane out cross validation is used for scoring. That is, in the first fold runs from LA and LB are used for training and runs from LC are used for testing. In the second fold, LA and LC are used for training and LB is used for testing. In the third fold, LB and LC are used for testing.
training and LA is used for testing. ROC curves are created for each run by plotting detection rate versus false alarm rate. The false alarm rate (FAR) is expressed as number of false alarms per square meter. An alarm location produced by the prescreener is considered a false alarm if it is farther than the halo radius from all ground truth locations. A halo radius of 50cm is used for all experiments. An alarm location is considered a correct hit (true positive) if it is closer than the halo radius to at least one ground truth location. If multiple alarms are within the halo radius of the same target, that target is considered detected and neither alarm is treated as a false alarm.

The normalized area under the curve (NAUC) value is used to summarize performance in terms of the ROC curve. The NAUC is computed by calculating the area under a ROC curve out to a certain FAR, referred to as the right hand side value. The calculated area is then divided by the right hand side value resulting in a number ranging from 0 to 1. A right hand side value of 0.1 false alarms per square meter is used for all experiments. For route clearance operations, a FAR of 0.005 is more realistic, but since this is a prescreening algorithm a larger FAR, resulting in higher recall, is desirable. A secondary classification algorithm will eliminate the extra alarms after reevaluation of the alarm locations using a more computationally expensive, and hopefully more accurate, classifier. Therefore, recall is more important than precision for the prescreener.

Table 4.3 lists the values of several parameters used for each of the cameras. These include the large and small cell sizes, distance ranges, and median pre-filter size for each HOG scale. As described in Section 4.4.1, two cell grids are used. A 5x5 grid with large cells, and a 4x8 grid with smaller cells. The median pre-filter sizes refer to the size of the median filter used before calculation of the HOG features at each scale, as described in Section 4.1.
TABLE 4.3
Camera type, resolution, distance ranges, large and small cell sizes, and median pre-filter sizes for Cameras A, B, C, D, and E used for prescreening evaluation.

<table>
<thead>
<tr>
<th>Camera</th>
<th>Camera Type and Resolution</th>
<th>Distance Limits</th>
<th>Large Cell W x H</th>
<th>Small Cell W x H</th>
<th>Median Pre-filter Sizes W x H</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Cooled MWIR 640x480</td>
<td>12-20</td>
<td>45 x 23</td>
<td>17 x 13</td>
<td>9x5, 13x7, 17x9</td>
</tr>
<tr>
<td>B</td>
<td>Uncooled LWIR 640x480</td>
<td>10-15</td>
<td>31 x 15</td>
<td>13 x 9</td>
<td>5x3, 9x5, 13x7</td>
</tr>
<tr>
<td>C</td>
<td>Cooled LWIR 640x512</td>
<td>19-27</td>
<td>31 x 13</td>
<td>13 x 7</td>
<td>3x1, 5x3, 9x5</td>
</tr>
<tr>
<td>D</td>
<td>Uncooled LWIR 640x480</td>
<td>12-18</td>
<td>19 x 9</td>
<td>9 x 5</td>
<td>3x1, 5x3, 9x5</td>
</tr>
<tr>
<td>E</td>
<td>Cooled LWIR 640x480</td>
<td>12-20</td>
<td>45 x 23</td>
<td>17 x 13</td>
<td>9x5, 13x7, 17x9</td>
</tr>
</tbody>
</table>

Unless stated otherwise, subsampling factors of 4 pixels horizontally, 2 pixels vertically, and 2 frames temporally are used for all experiments. The stochastic gradient descent (SGD) solver for linear SVM in the VLFeat [1] library is used for SVM training. The C parameter for SVM training is set to 0.01. When collecting the initial set of negative training samples, 16000 random world locations inside the lane boundaries are generated for each training run. These world locations are then projected into the imagery creating a much larger number of [world location, image coordinate, frame number] tuples. How much larger depends on how many frames see each world location. Next, the tuples are grouped by frame number, and these groups are randomly shuffled so that their ordering is random. From the random ordering, tuples are drawn in order, by group, one group at a time until 16000 tuples are collected. The feature vectors for these selected tuples are then extracted from the imagery and used as the negative training examples. This process of selecting negative examples minimizes the number frames that must be accessed and processed to collect the feature vectors as compared to simply selecting a random distance for each world location.
4.7 Restricting Training to Targets with Visible Signature

This section examines whether excluding targets which lack a visible signature in the infrared imagery can improve generalization performance. As mentioned in Section 1.1.2, there are two major difficulties with our raw data in terms of training a traditional sliding window detector. The first is that there are no bounding box coordinates for the targets. The second is that it is unknown whether a target has a visible signature in the infrared imagery. An enormous amount of manual labor would be needed to collect bounding boxes for the more than one million images containing known targets in the two data sets. Thus, no attempt to address the issue of training with bounding boxes versus without is made. However, the task of determining whether each target is visible in a run requires much less effort. Therefore, for each run captured by cmA, cmB, and cmE I manually verified whether each of the targets in the lane was visible. Target visibility rates across the runs varied from 100% to as little as 2% for runs captured near the diurnal crossover time periods. Using this information, targets without visible signatures could be excluded from the training process.

