Summer 2012

# A comparison of near-infrared and visible imaging for surveillance applications

Kathryn Nicole Rodhouse



Department:

# A COMPARISON OF NEAR-INFRARED AND VISIBLE IMAGING FOR SURVEILLANCE APPLICATIONS

**by**

**KATHRYN NICOLE RODHOUSE**

**A THESIS**

**Presented to Faculty of the Graduate School of the**

**MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**In Partial Fulfillment of the Requirements for the Degree**

**MASTER OF SCIENCE IN COMPUTER ENGINEERING**

**2012**

**Approved by**

**Dr. Steve E. Watkins, Advisor**

**Dr. R. Joe Stanley**

**Dr. Donald C. Wunsch**

# ABSTRACT

A computer vision approach is investigated which has low computational complexity and which compares near-infrared and visible image systems. The target application is a surveillance system for pedestrian and vehicular traffic. Near-infrared light has potential benefits including non-visible illumination requirements. Image-processing and intelligent classification algorithms for monitoring pedestrians are implemented in outdoor and indoor environments with frequent traffic.

The image set collected consists of persons walking in the presence of foreground as well as background objects at different times during the day. Image sets with non-person objects, e.g. bicycles and vehicles, are also considered. The complex, cluttered environments are highly variable, e.g. shadows and moving foliage. The system performance for near-infrared images is compared to that of traditional visible images.

The approach consists of thresholding an image and creating a silhouette of new objects in the scene. Filtering is used to eliminate noise. Twenty-four features are calculated by MATLAB© code for each identified object. These features are analyzed for usefulness in object discrimination. Minimal combinations of features are proposed and explored for effective automated discrimination. Features were used to train and test a variety of classification architectures.

The results show that the algorithm can effectively manipulate near-infrared images and that effective object classification is possible even in the presence of system noise and environmental clutter. The potential for automated surveillance based on near-infrared imaging and automated feature processing are discussed.

## ACKNOWLEDGEMENTS

It is a pleasure to thank those who made this thesis possible. I would like to express my utmost gratitude to Dr. Steve E. Watkins, for his encouragement, guidance and support at every step of this research. Without Dr. Watkins's unfailing confidence, this study would not have been successful. I wish to also thank Dr. R. Joe Stanley and Dr. Donald C. Wunsch for their boundless guidance as members of my graduate committee.

This work was supported by the Department of Electrical and Computer Engineering through a teaching assistantship. I owe my deepest gratitude for this support.

I am indebted to all of my peers who aided in this research, especially Kevin E. Robison. I also thank Dao Lam of the Missouri S&T Applied Computational Intelligence Laboratory and Dr. Randy H. Moss for the technical discussions they offered. The efforts of all of those researchers in the computer vision field, which aided me in this research, are also acknowledged. I would like to specifically thank Rui Xu for providing ssEAM architecture MATLAB code developed by Georgious Anagnostopoulous.

Lastly, I would like to thank my parents, sister and all my family for their love and support throughout this research. Their reassurance and trust contributed greatly to the success of this project.

# TABLE OF CONTENTS

Page

# LIST OF ILLUSTRATIONS

Figure Page

# LIST OF TABLES

Table Page

# 1. INTRODUCTION

Surveillance systems are used to collect information from an environment in order to monitor traffic or detect abnormal or critical situations. Such systems utilize visible or infrared technology to collect images of the observed environment. While thermal infrared imaging and visible imaging have both been explored extensively in security applications, near-infrared imaging has remained largely uncharted. In this research, an image manipulation algorithm and intelligent processing approach is developed and applied to near-infrared surveillance images.

Visible and infrared spectra provide different types of information to a surveillance system. While visible light systems can provide information similar to what the human eye would process, visible light is incapable of providing useful information in certain types of situations. For example, visible light systems will not provide valuable information in a foggy setting or a setting where the illumination is poor. Visible light provides non-optimal data in these extreme situations, so there exists a need for alternative imaging systems if those situations have a reasonable chance of occurring within the observed environment. Thermal infrared imaging is one such reliable system, because infrared information will be processed from an environment independent of the quality of the environmental light source(s).

Thermal infrared imaging can be particularly useful for surveillance of people or animals in an assortment of environments because living beings produce unique heat signatures. These heat signatures typically fall within the far-wavelength infrared range (~ 8 – 12μm) [1]. To characterize thermal infrared imaging usefulness imagine a

stationary camera used for surveillance of an environment. A reference image of the environment could be autonomously compared to the current environment to detect significant heat differences and therefore changes. When an animal or a person enters the environment, a thermal infrared image would show significant change. In outdoor applications such as game tracking systems or outdoor security systems, a reference image may not be necessary to track movement of animals or people because environmental heat intensity will likely differ significantly from the intensity of a person or animal. However, thermal infrared imaging may suffer in warm environments and is based on expensive camera hardware.

Security imaging systems must accommodate non-target activity and background variations (e.g. lighting or seasonal changes) and their implementation may be limited by hardware and operational expense. Simple imaging systems obtain useful information in a variety of environmental conditions, but a human observer or a computer system may have difficulty processing that information real-time [Friedrich, 2002]. Automation of image manipulation within surveillance systems allows for more effective surveillance at any given time by targeting the desired image. This filtering simplifies the observable image data to only critical data for the security system.

Currently, developed algorithms are used to detect abnormal or unsafe situations in visible light surveillance systems. Additionally, algorithms have been created for specific thermal infrared security systems, such as forest-fire detections systems [Arrue, 2000]. However, little has been done with near-infrared security systems. The high cost of thermal infrared cameras makes such surveillance systems unrealistic in many situations. Near-infrared cameras have been found to perform effectively in

environments that visible light cameras cannot due to lighting conditions, such as in smoky rooms [Sentenac, 2003]. Also, near-infrared (non-visible) illumination can be added for nighttime situations without alerting the target. Therefore, results of visible and thermal light algorithms could be applied within near-infrared security systems to determine if any benefits exist in the new system. This research explores the surveillance potential of near-infrared imaging.

In this work, algorithm methods with low computational requirements are applied to near-infrared and visible images of complex scenes. These methods include noise filtering as well as image manipulation operations. The image sets include outdoor and indoor environments. Pedestrian and vehicular target visibility is compared for near-infrared and visible images and is examined with system noise. Twenty-four target features are discussed as inputs for automated classification methods. These features are analyzed for classification effectiveness and reduced feature sets are proposed. Target features are used as inputs to two types of classification systems: 1) MLP neural network and 2) semi-supervised Ellipsoid ARTMAP. Classification results are used to compare visible and near-infrared system capabilities across a broad spectrum of environments.

In Section 2, a literature review is presented concerning computer vision research using visible, thermal infrared and near-infrared technologies. Section 3 presents the important research that inspired this work. The image sets collected for training and testing of the proposed classification systems are outlined in Section 4. An image processing algorithm for visible and near-infrared systems is presented in Section 5. Section 6 explains the feature analysis completed to produce a reduced feature set. Additionally, Section 6 describes the architecture developed for object classification.

Experimental results for feature analysis and architecture accuracy are explored in Section 7. These results show that effective surveillance is possible with near-infrared images even in the presence of environmental clutter and system noise.

# 2. LITERATURE REVIEW

Computer vision techniques aim to enable computers to process visual information similar to a brain. The brain processes information by extracting meaningful semantic features such as boundaries and shapes, but similar features can be difficult to process efficiently in computer systems [Zhang, 2010]. Historically techniques have been researched and refined to improve computer vision capabilities. According to Miller et al. [Miller, 2011], the scope of a computer vision problem for software-engineering applications includes 1) access, 2) transfer, 3) convert, 4) modify, and 5) analyze. Access is defined by the retrieval of image data. Transferring is the process of communicating retrieved image data. The conversion process maps data into the required format. Modification includes applying filters, cropping and transforming formatted data. The last scope point, analysis, encompasses using vision techniques to understand the image data.

## 2.1. BACKGROUND ON COMPUTER VISION

The computer vision field has grown significantly since its founding in the 1970s due to the complex nature of computer vision problems [Shah, 2002]. In the last decade, improved sensor and memory technologies have further fueled computer vision growth. Thermal infrared and near-infrared sensors are being explored as information providing alternatives for computer vision systems that operate in the visible spectrum. Improvements in computer technology have allowed the computer vision and computer graphics fields to become highly interrelated [Rockwood, 1999]. Computer graphics is

defined as the process of building a computer model and then displaying them with algorithms to produce an image. Similarly computer vision is the process of creating a computer model from an image. The increased interrelation of these fields has allowed many researchers to draw inspiration from both fields.

Feature-based techniques have also recently become a trend in the computer vision industry for the purpose of object recognition. These techniques allow particular aspects of an object to be analyzed. Some researches have utilized sensor integration to further computer vision systems. Such integration has been used to develop three dimensional environmental model spaces. Other types of sensor integration have allowed new types of sensory information to be incorporated into observed environmental models. For instance by using both infrared and visible wavelength sensors, both observational and thermal information from the environment can be preserved.

Various approaches have been demonstrated for the analysis of images. These approaches are highly dependent on application and scene complexity. They range from fixed algorithms for highly constrained situations, e.g. Stanley et al. [Stanley, 2006] to more flexible intelligent means to handle complex situations. Neural network based approaches are of much interest for handling highly variable information from multiple image features.

## 2.2. COMPUTER VISION APPLICATIONS

This section discusses recent computer vision research and applications completed for the purpose of object detection and surveillance. Neural network

processing is emphasized. Research herein was completed with a variety of sensors including visible, thermal infrared and near-infrared wavelength specific cameras.

**2.2.1. Small Target Detection in High Clutter.** Shirvaikar et al. [Shirvaikar, 1995] developed a neural network filter to detect very small targets in thermal infrared images. High-resolution aerial imagery of an environment was investigated. This algorithm eliminated the need for feature extraction, as thermal raw gray level intensities were descriptive enough to be inputted to the neural network. A backpropagation training algorithm was chosen to train the network. It was found that for small target tasks, a neural network filter performed very well in thermal infrared imagery.

**2.2.2. Multiple Neural Networks for Target Recognition**. Correia et al. [Correia, 2001] automatically detected motor vehicles in highly cluttered infrared images in this research. To perform object detection, several small multi-layer-perceptron neural networks were modularly combined into a larger neural network. This research demonstrates that tanks, trucks, cars, airplanes and helicopters were capable of being classified with thermal infrared images.

**2.2.3. Face Recognition in Dark Environments.** Through this research, Friedrich et al. [Friedrich, 2002] explore the capabilities that infrared sensors can provide to specific computer vision problems. This research reported that in certain situations thermal infrared imaging can provide more invariant images than a visible imaging counterpart, especially in poor lighting conditions. It is interesting to note that after preprocessing the image with a customized algorithm, Fredrich and Yeshurun then note the applicability of commonly used face detection methods used in the visible light

domain. Despite the differing spectral band information, commonly used algorithms could be used in both cases.

**2.2.4. Algorithmic Object Detection.** Sentenac et al. [Sentenac, 2004] demonstrate that customized algorithms can be used to detect abnormal and critical situations in a monitored environment. By utilizing low-cost cameras operating in the near-infrared spectral band along with an illuminating IR source, temperature characteristics of the environment can be measured [Sentenac, 2002]. Near-infrared sensors were selected because of their ability to measure features of fire appearance, load displacements, and smoke. Through specific feature extraction of the observed environment, algorithms could be developed to determine if a fire, smoke, or movement of cargo was present within an aircraft.

**2.2.5. Near-Infrared Face Detection.** In this research conducted by Dowdall et al [Dowdall, 2003], face detection and skin detection in near-infrared images is explored. This research notes that human skin and facial features have specific reflective qualities independent of the nationality of the human. This research also notes that near-infrared surveillance systems can remain unobtrusive and covert as the eye does not respond to near-infrared light. Ultimately this research demonstrates that the unique reflective qualities of certain objects can be exploited to develop optimal detection systems.

**2.2.6. Illumination Invariant Face Recognition.** Li et al. [Li, 2007] expands previous face detection research using near-infrared imaging by developing a face recognition system. Local binary patterns are calculated over the image to describe sub-windows within the image. The most indicative sub-windows or features are selected to construct a face-matching engine. Linear Discriminative Analysis, a statistical technique

of finding linear relationships between objects, was used on the features to determine which were most indicative of faces. The results of this research demonstrate the need for feature pruning to develop accurate and fast object recognition systems.

**2.2.7. Pattern Recognition for Detecting Human Heads.** Work completed by Bankman et al. [Bankman, 2008], performs segmentation on infrared image to target human heads for the purpose of military and civilian applications. The proposed algorithm utilizes thermal infrared imagery to locate the shape of a human head through radiance studies. The algorithm first locates target areas within the image of typical skin thermal intensities. Next a Gaussian filter is applied to the located blobs to smooth edges before extracted three features. The three extracted features include compactness (the measure of an object's circularity), chain code (a one-dimensional array describing the border of an object), and radial distances of the object (defined as the distance from each perimeter pixel to the centroid of the object). These features were selected in particular to detect round shapes such as human heads that would be detected in thermal imaging. The algorithm used two neural network classifiers to train the detection system.

**2.2.8. Recognizing Targets Using Artificial Neural Networks.** In this study completed by Aytac et al. [Aytac, 2009], multilayer artificial neural networks are used to identify patterns acquired from low-cost infrared sensors in an indoor environment. These patterns are described by infrared intensity measurements that help define the surface, geometry and location of the observed environment. Two training algorithms were used for the ANN, the back-propagation algorithm and the Levenberg-Marquadt algorithm. This work demonstrates that simple infrared sensors can be effectively trained to classify objects in a specific environment.

## 2.3. INTELLIGENT ANALYSIS APPROACHES

This section discusses intelligent analyses theorems that can be applied to computer vision problems.

**2.3.1. Neural Network Classifier.** Various target features within an image may be determined and feature vectors created as in the research by Watkins et al. [Watkins, 2009]. The target classification can be accomplished using these vectors as inputs to a backpropagation multi-layer perceptron network. Training from known input vectors, as from sample images or image models, provides the classification capability. Feature vectors from unknown images are classified by their similarity to the training set.

**2.3.2. Adaptive Resonance Theory.** Carpenter et al. [Carpenter, 2009] developed a set of algorithms called Adaptive Resonance Theory (ART) based on principles on cognitive theory of how the brain handles objects and events developed by Grossberg [Grossberg, 1980]. ART predicts links between Consciousness, Learning, Expectation, Attention, Resonance and Synchrony to describe how brains adapt in real time to a rapidly changing world. ART was developed to better emulate the way in which humans learn, quick adaptation to new experiences is emphasized without significantly forgetting previous experiences. ART algorithms can be used in large-scale applications such as database prediction, airplane design and autonomous adaptive robots. Neural networks based on ART principles are capable of clustering collections of input patterns. The first neural network ART developed was capable of clustering binary input patterns while unsupervised.

**2.3.3. ARTMAP.** ARTMAP networks are a continuation of the work done with Adaptive Resonance Theory [Carpenter, 2005]. ARTMAP allows supervised networks to be developed. Supervised learning requires a sample training set that has been pre-classified into the set of wanted output categories. ARTMAP systems have been found to have a simplicity of design and robust performance in a variety of applications including some in the computer vision domain. Many ARTMAP variations have been proposed throughout its history. Fuzzy ARTMAPs combines fuzzy logic and adaptive resonance theory by exploiting a similarity of fuzzy subsethood and ART learning principles [Carpenter, 1992]. Ellipsoid ARTMAP was developed based on the same ideas founded in Fuzzy ARTMAPS [Anagnostopoulos, 2001]. Fuzzy ARTMAPs aggregate input data using hyper-rectangles while Ellipsoid ARTMAPs use hyper-ellipsoids. Semi-supervised Ellipsoid ARTMAPs (ssEAM) and semi-supervised Fuzzy ARTMAPs (ssFAM) are further variations of ARTMAP architecture that is suitable for classification tasks [Anagnostopoulos, 2002]. Semi-supervised learning allows the architectures to have zero error after training while preserving restrained learning times as explained by Le et al. [Le, 2005]. Systems using ssEAM has been proved to quickly and robustly discriminate between classes of input data as demonstrated by Xu et al. [Xu, 2004] for cancer classification.

# 3. BACKGROUND AND APPROACH

Automatic surveillance systems can be used to collect information from an environment in order to monitor traffic or detect abnormal or critical situations. Current surveillance systems often utilize visible or infrared technology to collect information from the observed environment. While thermal infrared imaging and visible imaging have both been explored extensively in security applications, near-infrared imaging has remained largely uncharted.

## 3.1. CURRENT SURVEILLANCE APPROACHES

The research that inspired "A Comparison of Near-Infrared and Visible Imaging for Surveillance Applications" is detailed below.

**3.1.1. Pedestrian Bridge Surveillance.** In this research completed by Watkins et al. [Watkins, 2009], a computer vision surveillance system was created for the purpose of automatically detecting people moving across an outdoor pedestrian bridge. This project produced a surveillance system that was capable of real-time automated detection and tracking in a variable environment.

In the developed system, visible images were processed from an unmoving camera focusing on the bridge. After noise filtering was completed, a 'region-of-interest' was cropped from the images and was arbitrarily segmented into smaller vertical sections. Fourteen features known for working well in person-detection applications were extracted from each segmented section. The calculated features were then used to train a standard backpropagation multi-layer perceptrons (MLP) neural network, resulting

in a classifier capable of effectively detecting pedestrian traffic. Because the system was implemented outdoors, the images were highly variable due to weather patterns and lighting conditions. Image sequences with a variety of people at different times in the year and day were collected to account for the complex environment.

The system implementation was found to have high pedestrian detection accuracy despite the variable environment and system-introduced noise. The selected features and neural network architecture led to low computational complexity making real-time person detection a possibility for this system. This work was preceded by similar object detecting and tracking algorithms [Stanley, 2006] for an outdoor pedestrian bridge developed for use in an image-processing curriculum [Stanley, 2004].

**3.1.2. Multi-class Cancer Classification.** Through this research, Xu et al. [Xu, 2004] propose using a semi-supervised Ellipsoid ARTMAP (ssEAM) architecture developed by Anagnostopoulos et al. [Anagnostopoulos, 2002] for cancer classification. This architecture defines categories of data through embedded hyper-ellipsoids in the feature space. The semi-supervised portion of the architecture allows multiple classes to be contained within a training category. Semi-supervision within the architecture can be customized by setting a tolerance parameter. This parameter defines how often multiple classes can be contained within one category. This type of design allows both on-line and off-line learning, has low computational complexity, and can handle large amounts of data with many dimensions efficiently.

The developed ssEAM architecture was applied to a cancer classification problem. The problem encompassed two datasets. Each dataset had a small number of very multi-dimensional features belonging to a variety of classes. The results indicated

that ssEAM was capable of effectively classifying the data sets. Results also indicated that the number of features introduced to the architecture could affect the classification performance. In the first dataset, the classification rate deteriorated when all features were used and when a minimal number of features were used. In the second dataset, reduction of the number of features deteriorated the classification rate. The research stresses the importance of optimizing the feature set and the tolerance parameter of the ssEAM architecture in order to create an efficient classifier. This work was preceded by much research in Adaptive Resonance Theory as outlined in Section 2.3.

## 3.2. SURVEILLANCE METHODS USED IN THIS RESEARCH

As shown in the previous sections, significant surveillance research has been completed in the infrared and visible light domains. Detection systems in both domains have a variety of advantages and disadvantages. In order to explore the potential of near-infrared detection systems, the following algorithm is proposed according to the computer vision problem scope proposed by Miller et al. [Miller, 2011]. This algorithm is fashioned to intelligently process an image to determine if persons, bicycles or motor vehicles are present within the environment.

**3.2.1. Access.** In order to obtain image sequence data, a modified visible light camera was used to collect short near-infrared wavelength videos of the environment. A visible light camera of the same type also collected short videos of the environment for comparative uses. A variety of indoor and outdoor environments featuring a variety of subject traffic were imaged by both cameras. In depth descriptions of the environments manipulated for this research can be found in Section 4.

**3.2.2. Transfer.** For the purpose of this research, image sequences were manually transferred to the processing computer. Automated information transfer can be explored in future research similar to methods proposed by Watkins et al. [Watkins, 2009], so low computational complexity is considered in the scope of this project.

**3.2.3. Convert.** A video segmentation program was used to parse the information collected from the modified camera at a specified frame rate into separate image sequences. Visible RGB images were then converted to gray-scale. Near-infrared images were saved in two locations. In one location the near-infrared images were converted to gray scale, in the second the red channel of the near-infrared images was conserved. These three converted types of image sequences (gray-scale visible, gray-scale near-infrared, and red channel near-infrared) were then fed to an image modification algorithm to prepare for intelligent classification. The image frame conversion process is fully explained in Section 5.1.

**3.2.4. Modify.** The modification algorithm proposed in this research was inspired by the work completed by Watkins et al. [Watkins, 2009] for the purpose of pedestrian bridge surveillance.

Pixel noise is often prevalent in computer vision systems. Noise can be attributed to data transmission errors, image recording errors, poor lighting conditions, a variant environment, and many other factors. In this research, the proposed modification algorithm used median filtering to reduce the effects of pixel noise. The median filtering operation is outlined in Section 5.2.

After noise filtering, image sequences were compared to a reference image. A reference image was defined as the first image in a sequence that contained no abnormal

or critical surveillance situations. An absolute pixel difference between each image in a sequence and its corresponding reference image were found. The difference image process is shown in Section 5.3.

The prepared difference image was then thresholded to remove insignificant pixel differences that occur in the sequence image (Section 5.4). The resultant thresholded image was converted to a binary image, representative of important pixel changes.

Object segmentation was performed on the binary image by first finding blobs based on similar neighboring pixels. Median filtering was then used to preserve blob edge boundaries, working similarly as the Gaussian filter used by Bankman et al. [Bankman, 2008]. A basic hole-fill operation was applied to the blobs within the images. The object segmentation portion of the algorithm, as explained in Section 5.5, fully prepares the images for feature extracting and intelligent processing during the analysis portion of the computer vision problem scope.

**3.2.5. Analysis.** The analysis portion of the proposed research includes feature extraction of the modified image blobs and intelligent processing of the three different types of images (gray-scale visible, gray-scale near-infrared, and red channel near-infrared). The computer vision system was tasked to determine if a processed blob was a person, a bicycle, a motor vehicle, or insignificant clutter.

Twenty-four geometric and photometric object features were calculated for each blob of significant size located by the modification algorithm. The twenty-four calculated features include: height, width, aspect ratio, area, perimeter, convex hull area, solidity, compactness, horizontal centroid offset, vertical centroid offset, Euler number, skewness, kurtosis, the second, third and fourth order moments, ellipse major axis length,

ellipse minor axis length, ellipse eccentricity, ellipse orientation and the first, second, third and fourth Hu moments. The calculation process and purpose of each of these features is explained in Section 5.6. Fourteen of these features were calculated by Watkins et al. [Watkins, 2009]. The remaining ten features were selected because they do not require an excessive amount of computational resources and are commonly used in many computer vision problems.

To determine the most essential features for describing blobs within the image sequences feature discrimination was necessary. Linear Discriminative Analysis (LDA), also used by Li et al. [Li, 2007], was used to evaluate the importance of each feature to classification. The resultant feature rankings, shown in Section 7.1, were used to prune the twenty-four features to an optimal grouping per class. The LDA process is further explained in Section 6.1.

The feature vectors provided give intelligent architectures the ability to discriminate among clutter, persons, bicycles and motor vehicles. A basic MLP backpropagation neural network (MLP) and a semi-supervised Ellipsoidal ARTMAP (ssEAM) were selected for use as intelligent classifiers. The MLP network was selected to mimic previous research by Watkins et al. [Watkins, 2009] and the ssEAM network was selected for its proven capabilities of fast multi-class discrimination by Xu et al. [Xu, 2004].

In both the MLP and ssEAM architectures, feature vectors and corresponding class labels were delivered to the architecture for training purposes. The two networks were trained first with the entire original feature vectors (MLP-all, ssEAM-all) and with the LDA reduced feature vector (MLP-reduced, ssEAM-reduced). Each of the four

network templates (MLP-all, ssEAM-all, MLP-reduced, ssEAM-reduced) were trained using the gray-scale visible sequences, the gray-scale near-infrared sequences, and the red channel near-infrared sequences. Input parameters such as learning rate and the tolerance parameter that control the architectures were manipulated to find best case outputs. Classification yields of each network as well as train and test time statistics allowed for the visible and near-infrared capabilities to be compared as well as the performance of the MLP and ssEAM architectures.

To further compare the three image types, data fusion principles were explored. Optimal configurations of each of the four architectures templates (MLP-all, ssEAM-all, MLP-reduced, ssEAM-reduced) were trained and tested on all three image types. For instance, the MLP-all network trained for optimal use on visible gray-scale images was tested with the visible gray-scale images, the near-infrared gray-scale images and the near-infrared red channel images. The results of these tests give a measure of the uniqueness of each of the three types of image data.

Through analysis of reduced feature sets, neural network architecture, and data fusion many comparisons were able to be made between near-infrared and visible image surveillance capabilities. The MLP neural network architecture is described in Section 7.2. The ssEAM architecture is described in Section 7.3. The data fusion completed is outlined in Section 7.4.

# 4. IMAGE SET DATA COLLECTION

In order to develop the "A Comparison of Near-Infrared and Visible Imaging for Surveillance Applications" algorithm, a variety of images were collected. In varying environments, different persons, bicycles, and vehicles were observed. Image sequences were taken at different times of the day as well as during varying weather conditions. The following sections describe the equipment used for image collection as well as the observed environments.

## 4.1 IMAGING TECHNOLOGY

Two Canon Powershot G6 cameras were used for this research. One camera was unmodified to collect images within the visible wavelength spectrum. The second camera was modified to collect images within the near-infrared spectrum. The cameras were mounted to a modified standard camera tripod. The cameras were mounted such that the foci of the two lenses were separated by eight inches. Figure 4.1 shows the dual camera setup used for image collection.

The near-infrared capable camera was modified such that only information above the 720 nm wavelength would be collected. This modification required insertion of a low-pass CCD filter to eliminate the visible wavelength spectrum. Additionally removal of the existing hot mirror filter was required to allow near infrared light to pass to the CCD. All camera modification was completed by Ehab Eassa of Rochester, NY.

The two cameras were used consecutively to collect movie clips of 640x480 pixels. These videos were stored as .avi files. Collected .avi files were then segmented

into image sequences using VirtualDub-1.9.11 developed by Avery Lee [Lee, 2012]. Images were segmented at a one image per second frame rate and were stored as .jpeg images of 640x480 pixels. VirtualDub is licensed under the GNU General Public License (GPL).


Figure 4.1. Dual Camera Setup.

## 4.2 IMAGED ENVIRONMENTS

Seven different environments were observed within the developed image set to provide a variety of complex environments for object detection. This section describes each of the environments and typical objects found within. All images found in this

section are the three channel (red, green and blue or RGB) images taken directly from the cameras.

**4.2.1. Brick Wall Scene.** One series of images was taken of subjects passing by a brick wall. The images contained very little environmental clutter movement as no foliage was present in the scene. Sharp shadows and cloud movement were the only observed clutter. Image sequences in the environment included pedestrian, pet and bicycle traffic. An example image of this environment is shown in Figure 4.2.

Figure 4.2. Brick Wall Scene. (a) Near-Infrared Image. (b) Visible Image.

**4.2.2. Campus Building Scene.** A second series of images were taken by the side of a brick campus building. The scene contained reflective windows as well as a variety of foliage causing high amounts of environmental clutter movement. Pedestrian, pet and bicycle traffic were observed within the image sequences. All traffic passed in

front of a combination of brick walls and foliage. An example of a typical Campus Building Scene image is shown in Figure 4.3.

Figure 4.3. Campus Building Scene. (a) Near-Infrared Image. (b) Visible Image.

**4.2.3. Campus Library Scene.** An additional series of images, shown in Figure 4.4, were collected by a campus library building. This scenery also included highly reflective windows and foliage causing much environmental movement. Traffic in the forms of pedestrians, pets and bicycles were imaged passing in front of tall foliage.

**4.2.4. Pedestrian Bridge Scene.** A fifth series of images were taken of subjects on a pedestrian bridge. This exact bridge was used by Stanley et al. [Stanley, 2004] for development of an image processing curriculum. It was also used by Stanley et al. [Stanley, 2006] and Watkins et al. [Watkins, 2009] for traffic monitoring and surveillance. The bridge was surrounded by large trees and a variety of other foliage as shown in Figure 4.5. Bridge rails partially cover traffic moving across the bridge. Sharp shadows,

cloud movement, background traffic and foliage movement resulted in a highly variable environment. Bicycle and pedestrian traffic were observed within this image set.

Figure 4.4. Campus Library Scene. (a) Near-Infrared Image. (b) Visible Image.

**4.2.5. Indoor Hallway Scene.** In Figure 4.6, an example image of the Indoor Hallway Scene is shown. Sequences from this scene contain pedestrian and object traffic. This scene features painted cinder block walls and a reflective tile floor. Lighting to the room is provided by ceiling fluorescent lights and outdoor windows that are unobservable within the scene. The outdoor windows provide some lighting variations, but the environment is typically stagnant.

Figure 4.5. Pedestrian Bridge Scene. (a) Near-Infrared Image. (b) Visible Image.

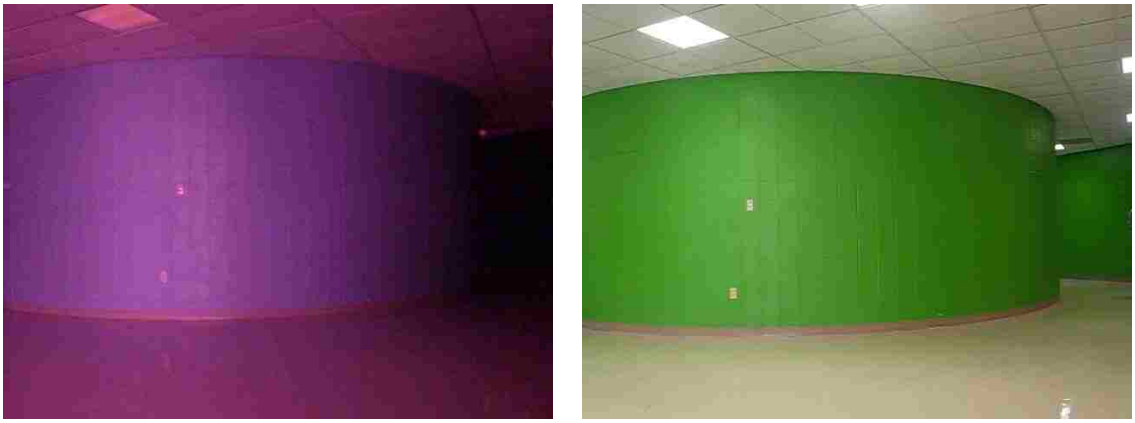

Figure 4.6. Indoor Hallway Scene. (a) Near-Infrared Image. (b) Visible Image.

**4.2.6. Urban Sidewalk Scene.** An urban sidewalk scene is observed in the sixth image sequence series. This sequence features a downtown street and sidewalk with very little foliage. Cloud movement and lighting variations were prominent in the image sequence series. A variety of pedestrian, bicycle and manned vehicle traffic were observed passing in front of buildings and background foliage. An example reference image of this environment is shown in Figure 4.7.

Figure 4.7. Urban Sidewalk Scene. (a) Near-Infrared Image. (b) Visible Image.

**4.2.7. Urban Street Scene.** The final environment observed features a street intersection in an urban area. The scene contains minimal foliage. Environmental clutter movement largely exists in the form of clouds. A large variety of vehicle traffic such as cars, trucks and motorcycles were imaged within these sequences. Vehicles were imaged passing in front of buildings, the street and other vehicles. Some environmental objects such as street signs and traffic lights partially block traffic traveling through the scene. An example image of the Urban Street Scene is shown in Figure 4.8.

Figure 4.8. Urban Street Scene. (a) Near-Infrared Image. (b) Visible Image.

# 5. IMAGE PROCESSING ALGORITHM DESCRIPTION

This section describes the algorithm used for "A Comparison of Near-Infrared and Visible Imaging for Surveillance Applications" as shown in Figure 5.1. Computational complexity is considered in this algorithm and was inspired by the results of the research by Watkins et al. [Watkins, 2009].

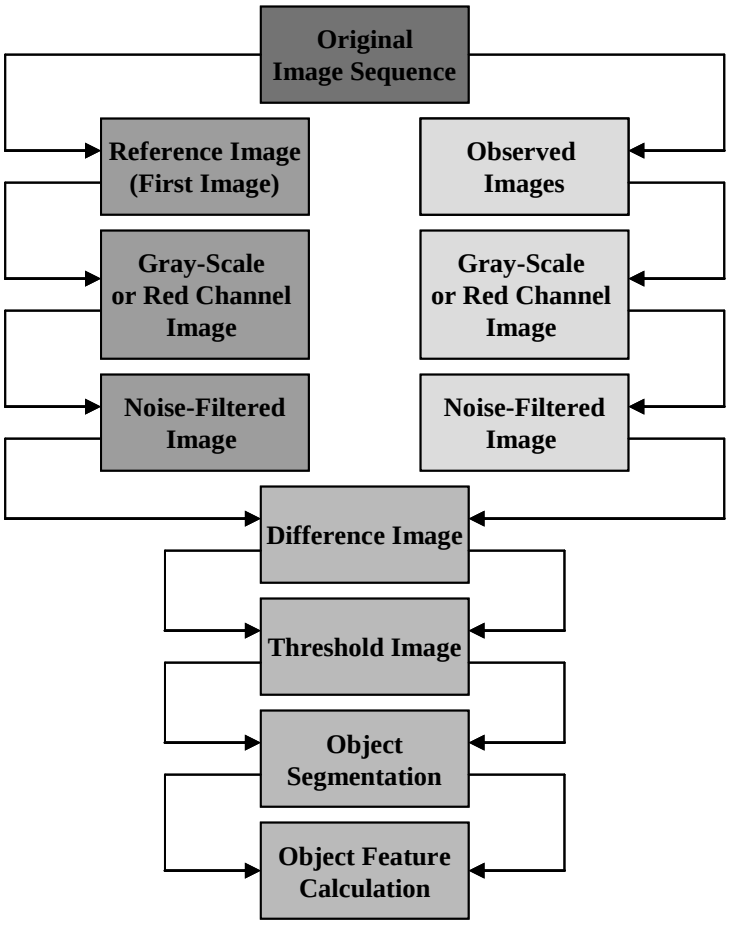


Figure 5.1. Algorithm Block Diagram.

Image sequences were acquired from seven different environments. Image frames within these sequences included pedestrian, bicycle and vehicle traffic as well as no traffic. The number of objects within the frame and directions of motion varied within the sequences. Image frames were processed as shown in the block diagram, Figure 5.1, to calculate feature vectors of objects within the frame that could later be used to determine what type of objects are present in the current frame. Sections 5.1 – 5.6 explain the processes behind the blocks from Figure 5.1. Section 5.7 holds a collection of images that visualize blocks from Figure 5.1. All algorithmic code can be found in Appendix B.

## 5.1. IMAGE FRAME CONVERSION

For visible image sequence processing, the sequences images are first converted to gray-scale in order to reduce the image complexity for processing purposes ('Gray-Scale or Red Channel Image' block in Figure 5.1). By converting to gray-scale, all pixel luminance information is preserved from the original infrared image while hue and saturation information is eradicated. Additionally the image becomes much simpler and therefore much faster to process since each pixel in color-scale images have to be represented by three values (red, green, and blue or RGB) while pixels in gray-scale images are only represented by a single value. The equation used to retain luminance information is shown in Equation 5.1. A typical gray-scale visible reference and non-reference image can be found in Figure A.44.

$$0.2989 \times (\text{Red}) + 0.5870 \times (\text{Green}) + 0.1140 \times (\text{Blue}) \qquad (5.1)$$

Two separate methods were used to process the near-infrared image sequences. The first method matches that of the visible image process by converting the near-infrared image to gray-scale. A typical gray-scale near-infrared reference and non-reference image can be found in Figure A.51. The second method only retains the red channel of the three channel RGB near-infrared image. Near-infrared light extends in wavelength from the red edge of the visible spectrum; therefore much information is stored in the red channel of the near-infrared image. An example of a typical red channel near-infrared reference and non-reference image can be found in Figure A.58.

## 5.2. FILTERING

All visible and near-infrared sequence images were median filtered over a five-by-five-pixel neighborhood in order to reduce noise ('Noise-Filtered' block in Figure 5.1). Median filtering is the process of replacing the value of a pixel with the median intensity value of the neighboring pixels. This filtering was selected in order to successfully reduce a variety of noise. Median filters are very successful at reducing salt and pepper noise; noise which results from errors in data transmission. These types of errors are common in many security systems where image data must be transferred from location to location for processing, which may be appropriate in future security applications . Additionally median filters can successfully reduce Gaussian noise, which is common in any imaging or data transmission system [Gonzales, 2008].

## 5.3. DIFFERENCE IMAGE

The next portion of the algorithm allows a comparison to occur between the reference image and the observed image (‘Difference Image’ block in Figure 5.1). Each pixel value of the median filtered reference image is subtracted from the pixel value of the median filtered non-reference image and the absolute value is then found. This subtraction method is illustrated by Equation 5.2.

$$\text{differenceImage}(i,j) = \text{absolute value}( \text{objectImage}(i,j) - \text{referenceImage}(i,j) ) \quad (5.2)$$

The resulting image represents the pixel differences between the reference and non-reference image. The difference image highlights environmental locations that have changed significantly in intensity over time. These environmental locations can be examined to locate target objects. An example of visible, gray-scale near-infrared, and red channel near-infrared difference images can be found in Figure A.46, Figure A.53 and Figure A.60.

## 5.4. THRESHOLDING

Thresholding of the difference image in this research is completed adaptively (‘Threshold Image’ block in Figure 5.1). In order to disregard insignificant changes from the reference image to the non-reference image, small pixel intensities are set to having no intensity. Then in order to highlight all significant changes, all other pixel intensities are transformed to the maximum pixel intensity.

To perform this thresholding, first the histogram of the difference image is plotted and analyzed. From the histogram, the lowest pixel intensity of the difference image was found as shown in Equation 5.3. Using the lowest pixel intensity, an upper limit is calculated to include the top ninety-five percent of pixels in terms of intensity. This upper limit is calculated through Equation 5.4. The lower limit is calculated by subtracting a pixel from the upper limit, which is shown in Equation 5.5. Using the upper limit, pixel intensities above and equal to that limit were set to the maximum intensity value (255). Using the lower limit, pixel intensities below and equal to that limit were set to the minimum intensity value (0). The equations to set these intensities are shown in Equation 5.6 and Equation 5.7. The thresholded images will highlight any significant changes in the environment including the introduction of target traffic. An example of visible, gray-scale near-infrared and red channel near-infrared thesholded images can be found in Figure A.47, Figure A.54 and Figure A.61.

$$i = i \quad \text{when} \quad H(i-1) = 0 \quad \text{and} \quad H(i) \neq 0 \tag{5.3}$$

$$\text{upperLimit} = (255 - i) * 0.05 \tag{5.4}$$

$$\text{lowerLimit} = \text{upperLimit} - 1 \tag{5.5}$$

$$\text{if} \quad \text{pixel} >= \text{upper} \quad \text{then} \quad \text{pixel} = 255 \tag{5.6}$$

$$\text{if} \quad \text{pixel} <= \text{lower} \quad \text{then} \quad \text{pixel} = 0 \tag{5.7}$$

## 5.5. OBJECT SEGMENTATION

Once the binary thresholded image is obtained, object segmentation must be completed ('Object Segmentation' block in Figure 5.1). The intent of object segmentation is to highlight a particular target type such as pedestrians to facilitate identification or classification. In particular, the binary comparison produces multiple "blobs" associated with aspects of the target (such as the silhouette of a face or the entire body). Once target blobs are grouped and highlighted, feature calculations can be performed on each located object.

In order to reduce small clutter within the image, a second median filtering operation with a five-by-five pixel neighborhood is performed on the thresholded image. This filtering operation will eliminate pixels that are not neighbored by twelve pixels that have the maximum intensity. Additionally, this filtering operation will help to fill holes in the object blobs. In existing object detection pre-processing algorithms, a flood fill of holes in the image is used to prepare the image for feature calculation [Friedrich, 2003]. In addition to median filtering, a basic hole fill algorithm is used to better define target objects moving through the environment. The hole fill algorithm fills modifies pixels of no intensity to have the maximum possible intensity when it is surrounded by pixels of the maximum intensity.

Once holes in the image have been filled, the different blobs within the image are found with a search of pixel connectivity to its eight neighbors. Different blobs are defined by the bounding box that contains them. Bounding boxes of blobs that overlap are merged together into a single bounding box. Additionally, bounding boxes of blobs that are within three pixels of each other are also merged. Once all close bounding boxes

are merged together, those boxes of insignificant sizes are eliminated from the target object group. At the completion of the object segmentation part of this algorithm, a list of target object bounding boxes are compiled. An example of visible, gray-scale near-infrared, and red channel near-infrared images containing target object bounding boxes can be found in Figure A.49, Figure A.56 and Figure A.63.

## 5.6. FEATURE CALCULATIONS

Discrimination of potential targets from non-target activity or clutter may be done by assessing or processing a number of features ('Object Feature Calculation' block in Figure 5.1). For an automated system, computer-based processing could alert users to potential targets of interest. Key considerations for a real-time automated system are the optimal set of features (in this case for near-infrared images) and the computational complexity associated with the processing. In this research, twenty-four feature vectors are computed for each target object that will later define the classification system for surveillance traffic. These twenty-four geometric and photometric object features were selected to allow further analysis to find the optimal group of features to pass to a classification system. The calculated features include: height, width, aspect ratio, area, perimeter, convex hull area, solidity, compactness, horizontal centroid offset, vertical centroid offset, Euler number, skewness, kurtosis, the second, third and fourth order moments, ellipse major axis length, ellipse minor axis length, ellipse eccentricity, ellipse orientation and the first, second, third and fourth Hu moments.

**5.6.1. Height (H).** Height is a scalar representing the number of pixels that define the height of the bounding boxes of objects.

**5.6.2. Width (W).** Width is a scalar representing the number of pixels that define the width of the bounding boxes of objects.

**5.6.3. Aspect Ratio (AR).** Aspect ratio is described by the proportionality between the object bounding boxes width and height. The equation defining aspect ratio can be found in Equation 5.8.

$$AR = H/W \tag{5.8}$$

**5.6.4. Area (A).** Area is a scalar that represents the total number of pixels that have an intensity value within the bounding boxes of objects.

**5.6.5. Perimeter (PER).** The perimeter of the object is a scalar value that represents the pixel distance around the boundary of the object. In the MATLAB function used for calculation, perimeter can not be calculated accurately for disconnected objects.

**5.6.6. Convex Hull Area (CA).** The convex hull area is a scalar that represents the total pixel area of the convex hull of the object. The convex hull is the smallest convex set that can contain the examined object. A convex set is characterized by a straight line segment joining any two points in the object while still remaining entirely in the object.

**5.6.7. Solidity (SLD).** Solidity is a proportional scalar value that takes into consideration the convex deficiency of the object. In particular, this scalar specifies the proportion of pixels in the object versus the pixels included in the convex hull. The equation defining solidity is found in Equation 5.9.

$$SLD = A/CA \tag{5.9}$$

**5.6.8. Compactness (CMP).** Compactness is a scalar proportion that indicates how dense the object is. Compactness is calculated by proportioning the area of the object and the perimeter of the object as shown in Equation 5.10.

$$CMP = A/P*P \tag{5.10}$$

**5.6.9. Horizontal Centroid Offset (COX).** The horizontal centroid offset is specified by a scalar value. This value represents the horizontal coordinate of the center of mass of the object.

**5.6.10. Vertical Centroid Offset (COY).** The vertical centroid offset is specified by a scalar value. This value represents the vertical coordinate of the center of mass of the object.

**5.6.11. Euler Number (EN).** The Euler number of the object is a scalar value that takes into considerations object count and object holes. The equation to calculate the Euler number can be found in Equation 5.11. Since part of the developed algorithm included filling in holes, this scalar will typically represent the number of blobs that make up the examined object.

$$EN = totalNumberObjects - totalNumberObjectHoles \tag{5.11}$$

**5.6.12. Skewness (SKW).** Skewness represents the way pixels are distributed within an object through the measurement of how asymmetric the data is around the sample mean. Data that is directed left of the mean has a negative skewness, while data that is directed right of the mean has a positive skewness. The equation defining skewness is shown in Equation 15.2. The mean of the data is defined as μ, the standard deviation is defined as σ and E(t) is defined as the expected value of the value t.

$$S = ( E (x-\mu)^{3} ) / ( \sigma^{3} ) \quad (5.12)$$

**5.6.13. Kurtosis (KUR).** Kurtosis also represents the way pixels are distributed within an object through the measurement of how outlier-prone a distribution is. Distributions that are more outlier-proned than the normal distribution have a kurtosis scalar value of greater than three. The normal distribution kurtosis value has a value of three. Distributions that are less outlier-proned have a kurtosis value less than three. The equation defining kurtosis is shown in Equation 5.13. The mean of the data is defined as μ, the standard deviation is defined as σ and E(t) is defined as the expected value of the value t.

$$K = ( E (x-\mu)^{4} ) / ( \sigma^{4} ) \quad (5.13)$$

**5.6.14. Order Moments (M2, M3, M4).** The second, third and fourth order moments return a quantitative measure of the shape of a set of points based on the order of the moment. The equation defining how to calculate the central moments is shown in

Equation 5.14. The order is defined as the value k and E(t) is defined as the expected value of the value t.

$$MK = E\,(x-\mu)^{k} \tag{5.14}$$

**5.6.15. Ellipse Major Axis Length (MJL).** The major axis length is calculated through examination of the ellipse that has the same second central moments as the object. The major axis length is the scalar value represented by the pixel length of the major axis of the calculated ellipse.

**5.6.16. Ellipse Minor Axis Length (MNL).** The minor axis length is calculated through examination of the ellipse that has the same second central moments as the object. The minor axis length is the scalar value represented by the pixel length of the minor axis of the calculated ellipse.

**5.6.17. Ellipse Eccentricity (ECC).** Eccentricity is calculated through examination of the ellipse that has the same second central moments as the object. Eccentricity is a scalar value that represents the eccentricity of the calculated ellipse. Eccentricity is described by the proportionality of the distance from the center of ellipse to the focus of the ellipse and the distance from the center of the ellipse to the vertex. This equation is defined by Equation 5.15.

$$ECC = ellipseFoci / MJL \tag{5.15}$$

**5.6.18. Ellipse Orientation (OR).** Orientation is calculated through examination of the ellipse that has the same second central moments as the object. Orientation is a scalar value that represents the angle between the x-axis and major axis of the calculated ellipse.

**5.6.19. Hu Moments (HU1, HU2, HU3, HU4).** The first, second, third and fourth order Hu invariant moments return a quantitative measure of the shape of a set of points based on the order of the moment that are invariant under different adaptations. The adaptations that orders of Hu moments are invariant to are rotation, translation, and scaling.

## 5.7. ALGORITHM EXAMPLE CASE

The images in this section demonstrate how the proposed image processing algorithm works. Figure 5.2 shows a reference image and an observed image taken directly from the near-infrared enabled camera. Figure 5.3 shows the image frame conversion step of the algorithm. In the image pictured, gray-scale conversion was performed. Figure 5.4 displays the noise-filtered reference and observed images. Figure 5.5 represents the difference image calculated between the observed and reference image. The histogram of the absolute difference image is shown in Figure 5.6. Figure 5.7 shows the threshold image obtained from histogram analysis. The methods used for object segmentation output the image as shown in Figure 5.8. The final important objects selected for feature vector calculation are bounded by boxes in Figure 5.9. In this example, two important objects were found, one representing a person object and one

representing a clutter object. Other image processing algorithm example cases are explored in Appendix A.

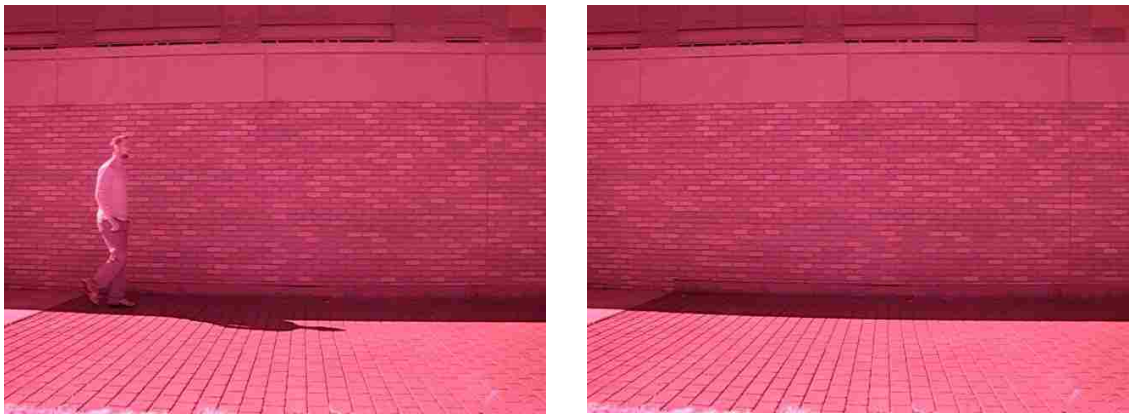

Figure 5.2. Original Near-Infrared Image. (a) Observed Image. (b) Reference Image.

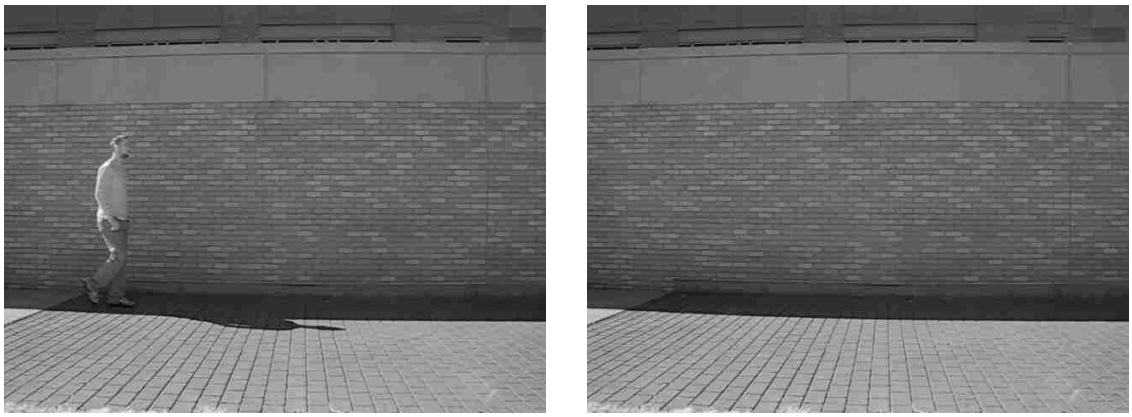

Figure 5.3. Gray-Scale Image. (a) Observed Image. (b) Reference Image.