The issue of target locations which lack visible signatures being included in the training data is very similar to the problem of training with label noise [70]. Label noise manifests as flipped labels, i.e. incorrectly labeled instances in the training data. The probability that a label is flipped can be uniform across classes, or can be class dependent. In the literature, several methods have been proposed to deal with label noise when training SVMs [71]–[73]. Most of these methods work by modifying the loss function. However, application of these techniques is problematic since the modified loss functions lead to non-convex optimization problems. Furthermore, most of the studies investigate the effect of label noise on classification accuracy,
whereas we are more interested in ranking accuracy. These techniques for addressing label noise when training SVMs are not studied in this dissertation, but do provide an interesting avenue for future work.

In terms of label noise, our process of excluding targets lacking visible signatures from the training data only addresses the issue of class 0 to class 1 flips. However, there will also be non-target objects in the lane that create signatures which mimic target signatures. These cases can be seen as class 1 to class 0 flips. Our exclusion process does nothing to address such cases, and these might be just as prevalent as the 0 to 1 flips. Tackling such cases in the same manner would involve either examining every negative example, which is not feasible, or an iterative approach, using a human expert, which alternates between training the SVM and having the human check the labels of the resulting most positive negative examples. The second possibility is yet another avenue for future study.

cmB and cmE were used for testing. No hard negative mining iterations were used in these experiments. Each test was repeated five times to account for the randomness in negative sample selection and SGD SVM training. The results for cmB are shown in Figure 4.8. The error bars correspond to a 95% confidence interval. The results for cmE are shown in Figure 4.9.

In the aggregate, i.e. the vertically averaged result for each lane, there is little difference between training with all targets versus training with only visible targets. For one or two runs the results are statistically different, but these differences average out for the most part. The results for Lane C show the largest difference between the two cases, and the all targets case performs better for both cameras. This suggests two possibilities. The first is that the issue of label noise is simply not significant with respect to this problem. The second is that a more
comprehensive approach to the problem, i.e. one that addresses both cases of flipped labels in a more principled way, is needed. There is also the issue that the task of determining whether a target is visible is highly subjective. For certain targets there is no definitive answer. It may be better to eliminate such hard decisions and leave the issue up to the classifier using the techniques for training with label noise mentioned previously.

An issue which has been neglected thus far is that, for our case, the labels for the test data are assigned using the same method used to assign labels to the training data. Therefore, if the training labels are expected to suffer from some amount of label noise, the test labels would also be expected to suffer from the same label noise. Thus, it is highly unlikely that efforts aimed at mitigating the effect of label noise during training would have a positive effect when performance is assessed on the label noise corrupted test data. This leads us to the conclusion that to gauge the benefit of methods aimed at improving generalization when training with label noise, we would need test data that is not corrupted by label noise. For our case, that means a human operator must evaluate the predictions. Furthermore, it means that the results in Figures 4.8 and 4.9 are probably not representative of the true effect of removing the non-visible targets from the set of positive training examples.

The conclusions drawn in the preceding paragraph lead to perhaps the biggest issue with label noise tolerant training algorithms, there is no good way to set the values of the parameters relating to the amount of label noise. Cross validation on the training data, the standard technique for setting such hyper-parameters, cannot be used precisely because all of the training data suffers from the label noise. This fact is often neglected, or simply ignored, by authors.
FIGURE 4.8. ROC curves for Camera B (cmB) when training with all targets (left) and training with only visible targets (right) on Lanes A, B, and C. Error bars represent 95% confidence interval. Legend entries are formatted: Run#: NAUC@0.1, and ± indicates uncertainty in NAUC at 95% confidence. All experiments were repeated five times to generate confidence intervals.
FIGURE 4.9. ROC curves for Camera E (cmE) when training with all targets (left) and training with only visible targets (right) on Lanes A, B, and C. Error bars represent 95% confidence interval. Legend entries are formatted: Run# : NAUC@0.1 ± indicates uncertainty in NAUC at 95% confidence. All experiments were repeated five times to generate confidence intervals.
4.8 Hard Negative Mining

This section explores whether hard negative mining improves the performance of the prescreener. Hard negative mining is a process employed by many object detection systems for training a SVM - though the process can be adapted to other classifiers - when the number of training examples is so large that it is impossible to process them all at one time [54]. Generally, the number of negative examples is orders of magnitude larger than the number of positive examples, as is the case for our problem. By using the knowledge that 1.) only the support vectors, i.e. the training samples with non-zero Lagrange multipliers, contribute to the SVM solution and 2.) only a small portion of the large set of negative examples would be support vectors if it were possible to solve the SVM objective function taking all examples into account, an iterative process can be formulated to approximate the solution. Here, $D$ denotes the set of all negative examples, $P$ denotes the set of all positive examples, and $\tau_i$ denotes the signed distance to the decision boundary - the output of Equation 2.20 - for example $i$ given a trained SVM model. The steps of the iterative process are:

**Hard Negative Mining Outline:**

1.) Select a random subset of negative examples, $C$, from $D$.

2.) Train a SVM using $P$ and $C$.

3.) Remove from $C$ all examples for which $\tau < -1$.

4.) Select new examples from $D$ for which $\tau > -1$ and add them to $C$.

   a. If no new examples can be found exit.