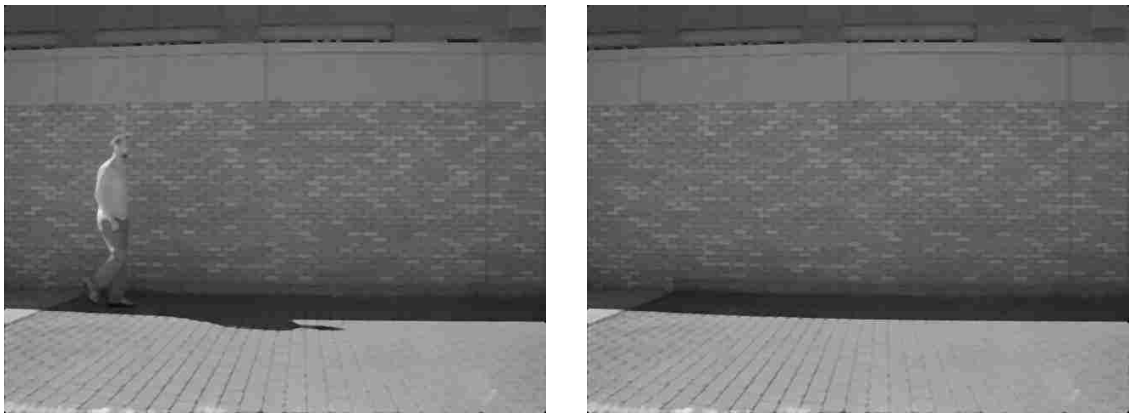

Figure 5.4. Noise-Filtered Image. (a) Observed Image. (b) Reference Image.

Figure 5.5. Absolute Difference Image.

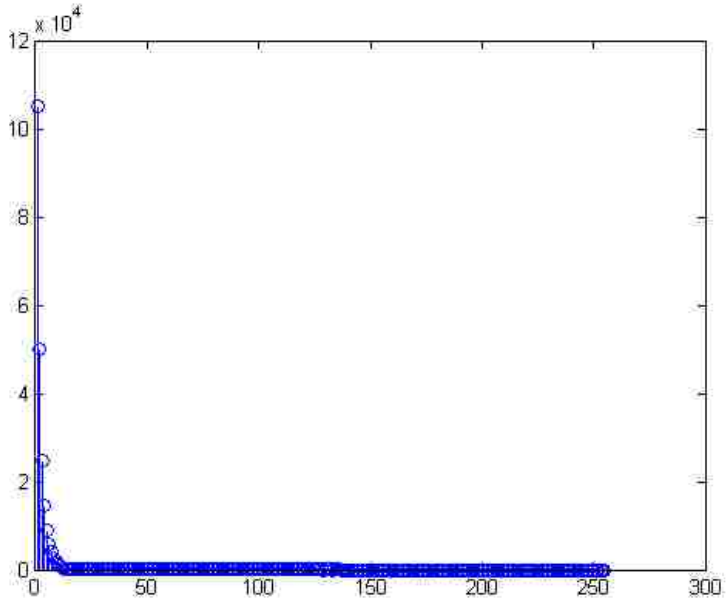


Figure 5.6. Histogram of Absolute Difference Image, Pixel Intensity vs. Pixel Count.

Figure 5.7. Threshold Image.

Figure 5.8. Object Segmentation Image.

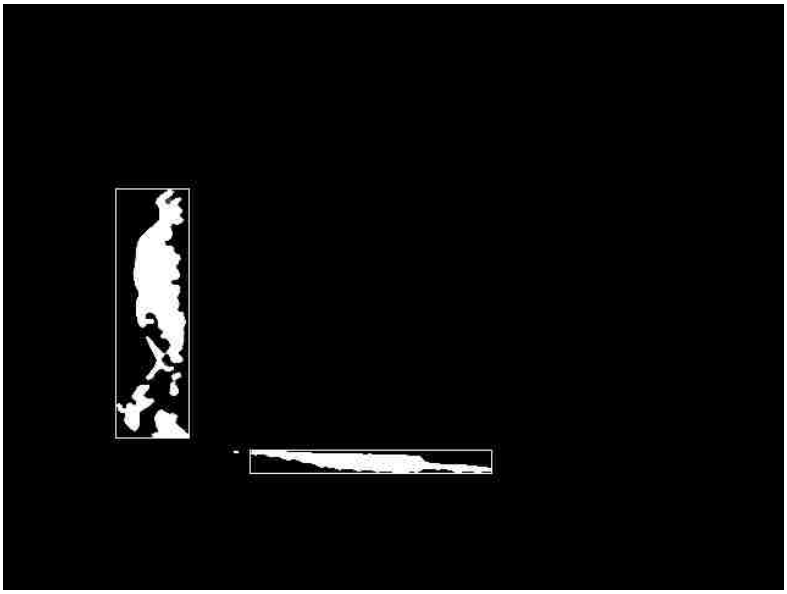

Figure 5.9. Algorithm Targeted Objects.

# 6. INTELLIGENT PROCESSING EXPERIMENTS

This section gives a detailed description of the experiments completed for "A Comparison of Near-Infrared and Visible Imaging for Surveillance Applications." Image sequences were collected from a variety of environments, which included pedestrian, bicycle and vehicle traffic. Individual image frames were processed to locate important 'blobs' or sub-images within the larger image frame. These 'blobs' can be classified as person, bicycle, vehicle or clutter objects. Twenty-four features were calculated for each 'blob' to be used for intelligent classification. The calculated features were then used to complete the following experimentations. Results and discussion of these experiments can be found in Section 7.

## 6.1. LINEAR DISCRIMINANT ANALYSIS

In order to correctly categorize 'blobs' into their respective classes of person, bicycle, vehicle or clutter, features are needed that are similar across an image set for a specific class, but are dissimilar to the same features of other classes. To determine which of the twenty-four collected features have the capability to discriminate among the four blob classes, Linear Discriminate Analysis (LDA) was performed. LDA is often used to reduce the dimensionality of a computer vision problem and in this research it is used to reduce the dimensionality of the applicable feature space. LDA reduces the dimensionality of the feature space by finding linear relationships of features across an image set.

In order to evaluate each feature's capability to classify, Single Feature Analysis, Forward Selection and Backward Elimination were used. Single Feature Analysis is defined as the process of training a classifier using an individual feature. Forward Selection is defined as the process of training a classifier by slowly adding best performing features. First, classifiers are trained with individual features. The individual feature allowing the best classification accuracy is preserved in the classifier and the process is repeated by adding the unused features. Backward Elimination is defined as the process of training a classifier by starting with the full set of features and slowly eliminating poor performing features. First, classifiers are trained by eliminating one feature from the feature set. The best performing classifier is preserved and recursively all remaining features are individually removed. Forward Selection and Backward Elimination give information as to how features and feature combinations affect classification capabilities.

In this research, fifty trials were completed on each feature combination being tested. Each trail randomly removed 10% of each class type from the train set and preserved them in a test set. Single Feature Analysis, Forward Selection and Backward Elimination were completed on the set. Results from these LDA techniques can be found in Section 7.1. A reduced feature set of six features was selected based on reported classifier accuracies. All LDA experimentation was performed by the computer vision researcher, Michael Ryan Bales. Bales has completed similar research in the visible light domain and was a valuable asset in analyzing the feature set.

## 6.2. MLP NEURAL NETWORK CLASSIFICATION

Both the original feature vectors described in Section 5.6 and the reduced feature vectors described in Section 6.1 were used to train and test a neural network classifier. The MLP [Hagan, 1996] architecture used for classification with the original feature vectors consisted of twenty-four input nodes, twenty-four nodes in a single hidden layer, and four output nodes corresponding to person, bicycle, vehicle and clutter (24x24x4). The architecture selected for classification with the reduced feature vectors consisted of six input nodes, six nodes in a single hidden layer and four output nodes (6x6x4).

In both MLP architectures the following settings were applied. Log sigmoid functions were selected for the activation function in both the hidden and output layers just as in previous work [Watkins, 2009]. Network weights were randomly initialized and were updated using a standard back-propagation algorithm. Input feature vectors were normalized by mapping to a range of [-1,1].

Feature vectors describing persons, bicycles, vehicles and clutter were read and used for training and testing of the aforementioned network architectures. These architectures were simulated using built-in MATLAB functionality. A total of 1489, 1448 and 1458 vectors were collected from the visible, gray-scale near-infrared and red channel near-infrared images respectively. A total of 40 videos were processed into image sequences. 36 videos contained persons, 18 contained bicycles, 10 contained vehicles and 40 contained clutter objects. Vectors were arranged into sets in the sequential order they were processed by the algorithm described in Section 5.

To test the accuracy of the developed architecture, ten-fold cross validation methodology was used to generate train and test sets [Kohavi, 1995]. This methodology

was implemented by taking the original forty videos and breaking them equally into ten groups. To ensure equal numbers of video types, first the vehicle videos were sorted, then the bicycle videos and finally the pedestrian videos. Ten experiments were created where one of the ten groups was implemented as the test set and the remaining nine groups encompassed the train set. All ten experiments were run on each of the two MLP architectures to better determine classifier accuracy.

The following learning rates were used to train the architectures: 0.04, 0.08…0.402. The following momentum values were used to train the architectures: 0.60, 0.64…0.96. The built-in MATLAB functionality broke the training set into a training and cross-validation group. Termination criteria for training included: 1) reaching a maximum of 30 epochs, 2) reaching the target accuracy goal of error rate less than 0.1%, 3) classification rate of cross validation set has not improved in 5 consecutive epochs, or 4) the classification rate is less than what had been found in a previous epoch.

Architectures were delivered feature vectors as well as class labels of the train sets for training purposes. Once training termination criteria had been reached, the trained network was tested on the train set and then the test set. Each of the two MLP architecture used had four output nodes. In order to determine the output classification, a winner-takes-all method was implemented. The highest ranking output node was determined to be the classification of the given input vector. Classification results were collected for both the training and testing sets. Overall classification results were reported in terms of precision and recall as introduced by Bar-Ilan et al. [Bar-Ilan, 1998]. The methods used to calculate precision and recall are shown in Equations 6.1 and 6.2. Experimental results for the MLP architectures can be found in Section 7.2.

precision = (true positive classifications) / (total classifications made of class) (6.1)

recall = (true positive classifications) / (count of objects in that class) (6.2)

## 6.3. ssEAM ARCHITECTURE CLASSIFICATION

Two semi-supervised Ellipsoid ARTMAP (ssEAM) architectures were trained and tested similarly to the MLP architectures defined in Section 6.4. As with the MLP architectures, ssEAM was trained first using the original feature set obtained in Section 5.6 and second with the reduced feature set obtained in Section 6.1. ssEAM used the exact same feature vector sets and ten-fold cross validation groups used in the MLP experimentations.

ssEAM performs differently from the MLP architecture in that it creates categories within a search space by creating hyper-ellipsoids. Each hyper-ellipsoid represents a category that could encompass a variety of feature vectors including those belonging to different classes. Instead of updating weights that connect nodes like MLP, ssEAM encodes learned information about the feature vectors into the locality of its hyper-ellipsoids.

ssEAM is controlled by a larger number of parameters than the MLP architecture. The semi-supervised portion of ssEAM is controlled by a category prediction error tolerance parameter (tolerance). This parameter can take values in the range of [0,1] and determines how often multiple classes can be contained in one category (hyper-ellipsoid). If tolerance is set to 0 only one class can be contained in a category (supervised learning) and if tolerance is set to 1 any number of any type of classes can be contained in a category (unsupervised learning). A baseline vigilance parameter (vigilance) lying

within the range of [0,1] controls the size of the created hyper-ellipsoids. Small vigilance values produce large categories or hyper-ellipsoids and large values produce smaller categories. When vigilance is set to 1, the EAM geometric representation is reduced to point categories with one point representing each feature vector. The shapes of the categories are controlled by mu, which defines eccentricity of the hyper-ellipsoids. Alpha and omega parameters are used to define activation values within the ssEAM.

Vectors were standardized by mapping them into the range of [0,1]. A maximum of five epochs was allowed for this architecture. Using fast learning enabled this small number of epochs. The following mu values were used: 0.2, 0.4…1. The following tolerance values were used: 0.1, 0.3…0.9. The following vigilance values were used: 0.1,0.3..0.9 and 0.92, 0.94, 0.95, 0.96, 0.98. An alpha value of 0.001 and an omega value of infinity were used.

Architectures were delivered feature vectors as well as class labels of the train sets for training purposes. ssEAM architecture accuracy is dependant on the order that feature vectors are read, so fifty different train feature vector orders were used to determine architecture potential. Once training was terminated, the trained ssEAM is first tested on the train set and then on the test set. The output of the ssEAM architecture is a label. Output label types match input label types. Classification results were collected for both the training and testing sets. Overall classification results were reported in terms of precision and recall as introduced by Bar-Ilan et al. [Bar-Ilan, 1998]. Experimental results for the ssEAM architectures can be found in Section 7.3.

## 6.4. DATA FUSION

While ssEAM and MLP classification results could give a good indication of the surveillance capabilities of near-infrared and visible systems, it is necessary to further explore the uniqueness of each system. In order to compare whether a visible system is significantly different from a near-infrared one, data fusion principles were explored.

In order to determine if a neural network or ssEAM architecture in one image domain was unique from that of another, trained architectures were tested on all three types of image sets. Optimal configurations found in Sections 6.2 and 6.3 for both the original feature vectors calculated in Section 5.6 and the reduced feature vectors calculated in Section 6.1 were tested on the entire feature vector sets not used for training. For instance, architectures trained with the visible light feature vectors were given gray-scale near-infrared feature vectors and then red channel near-infrared feature vectors as test sets. Classification results were once again reported in terms of precision and recall as introduced by Bar-Ilan et al. [Bar-Ilan, 1998]. Experimental results for these data fusion principles can be found in Section 7.4.

# 7. EXPERIMENTAL RESULTS

Section 6 described the different experimental approaches used to evaluate near-infrared and visible image filtering for surveillance applications. These approaches used feature vectors collected from an image set as described in Section 5. In this section, experimental results are illustrated and explored.

## 7.1. LINEAR DISCRIMINANT ANALYSIS RESULTS

Three methods of Linear Discriminative Analysis (LDA) were completed on the collected feature vectors. The three methods included were Single Feature Analysis, Forward Selection and Backward Elimination as described in Section 6.1. Table 7.1 illustrates the twenty-four features and corresponding abbreviations calculated for each 'blob' found in the image sets from the algorithm described in Section 5. The results for Single Feature Analysis can be found in Table 7.2. Results for Forward Selection and Backward Elimination can be found in Tables 7.3-7.5 and 7.6-7.8, respectively.

In Table 7.2, the 25% highest accuracy single features are highlighted. Five of the six highlighted features are similar in each of the three image domains: Width, Aspect Ratio, Horizontal Centroid Offset, the $2^{nd}$ Order Moment and the $3^{rd}$ Order Moment. Both the visible and near-infrared gray-scale image sets have a top performing $2^{nd}$ Order Moment feature while red channel near-infrared image sets have a top performing solidity feature. The ranges in accuracies indicate that feature importance varies between the image types. Standard deviations are also calculated and presented for each

feature in Table 7.2. High standard deviation rankings indicate which features were significantly different in importance across the image types.

Table 7.1. Table of Features and Abbreviations.

| Symbol | Feature | Symbol | Feature |
|---|---|---|---|
| H | Box Height | KUR | Kurtosis |
| W | Box Width | M2 | 2nd Order Moment |
| AR | Aspect Ratio (H/W) | M3 | 3rd Order Moment |
| A | Object Area | M4 | 4th Order Moment |
| PER | Object Perimeter | MJL | Ellipse Major Axis Length |
| CA | Convex Hull Area | MNL | Ellipse Minor Axis Length |
| SLD | Solidity (A/CA) | ECC | Ellipse Eccentricity |
| CMP | Compactness (A/P*P) | OR | Ellipse Orientation |
| COX | Horiz. Centroid Offset | HU1 | 1st Hu Moment |
| COY | Vert. Centroid Offset | HU2 | 2nd Hu Moment |
| EN | Euler Number | HU3 | 3rd Hu Moment |
| SKW | Skewness | HU4 | 4th Hu Moment |

Table 7.2. Results Obtained From Single Feature Analysis Using 50 Randomly Generated Data Sets.

| Feature | Visible | NIR – Gray | NIR – Red | Standard Deviation |
|---|---|---|---|---|
| H | 47.73% | 36.54% | 34.80% | 0.070 |
| W | **56.90%** | **60.64%** | **59.64%** | 0.019 |
| AR | **75.95%** | **80.46%** | **79.07%** | 0.023 |
| A | 39.53% | 55.67% | 52.03% | 0.085 |
| PER | 31.66% | 44.58% | 30.03% | 0.080 |
| CA | 26.81% | 30.29% | 33.08% | 0.031 |
| SLD | 50.12% | 57.11% | **54.40%** | 0.035 |
| CMP | 18.74% | 16.24% | 26.61% | 0.054 |
| COX | **56.24%** | **61.39%** | **58.76%** | 0.026 |
| COY | 45.49% | 35.82% | 36.66% | 0.054 |

Table 7.2. Results Obtained From Single Feature Analysis Using 50 Randomly Generated Data Sets. (cont.)

| Feature | Visible | NIR – Gray | NIR – Red | Standard Deviation |
|---|---|---|---|---|
| EN | 18.16% | 23.01% | 25.27% | 0.036 |
| SKW | 25.51% | 33.29% | 33.17% | 0.045 |
| KUR | 37.16% | 28.55% | 25.65% | 0.060 |
| M2 | **61.99%** | **59.51%** | **54.89%** | 0.036 |
| M3 | **56.61%** | **58.58%** | 53.52% | 0.026 |
| M4 | **58.15%** | **59.22%** | **53.96%** | 0.028 |
| MJL | 33.66% | 35.29% | 36.12% | 0.012 |
| MNL | 41.82% | 49.72% | 45.31% | 0.040 |
| ECC | 45.88% | 38.57% | 42.53% | 0.037 |
| OR | 45.21% | 30.29% | 43.95% | 0.083 |
| HU1 | 41.48% | 50.72% | 45.30% | 0.046 |
| HU2 | 27.33% | 47.05% | 35.01% | 0.099 |
| HU3 | 21.97% | 26.12% | 25.84% | 0.023 |
| HU4 | 19.26% | 24.39% | 20.70% | 0.026 |

Table 7.3. Results Obtained From Forward Search Analysis On Visible Images Using 50 Randomly Generated Data Sets.

| Ranked Feature | Accuracy After Addition | Change In Accuracy |
|---|---|---|
| AR | 75.95% | 75.95% |
| H | 84.30% | 8.35% |
| SKW | 85.54% | 1.24% |
| OR | 85.36% | -0.17% |
| COY | 85.09% | -0.27% |
| CA | 86.34% | 1.25% |
| HU4 | 86.42% | 0.07% |
| HU2 | 86.25% | -0.17% |
| HU3 | 85.83% | -0.42% |
| PER | 85.71% | -0.12% |
| W | 85.95% | 0.25% |
| ECC | 86.37% | 0.41% |
| A | 86.96% | 0.59% |

Table 7.3. Results Obtained From Forward Search Analysis On Visible Images Using 50 Randomly Generated Data Sets. (cont.)

| Ranked Feature | Accuracy After Addition | Change In Accuracy |
|---|---|---|
| MNL | 87.20% | 0.24% |
| EN | 88.47% | 1.26% |
| M2 | 89.99% | 1.52% |
| COX | 91.86% | 1.87% |
| KUR | 91.83% | -0.03% |
| HU1 | 91.90% | 0.07% |
| SLD | 92.62% | 0.71% |
| M3 | 92.69% | 0.08% |
| M4 | 92.87% | 0.17% |
| CMP | 92.62% | -0.25% |
| MJL | 92.00% | -0.62% |

Table 7.4. Results Obtained From Forward Search Analysis On Gray-Scale Near-Infrared Images Using 50 Randomly Generated Data Sets.

| Ranked Feature | Accuracy After Addition | Change In Accuracy |
|---|---|---|
| AR | 80.46% | 80.46% |
| H | 80.87% | 0.41% |
| SLD | 85.71% | 4.84% |
| M4 | 89.28% | 3.56% |
| PER | 89.25% | -0.02% |
| A | 89.17% | -0.08% |
| ECC | 89.17% | 0.00% |
| COX | 90.32% | 1.15% |
| MNL | 90.93% | 0.61% |
| COY | 91.47% | 0.54% |
| MJL | 91.91% | 0.44% |
| W | 92.10% | 0.19% |
| HU2 | 92.40% | 0.30% |
| CMP | 92.34% | -0.06% |
| M3 | 92.29% | -0.05% |
| HU4 | 92.22% | -0.07% |
| EN | 92.12% | -0.10% |

Table 7.4. Results Obtained From Forward Search Analysis On Gray-Scale Near-Infrared Images Using 50 Randomly Generated Data Sets. (cont.)

| Ranked Feature | Accuracy After Addition | Change In Accuracy |
|---|---|---|
| CA | 92.10% | -0.02% |
| KUR | 91.97% | -0.13% |
| SKW | 91.83% | -0.15% |
| HU1 | 91.72% | -0.11% |
| OR | 91.72% | 0.00% |
| HU3 | 91.43% | -0.29% |
| M2 | 90.66% | -0.77% |

Table 7.5. Results Obtained From Forward Search Analysis On Red Channel Near-Infrared Images Using 50 Randomly Generated Data Sets.

| Ranked Feature | Accuracy After Addition | Change In Accuracy |
|---|---|---|
| AR | 79.07% | 79.07% |
| H | 80.25% | 1.18% |
| HU1 | 80.77% | 0.51% |
| ECC | 83.59% | 2.82% |
| MJL | 86.00% | 2.42% |
| SLD | 86.72% | 0.72% |
| W | 87.21% | 0.49% |
| COX | 87.62% | 0.40% |
| MNL | 88.02% | 0.41% |
| COY | 88.34% | 0.32% |
| PER | 88.47% | 0.13% |
| A | 88.79% | 0.32% |
| OR | 89.13% | 0.33% |
| CMP | 89.13% | 0.00% |
| HU3 | 89.10% | -0.02% |
| HU4 | 89.28% | 0.18% |
| KUR | 89.44% | 0.16% |
| M3 | 89.60% | 0.15% |
| SKW | 89.51% | -0.09% |
| M4 | 89.48% | -0.03% |

Table 7.5. Results Obtained From Forward Search Analysis On Red Channel Near-Infrared Images Using 50 Randomly Generated Data Sets. (cont.)

| Ranked Feature | Accuracy After Addition | Change In Accuracy |
|---|---|---|
| M2 | 89.50% | 0.02% |
| HU2 | 89.25% | -0.25% |
| CA | 88.89% | -0.36% |
| EN | 88.80% | -0.09% |

As described by Tables 7.3 – 7.5, Aspect Ratio and Height were both selected first to be included in all three image types with Forward Search Analysis. 85% classification accuracy was achieved with the addition of three ranked features for both Visible and the Gray-Scale Near-Infrared, but the third ranked feature differs from Skewness and Solidity between the two. 85% classification accuracy was not achieved until the addition of five ranked features in the Red Channel Near-Infrared case. The highest accuracy achieved for the three cases of visible, gray-scale near-infrared and red channel near-infrared images were 92.87%, 92.40% and 89.60% respectively.

Backward Elimination Analysis as shown in Tables 7.6 – 7.8 also indicated that Aspect Ratio was selected to be most important in all three image types. Height still played an important role in both the Visible and the Gray-Scale Near-Infrared as did Solidity and the Fourth Order Moment, but not in the Red Channel Near-Infrared. 85% classification accuracy was achieved in visible, gray-scale infrared and red channel infrared with six, four and six features respectively. The highest accuracy achieved for the three cases of visible, gray-scale near-infrared and red channel near-infrared images were 93.38%, 92.35% and 90.03% respectively. These high accuracies were reached by the removal of a third, seventh and eighth feature resulting in set sizes of twenty-one,

seventeen and sixteen features. Alternatively in Forward Search the high accuracies were reached by the addition of a twenty-second, thirteenth and eighteenth feature.

The diversity in rankings as well as classification accuracies indicate that calculated features differ across the image types. These findings suggest that classification capabilities for the three image sets could differ. From these findings the following criteria for a reduced feature set is proposed. 25% of the original feature set results in at least an 85% accuracy for a linear classifier as shown from Tables 7.3 – 7.8. A reduced set for visible, gray-scale near-infrared and red channel near infrared is proposed based on the higher scoring classifier of six features from the Forward Search and Backward Elimination Analysis. The reduced sets selected can be found in Table 7.9.