5.) Repeat steps 2-4 if maximum number of iterations has not been reached.
Hard negative mining works well for problems with clearly defined objects and accurately labeled training data. That is, where no ambiguity exists as to whether a positive class instance is present in the set of images used for mining negative examples. For instance, in pedestrian detection a person is either present or not present. Some part of a person might be occluded, but there is no notion of partial expression of a person. That is, people do not fade in or fade out of the images. These conditions apply to most popular object detection data sets such as PASCAL VOC and ImageNet. Unfortunately, our data does not satisfy these requirements. The targets have a continuous scale of signature expression, varying from strongly expressed to non-existent, and the visual definition of what a target looks like is vague. Furthermore, non-target objects, which create signatures closely resembling target signatures, exist in the images used for mining negative examples. Even if humans were employed to cleanse the training data, a large amount of subjective judgment would be involved in determining 1.) whether a target is visible and 2.) which non-target signatures should be excluded because they resemble actual targets. These issues do not arise in most object detection problems. It could be particularly problematic if the hard mining process focuses too much on these non-target objects that look like targets, as this could skew the training data - that is, while such objects are relatively rare, they may tend to dominate the training data - causing the SVM to generalize poorly.

To investigate these issues, iterations of hard negative mining are run after the initial SVM training which uses a randomly selected set of negative examples. Hard negative examples are mined from each of the training runs. All images in a run are considered; however, windows centered on world locations within 1.25 meters of a ground truth location are excluded. The
mining process uses the same five step procedure outlined previously. The results for cmB are shown in Figure 4.10. The results for cmD are shown in Figure 4.11.

For cmB, hard negative mining slightly increases performance on LB, but performs worse on LA and LC. The results for LB and LC are within the 95% confidence interval bounds so the differences may not be significant. However, the difference on LA is much larger. Every single run on LA scores worse when hard negative mining is incorporated.

For cmD, hard negative mining performs worse on all three lanes. As with cmB, the results for LB and LC are within the 95% confidence interval bounds, but the difference on LA is larger. While not every run on LA scores worse, two of the runs, 4 and 9, see large drops in NAUC.

These experiments suggest that hard negative mining is not beneficial for this problem. This may be because the process is biasing the training, or because the selection of negative examples by scanning each image independently does not take into account the temporal correlation. That is, a particularly difficult false alarm may be included in the training data many times by the hard mining process depending on the number of frames in which it is seen. Further investigation is needed to definitively answer the question of why hard negative mining is detrimental.
FIGURE 4.10. ROC curves for Camera B (cmB) with hard mining (left) and without hard mining (right) on Lanes A, B, and C. Error bars represent 95% confidence interval. Legend entries are formatted: Run#: NAUC@0.1, and ± indicates uncertainty in NAUC at 95% confidence. All experiments were repeated five times to generate confidence intervals.
FIGURE 4.11. ROC curves for Camera D (cmD) with hard mining (left) and without hard mining (right) on Lanes A, B, and C. Error bars represent 95% confidence interval. Legend entries are formatted: Run#: NAUC@0.1, and ± indicates uncertainty in NAUC at 95% confidence. All experiments were repeated five times to generate confidence intervals.
4.9 Additive Kernels

This section explores whether the use of additive non-linear kernels, specifically, the histogram intersection and additive chi-square kernels, can improve the performance of the prescreener. The explicit feature maps described in Section 2.2.7 are used along with the LUT based approximate prediction described in Section 4.3, allowing for much faster training and prediction than a traditional kernel SVM. These two kernels have no hyper-parameters, but the output dimensionality of the feature map must be set. Thus, the only differences to the linear kernel SVM case are that the extracted feature vectors are mapped to the higher dimensional space before training, and that the LUT approximation is used during prediction.

Because the chi-square and histogram intersection kernels are 1-homogeneous, i.e. $K(cx, cy) = cK(x, y)$ for any constant $c$, L1 normalization of the features is used instead of L2 normalization as suggested in [29]. This results in $K(x, x) = d$. That is, the value of the kernel function when the two inputs are identical is always a fixed constant $d$. For the linear kernel which is 2-homogenous, i.e. $K(cx, cy) = c^2 K(x, y)$, L2 normalization yields $K(x, x) = d$.

The results for the histogram intersection kernel for cmB, cmD, and cmE are shown in Figures 4.12, 4.13, and 4.14, respectively. The results for the additive chi-square kernel for cmB, cmD, and cmE are shown in Figures 4.15, 4.16, and 4.17, respectively. Two values for the dimensionality of the resultant feature mapping are tested: 3 and 5 times larger than the original feature space.
FIGURE 4.12. ROC curves for Camera B (cmB) using histogram intersection kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C.
FIGURE 4.13. ROC curves for Camera D (cmD) using histogram intersection kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C.
FIGURE 4.14. ROC curves for Camera E (cmE) using histogram intersection kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C.
FIGURE 4.15. ROC curves for Camera B (cmB) using chi-square kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C.
FIGURE 4.16. ROC curves for Camera D (cmD) using chi-square kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C.
FIGURE 4.17. ROC curves for Camera E (cmE) using chi-square kernel explicit feature map with dimensionality increase of three (left) and five (right) on Lanes A, B, and C.
To help summarize the results, the NAUC @ 0.1 right hand side for the vertically averaged result from each camera and lane combination is shown in Table 4.4.

TABLE 4.4

Results for additive kernel tests. Numbers shown are NAUC @ 0.1 right hand side for the vertically averaged curve from each lane. HistX and ChiX indicate histogram intersection kernel and chi-square kernel with X factor dimensionality enlargement. Lin indicates linear kernel.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Camera B</th>
<th></th>
<th></th>
<th></th>
<th>Camera D</th>
<th></th>
<th></th>
<th></th>
<th>Camera E</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lane A</td>
<td>Lane B</td>
<td>Lane C</td>
<td>Lane A</td>
<td>Lane B</td>
<td>Lane C</td>
<td>Lane A</td>
<td>Lane B</td>
<td>Lane C</td>
<td>Lane A</td>
<td>Lane B</td>
</tr>
<tr>
<td>Hist3</td>
<td>0.865</td>
<td>0.743</td>
<td>0.582</td>
<td>0.880</td>
<td>0.866</td>
<td>0.657</td>
<td>0.791</td>
<td>0.659</td>
<td>0.560</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hist5</td>
<td>0.866</td>
<td>0.744</td>
<td>0.583</td>
<td>0.881</td>
<td>0.865</td>
<td>0.652</td>
<td>0.791</td>
<td>0.660</td>
<td>0.561</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chi3</td>
<td>0.865</td>
<td>0.748</td>
<td>0.583</td>
<td>0.883</td>
<td>0.864</td>
<td>0.654</td>
<td>0.791</td>
<td>0.669</td>
<td>0.558</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chi5</td>
<td>0.866</td>
<td>0.744</td>
<td>0.585</td>
<td>0.882</td>
<td>0.861</td>
<td>0.654</td>
<td>0.792</td>
<td>0.663</td>
<td>0.559</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lin</td>
<td>0.863</td>
<td>0.741</td>
<td>0.569</td>
<td>0.888</td>
<td>0.870</td>
<td>0.645</td>
<td>0.790</td>
<td>0.655</td>
<td>0.544</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As seen in Table 4.4, no significant performance difference was seen between the histogram intersection and chi-square kernels for any of the cameras or lanes. There was also no significant performance difference between dimensionality enlargement factors of 3 and 5. For cmB and cmE, both the histogram intersection and chi-square kernel gave a small bump, roughly 1.5 percent, in performance for Lane C versus the linear kernel. This small improvement is probably not worth the added computational expense.

While disappointing, these results are not unexpected. In the paper describing their explicit feature maps [29], Vedaldi and Zisserman report little difference between the histogram intersection and chi-square kernels, as well as little difference between dimensionality enlargement factors. They show significant gains over the linear kernel on two datasets: Caltech-101 and the INRIA pedestrian benchmark. However, recent research has shown that the histogram intersection and chi-square kernels do not improve pedestrian detection performance when better evaluation metrics are used [74]. Thus, as with many machine
learning techniques, the usefulness of additive non-linear kernels seems to be very much problem dependent.

4.10 Comparison to Ensemble of Size-Contrast Filters Prescreening Algorithm

This section compares the proposed MS-HOG prescreener, referred to MSHP, to the ensemble of size-contrast filters prescreener, referred to as ESCP, described in [9], [51]. Results for all five cameras: cmA, cmB, cmC, cmD, and cmE are presented in Figures 4.18 through 4.22 respectively. ESCP is trained for 75 epochs, and Contrast Limited Adaptive Histogram Equalization (CLAHE) preprocessing is performed prior to ESCP training in order to eliminate large contrast differences between the runs.

MSHP is significantly faster to train than ESCP. On a system with an Intel Core i7-3930K CPU, 64 GB of RAM, and NVIDIA GeForce GTX 580 and GTX 780 GPUs, ESCP requires roughly 20 hours to train all three folds for cmB. This does not include the time for image contrast enhancement using CLAHE. CLAHE significantly improves ESCP’s performance when runs have large differences in contrast, as is the case for the December 2013 dataset of which cmA, cmB, and cmE are part. ESCP is multithreaded, and uses both GPUs heavily. MSHP on the other hand, requires only 30 minutes to train, and does not use the GPUs at all. It also does not require CLAHE preprocessing. Due to the slow training of ESCP, it is run only a single time for each experiment even though there is variation between runs due to the randomness of genetic algorithm training. While MSHP is faster to train, it is much slower to test. On the same Intel system, MSHP can process 6 frames per second on cmB when using the linear kernel and multithreading. ESCP can easily process 150+ frames per second. The slower processing speed of MSHP will hopefully be offset by its improved detection performance.
FIGURE 4.18. ROC curves for Camera A (cmA) for proposed MS-HOG prescreener (left) and Ensemble of Size-Contrast Filters prescreener (right) on Lanes A, B, and C.
FIGURE 4.19. ROC curves for Camera B (cmB) for proposed MS-HOG prescreener (left) and Ensemble of Size-Contrast Filters prescreener (right) on Lanes A, B, and C.
FIGURE 4.20. ROC curves for Camera C (cmC) for proposed MS-HOG prescreener (left) and Ensemble of Size-Contrast Filters prescreener (right) on Lanes A, B, and C.
Figure 4.21. ROC curves for Camera D (cmD) for proposed MS-HOG prescreener (left) and Ensemble of Size-Contrast Filters prescreener (right) on Lanes A, B, and C.
FIGURE 4.22. ROC curves for Camera E (cmE) for proposed MS-HOG prescreener (left) and Ensemble of Size-Contrast Filters prescreener (right) on Lanes A, B, and C.
To help summarize the results, the NAUC @ 0.1 right hand side for the vertically averaged result of each lane for each camera and algorithm combination is shown in Table 4.5.