Table 7.6. Results Obtained From Backward Elimination Analysis On Visible Images Using 50 Randomly Generated Data Sets.

| **Ranked Feature** | **Accuracy After Elimination** | **Change In Accuracy** |
|---|---|---|
| MJL | 92.62% | 92.62% |
| M2 | 92.87% | 0.25% |
| CMP | 93.38% | 0.51% |
| HU3 | 93.38% | 0.00% |
| OR | 93.36% | -0.02% |
| KUR | 93.33% | -0.02% |
| HU2 | 93.26% | -0.07% |
| CA | 92.94% | -0.32% |
| SKW | 93.29% | 0.35% |
| COY | 93.07% | -0.22% |
| HU4 | 92.74% | -0.32% |
| HU1 | 92.47% | -0.28% |

Table 7.6. Results Obtained From Backward Elimination Analysis On Visible Images Using 50 Randomly Generated Data Sets. (cont.)

| Ranked Feature | Accuracy After Elimination | Change In Accuracy |
|---|---|---|
| A | 92.20% | -0.27% |
| ECC | 91.51% | -0.69% |
| MNL | 91.14% | -0.37% |
| PER | 90.45% | -0.68% |
| EN | 89.59% | -0.86% |
| COX | 88.39% | -1.20% |
| W | 87.23% | -1.16% |
| M3 | 84.82% | -2.41% |
| M4 | 81.27% | -3.55% |
| SLD | 84.30% | 3.03% |
| H | 75.95% | -8.35% |
| AR | 75.95% | 0.00% |

Table 7.7. Results Obtained From Backward Elimination Analysis On Gray-Scale Near-Infrared Images Using 50 Randomly Generated Data Sets.

| Ranked Feature | Accuracy After Elimination | Change In Accuracy |
|---|---|---|
| M2 | 91.43% | 91.43% |
| HU4 | 91.81% | 0.38% |
| PER | 92.10% | 0.29% |
| CMP | 92.14% | 0.04% |
| OR | 92.16% | 0.02% |
| HU1 | 92.29% | 0.13% |
| HU3 | 92.35% | 0.06% |
| MJL | 92.14% | -0.21% |
| M3 | 92.14% | 0.00% |
| CA | 92.12% | -0.02% |
| SKW | 92.12% | 0.00% |
| KUR | 92.33% | 0.21% |
| HU2 | 92.17% | -0.17% |
| EN | 91.65% | -0.52% |
| W | 91.37% | -0.28% |

Table 7.7. Results Obtained From Backward Elimination Analysis On Gray-Scale Near-Infrared Images Using 50 Randomly Generated Data Sets. (cont.)

| Ranked Feature | Accuracy After Elimination | Change In Accuracy |
|---|---|---|
| COY | 90.83% | -0.54% |
| MNL | 90.24% | -0.59% |
| A | 90.07% | -0.17% |
| COX | 89.00% | -1.06% |
| ECC | 89.28% | 0.27% |
| M4 | 85.71% | -3.56% |
| SLD | 80.87% | -4.84% |
| H | 80.46% | -0.41% |
| AR | 80.46% | 0.00% |

Table 7.8. Results Obtained From Backward Elimination Analysis On Red Channel Near-Infrared Images Using 50 Randomly Generated Data Sets.

| Ranked Feature | Accuracy After Elimination | Change In Accuracy |
|---|---|---|
| SKW | 89.48% | 89.48% |
| M3 | 89.50% | 0.02% |
| OR | 89.59% | 0.09% |
| KUR | 89.63% | 0.04% |
| HU1 | 89.61% | -0.02% |
| CMP | 89.65% | 0.04% |
| SLD | 89.57% | -0.08% |
| MJL | 90.03% | 0.45% |
| M4 | 89.78% | -0.25% |
| PER | 89.65% | -0.14% |
| HU3 | 89.65% | 0.00% |
| EN | 89.38% | -0.27% |
| HU2 | 89.33% | -0.05% |
| CA | 89.22% | -0.11% |
| W | 88.75% | -0.47% |
| COY | 87.39% | -1.37% |
| H | 86.50% | -0.88% |
| COX | 85.39% | -1.11% |

Table 7.8. Results Obtained From Backward Elimination Analysis On Red Channel Near-Infrared Images Using 50 Randomly Generated Data Sets. (cont.)

| Ranked Feature | Accuracy After Elimination | Change In Accuracy |
|---|---|---|
| M2 | 85.12% | -0.27% |
| HU4 | 84.51% | -0.61% |
| A | 78.87% | -5.65% |
| ECC | 78.28% | -0.59% |
| MNL | 79.07% | 0.79% |
| AR | 79.07% | 0.00% |
| M2 | 85.12% | -0.27% |
| HU4 | 84.51% | -0.61% |
| A | 78.87% | -5.65% |

Table 7.9. Reduced Feature Vectors Selected.

| Image Type | Feature Set | Analysis Source |
|---|---|---|
| Visible | AR, H, SLD, M4, M3, W | Backward Elimination |
| NIR-Gray | AR, H, SLD, M4, PER, A | Forward Search |
| NIR-Red | AR, H, HU1,ECC, MJL, SLD | Forward Search |

## 7.2. MLP NEURAL NETWORK RESULTS

Two MLP neural network architectures were explored through this research. The first architecture of size 24x24x4 used the original feature vectors calculated in Section 5.6. The second architecture of size 6x6x4 used the reduced feature sets indicated by Table 7.9. Ten-fold cross validation was used to calculate the classification accuracy. Classification results were reported in terms of precision and recall as introduced by Bar-Ilan et al. [Bar-Ilan, 1998]. Architectures were ranked by averaging precision and recall over the ten experiments. Precision and recall scores were then combined to find the best

scoring architectures. This section lists the results collected using the trained neural network architectures.

Tables 7.10, 7.12 and 7.14 show the best architecture accuracy found for visible, gray-scale near-infrared and red channel near-infrared image sets respectively using the original feature vectors. Tables 7.16, 7.18 and 7.20 show the best architecture accuracy found using the reduced feature vectors. The first column in these tables represents the experiment number (ranging from 1 to 10). The second column represents the precision and recall calculated for person objects. The third, fourth and fifth columns pertain to precision and recall calculated for bicycle, vehicle and clutter objects. The final two rows show the class mean and standard deviation values for precision and recall. These final rows summarize the estimated classification capabilities of the architecture described.

Tables 7.11, 7.13 and 7.15 show the top five ranked architectures trained for visible, gray-scale near-infrared and red channel near-infrared image sets using the original feature vectors. Similarly Tables 7.17, 7.19 and 7.21 show the top five ranked architectures trained using the reduced feature vectors. The first row in these tables represents the ranking of the architecture. The second row designates the learning rate used for training the neural network. The third row designates the momentum parameter used for training. The fourth row indicates the average precision and recall calculated for person objects across all ten experiments. Similarly the fifth, sixth and seventh rows indicate the average precision and recall calculated for bicycle, vehicle and clutter objects. The eighth row shows the total time in seconds needed for training. The ninth row indicates the total time in seconds needed to classify the test set. The final column in these tables indicate the average time needed for classification of the test set across the

top five ranking architectures. These average precision and recall calculations indicate the estimated classification capabilities of the architecture for a given class.

Table 7.10. Precision/Recall Obtained From Highest Performing Classifier Trained Using Original Feature Vector For Visible Images. Learning Rate = 0.24, Momentum = 0.72

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 71.0 / 89.1 | 69.2 / 31.0 | 200 / 92.9 | 85.7 / 92.3 |
| **2** | 56.0 / 96.2 | 35.7 / 19.2 | 25.0 / 100 | 94.9 / 54.4 |
| **3** | 73.2 / 66.1 | 0 / 0 | 80.0 / 100 | 84.4 / 89.0 |
| **4** | 92.3 / 90.6 | 78.9 / 60.0 | 100 / 100 | 76.2 / 91.4 |
| **5** | 82.5 / 91.2 | 90.0 / 50.0 | 55.6 / 83.3 | 72.7 / 69.6 |
| **6** | 70.8 / 91.9 | 80.0 / 26.7 | 100 / 100 | 86.2 / 84.8 |
| **7** | 82.8 / 82.8 | 92.9 / 86.7 | 87.5 / 77.8 | 82.2 / 84.5 |
| **8** | 43.6 / 92.3 | 28.6 / 14.8 | 43.5 / 100 | 100 / 58.0 |
| **9** | 62.2 / 80.7 | 0 / 0 | 64.3 / 90.0 | 68.1 / 56.1 |
| **10** | 87.1 / 91.0 | 69.2 / 75.0 | 100 / 73.3 | 80.8 / 79.7 |
| **Mean** | 72.2 / 87.2 | 60.5 / 36.3 | 75.6 / 91.7 | 83.1 / 76.0 |
| **Std Dev** | 15.0 / 8.7 | 31.8 / 30.4 | 27.2 / 10.3 | 9.6 / 15.2 |

Table 7.11. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Original Feature Vector For Visible Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Learn Rate** | 0.24 | 0.36 | 0.04 | 0.28 | 0.08 | |
| **Mom.** | 0.72 | 0.8 | 0.72 | 0.68 | 0.96 | |
| **Person** | 72.2 / 87.2 | 76.9 / 82.6 | 72.1 / 86.3 | 73.9 / 83.2 | 77.2 / 78.7 | 74.5 / 83.6 |
| **Bicycle** | 60.5 / 36.3 | 73.8 / 39.8 | 58.4 / 33.7 | 66.3 / 40.3 | 67.8 / 39.4 | 65.4 / 37.9 |
| **Vehicle** | 75.6 / 91.7 | 71.1 / 72.7 | 83.0 / 85.5 | 75.7 / 76.4 | 75.7 / 64.6 | 76.2 / 78.2 |
| **Clutter** | 83.1 / 76.0 | 78.5 / 86.0 | 81.6 / 80.4 | 79.6 / 82.3 | 78.8 / 88.8 | 80.2 / 82.7 |
| **Train Time** | 35.231 | 35.6392 | 33.8697 | 34.4431 | 36.6028 | 35.15716 |
| **Test Time** | 0.025911 | 0.026556 | 0.026009 | 0.026087 | 0.025342 | 0.025981 |

Table 7.12. Precision/Recall Obtained From Highest Performing Classifier Trained Using Original Feature Vector For Gray-Scale Near-Infrared Images. Learning Rate = 0.28, Momentum = 0.68

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 64.8 / 93.3 | 0 / 0 | 100 / 100 | 94.6 / 77.8 |
| **2** | 65.9 / 93.1 | 87.5 / 28.0 | 100 / 100 | 77.8 / 68.9 |
| **3** | 45.6 / 50.0 | 0 / 0 | 8.8 / 100 | 95.1 / 63.9 |
| **4** | 55.3 / 71.2 | 0 / 0 | 0 / 0 | 39.0 / 88.9 |
| **5** | 95.5 / 84.2 | 0 / 0 | 10.3 / 100 | 33.3 / 63.6 |
| **6** | 69.5 / 66.1 | 24.4 / 58.8 | 100 / 38.1 | 78.7 / 69.6 |
| **7** | 90.5 / 98.5 | 53.8 / 58.3 | 100 / 40.0 | 76.5 / 72.2 |
| **8** | 84.4 / 79.4 | 80.8 / 67.7 | 66.7 / 100 | 77.8 / 80.8 |
| **9** | 95.5 / 62.7 | 54.5 / 78.3 | 65.9 / 93.1 | 61.9 / 63.4 |
| **10** | 84.0 / 93.7 | 66.7 / 100 | 100 / 95.5 | 95.3 / 68.3 |
| **Mean** | 75.1 / 79.2 | 36.8 / 39.1 | 65.2 / 76.7 | 73.0 / 71.8 |
| **Std Dev** | 17.4 / 16.2 | 35.9 / 38.1 | 42.8 / 36.6 | 22.1 / 834 |

Table 7.13. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Original Feature Vector For Gray-Scale Near-Infrared Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Learn Rate** | 0.28 | 0.24 | 0.04 | 0.16 | 0.28 | |
| **Mom.** | 0.68 | 0.68 | 0.84 | 0.8 | 0.64 | |
| **Person** | 75.1 / 79.2 | 70.0 / 77.0 | 68.3 / 71.8 | 69.0 / 75.1 | 72.1 / 73.4 | 70.9 / 75.3 |
| **Bicycle** | 46.0 / 39.1 | 46.4 / 32.3 | 44.7 / 36.5 | 51.7 / 35.6 | 55.1 / 19.6 | 48.8 / 32.6 |
| **Vehicle** | 72.4 / 76.7 | 85.5 / 79.8 | 85.2 / 73.4 | 80.0 / 67.8 | 81.0 / 67.4 | 80.8 / 73.0 |
| **Clutter** | 73.0 / 71.7 | 65.2 / 67.7 | 67.1 / 70.5 | 66.4 / 67.3 | 65.5 / 77.8 | 67.5 / 71.0 |
| **Train Time** | 34.4717 | 36.3863 | 38.745 | 40.8434 | 44.1187 | 38.91302 |
| **Test Time** | 0.025463 | 0.026078 | 0.026007 | 0.026708 | 0.026668 | 0.026185 |

Table 7.14. Precision/Recall Obtained From Highest Performing Classifier Trained Using Original Feature Vector For Red Channel Near-Infrared Images. Learning Rate = 0.4, Momentum = 0.92

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 53.3 / 96.6 | 63.6 / 23.3 | 62.5 / 83.3 | 96.7 / 46.0 |
| **2** | 77.1 / 80.4 | 75.0 / 60.0 | 100 / 66.7 | 71.7 / 75.0 |
| **3** | 80.0 / 76.7 | 43.8 / 28.0 | 100 / 80.0 | 84.3 / 93.9 |
| **4** | 62.5 / 54.4 | 66.7 / 46.7 | 100 / 100 | 36.8 / 60.9 |
| **5** | 94.6 / 73.6 | 76.1 / 80.0 | 28.6 / 100 | 78.0 / 86.5 |
| **6** | 31.8 / 36.8 | 8.3 / 7.1 | 100 / 83.3 | 52.6 / 50.6 |
| **7** | 94.3 / 95.7 | 38.5 / 83.3 | 100 / 18.2 | 95.0 / 73.1 |
| **8** | 65.5 / 65.5 | 47.6 / 80.0 | 50.0 / 50.0 | 81.3 / 60.0 |
| **9** | 87.7 / 79.4 | 100 / 25.0 | 63.6 / 29.2 | 55.6 / 85.4 |
| **10** | 86.0 / 67.2 | 52.2 / 80.0 | 100 / 83.3 | 45.2 / 66.7 |
| **Mean** | 72.4 / 73.5 | 57.2 / 51.3 | 80.5 / 69.4 | 69.7 / 69.8 |
| **Std Dev** | 19.6 / 18.6 | 25.1 / 28.9 | 26.9 / 28.3 | 21.0 / 15.8 |

Table 7.15. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Original Feature Vector For Red Channel Near-Infrared Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Learn Rate** | 0.4 | 0.4 | 0.08 | 0.4 | 0.2 | |
| **Mom.** | 0.92 | 0.84 | 0.72 | 0.8 | 0.96 | |
| **Person** | 72.4 / 73.5 | 70.6 / 82.2 | 72.0 / 73.1 | 72.5 / 71.5 | 77.7 / 71.7 | 73.0 / 74.4 |
| **Bicycle** | 57.2 / 51.3 | 53.0 / 39.4 | 40.0 / 42.1 | 48.5 / 33.5 | 42.7 / 47.2 | 48.4 / 42.7 |
| **Vehicle** | 80.5 / 69.4 | 80.1 / 76.6 | 89.2 / 72.5 | 75.1 / 80.0 | 82.2 / 58.2 | 81.4 / 71.3 |
| **Clutter** | 69.7 / 69.8 | 69.8 / 66.5 | 69.5 / 70.3 | 69.5 / 75.7 | 67.4 / 76.7 | 69.2 / 71.8 |
| **Train Time** | 36.9141 | 34.4084 | 36.4739 | 34.5238 | 37.0987 | 35.88378 |
| **Test Time** | 0.026929 | 0.026233 | 0.026544 | 0.028215 | 0.025684 | 0.026721 |

Table 7.16. Precision/Recall Obtained From Highest Performing Classifier Trained Using Reduced Feature Vector For Visible Images. Learning Rate = 0.08, Momentum = 0.8

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 60.2 / 85.5 | 0 / 0 | 100 / 42.9 | 66.0 / 79.5 |
| **2** | 69.0 / 92.5 | 66.7 / 23.1 | 50 / 100 | 91.7 / 85.4 |
| **3** | 77.0 / 75.8 | 25 / 29.4 | 33.3 / 100 | 88.9 / 79.1 |
| **4** | 68 / 96.2 | 42.9 / 12 | 100 / 100 | 87.1 / 77.1 |
| **5** | 75.6 / 59.6 | 66.7 / 33.3 | 50 / 66.7 | 47.6 / 87.0 |
| **6** | 56.9 / 100 | 100 / 6.7 | 100 / 100 | 100 / 76.3 |
| **7** | 74.1 / 93.8 | 26.7 / 26.7 | 80 / 44.4 | 96.6 / 78.9 |
| **8** | 50 / 80.8 | 63.6 / 25.9 | 90.9 / 100 | 80.9 / 79.7 |
| **9** | 68.6 / 84.2 | 100 / 26.7 | 87.0 / 100 | 67.3 / 61.4 |
| **10** | 82.4 / 94.4 | 50 / 91.7 | 78.9 / 100 | 97.9 / 62.2 |
| **Mean** | 68.2 / 86.3 | 54.1 / 27.5 | 77.0 / 85.4 | 82.4 / 76.7 |
| **Std Dev** | 9.9 / 12.0 | 32.1 / 25.0 | 24.2 / 24.3 | 17.0 / 8.5 |

Table 7.17. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Reduced Feature Vector For Visible Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Learn Rate** | 0.08 | 0.32 | 0.24 | 0.2 | 0.28 | |
| **Mom.** | 0.8 | 0.72 | 0.92 | 0.8 | 0.84 | |
| **Person** | 68.1 / 86.3 | 67.2 / 82.9 | 68.3 / 88.6 | 68.7 / 91.6 | 69.1 / 80.0 | 68.3 / 71.5 |
| **Bicycle** | 54.1 / 27.5 | 50.1 / 27.4 | 51.5 / 24.5 | 49.9 / 32.0 | 44.9 / 28.2 | 50.1 / 27.9 |
| **Vehicle** | 77.0 / 85.4 | 84.7 / 83.6 | 78.0 / 89.0 | 75.0 / 77.0 | 82.9 / 86.0 | 79.5 / 84.2 |
| **Clutter** | 82.4 / 76.7 | 81.1 / 78.0 | 81.6 / 72.5 | 86.2 / 73.5 | 81.6 / 80.0 | 82.6 / 76.1 |
| **Test Time** | 6.07948 | 5.93825 | 5.71151 | 5.85354 | 5.91678 | 5.899910 |
| **Train Time** | 0.026329 | 0.024218 | 0.025243 | 0.024581 | 0.024188 | 0.024910 |

Table 7.18. Precision/Recall Obtained From Highest Performing Classifier Trained Using Reduced Feature Vector For Gray-Scale Near-Infrared Images. Learning Rate = 0.4, Momentum = 0.72

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 52.3 / 60 | 0 / 0 | 60 / 30 | 40.3 / 55.6 |
| **2** | 57.5 / 86.2 | 66.7 / 8 | 85.7 / 100 | 81.1 / 70.5 |
| **3** | 56.1 / 74.2 | 0 / 0 | 16.7 / 33.3 | 63.6 / 57.4 |
| **4** | 53.5 / 64.4 | 0 / 0 | 100 / 75 | 35.5 / 61.1 |
| **5** | 78.5 / 67.1 | 78.6 / 30.6 | 28.6 / 100 | 17.6 / 54.5 |
| **6** | 47.1 / 91.9 | 6.25 / 5.9 | 100 / 42.9 | 78.3 / 26.1 |
| **7** | 65.5 / 55.9 | 50 / 8.3 | 100 / 90 | 20.5 / 44.4 |
| **8** | 57.7 / 88.2 | 0 / 0 | 62.5 / 100 | 69.5 / 78.8 |
| **9** | 91.7 / 65.7 | 50 / 69.6 | 77.8 / 48.3 | 54.8 / 82.9 |
| **10** | 73.0 / 88.4 | 50 / 35.7 | 100 / 81.8 | 68.8 / 55 |
| **Mean** | 63.3 / 74.2 | 30.1 / 15.8 | 73.1 / 70.1 | 53.0 / 58.6 |
| **Std Dev** | 13.9 / 13.4 | 31.7 / 22.9 | 30.8 / 28.7 | 23.2 / 16.5 |

Table 7.19. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Reduced Feature Vector For Gray-Scale Near-Infrared Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Learn Rate** | 0.4 | 0.28 | 0.08 | 0.08 | 0.24 | |
| **Mom.** | 0.72 | 0.92 | 0.68 | 0.92 | 0.92 | |
| **Person** | 63.3 / 74.2 | 59.6 / 75.1 | 58.7 / 64.1 | 58.5 / 78.6 | 55.3 / 75.0 | 59.1 / 73.4 |
| **Bicycle** | 33.5 / 15.8 | 19.8 / 15.7 | 34.1 / 14.0 | 12.3 / 7.1 | 27.0 / 10.2 | 25.3 / 12.6 |
| **Vehicle** | 73.1 / 70.1 | 89.4 / 69.1 | 77.1 / 77.7 | 82.0 / 77.8 | 79.7 / 74.0 | 80.3 / 73.8 |
| **Clutter** | 53.0 / 58.6 | 60.6 / 52.0 | 56.5 / 55.8 | 72.3 / 48.9 | 69.2 / 43.2 | 62.3 / 51.7 |
| **Train Time** | 5.72845 | 6.25544 | 5.98835 | 5.80661 | 5.93136 | 5.942042 |
| **Test Time** | 0.023626 | 0.0244 | 0.24405 | 0.24048 | 0.023647 | 0.024025 |

Table 7.20. Precision/Recall Obtained From Highest Performing Classifier Trained Using Reduced Feature Vector For Red Channel Near-Infrared Images. Learning Rate = 0.28, Momentum = 0.88

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 40.6 / 72.9 | 9.5 / 13.3 | 0 / 0 | 75.0 / 19.0 |
| **2** | 58.1 / 78.3 | 50.0 / 13.3 | 0 / 0 | 44.4 / 27.3 |
| **3** | 62.1 / 24.7 | 11.8 / 72.0 | 0 / 0 | 80.0 / 10.5 |
| **4** | 49.1 / 60.9 | 0 / 0 | 0 / 0 | 100 / 17.4 |
| **5** | 50.0 / 1.4 | 84.2 / 72.7 | 3.5 / 100 | 0 / 0 |
| **6** | 24.7 / 35.1 | 0 / 0 | 0 / 0 | 64 /59.3 |
| **7** | 68.1 / 46.4 | 0 / 0 | 0 / 0 | 32.8 / 84.6 |
| **8** | 29.6 / 72.4 | 25.0 / 4.0 | 13.6 / 50.0 | 100 / 18.5 |
| **9** | 36.2 / 66.7 | 0 / 0 | 0 / 0 | 100 / 9.8 |
| **10** | 46.7 / 21.9 | 5.6 / 33.3 | 0 / 0 | 80.0 / 19.0 |
| **Mean** | 46.5 / 48.0 | 18.6 / 20.9 | 1.7 / 15.0 | 67.6 / 26.5 |
| **Std Dev** | 14.0 / 26.3 | 27.9 / 29.1 | 4.3 / 33.7 | 33.1 / 25.7 |

Table 7.21. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Reduced Feature Vector For Red Channel Near-Infrared Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Learn Rate** | 0.28 | 0.24 | 0.04 | 0.4 | 0.36 | |
| **Mom.** | 0.88 | 0.68 | 0.64 | 0.96 | 0.72 | |
| **Person** | 46.5 / 48.0 | 47.7 / 40.9 | 43.1 / 52.5 | 49.1 / 26.8 | 48.6 / 54.2 | 47.0 / 44.5 |
| **Bicycle** | 23.3 / 20.9 | 11.4 / 24.6 | 8.4 / 12.5 | 14.6 / 21.6 | 7.7 / 9.5 | 13.1 / 17.8 |
| **Vehicle** | 2.1 / 15.0 | 7.1 / 21.7 | 16.0 / 12.1 | 12.6 / 22.1 | 5.2 / 16.2 | 8.6 / 17.4 |
| **Clutter** | 67.6 / 26.5 | 61.8 / 32.1 | 61.7 / 39.7 | 51.2 / 47.6 | 66.7 / 36.0 | 61.8 / 36.3 |
| **Train Time** | 6.3246 | 6.10261 | 6.18044 | 6.1102 | 6.08352 | 6.160274 |
| **Test Time** | 0.02559 | 0.024762 | 0.036363 | 0.025142 | 0.024647 | 0.027301 |

### 7.3. ssEAM CLASSIFICATION RESULTS

Two ssEAM neural network architectures were explored through this research. The first architecture used the original feature vectors calculated in Section 5.6. The second architecture used the reduced feature sets indicated by Table 7.9. Ten-fold cross validation was used to calculate the accuracy of each classification system. Classification results were reported in terms of precision and recall as introduced by Bar-Ilan et al. [Bar-Ilan, 1998]. Precision and recall were used to determine the highest scoring feature vector order for each architecture variation. Highest scoring orders were averaged over the ten experiments to find the architecture variations with the most classification potential. This section lists the results collected using the trained ssEAM architectures.