<table>
<thead>
<tr>
<th>Camera</th>
<th>Lane A</th>
<th>Lane B</th>
<th>Lane C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSHP</td>
<td>ESCP</td>
<td>MSHP</td>
</tr>
<tr>
<td>A</td>
<td>0.807</td>
<td>0.656</td>
<td>0.790</td>
</tr>
<tr>
<td>B</td>
<td>0.865</td>
<td>0.709</td>
<td>0.748</td>
</tr>
<tr>
<td>C</td>
<td>0.858</td>
<td>0.837</td>
<td>0.815</td>
</tr>
<tr>
<td>D</td>
<td>0.883</td>
<td>0.842</td>
<td>0.864</td>
</tr>
<tr>
<td>E</td>
<td>0.791</td>
<td>0.711</td>
<td>0.669</td>
</tr>
</tbody>
</table>

As seen in Table 4.5, MSHP outperforms ESCP for every camera and lane combination. The largest improvements are seen with cmA, cmB, and cmE, which are the cameras from the December 2013 data collection. Because of the time of year, the target signatures in many of the runs from the December 2013 data collection have much lower contrast, as compared to runs from the May 2012 data collection. Thus, cmA, cmB, and cmE would be expected to show the largest gains.

In terms of individual run performance, MSHP bested ESCP in all runs for cmA, cmB, and cmE. For cmC, there were three runs from Lane A where MSHP was outperformed by ESCP. Both algorithms score well on those three runs, with NAUC above 0.91, and the differences are less than 0.02 NAUC. The difference is larger than the 95% confidence interval for only one of the three runs. On cmD, MSHP was outperformed by ESCP on three runs from Lane C. For one of those runs the difference was in excess of 0.12 NAUC, which is a considerable drop. Further investigation is needed to determine the reason for the drop. A likely candidate is geo-
referencing error. The MSHP algorithm has higher sensitivity to registration noise than ESCP because it does not learn the radius for mean-shift from the training data. Instead, it uses a fixed 50cm radius. Having MS-HOG learn the radius for mean-shift clustering from the training data, or replacing mean-shift with another clustering algorithm, is an avenue for future work.

Because the MS-HOG prescreening algorithm trains a classifier using training data, special care must be taken when training a secondary classifier to follow it. This is different than for the case of a prescreener which does not learn parameters from training data, as is common in GPR and IR BEH detection. The correct procedure is discussed in Section 3 of the Appendix.

A common criticism of image feature based SVM classifiers is that they are a black box, and that it is hard to get an intuitive sense of how a trained classifier makes its decisions. To try to gain insight into how the MS-HOG prescreener models are working, we attempt to visualize models learned using the linear kernel. This analysis is presented in Section 2 of the Appendix.
Chapter 5: Conclusions

5.1 Summary

A new method for geo-referencing infrared imagery collected from a single camera mounted on a moving platform with an attached INS, referred to as Flat Earth 2.0, was proposed, developed, and evaluated. Flat Earth 2.0 was able to achieve accuracies equal to SfM at a significantly reduced computational cost, as long as the observed scene roughly matched the model’s underlying assumptions. The ability to scale the computations based on the number of pixels that need to be geo-referenced makes it feasible for real-time operation. Even for less than ideal scene conditions, Flat Earth 2.0 was able to outperform the older Flat Earth 1.0 model in terms of accuracy.

A new prescreening algorithm for detection of BEHs in FL-IR imagery was presented. The proposed algorithm used a sliding window SVM classifier coupled with MS-HOG features extracted in a cell-structured manner from the detection window. It significantly outperformed the current state-of-the-art FL-IR BEH prescreening algorithm, which uses an ensemble of size-contrast filters trained via genetic algorithm, on multiple cameras and datasets. The largest improvements were seen in difficult, low-contrast conditions. It was also shown how boxcar images could be used in place of integral images for a detector utilizing a small number of fixed cell sizes, and how the LUT additive kernel approximate prediction technique could be combined with explicit feature maps to allow fast training and prediction of non-linear additive kernels such as histogram intersection and chi-square. The effects of non-linear additive kernels, hard
negative mining, and removal of non-visible targets during training on prescreener performance were investigated.

5.2 Future Work

In regards to the Flat Earth 2.0 model, the obvious area for future research is to develop a process for identifying the ground plane within the detection region of interest and removing keypoint matches that lie outside of it when computing the frame to frame homographies. There are a number of ways this could be performed. One is to use a road finding algorithm to identify pixels in the image belonging to the road surface. Another possible approach is to employ ground plane detection techniques from mobile robotics. However, these techniques generally rely on the presence of a single dominant horizontal plane interspersed with many near vertical planes, as would be found in indoor environments. The outdoor environments typical of our datasets have much different configurations, and do not have such perfectly planar surfaces, nor such clear distinctions between planar surfaces.