Tables 7.22, 7.24 and 7.26 show the best architecture accuracy found for visible, gray-scale near-infrared and red channel near-infrared image sets respectively using the original feature vectors. Tables 7.28, 7.30 and 7.32 show the best architecture accuracy found using the reduced feature vectors. The first column in these tables represents the experiment number (ranging from 1 to 10). The second column represents the precision and recall calculated for person objects. The third, fourth and fifth columns pertain to precision and recall calculated for bicycle, vehicle and clutter objects. The final two rows show the class mean and standard deviation values for precision and recall. These final rows summarize the estimated classification capabilities of the architecture described.

Tables 7.23, 7.25 and 7.27 show the top five ranked architectures trained for visible, gray-scale near-infrared and red channel near-infrared image sets using the original feature vectors. Similarly Tables 7.29, 7.31 and 7.33 show the top five ranked architectures trained using the reduced feature vectors. The first row in these tables

represents the ranking of the architecture. The second row designates the eccentricity of the hyper-ellipsoids (mu) used for categorizing. The third and fourth rows designate the tolerance and vigilance parameters used for training. The fifth row indicates the average precision and recall calculated of the maximum scoring feature vector order of person objects across all ten experiments. Similarly the sixth, seventh and eighth rows indicate the average precision and recall calculated for bicycle, vehicle and clutter objects. The ninth row shows the total time in seconds needed for training. The tenth row indicates the total time in seconds needed to classify the test set. The final column in these tables indicate the average time needed for classification of the test set across the top five ranking architectures. These average precision and recall calculations indicate the estimated classification capabilities of the architecture for a given class.

Table 7.22. Precision/Recall Obtained From Highest Performing Classifier Trained Using Original Feature Vector For Visible Images. Mu = 0.8, Tolerance = 0.3, Vigilance = 0.5

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 44.8 / 54.5 | 38.9 / 24.1 | 16.7 / 21.4 | 41.2 / 35.9 |
| **2** | 37.3 / 71.7 | 23.1 / 11.5 | 27.3 / 50.0 | 72.6 / 43.7 |
| **3** | 66.7 / 54.8 | 17.6 / 17.6 | 25.0 / 50.0 | 71.4 / 76.9 |
| **4** | 53.2 / 47.2 | 30.8 / 32.0 | 43.8 / 63.6 | 37.1 / 37 |
| **5** | 75.0 / 52.6 | 42.9 / 50.0 | 40.0 / 33.3 | 39.5 / 65.2 |
| **6** | 46.2 / 48.6 | 85.7 / 40.0 | 100 / 33.3 | 65.8 / 88.1 |
| **7** | 50.0 / 51.6 | 18.8 / 20.0 | 100 / 22.2 | 60.0 / 63.4 |
| **8** | 21.6 / 42.3 | 77.8 / 25.9 | 100 / 10.0 | 69.0 / 71.0 |
| **9** | 51.5 / 61.4 | 35.0 / 46.7 | 16.7 / 10.0 | 59.2 / 50.9 |
| **10** | 57.1 / 36.0 | 22.5 / 75.0 | 50.0 / 13.3 | 54.4 / 66.2 |
| **Mean** | 50.3 / 52.1 | 39.3 / 34.3 | 51.9 / 20.7 | 57.0 / 59.8 |
| **Std Dev** | 14.8 / 9.9 | 24.0 / 19.1 | 34.9 / 18.7 | 13.5 / 17.4 |

Table 7.23. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Original Feature Vector For Visible Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Mu** | 0.8 | 0.4 | 0.6 | 0.2 | 0.2 | |
| **Tolerance** | 0.3 | 0.5 | 0.3 | 0.3 | 0.1 | |
| **Vigilance** | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | |
| **Person** | 50.3 / 52.1 | 48.5 / 51.9 | 50.5 / 53.2 | 49.4 / 49.6 | 49.6 / 53.8 | 49.7 / 52.1 |
| **Bicycle** | 39.3 / 34.3 | 43.2 / 28.4 | 33.9 / 34.5 | 37.1 / 34.2 | 32.6 / 33.7 | 37.2 / 33.0 |
| **Vehicle** | 51.9 / 20.7 | 58.7 / 19.8 | 48.8 / 25.4 | 46.4 / 31.0 | 49.3 / 29.2 | 51.0 / 27.2 |
| **Clutter** | 57.0 / 59.8 | 58.6 / 65.2 | 60.7 / 60.7 | 57.5 / 60.6 | 58.5 / 57.7 | 58.5 / 60.8 |
| **Train Time** | 1.15675 | 0.514684 | 1.18358 | 1.23763 | 1.79881 | 1.178291 |
| **Test Time** | 0.041204 | 0.022007 | 0.046751 | 0.050117 | 0.065393 | 0.045094 |

Table 7.24. Precision/Recall Obtained From Highest Performing Classifier Trained Using Original Feature Vector For Gray-Scale Near-Infrared Images. Mu = 0.4, Tolerance = 0.5, Vigilance = 0.5

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 51.4 / 73.3 | 45.3 / 9.7 | 66.7 / 20.0 | 36.8 / 31.1 |
| **2** | 45.3 / 58.6 | 22.2 / 16.0 | 100 / 16.7 | 52.9 / 44.3 |
| **3** | 54.7 / 75.8 | 50.0 / 18.2 | 25.0 / 33.3 | 72.3 / 55.7 |
| **4** | 59.0 / 78.0 | 27.3 / 10.7 | 70.0 / 58.3 | 46.7 / 38.9 |
| **5** | 66.7 / 63.2 | 70.0 / 19.4 | 20.0 / 25.0 | 24.0 / 54.5 |
| **6** | 43.6 / 77.4 | 30.8 / 33.3 | 75.0 / 14.3 | 73.7 / 40.6 |
| **7** | 78.0 / 67.6 | 30.8 / 33.3 | 100 / 10.0 | 56.3 / 100 |
| **8** | 29.6 / 70.6 | 62.5 / 16.1 | 50.0 / 20.0 | 75.8 / 48.1 |
| **9** | 40.4 / 62.7 | 22.2 / 26.1 | 100 / 6.9 | 30.8 / 19.5 |
| **10** | 67.4 / 63.2 | 23.8 / 35.7 | 50.0 / 9.1 | 66.7 / 76.7 |
| **Mean** | 53.6 / 69.0 | 38.0 / 22.1 | 65.7 / 21.4 | 53.6 / 50.9 |
| **Std Dev** | 14.6 / 7.0 | 17.2 / 9.9 | 29.6 / 15.2 | 18.7 / 23.1 |

Table 7.25. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Original Feature Vector For Gray-Scale Near-Infrared Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Mu** | 0.4 | 1 | 0.8 | 0.2 | 1 | |
| **Tolerance** | 0.5 | 0.5 | 0.5 | 0.5 | 0.3 | |
| **Vigilance** | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | |
| **Person** | 53.6 / 69.0 | 52.4 / 74.7 | 52.4 / 71.9 | 52.7 / 66.6 | 54.0 / 64.1 | 53.0 / 69.3 |
| **Bicycle** | 38.0 / 22.1 | 36.7 / 15.8 | 35.4 / 16.9 | 29.2 / 23.5 | 37.2 / 24.1 | 35.3 / 20.5 |
| **Vehicle** | 65.7 / 21.4 | 72.5 / 15.3 | 62.0 / 23.8 | 60.1 / 22.4 | 43.6 / 32.2 | 60.8 / 23.0 |
| **Clutter** | 53.6 / 50.9 | 57.2 / 48.5 | 54.8 / 46.4 | 55.2 / 48.3 | 50.6 / 50.9 | 54.3 / 49.0 |
| **Train Time** | 0.514138 | 0.365545 | 0.438471 | 0.530506 | 1.060988 | 0.58193 |
| **Test Time** | 0.02256 | 0.015557 | 0.018852 | 0.023501 | 0.037306 | 0.023555 |

Table 7.26. Precision/Recall Obtained From Highest Performing Classifier Trained Using Original Feature Vector For Red Channel Near-Infrared Images. Mu = 0.8, Tolerance = 0.5, Vigilance = 0.5

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 44.6 / 62.7 | 33.3 / 16.7 | 25.0 / 8.3 | 57.6 / 54.0 |
| **2** | 49.1 / 56.5 | 21.4 / 20.0 | 50.0 / 33.3 | 54.3 / 43.2 |
| **3** | 48.9 / 60.3 | 46.7 / 28.0 | 14.3 / 20.0 | 71.0 / 62.3 |
| **4** | 47.9 / 76.1 | 62.5 / 16.7 | 100 / 16.7 | 50.0 / 52.2 |
| **5** | 56.1 / 44.4 | 56.3 / 20.5 | 25.0 / 25.0 | 34.7 / 70.3 |
| **6** | 45.7 / 75.4 | 57.1 / 28.6 | 50.0 / 4.2 | 73.6 / 65.4 |
| **7** | 71.2 / 60.9 | 22.2 / 33.3 | 100 / 9.1 | 41.7 / 57.7 |
| **8** | 21.5 / 48.3 | 45.5 / 20.0 | 100 / 8.3 | 66.0 / 53.8 |
| **9** | 54.5 / 63.2 | 28.6 / 30.0 | 50.0 / 12.5 | 52.5 / 51.2 |
| **10** | 55.0 / 51.6 | 6.7 / 13.3 | 100 / 4.2 | 25.0 / 33.3 |
| **Mean** | 49.5 / 59.9 | 38.0 / 22.7 | 61.4 / 14.2 | 52.6 / 54.3 |
| **Std Dev** | 12.4 / 10.4 | 18.4 / 6.7 | 35.3 / 9.6 | 15.6 / 10.7 |

Table 7.27. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Original Feature Vector For Red Channel Near-Infrared Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Mu** | 0.8 | 0.8 | 0.2 | 0.6 | 0.4 | |
| **Tolerance** | 0.5 | 0.3 | 0.3 | 0.5 | 0.5 | |
| **Vigilance** | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | |
| **Person** | 49.5 / 60.0 | 49.2 / 60.9 | 53.5 / 56.9 | 48.2 / 63.9 | 47.5 / 52.4 | 49.6 / 58.8 |
| **Bicycle** | 38.0 / 22.7 | 37.8 / 32.4 | 34.8 / 32.7 | 37.3 / 24.1 | 33.9 / 24.7 | 36.4 / 27.3 |
| **Vehicle** | 61.4 / 14.2 | 46.6 / 18.3 | 38.7 / 25.4 | 53.9 / 19.6 | 53.6 / 21.2 | 50.9 / 19.8 |
| **Clutter** | 52.6 / 54.3 | 54.3 / 48.4 | 51.3 / 53.5 | 52.9 / 45.1 | 51.6 / 55.9 | 52.6 / 51.4 |
| **Train Time** | 0.21865 | 0.579652 | 0.604578 | 0.234865 | 0.258831 | 0.379315 |
| **Test Time** | 0.009828 | 0.021997 | 0.02458 | 0.009942 | 0.011101 | 0.01549 |

Table 7.28. Precision/Recall Obtained From Highest Performing Classifier Trained Using Reduced Feature Vector For Visible Images. Mu =0.4, Tolerance = 0.3, Vigilance = 0.98

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 58.9 / 78.2 | 53.3 / 27.6 | 100 / 71.4 | 69.2 / 69.2 |
| **2** | 52.2 / 67.9 | 36.4 / 15.4 | 45.5 / 83.3 | 80.4 / 75.7 |
| **3** | 86.8 / 53.3 | 14.3 / 11.8 | 75.0 / 75.0 | 68.6 / 89.0 |
| **4** | 72.7 / 75.5 | 30.8 / 16.0 | 78.6 / 100 | 69.0 / 82.9 |
| **5** | 75.4 / 86.0 | 50.0 / 536 | 27.3 / 50.0 | 69.2 / 78.3 |
| **6** | 77.8 / 75.7 | 6.1 / 73.3 | 94.1 / 76.2 | 88.5 / 91.5 |
| **7** | 71.4 / 62.5 | 27.8 / 33.3 | 100 / 22.2 | 71.1 / 83.1 |
| **8** | 30.2 / 61.5 | 100 / 7.4 | 18.8 / 30.0 | 77.0 / 68.1 |
| **9** | 51.8 / 77.2 | 21.1 / 26.7 | 95.0 / 30.0 | 61.1 / 38.6 |
| **10** | 74.6 / 52.8 | 20.0 / 50.0 | 86.7 / 86.7 | 67.1 / 74.3 |
| **Mean** | 65.2 / 69.0 | 41.5 / 26.7 | 72.1 / 62.5 | 72.1 / 75.1 |
| **Std Dev** | 16.7 / 11.2 | 25.8 / 21.1 | 30.5 / 27.4 | 7.8 / 15.0 |

Table 7.29. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Reduced Feature Vector For Visible Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Mu** | 0.4 | 0.8 | 0.8 | 0.6 | 1 | |
| **Tolerance** | 0.3 | 0.3 | 0.1 | 0.3 | 0.5 | |
| **Vigilance** | 0.98 | 0.98 | 0.98 | 0.98 | 0.95 | |
| **Person** | 65.2 / 69.0 | 66.3 / 67.9 | 64.9 / 68.1 | 64.2 / 68.9 | 61.3 / 64.3 | 64.4 / 67.6 |
| **Bicycle** | 41.5 / 26.7 | 40.6 / 27.3 | 36.4 / 35.5 | 35.7 / 26.8 | 36.2 / 33.9 | 38.1 / 30.1 |
| **Vehicle** | 72.1 / 62.5 | 70.6 / 62.7 | 73.7 / 55.9 | 74.7 / 60.6 | 69.0 / 66.3 | 72.0 / 61.6 |
| **Clutter** | 72.1 / 75.1 | 72.1 / 75.6 | 72.5 / 71.2 | 71.8 / 74.3 | 71.2 / 70.5 | 72.0 / 73.3 |
| **Test Time** | 0.836135 | 0.943319 | 1.1024 | 0.955412 | 0.78199 | 0.923851 |
| **Train Time** | 0.037501 | 0.040259 | 0.046699 | 0.038834 | 0.033232 | 0.039305 |

Table 7.30. Precision/Recall Obtained From Highest Performing Classifier Trained Using Reduced Feature Vector For Gray-Scale Near-Infrared Images. Mu = 0.4, Tolerance = 0.3, Vigilance =0.98

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 58.7 / 85.3 | 59.1 / 41.9 | 50.0 / 60.0 | 77.8 / 31.1 |
| **2** | 47.0 / 53.4 | 20.7 / 24.0 | 83.3 / 83.3 | 63.3 / 50.8 |
| **3** | 69.9 / 82.3 | 20.0 / 4.5 | 42.9 / 100 | 76.2 / 78.7 |
| **4** | 67.1 / 79.7 | 50.0 / 32.1 | 100 / 100 | 76.5 / 72.2 |
| **5** | 80.0 / 73.7 | 38.5 / 27.8 | 18.2 / 100 | 33.3 / 27.3 |
| **6** | 73.7 / 67.7 | 52.9 / 52.9 | 95.5 / 100 | 79.5 / 84.1 |
| **7** | 75.7 / 82.4 | 20.0 / 25.0 | 100 / 30.0 | 81.3 / 72.2 |
| **8** | 39.3 / 70.6 | 42.1 / 25.8 | 71.4 / 50.0 | 77.5 / 59.6 |
| **9** | 67.1 / 85.1 | 55.0 / 47.8 | 77.8 / 48.3 | 62.2 / 56.1 |
| **10** | 79.4 / 85.3 | 31.3 / 35.7 | 95.0 / 86.4 | 83.0 / 73.3 |
| **Mean** | 65.8 / 76.5 | 39.0 / 31.8 | 73.4 / 75.8 | 71.0 / 60.5 |
| **Std Dev** | 13.6 / 10.3 | 15.3 / 13.8 | 27.9 / 26.4 | 15.0 / 19.5 |

Table 7.31. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Reduced Feature Vector For Gray-Scale Near-Infrared Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Mu** | 0.4 | 0.6 | 0.2 | 0.8 | 0.4 | |
| **Tolerance** | 0.3 | 0.3 | 0.1 | 0.1 | 0.1 | |
| **Vigilance** | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | |
| **Person** | 65.8 / 76.5 | 65.4 / 77.8 | 65.7 / 73.8 | 64.8 / 77.1 | 66.3 / 75.5 | 65.6 / 76.2 |
| **Bicycle** | 39.0 / 31.8 | 39.5 / 30.0 | 38.2 / 35.0 | 36.2 / 28.4 | 36.3 / 32.3 | 37.8 / 31.5 |
| **Vehicle** | 73.4 / 75.8 | 71.1 / 78.6 | 70.2 / 79.6 | 71.2 / 79.1 | 67.8 / 75.1 | 70.7 / 77.7 |
| **Clutter** | 71.0 / 60.5 | 71.2 / 59.2 | 70.0 / 59.4 | 72.7 / 61.9 | 72.2 / 62.0 | 71.4 / 60.6 |
| **Train Time** | 1.015981 | 0.968147 | 1.080609 | 1.048993 | 1.053158 | 1.034474 |
| **Test Time** | 0.041947 | 0.044268 | 0.046168 | 0.050359 | 0.046239 | 0.045796 |

Table 7.32. Precision/Recall Obtained From Highest Performing Classifier Trained Using Reduced Feature Vector For Red Channel Near-Infrared Images. Mu = 0.2, Tolerance = 0.1, Vigilance = 0.98

| Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|
| **1** | 55.6 / 76.3 | 44.4 / 53.3 | 60.0 / 50.0 | 86.5 / 50.8 |
| **2** | 56.9 / 80.4 | 37.5 / 40.0 | 100 / 33.3 | 61.5 / 36.4 |
| **3** | 60.0 / 67.1 | 21.1 / 16.0 | 50.0 / 60.0 | 78.2 / 75.4 |
| **4** | 60.9 / 84.8 | 50.0 / 26.7 | 83.3 / 41.7 | 64.0 / 69.6 |
| **5** | 75.8 / 69.4 | 14.8 / 9.1 | 6.3 / 50.0 | 65.6 / 56.8 |
| **6** | 62.5 / 61.4 | 23.3 / 50.0 | 94.7 / 75.0 | 80.3 / 70.4 |
| **7** | 76.3 / 84.1 | 18.2 / 16.7 | 63.6 / 63.6 | 75.0 / 57.7 |
| **8** | 38.3 / 79.3 | 35.7 / 20.0 | 41.7 / 41.7 | 82.2 / 56.9 |
| **9** | 71.9 / 80.7 | 47.8 / 55.0 | 64.0 /66.7 | 66.7 / 48.8 |
| **10** | 78.7 / 75.0 | 27.3 / 60.0 | 90.5 / 79.2 | 55.6 / 23.8 |
| **Mean** | 63.7 / 75.9 | 32.0 / 34.7 | 65.4 / 56.1 | 71.6 / 54.6 |
| **Std Dev** | 12.4 / 7.7 | 12.8 / 19.1 | 28.6 / 15.2 | 10.2 / 15.8 |

Table 7.33. Average Precision/Recall Obtained From Highest Five Performing Classifiers Trained Using Reduced Feature Vector For Red Channel Near-Infrared Images.

| Rank | 1 | 2 | 3 | 4 | 5 | Mean |
|---|---|---|---|---|---|---|
| **Mu** | 0.2 | 0.8 | 0.4 | 0.6 | 0.4 | |
| **Tolerance** | 0.1 | 0.1 | 0.1 | 0.1 | 0.3 | |
| **Vigilance** | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | |
| **Person** | 63.7 / 75.9 | 64.6 / 24.8 | 64.0 / 74.4 | 65.2 / 75.2 | 63.6 / 75.0 | 64.2 / 75.0 |
| **Bicycle** | 32.0 / 34.7 | 30.1 / 27.3 | 32.3 / 29.1 | 30.0 / 28.3 | 37.3 / 32.8 | 32.4 / 30.4 |
| **Vehicle** | 65.4 / 56.1 | 67.0 / 61.4 | 62.5 / 63.9 | 65.0 / 56.7 | 55.7 / 62.3 | 63.1 / 60.1 |
| **Clutter** | 71.6 / 54.6 | 68.5 / 59.3 | 69.5 / 56.3 | 71.1 / 59.1 | 65.9 / 55.1 | 69.3 / 56.9 |
| **Train Time** | 1.095165 | 1.103951 | 1.125585 | 1.099705 | 1.045717 | 1.094025 |
| **Test Time** | 0.012449 | 0.049011 | 0.046775 | 0.047445 | 0.042797 | 0.046563 |

## 7.4. PREDICTION ERROR EXAMPLES

As indicated in Sections 7.4 and 7.5, the classification architectures developed in this research did not achieve 100% accuracy. An example misclassification of each of the four object classes are shown in Figures 7.1, 7.2, 7.3 and 7.4. These examples show some common reasons for misclassification.

Figure 7.1 demonstrates that person objects can bend and move in unexpected ways that may cause confusion in a classifier. In this example case, the person imaged is pulling on a dog and is leaning abnormally. The image processing algorithm did not segment the dog and the person, which could also cause confusion for the classifier.

Figure 7.2 reiterates the importance of object orientation for correct classification. In this image, a person is riding a bicycle, but is facing the camera head-on. This bicycle object was typically labeled as a person object.

Figure 7.3 shows another example of the image processing algorithm failing to segment images that are very close in proximity. In this case, a person and a vehicle

object are separated into one object boundary. This would cause confusion to the classification architecture as it would not match previously found patterns for vehicle or person classes. Oftentimes this object would be described as a clutter object.

Figure 7.1. Example of Person Misclassification. (a) Original Image. (b) Object Segmented Image.

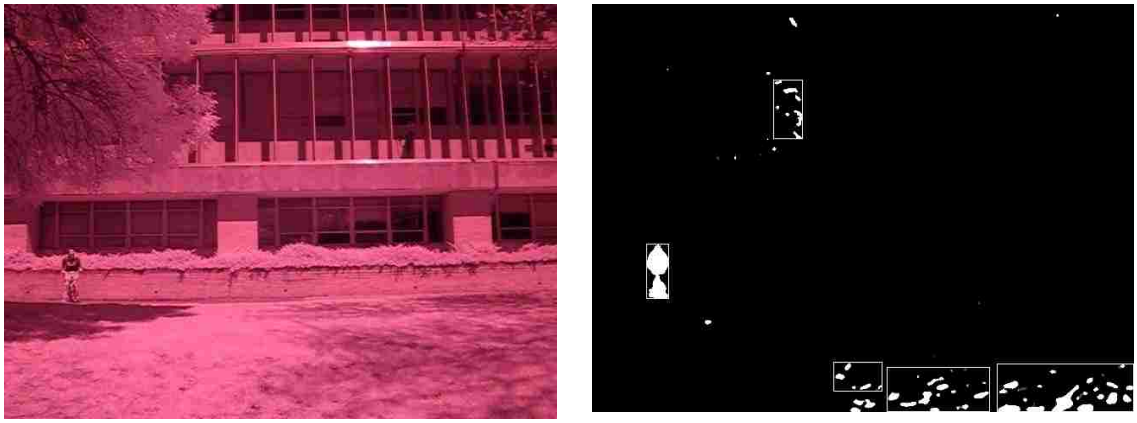

Figure 7.2. Example of Bicycle Misclassification. (a) Original Image. (b) Object Segmented Image.

Figure 7.3. Example of Vehicle Misclassification. (a) Original Image. (b) Object Segmented Image.

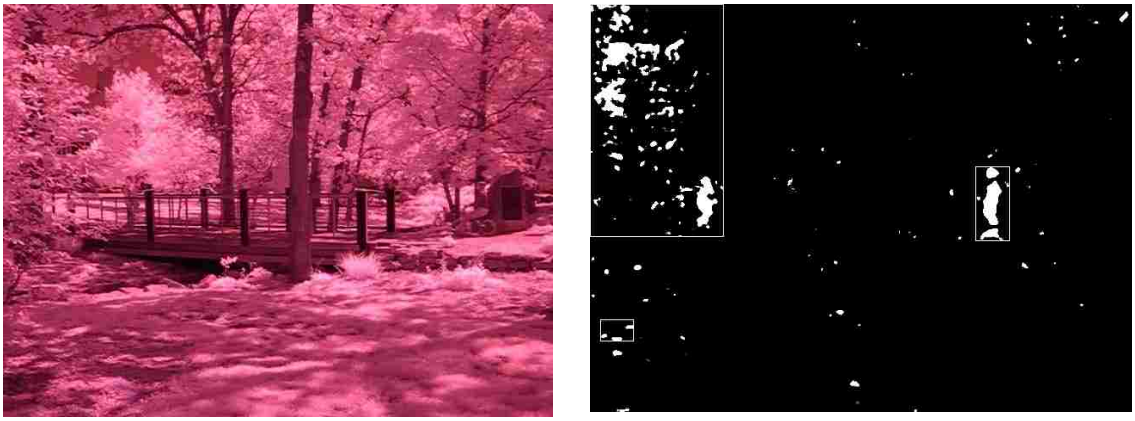

Figure 7.4. Example of Clutter Misclassification. (a) Original Image. (b) Object Segmented Image.

In the final figure, Figure 7.4, issues with the segmentation algorithm are once again observed. A clutter and a person object are classified in this case as one object. This object was most often labeled as a person object. It is noted that in some cases both the near-infrared and visible images are incapable of successfully segmenting objects with the developed processing algorithm.

While these examples indicate some weaknesses in the classification architectures and the image processing algorithm, these weaknesses would not make surveillance applications impossible. Oftentimes surveillance systems run real-time, which would make the noted weaknesses in this research less significant as objects will naturally move around an environment changing both their orientation to the camera and other environmental objects.

## 7.5. DATA FUSION RESULTS

Some principles of data fusion were used in this research to determine the uniqueness of the architectures found in Sections 7.2 and 7.4. Through this experimentation the highest classification architectures found were tested on all three types of image sets. Classification results were reported in terms of precision and recall.

Tables 7.34, 7.35 and 7.36 indicate the classification performance of the highest ranking MLP architectures for each image type using the original feature vectors on the other two image types. The first column indicates the image type and references the table where it is outlined. The first column is broken into two sub columns. The first of the two sub-columns indicates the label of the image type. The second sub-column designates whether the information presented is the initial data or a difference taken between the initial data collected from the training image type and the initial data collected from the new image type. Other column labels indicate the class of data being observed in terms of precision and recall. Similarly Tables 7.37, 7.38 and 7.39 examine the performance of the highest ranking MLP architectures using the reduced feature vectors. Tables 7.40, 7.41 and 7.42 and Tables 7.43, 7.44 and 7.45 illustrate the

performance of the highest ranking ssEAM architectures using the original and reduced feature vectors, respectively.

Table 7.34. Precision/Recall Obtained From Highest Performing MLP Classifier Trained Using Original Feature Vector From Visible Images (Table 7.10).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **Visible** | **Mean** | 72.2 / 87.2 | 60.5 / 36.3 | 75.6 / 91.7 | 83.1 / 76.0 |
| **NIR-Gray** | **1** | 53.3 / 42.7 | 22.2 / 12.9 | 80.0 / 80.0 | 49.3 / 80.0 |
| | **2** | 70.9 / 67.2 | 34.6 / 67.2 | 100 / 100 | 63.5 / 65.6 |
| | **3** | 90.9 / 64.5 | 25.0 / 9.1 | 60.0 / 100 | 56.0 / 83.6 |
| | **4** | 69.1 / 94.9 | 33.3 / 7.1 | 0 / 0 | 46.7 / 77.8 |
| | **5** | 100 / 50.0 | 0 / 0 | 0 / 0 | 11.8 / 90.9 |
| | **6** | 68.6 / 77.4 | 20.0 / 5.9 | 100 / 85.7 | 81.6 / 89.9 |
| | **7** | 87.9 / 42.6 | 66.7 / 16.7 | 0 / 0 | 23.6 / 94.4 |
| | **8** | 70.0 / 61.8 | 75.0 / 58.1 | 90.9 / 100 | 72.6 / 86.5 |
| | **9** | 87.9 / 43.3 | 75.0 / 39.1 | 60.0 / 93.1 | 37.1 / 63.4 |
| | **10** | 93.0 / 84.2 | 48.3 / 100 | 100 / 68.2 | 88.5 / 90.0 |
| | **Mean** | 79.2 / 62.9 | 40.0 / 28.5 | 84.4 / 62.7 | 53.1 / 82.2 |
| | **Difference** | 7.0 / 24.3 | 20.5 / 7.8 | 8.8 / 29.0 | 30.0 / 6.2 |
| **NIR-Red** | **1** | 51.9 / 93.2 | 0 / 0 | 37.5 / 25.0 | 74.5 / 55.6 |
| | **2** | 59.6 / 69.6 | 41.2 / 46.7 | 23.1 / 100 | 56.3 / 40.9 |
| | **3** | 91.7 / 75.3 | 50.0 / 48.0 | 22.7 / 100 | 83.8 / 81.6 |
| | **4** | 59.7 / 87.0 | 85.7 / 20.0 | 100 / 83.3 | 70.4 / 82.6 |
| | **5** | 80.0 / 88.9 | 0 / 0 | 23.5 / 100 | 60.7 / 91.9 |
| | **6** | 70.5 / 75.4 | 0 / 0 | 0 / 0 | 76.5 / 92.6 |
| | **7** | 94.1 / 92.8 | 100 / 33.3 | 100 / 45.5 | 61.0 / 96.2 |
| | **8** | 64.0 / 55.2 | 46.7 / 56.0 | 0 / 0 | 71.1 / 83.1 |
| | **9** | 90.9 / 35.1 | 0 / 0 | 70.6 / 100 | 39.5 / 82.9 |
| | **10** | 84.4 / 59.4 | 33.3 / 13.3 | 85.7 / 25.0 | 21.2 / 66.7 |
| | **Mean** | 75.7 / 73.2 | 39.7 / 21.7 | 57.9 / 57.9 | 61.5 / 77.4 |
| | **Difference** | 3.5 / 14.0 | 20.0 / 14.6 | 17.7 / 33.8 | 21.6 / 1.4 |

Table 7.35. Precision/Recall Obtained From Highest Performing MLP Classifier Trained Using Original Feature Vector From NIR-Gray Images (Table 7.12).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **NIR-Gray** | **Mean** | 75.1 / 79.2 | 36.8 / 39.1 | 65.2 / 76.7 | 73.0 / 71.8 |
| **Visible** | **1** | 63.6 / 76.4 | 10.0 / 3.4 | 92.9 / 92.9 | 70.2 / 84.6 |
| | **2** | 50.5 / 96.2 | 0 / 0 | 85.7 / 100 | 97.1 / 64.1 |
| | **3** | 75.4 / 74.2 | 0 / 0 | 100 / 100 | 77.9 / 89.0 |
| | **4** | 78.2 / 81.1 | 61.9 / 52.0 | 0 / 0 | 54.2 / 74.3 |
| | **5** | 67.1 / 100 | 0 / 0 | 83.3 / 83.3 | 100 / 56.5 |
| | **6** | 46.4 / 70.3 | 0 / 0 | 100 / 61.9 | 74.6 / 79.7 |
| | **7** | 60.9 / 87.5 | 5.3 / 6.7 | 100 / 11.1 | 97.9 / 64.8 |
| | **8** | 40.0 / 100 | 81.0 / 63.0 | 76.9 / 100 | 97.0 / 46.4 |
| | **9** | 73.4 / 82.5 | 56.3 / 60.0 | 58.8 / 50.0 | 65.4 / 60.0 |
| | **10** | 63.2 / 67.4 | 100 / 8.3 | 93.3 / 93.3 | 62.0 / 66.2 |
| | **Mean** | 61.9 / 83.6 | 39.3 / 19.3 | 87.9 / 69.3 | 79.6 / 68.5 |
| | **Difference** | 13.2 / 4.4 | 2.5 / 19.8 | 22.7 / 7.4 | 6.6 / 3.3 |
| **NIR-Red** | **1** | 40.6 / 72.9 | 0 / 0 | 66.7 / 16.7 | 65.9 / 42.9 |
| | **2** | 66.7 / 82.6 | 57.1 / 26.7 | 23.1 / 100 | 71.0 / 50.0 |
| | **3** | 68.4 / 74.0 | 20.0 / 8.0 | 41.7 / 100 | 90.0 / 91.2 |
| | **4** | 49.5 / 100 | 0 / 0 | 0 / 0 | 83.3 / 65.2 |
| | **5** | 75.0 / 83.3 | 0 / 0 | 50.0 / 75.0 | 42.9 / 81.1 |
| | **6** | 65.9 / 98.2 | 2.3 / 7.1 | 100 / 8.3 | 100 / 56.8 |
| | **7** | 71.0 / 71.0 | 100 / 16.7 | 76.9 / 90.9 | 41.2 / 53.8 |
| | **8** | 52.0 / 89.7 | 54.1 / 80.0 | 70.0 / 58.3 | 94.1 / 49.2 |
| | **9** | 64.0 / 96.5 | 0 / 0 | 41.9 / 75.0 | 84.6 / 26.8 |
| | **10** | 87.9 / 45.3 | 60.0 / 40.0 | 100 / 45.8 | 22.9 / 76.2 |
| | **Mean** | 64.1 / 81.4 | 36.7 / 17.8 | 63.4 / 57.0 | 69.5 / 59.3 |
| | **Difference** | 11.0 / 2.2 | 0.1 / 21.3 | 1.8 / 19.7 | 3.5 / 12.5 |

Table 7.36. Precision/Recall Obtained From Highest Performing MLP Classifier Trained Using Original Feature Vector From NIR-Red Images (Table 7.14).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **NIR-Red** | **Mean** | 72.4 / 73.5 | 57.2 / 51.3 | 80.5 / 69.4 | 69.7 / 69.8 |
| **Visible** | **1** | 58.1 / 65.5 | 85.7 / 20.7 | 71.4 / 71.4 | 63.0 / 87.2 |
| | **2** | 54.4 / 92.5 | 88.9 / 30.8 | 85.7 / 100 | 91.5 / 72.8 |
| | **3** | 52.1 / 80.6 | 0 / 0 | 100 / 100 | 78.4 / 63.7 |
| | **4** | 56.1 / 69.8 | 50.0 / 36.0 | 100 / 100 | 65.5 / 100 |
| | **5** | 66.7 / 70.2 | 33.3 / 16.7 | 80.0 / 66.7 | 43.3 / 56.5 |
| | **6** | 53.1 / 45.9 | 50.0 / 33.3 | 95.5 / 100 | 73.5 / 84.7 |
| | **7** | 77.6 / 92.2 | 27.3 / 20.0 | 80.0 / 89.9 | 83.9 / 73.2 |
| | **8** | 38.9 / 53.8 | 45.5 / 18.5 | 100 / 20.0 | 65.1 / 78.3 |
| | **9** | 37.0 / 17.5 | 50.0 / 26.7 | 100 / 30.0 | 38.9 / 73.7 |
| | **10** | 71.4 / 39.3 | 76.9 / 83.3 | 100 / 93.3 | 50.0 / 77.0 |
| | **Mean** | 56.5 / 62.7 | 50.8 / 28.6 | 91.3 / 77.0 | 65.3 / 72.1 |
| | **Difference** | 15.9 / 10.8 | 6.4 / 22.7 | 10.8 / 7.6 | 4.4 / 2.3 |
| **NIR-Gray** | **1** | 62.7 / 62.7 | 11.1 / 3.2 | 100 / 100 | 50.7 / 75.6 |
| | **2** | 77.8 / 60.3 | 30.0 / 60.0 | 100 / 100 | 73.5 / 59.0 |
| | **3** | 81.4 / 56.4 | 66.7 / 9.1 | 100 / 100 | 59.6 / 96.7 |
| | **4** | 77.6 / 76.3 | 33.3 / 7.1 | 100 / 41.7 | 27.1 / 72.2 |
| | **5** | 100 / 77.6 | 0 / 0 | 25.0 / 50.0 | 15.8 / 81.8 |
| | **6** | 76.5 / 62.9 | 25.0 / 17.6 | 100 / 100 | 70.6 / 87.0 |
| | **7** | 83.9 / 69.2 | 50.0 / 50.0 | 80.0 / 40.0 | 28.6 / 55.6 |
| | **8** | 74.2 / 67.6 | 76.2 / 51.6 | 100 / 50.0 | 67.1 / 90.4 |
| | **9** | 87.5 / 31.3 | 91.7 / 47.8 | 16.7 / 3.4 | 32.2 / 92.7 |
| | **10** | 84.5 / 63.2 | 5.0 / 7.1 | 0 / 0 | 44.0 / 73.3 |
| | **Mean** | 80.6 / 62.8 | 38.9 / 25.4 | 72.2 / 58.5 | 46.9 / 78.4 |
| | **Difference** | 8.2 / 10.7 | 18.3 / 25.9 | 8.3 / 10.9 | 22.8 / 8.6 |

Table 7.37. Precision/Recall Obtained From Highest Performing MLP Classifier Trained Using Reduced Feature Vector From Visible Images (Table 7.16).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **Visible** | **Mean** | 68.2 / 86.3 | 54.1 / 27.5 | 77.0 / 85.4 | 82.4 / 76.7 |
| **NIR-Gray** | **1** | 64.6 / 97.3 | 0 / 0 | 57.1 / 40.0 | 78.9 / 66.7 |
| | **2** | 70.8 / 58.6 | 40.5 / 60.0 | 42.9 / 100 | 76.5 / 63.9 |
| | **3** | 66.7 / 51.6 | 56.0 / 63.6 | 7.9 / 100 | 100 / 6.7 |
| | **4** | 61.4 / 86.4 | 0 / 0 | 100 / 8.3 | 60.0 / 66.7 |
| | **5** | 92.6 / 82.9 | 12.5 / 2.8 | 18.2 / 100 | 31.0 / 81.8 |
| | **6** | 49.2 / 100 | 0 / 0 | 9.1 / 4.8 | 100 / 20.3 |
| | **7** | 76.7 / 97.1 | 100 / 25.0 | 0 / 0 | 47.4 / 50.0 |
| | **8** | 73.9 / 50.0 | 25.0 / 6.5 | 9.2 / 60.0 | 87.1 / 51.9 |
| | **9** | 91.5 / 64.2 | 0 / 0 | 33.0 / 100 | 31.8 / 17.1 |
| | **10** | 92.9 / 83.2 | 31.3 / 71.4 | 91.3 / 95.5 | 96.1 / 81.7 |
| | **Mean** | 74.0 / 77.1 | 26.5 / 22.9 | 36.9 / 60.9 | 70.9 / 56.1 |
| | **Difference** | 5.8 / 9.2 | 27.6 / 4.6 | 40.1 / 24.5 | 11.5 / 20.6 |
| **NIR-Red** | **1** | 51.5 / 89.8 | 8.3 / 3.3 | 71.4 / 83.3 | 80.0 / 44.4 |
| | **2** | 67.9 / 82.6 | 91.7 / 73.3 | 25.0 / 100 | 75.0 / 47.3 |
| | **3** | 71.7 / 90.4 | 38.1 / 32.0 | 0 / 0 | 84.6 / 77.2 |
| | **4** | 51.7 / 97.8 | 0 / 0 | 0 / 0 | 58.8 / 43.5 |
| | **5** | 87.7 / 69.4 | 56.5 / 88.6 | 0 / 0 | 93.5 / 78.4 |
| | **6** | 51.6 / 56.1 | 0 / 0 | 0 / 0 | 71.0 / 81.5 |
| | **7** | 73.6 / 97.1 | 8.3 / 8.3 | 0 / 0 | 86.7 / 50.0 |
| | **8** | 48.9 / 79.3 | 63.2 / 48.0 | 63.6 / 58.3 | 88.9 / 73.8 |
| | **9** | 59.1 / 22.8 | 32.6 / 75.0 | 45.1 / 95.8 | 91.3 / 51.2 |
| | **10** | 93.8 / 93.8 | 56.3 / 60.0 | 0 / 0 | 35.9 / 66.7 |
| | **Mean** | 65.8 / 77.9 | 38.9 / 35.5 | 20.5 / 33.7 | 76.6 / 61.4 |
| | **Difference** | 2.4 / 8.4 | 18.6 / 11.4 | 56.5 / 51.7 | 5.8 / 15.3 |

Table 7.38. Precision/Recall Obtained From Highest Performing MLP Classifier Trained Using Reduced Feature Vector From NIR-Gray Images (Table 7.18).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **NIR-Gray** | **Mean** | 63.3 / 74.2 | 30.1 / 15.8 | 73.1 / 70.1 | 53.0 / 58.6 |
| **Visible** | **1** | 46.3 / 56.4 | 0 / 0 | 83.3 / 71.4 | 57.4 / 79.5 |
| | **2** | 48.6 / 67.9 | 0 / 0 | 66.7 / 100 | 81.0 / 78.6 |
| | **3** | 38.3 / 74.2 | 0 / 0 | 100 / 100 | 68.0 / 37.4 |
| | **4** | 53.0 / 83.0 | 0 / 0 | 84.6 / 100 | 57.1 / 45.7 |
| | **5** | 64.6 / 93.0 | 0 / 0 | 0 / 0 | 63.2 / 52.2 |
| | **6** | 2.5 / 5.4 | 0 / 0 | 100 / 81.0 | 56.8 / 91.5 |
| | **7** | 39.1 / 28.1 | 0 / 0 | 50.0 / 33.3 | 49.0 / 67.6 |
| | **8** | 41.5 / 65.4 | 44.4 / 29.6 | 18.8 / 60.0 | 80.5 / 47.8 |
| | **9** | 64.5 / 70.2 | 62.5 / 33.3 | 28.2 / 55.0 | 65.0 / 45.6 |
| | **10** | 64.8 / 66.3 | 19.1 / 75.0 | 58.3 / 93.3 | 100 / 37.8 |
| | **Mean** | 47.3 / 61.0 | 12.6 / 13.8 | 59.0 / 69.4 | 67.8 / 58.4 |
| | **Difference** | 16.0 / 13.2 | 17.5 / 2.0 | 14.1 / 0.7 | 14.8 / 0.2 |
| **NIR-Red** | **1** | 38.7 / 61.0 | 0 / 0 | 75.0 / 50.0 | 58.6 / 54.0 |
| | **2** | 58.9 / 93.5 | 0 / 0 | 100 / 100 | 81.8 / 40.9 |
| | **3** | 59.0 / 67.1 | 0 / 0 | 100 / 80.0 | 69.3 / 77.2 |
| | **4** | 40.5 / 65.2 | 0 / 0 | 75.0 / 25.0 | 32.0 / 34.8 |
| | **5** | 94.1 / 66.7 | 27.3 / 13.6 | 57.1 / 100 | 37.7 / 78.4 |
| | **6** | 31.8 / 47.4 | 0 / 0 | 100 / 70.8 | 55.2 / 45.7 |
| | **7** | 66.7 / 29.0 | 23.1 / 25.0 | 83.3 / 45.5 | 21.7 / 57.7 |
| | **8** | 50.0 / 96.6 | 66.7 / 40.0 | 85.7 / 50.0 | 88.7 / 72.3 |
| | **9** | 76.4 / 96.5 | 100 / 40.0 | 80.0 / 16.7 | 57.9 / 80.5 |
| | **10** | 88.9 / 100 | 100 / 33.3 | 72.4 / 87.5 | 77.8 / 66.7 |
| | **Mean** | 60.5 / 72.3 | 31.7 / 15.2 | 82.9 / 62.5 | 58.1 / 60.8 |
| | **Difference** | 2.8 / 1.9 | 1.6 / 0.6 | 9.8 / 7.6 | 5.1 / 2.2 |

Table 7.39. Precision/Recall Obtained From Highest Performing MLP Classifier Trained Using Reduced Feature Vector From NIR-Red Images (Table 7.20).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **NIR-Red** | **Mean** | 46.5 / 48.0 | 18.6 / 20.9 | 1.7 / 15.0 | 67.6 / 26.5 |
| **Visible** | **1** | 52.2 / 63.6 | 0 / 0 | 87.5 / 50.0 | 65.2 / 76.9 |
| | **2** | 57. / 83.0 | 60.0 / 11.5 | 42.9 / 100 | 90.3 / 81.6 |
| | **3** | 45.9 / 45.2 | 0 / 0 | 50.0 / 75.0 | 63.0 / 69.2 |
| | **4** | 57.3 / 96.2 | 28.6 / 8.0 | 90.0 / 81.8 | 88.9 / 45.7 |
| | **5** | 67.2 / 68.4 | 19.0 / 22.2 | 83.3 / 83.3 | 57.9 / 47.8 |
| | **6** | 43.8 / 46.8 | 0 / 0 | 95.2 / 95.2 | 72.4 / 71.2 |
| | **7** | 95.6 / 67.2 | 73.7 / 93.3 | 75.0 / 66.7 | 77.0 / 94.4 |
| | **8** | 39.4 / 50.0 | 33.3 / 29.6 | 16.9 / 100 | 93.8 / 21.7 |
| | **9** | 61.8 / 59.6 | 30.0 / 40.0 | 48.2 / 70.0 | 51.1 / 40.3 |
| | **10** | 76.6 / 95.5 | 28.6 / 33.3 | 62.5 / 100 | 100 / 55.4 |
| | **Mean** | 59.8 / 68.6 | 27.3 / 23.8 | 65.2 / 82.2 | 76.0 / 60.4 |
| | **Difference** | 13.3 / 20.5 | 8.7 / 2.9 | 63.5 / 67.2 | 8.4 / 33.9 |
| **NIR-Gray** | **1** | 62.5 / 93.3 | 0 / 0 | 90.9 / 100 | 83.8 / 68.9 |
| | **2** | 63.8 / 87.9 | 53.6 / 60.0 | 100 / 100 | 97.2 / 57.4 |
| | **3** | 40.6 / 66.1 | 0 / 0 | 13.6 / 100 | 52.9 / 14.8 |
| | **4** | 50.0 / 79.7 | 0 / 0 | 100 / 33.3 | 33.3 / 27.8 |
| | **5** | 100 / 81.6 | 0 / 0 | 0 / 0 | 55.0 / 100 |
| | **6** | 63.3 / 100 | 0 / 0 | 100 / 66.7 | 92.6 / 72.5 |
| | **7** | 47.8 / 16.2 | 87.5 / 58.3 | 58.8 / 100 | 5.0 / 16.7 |
| | **8** | 84.6 / 64.7 | 56.4 / 71.0 | 27.0 / 100 | 100 / 48.1 |
| | **9** | 91.4 / 47.8 | 46.7 / 30.4 | 59.2 / 100 | 37.7 / 56.1 |
| | **10** | 71.7 / 69.5 | 100 / 42.9 | 0 / 0 | 43.8 / 65.0 |
| | **Mean** | 67.6 / 70.7 | 34.4 / 26.3 | 55.0 / 70.0 | 60.1 / 52.7 |
| | **Difference** | 21.1 / 22.7 | 15.8 / 5.3 | 53.2 / 55.0 | 7.5 / 26.2 |

Table 7.40. Precision/Recall Obtained From Highest Performing ssEAM Classifier Trained Using Original Feature Vector From Visible Images (Table 7.22).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **Visible** | **Mean** | 50.3 / 52.1 | 39.3 / 34.3 | 51.9 / 20.7 | 57.0 / 59.8 |
| **NIR-Gray** | **1** | 52.3 / 30.7 | 27.3 / 9.7 | 100 / 10.0 | 33.0 / 68.9 |
| | **2** | 40.0 / 41.4 | 37.5 / 12.0 | 50.0 / 16.7 | 44.8 / 49.2 |
| | **3** | 44.6 / 46.8 | 14.3 / 4.5 | 33.3 / 33.3 | 42.9 / 44.3 |
| | **4** | 53.7 / 37.3 | 75.0 / 10.7 | 0 / 0 | 14.9 / 55.6 |
| | **5** | 62.3 / 43.4 | 77.8 / 19.4 | 0 / 0 | 12.3 / 63.6 |
| | **6** | 41.2 / 45.2 | 11.1 / 5.9 | 100 / 4.8 | 44.0 / 68.1 |
| | **7** | 70.0 / 41.2 | 50.0 / 16.7 | 0 / 0 | 24.1 / 77.8 |
| | **8** | 25.0 / 35.3 | 40.0 / 12.9 | 50.0 / 10.0 | 47.6 / 57.7 |
| | **9** | 44.4 / 47.8 | 37.5 / 13.0 | 40.0 / 6.9 | 29.9 / 48.8 |
| | **10** | 54.9 / 41.1 | 19.2 / 35.7 | 50.0 / 4.5 | 33.3 / 46.7 |
| | **Mean** | 48.8 / 41.0 | 39.0 / 14.1 | 42.3 / 8.6 | 33.7 / 58.1 |
| | **Difference** | 1.5 / 11.1 | 0.3 / 20.3 | 9.6 / 12.1 | 23.3 / 1.7 |
| **NIR-Red** | **1** | 42.0 / 35.6 | 69.2 / 30.0 | 0 / 0 | 41.1 / 61.9 |
| | **2** | 43.5 / 43.5 | 37.5 / 20.0 | 33.3 / 33.3 | 39.6 / 43.2 |
| | **3** | 31.8 / 38.4 | 30.0 / 12.0 | 37.5 / 60.0 | 46.5 / 40.4 |
| | **4** | 39.0 / 34.8 | 54.5 / 20.0 | 66.7 / 33.3 | 17.4 / 34.8 |
| | **5** | 51.1 / 33.3 | 52.0 / 29.5 | 0 / 0 | 28.6 / 54.1 |
| | **6** | 43.1 / 49.1 | 20.0 / 7.1 | 10.0 / 4.2 | 47.3 / 53.1 |
| | **7** | 67.4 / 44.9 | 50.0 / 16.7 | 0 / 0 | 21.7 / 50.0 |
| | **8** | 37.0 / 69.0 | 50.0 / 12.0 | 0 / 0 | 58.9 / 50.8 |
| | **9** | 43.4 / 40.4 | 0 / 0 | 100 / 56.1 | 33.3 / 56.1 |
| | **10** | 53.2 / 39.1 | 20.0 / 6.7 | 50.0 / 12.5 | 36.8 / 71.4 |
| | **Mean** | 45.1 / 42.8 | 38.3 / 15.4 | 29.8 / 14.7 | 36.1 / 51.6 |
| | **Difference** | 5.2 / 9.3 | 1.0 / 19.0 | 22.2 / 6.0 | 20.9 / 8.2 |