Replacing the perspective projection in the FE2 DLTPT approach with another mathematical transformation or learning a different perspective projection for sub-sections of the image are also possibilities. The latter option is actually a popular technique in image stitching [75], [76]. Traditional image stitching relies on homographies to warp images to a common image plane for creation of large panoramas. This works when the camera undergoes only rotation or if the scene is planar. However, since many users of stitching software are unaware of these requirements, photos used for stitching often violate them. Thus, much research has been done on adaptive techniques, such as piece-wise homographic warping, for handling such cases.
An interesting future direction for the MS-HOG prescreener is to have it learn the radius for mean-shift clustering from the training data, or even replace mean-shift with another clustering algorithm. This would allow the prescreener to adapt to the geo-referencing error distribution. Another research direction is the use of label noise tolerant training algorithms. However, accurate evaluation of such techniques is quite problematic, as mentioned in Section 4.7. The application of the MS-HOG prescreener to FL-GPR data is also of interest. Adapting the MS-HOG prescreener algorithm to FL-GPR data should require little modification since FL-GPR data can easily be converted into a real-valued image. Furthermore, the image to world coordinate mapping can be modeled using the same techniques used for FL-IR. The ensemble of size-contrast filters prescreening algorithm, which, like the MS-HOG algorithm, was developed with infrared imagery in mind, has shown good performance in the FL-GPR domain. It even bested a FL-GPR specific prescreening algorithm in [9]. Thus, the MS-HOG algorithm should perform quite well for that task.
Chapter 6: Appendix

6.1 Iterative Linear-LS Triangulation for Many Points at Once

In Section 3.2 it is mentioned that the iterative linear-LS triangulation algorithm can be applied to many independent points at once even with varying numbers of equations per point. MATLAB code for this is presented here. To use this code, the terms from the equations arising for each pixel position from each view must be arranged into three variables: Q, R, and S. Q is a 3x1 cell array. Each cell of Q contains an N by 2M matrix, where N is the number of pixels in the image - the code solves for the unknown 3D world location [X Y Z] at each pixel - and M is the maximum number of views for any pixel. Recall the notation for the perspective projection mapping of world coordinate [X Y Z] to image coordinate [U V]:

\[
\begin{bmatrix}
U \\
V \\
1
\end{bmatrix} =
\begin{bmatrix}
A & B & C & D \\
E & F & G & H \\
I & J & K & L
\end{bmatrix}
\begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]

\[X(UI - A) + Y(UJ - B) + Z(UK - C) = D - UL\]
\[X(VI - E) + Y(VJ - F) + Z(VK - G) = H - VL\]

Remember, two equations for the three variables X, Y, and Z arise from each view. The first cell of Q collects the multipliers for X’s. The second cell of Q collects the multipliers for Y’s. The third cell of Q collects the multipliers for Z’s. In the following equations, the single subscript on A through L indexes the view. The two subscripts on U and V index the pixel position in the image and the view respectively. Matrix R collects the target values. S is a 2M by 4 matrix that
contains the I through L terms for each view. If a pixel does not have equations for a view, simply fill in the corresponding entries in Q and R with zero.

\[
Q(1) = \begin{bmatrix}
A_0 - U_{0,0}I_0 & E_0 - V_{0,0}I_0 & A_1 - U_{0,1}I_1 & E_1 - V_{0,1}I_1 & \cdots \\
A_0 - U_{1,0}I_0 & E_0 - V_{1,0}I_0 & A_1 - U_{1,1}I_1 & E_1 - V_{1,1}I_1 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}
\]

\[
Q(2) = \begin{bmatrix}
B_0 - U_{0,0}J_0 & F_0 - V_{0,0}J_0 & B_1 - U_{0,1}J_1 & F_1 - V_{0,1}J_1 & \cdots \\
B_0 - U_{1,0}J_0 & F_0 - V_{1,0}J_0 & B_1 - U_{1,1}J_1 & F_1 - V_{1,1}J_1 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}
\]

\[
Q(3) = \begin{bmatrix}
C_0 - U_{0,0}K_0 & G_0 - V_{0,0}K_0 & C_1 - U_{0,1}K_1 & G_1 - V_{0,1}K_1 & \cdots \\
C_0 - U_{1,0}K_0 & G_0 - V_{1,0}K_0 & C_1 - U_{1,1}K_1 & G_1 - V_{1,1}K_1 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}
\]

\[
R = \begin{bmatrix}
U_{0,0}L_0 - D_0 & V_{0,0}L_0 - H_0 & U_{0,1}L_1 - D_1 & V_{0,1}L_1 - H_1 & \cdots \\
U_{1,0}L_0 - D_0 & V_{1,0}L_0 - H_0 & U_{1,1}L_1 - D_1 & V_{1,1}L_1 - H_1 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}
\]

\[
S = \begin{bmatrix}
I_0 & J_0 & K_0 & L_0 \\
I_0 & J_0 & K_0 & L_0 \\
I_1 & J_1 & K_1 & L_1 \\
I_1 & J_1 & K_1 & L_1 \\
\vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}
\]

Once the Q, R, and S variables are constructed, simply call the function \textit{IterativeLinearLS} given below. The parameter \textit{miters} sets the maximum number of iterations. This code is able to solve the independent least squares problems simultaneously by separating the X, Y, and Z multipliers into different matrices. This allows the solutions to be computed via the Moore-Penrose pseudoinverse method using a series of matrix operations as shown in the \textit{solveNx3} function below. If the difference in triangulated position for a pixel from one iteration to the next falls below 1cm, that pixel is excluded from further iterations in order to minimize processing time.