Table 7.41. Precision/Recall Obtained From Highest Performing ssEAM Classifier Trained Using Original Feature Vector From NIR-Gray Images (Table 7.24).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **NIR-Gray** | **Mean** | 53.6 / 69.0 | 38.0 / 22.1 | 65.7 / 21.4 | 53.6 / 50.9 |
| **Visible** | **1** | 46.1 / 74.5 | 22.2 / 6.9 | 50.0 / 21.4 | 37.9 / 28.2 |
| | **2** | 35.4 / 87.0 | 19.0 / 15.4 | 25.0 / 33.3 | 56.5 / 12.6 |
| | **3** | 50.0 / 64.5 | 15.4 / 11.8 | 0 / 0 | 62.5 / 44.0 |
| | **4** | 47.1 / 62.3 | 50.0 / 12.0 | 71.4 / 45.5 | 41.7 / 42.9 |
| | **5** | 57.9 / 57.9 | 30.0 / 16.7 | 50.0 / 33.3 | 25.0 / 30.4 |
| | **6** | 31.0 / 48.6 | 17.6 / 20.0 | 100 / 4.8 | 58.8 / 50.8 |
| | **7** | 47.6 / 62.5 | 33.3 / 26.7 | 25.0 / 22.2 | 52.4 / 31.0 |
| | **8** | 17.7 / 53.8 | 100 / 11.1 | 4.8 / 10.0 | 50.0 / 20.3 |
| | **9** | 40.4 / 66.7 | 40.0 / 26.7 | 22.2 / 10.0 | 38.7 / 21.1 |
| | **10** | 44.3 / 48.3 | 8.3 / 16.7 | 100 / 6.7 | 42.3 / 29.7 |
| | **Mean** | 41.8 / 62.6 | 33.6 / 16.4 | 44.8 / 18.7 | 46.6 / 31.1 |
| | **Difference** | 11.8 / 6.4 | 4.4 / 5.7 | 20.9 / 2.7 | 4.3 / 19.8 |
| **NIR-Red** | **1** | 42.9 / 61.0 | 40.0 / 13.3 | 60.0 / 25.0 | 43.8 / 33.3 |
| | **2** | 50. 8 / 71.7 | 0 / 0 | 33.3 / 33.3 | 69.0 / 45.5 |
| | **3** | 40.2 / 67.1 | 9.5 / 8.0 | 11.1 / 20.0 | 63.6 / 30.7 |
| | **4** | 41.8 / 60.9 | 50.0 / 6.7 | 71.4 / 41.7 | 22.6 / 30.4 |
| | **5** | 51.2 / 58.3 | 75.0 / 20.5 | 0 / 0 | 34.8 / 43.2 |
| | **6** | 40.0 / 73.7 | 33.3 / 25.0 | 100 / 9.1 | 68.7 / 40.7 |
| | **7** | 57.1 / 52.2 | 33.3 / 25.0 | 100 / 9.1 | 15.2 / 19.2 |
| | **8** | 26.5 / 75.9 | 40.0 / 16.0 | 100 / 16.7 | 64.7 / 33.8 |
| | **9** | 48.8 / 70.2 | 27.3 / 15.0 | 50.0 / 8.3 | 43.2 / 39.0 |
| | **10** | 51.7 / 71.9 | 15.4 / 13.3 | 60.0 / 12.5 | 15.4 / 9.5 |
| | **Mean** | 45.1 / 66.3 | 31.4 / 14.6 | 53.6 / 17.1 | 44.1 / 32.6 |
| | **Difference** | 8.5 / 2.7 | 7.0 / 7.5 | 12.1 / 4.3 | 6.8 / 18.3 |

Table 7.42. Precision/Recall Obtained From Highest Performing ssEAM Classifier Trained Using Original Feature Vector From NIR-Red Images (Table 7.26).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **NIR-Red** | **Mean** | 49.5 / 59.9 | 38.0 / 22.7 | 61.4 / 14.2 | 52.6 / 54.3 |
| **Visible** | **1** | 42.2 / 49.1 | 9.1 / 3.4 | 57.1 / 28.6 | 36.5 / 48.7 |
| | **2** | 39.4 / 49.1 | 15.4 / 7.7 | 7.7 / 16.7 | 68.7 / 55.3 |
| | **3** | 47.1 / 51.6 | 14.3 / 17.6 | 22.2 / 50.0 | 57.7 / 45.1 |
| | **4** | 47.6 / 56.6 | 25.0 / 8.0 | 71.4 / 45.5 | 31.8 / 40.0 |
| | **5** | 72.1 / 54.4 | 22.2 / 11.1 | 16.7 / 33.3 | 29.7 / 47.8 |
| | **6** | 30.5 / 48.6 | 18.2 / 13.3 | 50.0 / 4.8 | 58.5 / 52.5 |
| | **7** | 43.5 / 46.9 | 41.2 / 46.7 | 12.5 / 11.1 | 44.6 / 35.2 |
| | **8** | 21.3 / 65.4 | 37.5 / 11.1 | 100 / 10.0 | 52.8 / 27.5 |
| | **9** | 51.7 / 52.6 | 30.0 / 20.0 | 31.6 / 30.0 | 40.0 / 49.1 |
| | **10** | 48.3 / 48.3 | 5.3 / 8.3 | 66.7 / 13.3 | 39.2 / 39.2 |
| | **Mean** | 44.4 / 52.3 | 21.8 / 14.7 | 43.6 / 24.3 | 47.0 / 44.1 |
| | **Difference** | 5.1 / 7.6 | 16.2 / 8.0 | 17.8 / 10.1 | 5.6 / 10.2 |
| **NIR-Gray** | **1** | 51.4 / 50.7 | 41.2 / 22.6 | 10.0 / 10.0 | 26.8 / 33.3 |
| | **2** | 49.3 / 60.3 | 16.7 / 8.0 | 18.2 / 33.3 | 47.9 / 37.7 |
| | **3** | 54.5 / 67.7 | 42.9 / 13.6 | 0 / 0 | 62.0 / 50.8 |
| | **4** | 58.5 / 52.5 | 41.7 / 17.8 | 50.0 / 41.7 | 20.0 / 38.9 |
| | **5** | 64.8 / 46.1 | 50.0 / 25.0 | 0 / 0 | 16.7 / 63.6 |
| | **6** | 40.2 / 60.0 | 21.4 / 17.6 | 100 / 4.8 | 41.8 / 33.3 |
| | **7** | 73.0 / 39.7 | 15.4 / 16.7 | 33.3 / 10.0 | 34.0 / 88.9 |
| | **8** | 31.7 / 58.8 | 23.5 / 12.9 | 100 / 10.0 | 40.0 / 30.8 |
| | **9** | 41.8 / 49.3 | 27.8 / 21.7 | 100 / 3.4 | 21.8 / 29.3 |
| | **10** | 54.7 / 43.2 | 13.6 / 21.4 | 33.3 / 9.1 | 40.5 / 53.3 |
| | **Mean** | 52.0 / 52.8 | 29.4 / 17.7 | 44.5 / 12.2 | 35.2 / 46.0 |
| | **Difference** | 2.5 / 7.1 | 8.6 / 5.0 | 16.9 / 2.0 | 17.4 / 8.3 |

Table 7.43. Precision/Recall Obtained From Highest Performing ssEAM Classifier Trained Using Reduced Feature Vector From Visible Images (Table 7.28).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **Visible** | **Mean** | 65.2 / 69.0 | 41.5 / 26.7 | 72.1 / 62.5 | 72.1 / 75.1 |
| **NIR-Gray** | **1** | 0 / 0 | 50.0 / 3.2 | 0 / 0 | 27.7 / 97.8 |
| | **2** | 100 / 1.7 | 0 / 0 | 0 / 0 | 40.9 / 100 |
| | **3** | 50.0 / 1.6 | 0 / 0 | 0 / 0 | 41.1 / 98.4 |
| | **4** | 0 / 0 | 0 / 0 | 0 / 0 | 15.4 / 100 |
| | **5** | 0 / 0 | 0 / 0 | 0 / 0 | 8.7 / 100 |
| | **6** | 0 / 0 | 0 / 0 | 0 / 0 | 40.8 / 100 |
| | **7** | 100 / 1.5 | 0 / 0 | 0 / 0 | 16.8 / 100 |
| | **8** | 0 / 0 | 0 / 0 | 0 / 0 | 40.9 / 100 |
| | **9** | 0 / 0 | 0 / 0 | 0 / 0 | 25.6 / 100 |
| | **10** | 0 / 0 | 0 / 0 | 0 / 0 | 31.4 / 100 |
| | **Mean** | 25.0 / 0.5 | 5.0 / 0.3 | 0 / 0 | 28.9 / 99.6 |
| | **Difference** | 40.2 / 68.5 | 36.5 / 26.4 | 72.1 / 62.5 | 43.2 / 24.5 |
| **NIR-Red** | **1** | 66.7 / 3.4 | 0 / 0 | 0 / 0 | 38.5 / 98.4 |
| | **2** | 0 / 0 | 0 / 0 | 0 / 0 | 40.7 / 100 |
| | **3** | 20.0 / 1.4 | 0 / 0 | 0 / 0 | 51.9 / 96.5 |
| | **4** | 100 / 2.2 | 0 / 0 | 0 / 0 | 20.9 / 100 |
| | **5** | 100 / 1.4 | 0 / 0 | 0 / 0 | 23.7 / 100 |
| | **6** | 0 / 0 | 0 / 0 | 0 / 0 | 46.0 / 100 |
| | **7** | 0 / 0 | 0 / 0 | 0 / 0 | 22.0 / 100 |
| | **8** | 0 / 0 | 0 / 0 | 0 / 0 | 50.0 / 100 |
| | **9** | 50.0 / 1.8 | 0 / 0 | 0 / 0 | 28.6 / 97.6 |
| | **10** | 0 / 0 | 0 / 0 | 0 / 0 | 16.9 / 100 |
| | **Mean** | 33.7 / 1.0 | 0 / 0 | 0 / 0 | 33.9 / 99.2 |
| | **Difference** | 31.5 / 68.0 | 41.5 / 26.7 | 72.1 / 62.5 | 38.2 / 24.1 |

Table 7.44. Precision/Recall Obtained From Highest Performing ssEAM Classifier Trained Using Reduced Feature Vector From NIR-Gray Images (Table 7.30).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **NIR-Gray** | **Mean** | 65.8 / 76.5 | 39.0 / 31.8 | 73.4 / 75.8 | 71.0 / 60.5 |
| **Visible** | **1** | 40.1 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **2** | 29.0 / 100 | 0 / 0 | 0 / 0 | 100 / 4.9 |
| | **3** | 35.5 / 98.4 | 0 / 0 | 0 / 0 | 50. 0 / 1.1 |
| | **4** | 43.1 / 100 | 0 / 0 | 0 / 0 | 100 / 2.9 |
| | **5** | 55.3 / 100 | 0 / 0 | 0 / 0 | 100 / 4.3 |
| | **6** | 28.0 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **7** | 41.3 / 100 | 0 / 0 | 0 / 0 | 100 / 5.6 |
| | **8** | 19.4 / 96.2 | 0 / 0 | 0 / 0 | 66.7 / 2.9 |
| | **9** | 38.8 / 100 | 0 / 0 | 0 / 0 | 100 / 3.5 |
| | **10** | 46.8 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **Mean** | 37.7 / 99.5 | 0 / 0 | 0 / 0 | 61.7 / 2.5 |
| | **Difference** | 28.1 / 23.0 | 39.0 / 31.8 | 73.4 / 75.8 | 9.3 / 58.0 |
| **NIR-Red** | **1** | 35.4 / 96.6 | 0 / 0 | 0 / 0 | 33.3 / 1.6 |
| | **2** | 43.0 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **3** | 34.0 / 98.6 | 0 / 0 | 0 / 0 | 80.0 / 3.5 |
| | **4** | 41.4 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **5** | 45.9 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **6** | 32.9 / 100 | 0 / 0 | 0 / 0 | 100 / 3.7 |
| | **7** | 58.5 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **8** | 22.1 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **9** | 40.0 / 98.2 | 0 / 0 | 0 / 0 | 50.0 / 2.4 |
| | **10** | 51.6 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **Mean** | 40.4 / 99.3 | 0 / 0 | 0 / 0 | 26.3 / 1.1 |
| | **Difference** | 25.4 / 22.8 | 39.0 / 31.8 | 73.4 / 75.8 | 44.7 / 59.4 |

Table 7.45. Precision/Recall Obtained From Highest Performing ssEAM Classifier Trained Using Reduced Feature Vector From NIR-Red Images (Table 7.32).

| | Case | Person | Bicycle | Vehicle | Clutter |
|---|---|---|---|---|---|
| **NIR-Red** | **Mean** | 63.7 / 75.9 | 32.0 / 34.7 | 65.4 / 56.1 | 71.6 / 54.6 |
| **Visible** | **1** | 40.7 / 100 | 0 / 0 | 0 / 0 | 100 / 5.1 |
| | **2** | 29.0 / 100 | 0 / 0 | 0 / 0 | 100 / 4.9 |
| | **3** | 35.5 / 98.4 | 0 / 0 | 0 / 0 | 50.0 / 1.1 |
| | **4** | 43.1 / 100 | 0 / 0 | 0 / 0 | 100 / 2.9 |
| | **5** | 55.3 / 100 | 0 / 0 | 0 / 0 | 100 / 4.3 |
| | **6** | 28.0 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **7** | 41.3 / 100 | 0 / 0 | 0 / 0 | 100 / 5.6 |
| | **8** | 19.4 / 96.2 | 0 / 0 | 0 / 0 | 66.7 / 2.9 |
| | **9** | 38.8 / 100 | 0 / 0 | 0 / 0 | 100 / 3.5 |
| | **10** | 46.8 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **Mean** | 37.8 / 99.5 | 0 / 0 | 0 / 0 | 71.7 / 3.0 |
| | **Difference** | 25.9 / 23.6 | 32 / 34.7 | 65.4 / 56.1 | 0.1 / 51.6 |
| **NIR-Gray** | **1** | 47.2 / 100 | 0 / 0 | 0 / 0 | 50.0 / 2.2 |
| | **2** | 38.7 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **3** | 41.8 / 98.4 | 0 / 0 | 0 / 0 | 50.0 / 1.6 |
| | **4** | 50.4 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **5** | 60.3 / 100 | 0 / 0 | 0 / 0 | 100 / 9.1 |
| | **6** | 36.9 / 100 | 0 / 0 | 0 / 0 | 100 / 1.4 |
| | **7** | 63.0 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **8** | 27.0 / 100 | 0 / 0 | 0 / 0 | 100 / 1.9 |
| | **9** | 41.9 / 100 | 0 / 0 | 0 / 0 | 0 / 0 |
| | **10** | 50.5 / 100 | 0 / 0 | 0 / 0 | 100 / 5.0 |
| | **Mean** | 45.8 / 99.8 | 0 / 0 | 0 / 0 | 50. 0 / 2.1 |
| | **Difference** | 17. 9 / 23.9 | 32.0 / 34.7 | 65.4 / 52.5 | 21.6 / 52.5 |

## 7.6. DISCUSSION

The architectures described in Sections 7.3 and 7.4 did not have 100% accuracy at classifying the classes of persons, bicycles, vehicles and clutter. Accuracy discrepancies can be attributed to the variety of environments used and environmental noise and clutter. Accuracy discrepancies can also be attributed to imperfect object segmentation by the processing algorithm defined in Chapter 5 (as shown in Section 7.4). The experiments performed in this research were intended to explore surveillance capabilities across a spectrum of situations rather than optimizing architectures for a specific situation.

The ssEAM and MLP neural network architectures tended to perform well on both near-infrared and visible images despite the variability of the observed environments. Typically vehicle and bicycle classification performances were lower than clutter and person classification. This could be attributed to the range of views imaged of vehicles and bicycles. Vehicles and bicycles look significantly different at different angles while persons tend to look similar despite angle variations. A wider variety of person objects were imaged in the image set than bicycle or vehicles, which could also have caused accuracy differences. The ssEAM architectures appeared to evenly divide the sample space defining classification causing the precision and recall values across the four classes to be more consistent than those in the MLP neural network architecture. Both architectures appear to have the potential to classify well when optimized. Test times in both architectures were insignificant, allowing potential for real-time implementation.

Differences between red-channel and gray-scale near-infrared images were fairly minute. Reduced feature sets degraded performance in the MLP neural network

architectures, but improved performance in ssEAM architectures. As noted in previous literature review, feature count effects accuracy in ssEAM architecture, so further feature analysis would be needed to optimize this architecture performance. Train time was improved with reduced features sets, but test times remained small across the board.

Both the LDA feature analysis and data fusion experiments showed that feature importance for classification varied across the three image types. The ssEAM architectures trained on one image type did not perform well on the other two image types. The MLP neural network architectures trained performed better on the other image types than the ssEAM architectures did. Data fusion experiments show that visible images and gray-scale near-infrared images compare as they performed well at classifying the other. Experiments also show that gray-scale near-infrared images compare to red channel near-infrared images.

Overall the results show that near-infrared images have the potential to perform just as well in classification applications as visible images. Benefits of near-infrared imagery as mentioned before justify future research in near-infrared surveillance applications.

# 8. CONCLUSIONS AND FUTURE WORK

## 8.1. CONCLUSIONS

In this research, a computer vision approach was investigated for comparison of near-infrared and visible light systems. Near-infrared systems have been found to perform better than visible light systems in some situations such as smoky rooms and environments with poor illumination. Surveillance of pedestrian and vehicular traffic was the application area and the intended system was designed for low computational complexity and hardware economy. In particular, the approach was constrained to standard image processing operations and single-perspective, gray-scale images.

A collection of image sequences from seven different environments were collected from identical cameras operating in visible and near-infrared wavelengths. Both visible and near-infrared images were converted to gray-scale. The near-infrared images were also filtered to only contain red channel information. The image sequences were filtered to find variable areas that could contain person, bicycle, vehicle or clutter objects. A variety of features were calculated across the located areas. The three types of converted images were compared by using Linear Discriminant Analysis across their feature sets and by using the calculated testing accuracies to assess their utility in target detection. Two classification architectures, a MLP backpropagation neural network and a semi-supervised Ellipsoid ARTMAP (ssEAM), were trained and tested for accuracy using the calculated features. Data fusion was also performed on the three sets of data to further analyze surveillance capabilities.

Image sequences collected showed significant differences between visible and near-infrared images. Human skin was noted to have a unique reflective quality in all environments imaged in the near-infrared wavelength, including the indoor environment. It was also noted that variable environmental clutter, such as foliage, imaged differently in the near-infrared and visible domains due to specific reflective qualities of the observed material. Once converted, gray-scale visible and gray-scale near-infrared images were somewhat comparable. Red channel images looked significantly different from the gray-scale images.

Experimental results show that all three converted image types were capable of classification. Because the image sets were taken in a large variety of environments that were cluttered and highly variable, architecture optimization was not a priority; rather a general testing of the potential capabilities of each image set was the research concentration. Linear Discriminant Analysis results showed that critical features differed across the image sets. Features selected for visible and gray-scale near-infrared images compared while red channel near-infrared images significantly different. Aspect ratio was commonly selected by all image type analysis. This analysis along with data fusion analysis showed that performance from gray-scale near-infrared images and visible images were most comparable.

Results show that the MLP backpropagation neural network performed best with the original feature set of twenty-four features per object. The ssEAM architecture performed better with the reduced feature set of six features per object. Results from both architectures across the board were comparable, but ssEAM tended to collectively

classify better. Testing time for both architectures were relatively low, hence both have the potential to run in real-time.

The results show that the three data sets are all capable of reasonably performing object detection. Findings suggest that gray-scale near-infrared light systems can be used to classify persons and vehicles for surveillance with similar approaches as those used for visible light systems. Near-infrared imaging has been found in other research to perform well in environments that visible imaging cannot such as in poor lighting conditions or in smoky or foggy environments. Significant benefits could exist by applying visible imaging technologies to near-infrared surveillance applications.

## 8.2. FUTURE WORK

This experimental research proves the potential of near-infrared surveillance systems. This potential can and should be further explored to determine the capabilities of near-infrared object detection systems. A first step to exploring this potential is optimizing intelligent classification architecture for a specific environment. In this research, a variety of environments were examined, restricting the capabilities of a learning system. It would be interesting to draw comparisons between near-infrared and visible imaging systems with an optimized architecture.

Real-time processing and detection was not completed in this research. While the algorithmic processes created were intended for low computational complexity, the algorithms should be tested to determine real-time capabilities of near-infrared and visible light systems. Real-time testing could present some interesting findings,

especially when testing occurs in outdoor environments, as previous research has asserted that near-infrared performs better in certain outdoor environmental conditions.

A combined sensor system could also be examined, which may have the potential to improve current visible light surveillance applications without degrading existing capabilities. Experiments to determine a combined system's potential could include observing a classification confusion matrix to explore if one imaging sensor is best able to discriminate a particular class.

# APPENDIX A

# EXAMPLE OUTPUT IMAGES

Appendix A further describes the visible and near-infrared images captured for use in “A Comparison of Near-Infrared and Visible Image Filtering for Surveillance Applications”. Images were captured in a variety of environments including: 1) Brick Wall Scene, 2) Campus Building Scene, 3) Campus Library Scene, 4) Pedestrian Bridge Scene, 5) Indoor Hallway Scene, 6) Urban Sidewalk Scene, and 7) Urban Street Scene. A total of forty image sequences were collected from theses environments that contained a variety of person, bicycle, vehicle and clutter objects. In order to perform ten-fold cross-validation, the forty image sequences were separated into ten experiment groups. Table A.1 shows the experiment number and observed objects for each image sequence. The figures contained in this Appendix illustrate example visible and near-infrared images of the observed sequences. Figures in “Example Cases From Image Sequences” show an example case of a visible and a near-infrared image from each image sequence described in Table A.1. Figures shown in “Sample Image Sequence” show a short portion of a near-infrared and visible image sequence. The final section “Sample Image Processing Algorithm” indicates the output at each stage of the processing algorithm described in Section 5 for a gray-scale visible, gray-scale near-infrared and a red channel near-infrared image.

Table A.1. Description of Image Sequences.

| Image Sequence | Environment Type | Objects | | | | Experiment Number |
|---|---|---|---|---|---|---|
| | | Person | Bicycle | Vehicle | Clutter | |
| 1 | 1 | X | | | X | 9 |
| 2 | 1 | X | | | X | 3 |
| 3 | 1 | X | X | | X | 8 |
| 4 | 1 | X | | | X | 4 |
| 5 | 1 | X | | | X | 1 |
| 6 | 1 | X | X | | X | 5 |
| 7 | 1 | X | X | | X | 9 |
| 8 | 2 | X | X | | X | 1 |
| 9 | 2 | X | | | X | 5 |
| 10 | 2 | X | X | | X | 7 |
| 11 | 2 | X | | | X | 7 |
| 12 | 2 | X | X | | X | 3 |
| 13 | 2 | X | X | | X | 1 |
| 14 | 3 | X | | | X | 9 |
| 15 | 3 | X | X | | X | 4 |
| 16 | 3 | X | | | X | 5 |
| 17 | 3 | X | | | X | 7 |
| 18 | 3 | X | | | X | 6 |
| 19 | 3 | X | | | X | 10 |
| 20 | 3 | X | X | | X | 6 |
| 21 | 3 | X | X | | X | 10 |
| 22 | 4 | X | | | X | 8 |
| 23 | 4 | X | | | X | 6 |
| 24 | 4 | X | X | | X | 3 |
| 25 | 4 | X | X | | X | 3 |
| 26 | 4 | X | X | | X | 4 |
| 27 | 5 | X | | | X | 10 |
| 28 | 5 | X | | | X | 2 |
| 29 | 5 | X | | | X | 8 |
| 30 | 5 | X | | | X | 2 |
| 31 | 6 | X | | X | X | 1 |
| 32 | 6 | | X | X | X | 2 |
| 33 | 6 | X | | X | X | 3 |
| 34 | 6 | X | | X | X | 4 |
| 35 | 6 | | X | X | X | 5 |
| 36 | 6 | | X | X | X | 6 |
| 37 | 6 | X | | X | X | 7 |
| 38 | 6 | X | | X | X | 8 |
| 39 | 7 | | X | X | X | 9 |
| 40 | 7 | | X | X | X | 10 |

## EXAMPLE CASES FROM IMAGE SEQUENCES

Figure A.1. Example of Sequence 1. (a) Near-Infrared Image. (b) Visible Image.

Figure A.2. Example of Sequence 2. (a) Near-Infrared Image. (b) Visible Image.