131
function X = IterativeLinearLS(Q,R,S,miters)

% initial solution with all weights=1
X = [ solveNx3(Q,R); ones(1,size(R,1)); ];

% iteratively refine
idx = true(1,size(X,2));
for i=1:miters
    w = realsqrt(1.0./abs(S*X(:,idx)))';
    Qw = cell(1,3);
    Qw{1} = Q{1}(idx,:).*w;
    Qw{2} = Q{2}(idx,:).*w;
    Qw{3} = Q{3}(idx,:).*w;
    Rw = R(idx,:).*w;
    Xn = solveNx3(Qw,Rw);
    diff = sum((X(1:3,idx)-Xn).^2);
    X(1:3,idx) = Xn;
    ind = find(idx);
    idx(ind(diff<0.01)) = false;
end
end

function X = solveNx3(A,B)

a{1} = sum(A{1}.*A{1},2)'; % cross terms
a{2} = sum(A{1}.*A{2},2)';
A{3} = sum(A{1}.*A{3},2)';
A{4} = sum(A{2}.*A{2},2)';
A{5} = sum(A{2}.*A{3},2)';
A{6} = sum(A{3}.*A{3},2)';
A{7} = sum(A{1}.*B,2)';
A{8} = sum(A{2}.*B,2)';
A{9} = sum(A{3}.*B,2)';

% inverse of 3x3 matrix:
% | a{1} a{2} a{3} |
% | a{2} a{4} a{5} | =>
% | a{3} a{5} a{6} |
% |
% | c0 c1 c2 | * (1/det)
% | c1 c3 c4 |
% | c2 c4 c5 |

% inverse of 3x3 matrix:
% | a(1) a(2) a(3) |
% | a(2) a(4) a(5) | =>
% | a(3) a(5) a(6) |
% |
% | c0 c1 c2 | * (1/det)
% | c1 c3 c4 |
% | c2 c4 c5 |

X = bsxfun(@times, ... [ c0.*a{7} + c1.*a{8} + c2.*a{9}; ... c1.*a{7} + c3.*a{8} + c4.*a{9}; ... c2.*a{7} + c4.*a{8} + c5.*a{9}; ], idet);
end
6.2 Visualizing the MS-HOG SVM Classifier

To better understand what the MG-HOG prescreener is learning, we attempt to visualize the resulting SVM classifiers. This is easiest for the linear kernel case because the explicit form of the weight vector is known. Since all of the HOG features take on positive values in the range 0-1, the magnitude and sign of the corresponding weight for each feature gives a good idea of its importance for classification.

To visualize the classifier weights, images for each cell grid, i.e. the 5x5 coarse grid and the 4x8 fine grid, are created separately. For each grid image, the weights are organized visually by cell. For each cell, there are a total of 72 weight values corresponding to the 72 image features - 12 orientations for each of the 3 HOG scales = 36, and each of those 36 values is normalized by two methods 1.) energy across all scales and orientations 2.) energy across matching scale and orientation only. These 72 weights are organized in a 6 by 12 rectangle, where the first six columns are for normalization scheme 1 and the second six columns are for normalization scheme 2. Within the six columns, columns 1-2 correspond to the first HOG scale, columns 3-4 correspond to the second HOG scale, and columns 5-6 correspond to the third HOG scale. Within each set of two columns, the values for each gradient orientation are arranged from top->bottom, left->right starting at orientation -180 degrees (large positive gradient from right to left) and then proceeding in order counter-clockwise around the unit circle: -180, -90, 0, 90, 180. White pixels in the image indicate large positive weights. Black pixels indicate large negative weights. Gray indicates zero. Figure 4.23 shows the visualization of prescreener models learned on cmA (top) and cmB (bottom).
FIGURE 4.23. Visualization of two trained linear kernel SVM classifiers. Top two images: cmA. Bottom two images: cmB. White indicates positive weights. Black indicates negative weights.
Looking at the top most image of Figure 4.23, it is clear that the classifier is looking for strong vertical gradients across all three scales in the center cell of the coarse 5x5 grid. Furthermore, it is favoring the features normalized using the second normalization scheme, suggesting that it does not care about strong gradients in other orientations. In the cells immediately above and below the center cell, these same orientations are given negative weights. In the cells to the far left and right, horizontal gradients at the finest scale are preferred. Thus, as might be expected, the classifier is looking for horizontally elongated objects in the center of the detection window that have vertical edges on the left and right.

The second image in Figure 4.23, which visualizes the finer 4x8 grid, tells much the same story. The interesting observation is that it favors opposite vertical gradient directions in the cells directly above and below the center of the detection window. That is, the two center cells in the second row favor the -90 degree orientation, but the center cells in the third row favor the 90 degree orientation. Because the infrared images are adjusted so that targets should appear warmer than the background, one would expect the transition from below, i.e. upward from background to target, to be black to white (90 degrees) and the next upward transition from the target to background to be white to black (-90 degrees), which is exactly the relationship the classifier is favoring. The reason why this is observed in the fine 4x8 grid and not the coarse 5x5 grid is that the center cell of the 5x5 grid completely overlaps both transition zones. Therefore, it cannot discriminate between dark on bright and bright on dark objects. For the 4x8 grid, the cells in the second and third row overlap only one of the transition zones, allowing the classifier to discriminate between these cases.
Unsurprisingly, the bottom two images in Figure 4.23 tell much the same story as the top two. Thus, the classifier seems to be learning exactly what one would expect.

### 6.3 Training a Secondary Classifier

Because the proposed prescreening algorithm requires learning parameters from the training data, some care is needed when training a secondary classifier to follow it. For prescreening algorithms that do not learn parameters from the training data, the simple approach of running the prescreener on each training run and using the resultant alarm locations as the training input to the secondary classifier will work just fine. The reason why this works is because the distribution of alarm locations from the training data would be expected to match the distribution of alarms from unseen data because no fitting to the training data took place. However, for a prescreener that is tuned to the training data, this simple approach will not work because the distribution of alarms from the training data would not be expected to match the distribution of alarms from unseen data.

The technique of training a classifier to use the output of one or more prior classifiers as input for classification or prediction is referred to as **stacking** and **blending** in the literature, and is a well-studied problem. The important principle from stacking that applies to our case is that the secondary classifier must be trained on out-of-bag samples from the prescreener. That is, the input to the secondary classifier must be alarm locations from the prescreener generated on unseen data. To achieve this, the training runs should be split into several groups, ideally grouped by lane if lane based cross validation is the final evaluation metric, for cross validation. Each group should be held out one time while the prescreener is trained on the remaining groups. The resulting prescreener model should then be used to generate alarm locations, i.e.
the out-of-bag samples, on the held out group of runs. After repeating this process for each
group of runs, the out-of-bag alarm locations from all runs can be used as the training data for
the secondary classifier.

6.4 Estimation of Plane Normal Vector and Depth to Plane

This section describes a method for calculating the normal vector to the plane and the
depth to the plane for a plane viewed by a pinhole camera from two different locations, \( Q_1 \) and
\( Q_2 \), given point correspondences between the images and the rotation, \( R \), and translation, \( T \),
of the camera from \( Q_1 \) to \( Q_2 \). The normal vector and depth are solved in \( Q_1 \) relative
coordinates.

Starting with the equation for a homography, where \( K \) is the intrinsic parameter matrix
of the camera, \( [U \ V \ 1] \) is a pixel coordinate, \( n \) is the normal vector to the plane, and \( \vec{d} \) is
the depth to the plane.

\[
\begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix}_{Q_2} = H \begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix}_{Q_1} = K \left( R - \frac{Tn^T}{d} \right) K^{-1} \begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix}_{Q_1}
\]

\[
\begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix}_{Q_2} = KRK^{-1} \begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix}_{Q_1} - K \left( \frac{Tn^T}{d} \right) K^{-1} \begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix}_{Q_1}
\]
And rewriting with the following substitutions.

\[
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix} = K R K^{-1}
\begin{bmatrix}
u \\
v \\
v
\end{bmatrix}_{Q^1}
\begin{bmatrix}
d \\
\bar{d}
\end{bmatrix} = KT
\begin{bmatrix}
g \\
h
\end{bmatrix} = \frac{n}{d}
\begin{bmatrix}
j \\
\bar{j}
\end{bmatrix} = K^{-1}
\begin{bmatrix}
u \\
v
\end{bmatrix}_{Q^1}
\]

\[
\begin{bmatrix}
u \\
v \\
v
\end{bmatrix}_{Q^2} =
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix} -
\begin{bmatrix}
d \\
g \\
f
\end{bmatrix} [i, j, k, l] \Leftrightarrow
\begin{bmatrix}
u \\
v \\
v
\end{bmatrix}_{Q^2} =
\begin{bmatrix}
a - (d g j + d h k + d i l) \\
b - (e g j + e h k + e i l) \\
c - (f g j + f h k + f i l)
\end{bmatrix}
\]

\[
u(c - (f g j + f h k + f i l)) = a - (d g j + d h k + d i l)
\]

\[
v(c - (f g j + f h k + f i l)) = b - (e g j + e h k + e i l)
\]

\[
u c - a = g j (u f - d) + h k (u f - d) + i l (u f - d)
\]

\[
v c - b = g j (v f - e) + h k (v f - e) + i l (v f - e)
\]

Yields a closed form solution via least squares \( A x = b \), shown below, where \( [ ]_N \) indicates that all pixel-wise terms inside the brackets arise from pixel correspondence \( N \). Each pixel correspondence contributes two equations.
$$A = \begin{bmatrix} [j(uf - d)]_N & [k(uf - d)]_N & [l(uf - d)]_N \\ [j(vf - e)]_N & [k(vf - e)]_N & [l(vf - e)]_N \\ \vdots & \vdots & \vdots \end{bmatrix} \quad b = \begin{bmatrix} [uc - a]_N \\ [vc - b]_N \end{bmatrix}$$

$$\tilde{d} = ||x|| \quad \text{and} \quad n = \frac{x}{||x||}$$

As in the case of iterative linear-LS triangulation, an iterative scheme can be used to more closely minimize the true projective error. At each iteration, the weight for each of the two equations arising from pixel correspondence $N$ is set equal to the following value. The iterative process is terminated when the change in $x$ between iterations falls below a threshold. For the first iteration all weights are set equal to 1. $|| \cdot ||$ indicates absolute value.

$$W_N = \frac{1}{[c - (fgj + fhk + fil)]_N}$$
REFERENCES


Kevin Eliot Stone was born on March 6, 1984 in Warrensburg, MO. He received dual Bachelor’s Degrees in Electrical and Computer Engineering, a Master’s Degree in Computer Engineering, and a Ph.D. in Electrical and Computer Engineering all from the University of Missouri – Columbia. He was a research assistant to Dr. James Keller between 2007-2014. His research interests include computer vision, image and video processing, machine learning, and computational intelligence.