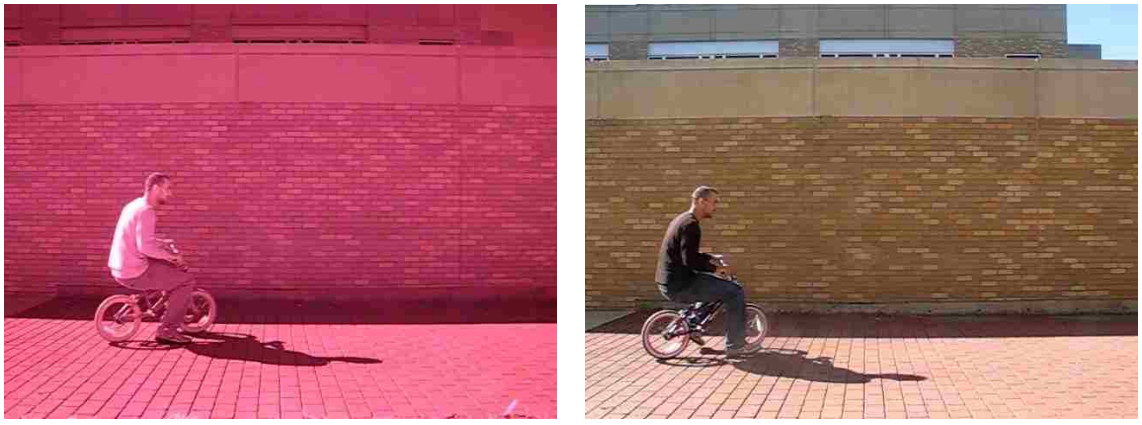

Figure A.3. Example of Sequence 3. (a) Near-Infrared Image. (b) Visible Image.

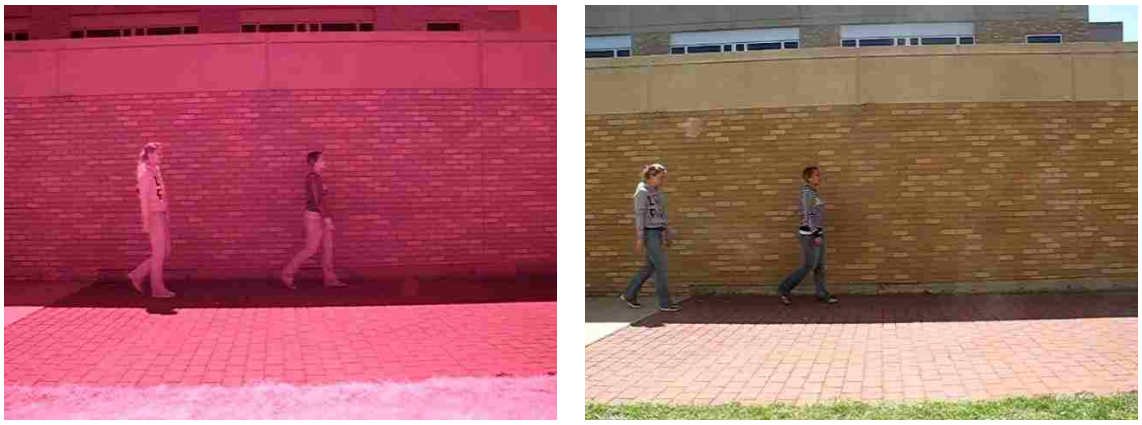

Figure A.4. Example of Sequence 4. (a) Near-Infrared Image. (b) Visible Image.

Figure A.5. Example of Sequence 5. (a) Near-Infrared Image. (b) Visible Image.

Figure A.6. Example of Sequence 6. (a) Near-Infrared Image. (b) Visible Image.

Figure A.7 Example of Sequence 7. (a) Near-Infrared Image. (b) Visible Image.

Figure A.8. Example of Sequence 8. (a) Near-Infrared Image. (b) Visible Image.

Figure A.9. Example of Sequence 9. (a) Near-Infrared Image. (b) Visible Image.

Figure A.10. Example of Sequence 10. (a) Near-Infrared Image. (b) Visible Image.

Figure A.11. Example of Sequence 11. (a) Near-Infrared Image. (b) Visible Image.

Figure A.12. Example of Sequence 12. (a) Near-Infrared Image. (b) Visible Image.

Figure A.13. Example of Sequence 13. (a) Near-Infrared Image. (b) Visible Image.

Figure A.14. Example of Sequence 14. (a) Near-Infrared Image. (b) Visible Image.

Figure A.15. Example of Sequence 15. (a) Near-Infrared Image. (b) Visible Image.

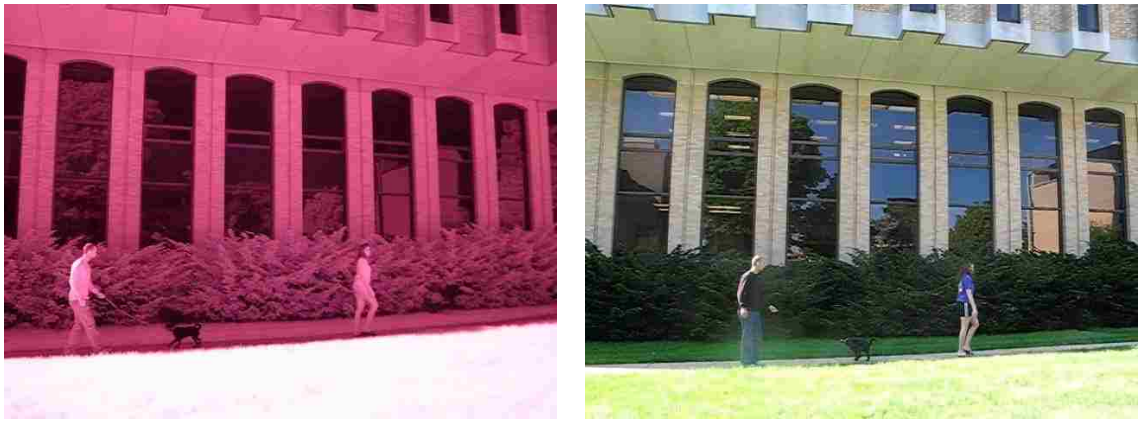

Figure A.16. Example of Sequence 16. (a) Near-Infrared Image. (b) Visible Image.

Figure A.17. Example of Sequence 17. (a) Near-Infrared Image. (b) Visible Image.

Figure A.18. Example of Sequence 18. (a) Near-Infrared Image. (b) Visible Image.

Figure A.19. Example of Sequence 19. (a) Near-Infrared Image. (b) Visible Image.

Figure A.20. Example of Sequence 20. (a) Near-Infrared Image. (b) Visible Image.

Figure A.21. Example of Sequence 21. (a) Near-Infrared Image. (b) Visible Image.

Figure A.22. Example of Sequence 22. (a) Near-Infrared Image. (b) Visible Image.

Figure A.23. Example of Sequence 23. (a) Near-Infrared Image. (b) Visible Image.

Figure A.24. Example of Sequence 24. (a) Near-Infrared Image. (b) Visible Image.

Figure A.25. Example of Sequence 25. (a) Near-Infrared Image. (b) Visible Image.

Figure A.26. Example of Sequence 26. (a) Near-Infrared Image. (b) Visible Image.

Figure A.27. Example of Sequence 27. (a) Near-Infrared Image. (b) Visible Image.

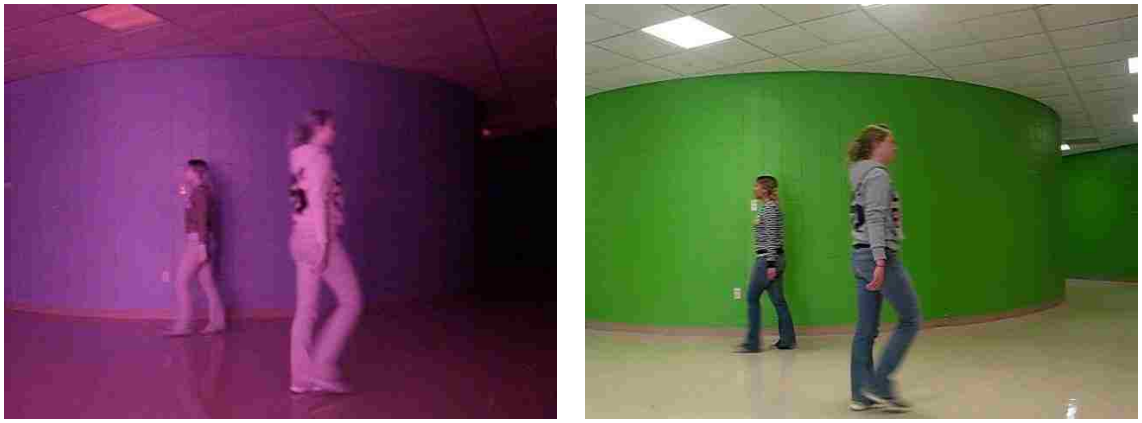

Figure A.28. Example of Sequence 28. (a) Near-Infrared Image. (b) Visible Image.

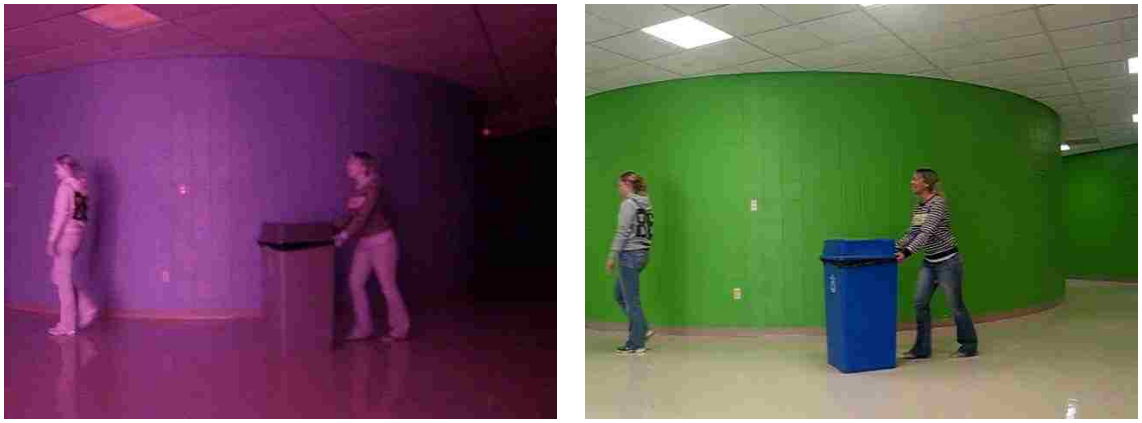

Figure A.29. Example of Sequence 29. (a) Near-Infrared Image. (b) Visible Image.

Figure A.30. Example of Sequence 30. (a) Near-Infrared Image. (b) Visible Image.

Figure A.31. Example of Sequence 31. (a) Near-Infrared Image. (b) Visible Image.

Figure A.32. Example of Sequence 32. (a) Near-Infrared Image. (b) Visible Image.

Figure A.33. Example of Sequence 33. (a) Near-Infrared Image. (b) Visible Image.

Figure A.34. Example of Sequence 34. (a) Near-Infrared Image. (b) Visible Image.

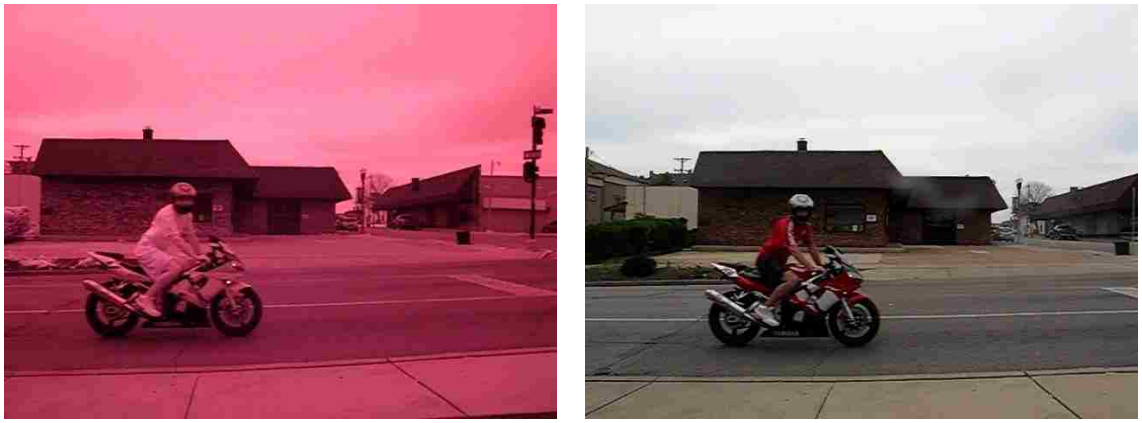

Figure A.35. Example of Sequence 35. (a) Near-Infrared Image. (b) Visible Image.

Figure A.36. Example of Sequence 36. (a) Near-Infrared Image. (b) Visible Image.

Figure A.37. Example of Sequence 37. (a) Near-Infrared Image. (b) Visible Image.

Figure A.38. Example of Sequence 38. (a) Near-Infrared Image. (b) Visible Image.

Figure A.39. Example of Sequence 39. (a) Near-Infrared Image. (b) Visible Image.

Figure A.40.  Example of Sequence 40. (a) Near-Infrared Image. (b) Visible Image.

**SAMPLE IMAGE SEQUENCE**

Figure A.41. Near-Infrared Image Sequence. (a) Reference Image. (b) Image 2.

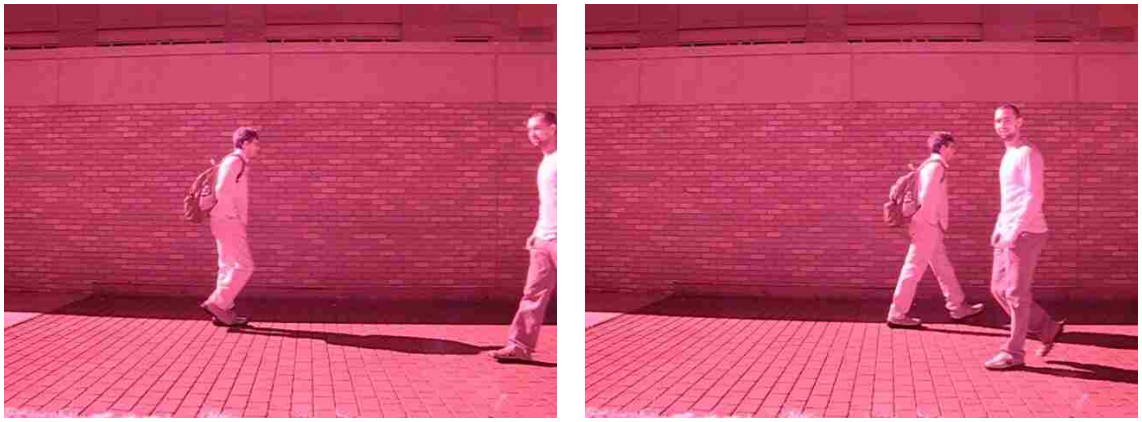

Figure A.41. Near-Infrared Image Sequence. (cont.) (c) Image 3. (d) Image 4.

Figure A.41. Near-Infrared Image Sequence. (cont.) (e) Image 5. (f) Image 6.

Figure A.42.  Visible Image Sequence. (a) Reference Image. (b) Image 2.

Figure A.42. Visible Image Sequence. (cont.)  (c) Image 3. (d) Image 4.

Figure A.42.  Visible Image Sequence. (cont.)  (e) Image 5. (f) Image 6.

## SAMPLE IMAGE PROCESSING ALGORITHM

Figure A.43. Original Visible Image. (a) Observed Image. (b) Reference Image.

Figure A.44. Gray-Scale Visible Image. (a) Observed Image. (b) Reference Image.

Figure A.45. Noise-Filtered Visible Image. (a) Observed Image. (b) Reference Image.

Figure A.46. Visible Difference Image.

Figure A.47. Visible Threshold Image.

Figure A.48. Visible Object Segmentation Image.

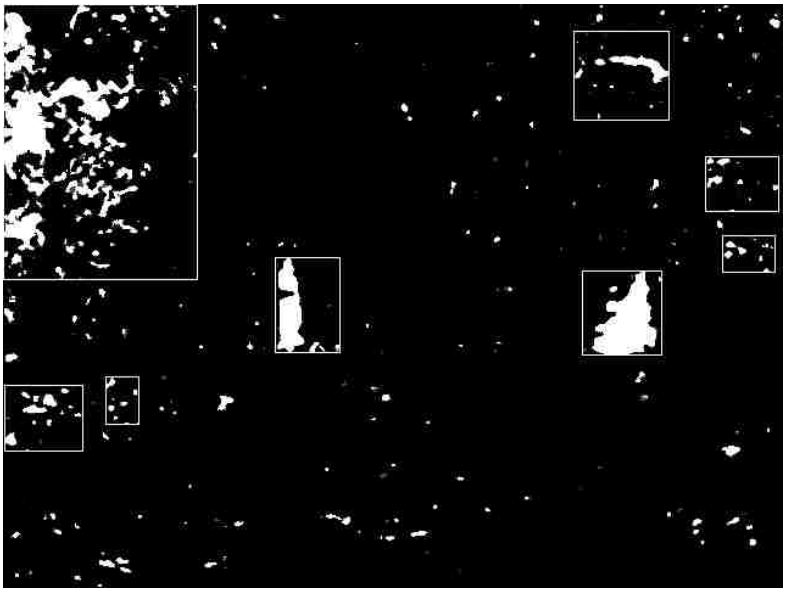

Figure A.49. Algorithm Targeted Visible Objects.

Figure A.50. Original Near-Infrared Image. (a) Observed Image. (b) Reference Image.

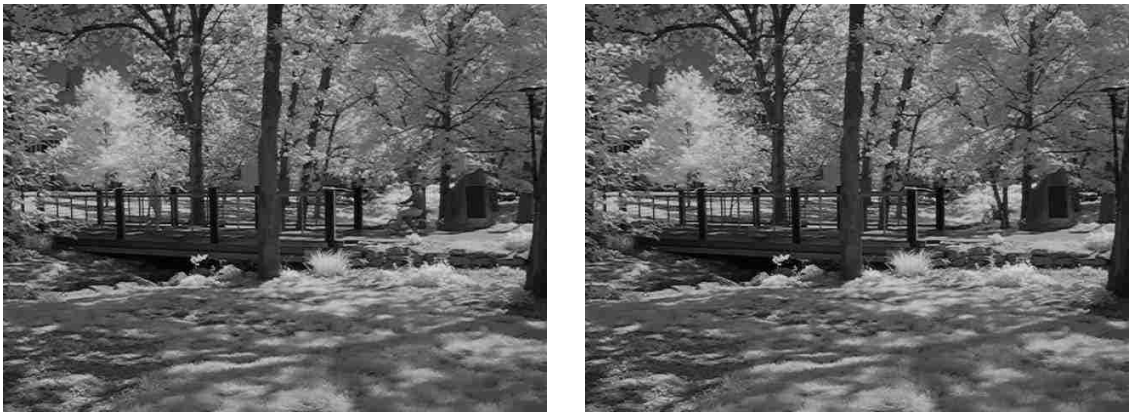

Figure A.51. Gray-Scale Near-Infrared Image. (a) Observed Image. (b) Reference Image.

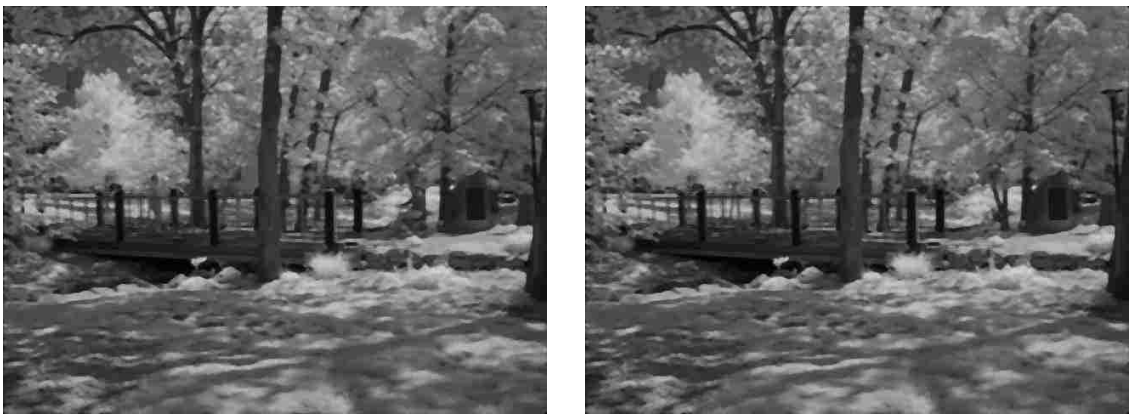

Figure A.52. Noise-Filtered Gray-Scale Near-Infrared Image. (a) Observed Image. (b) Reference Image.

Figure A.53. Gray-Scale Near-Infrared Difference Image.

Figure A.54. Gray-Scale Near-Infrared Threshold Image.

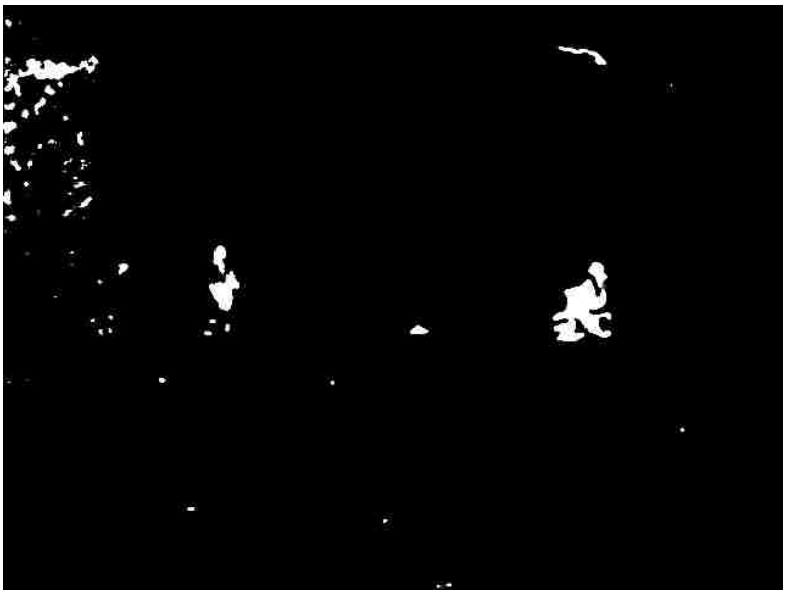

Figure A.55. Gray-Scale Near-Infrared Object Segmentation Image.

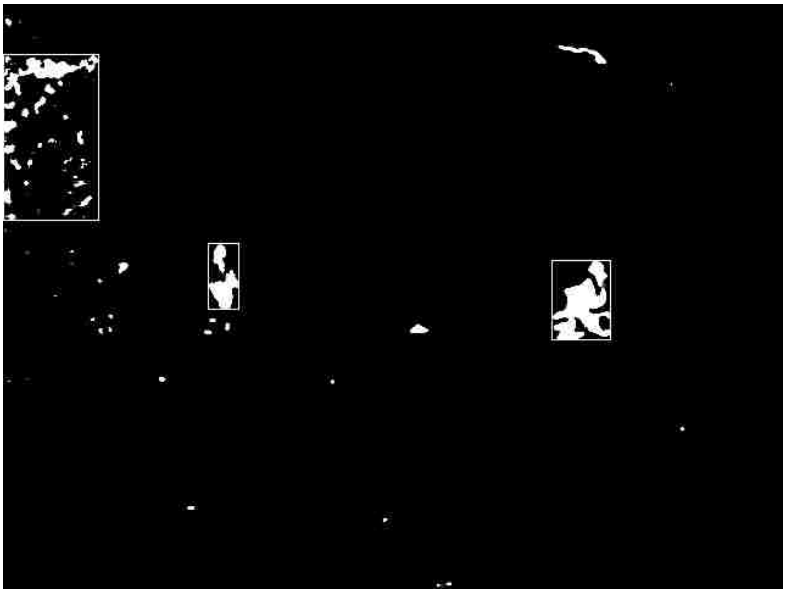

Figure A.56. Gray-Scale Near-Infrared Final Segmented Image.

Figure A.57. Original Near-Infrared Image. (a) Observed Image. (b) Reference Image.

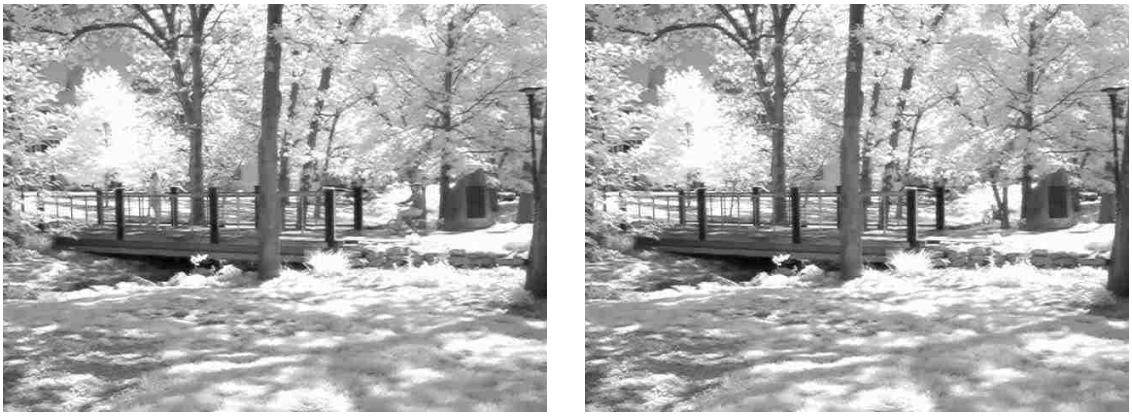

Figure A.58. Red-Channel Near-Infrared Image. (a) Observed Image. (b) Reference Image.

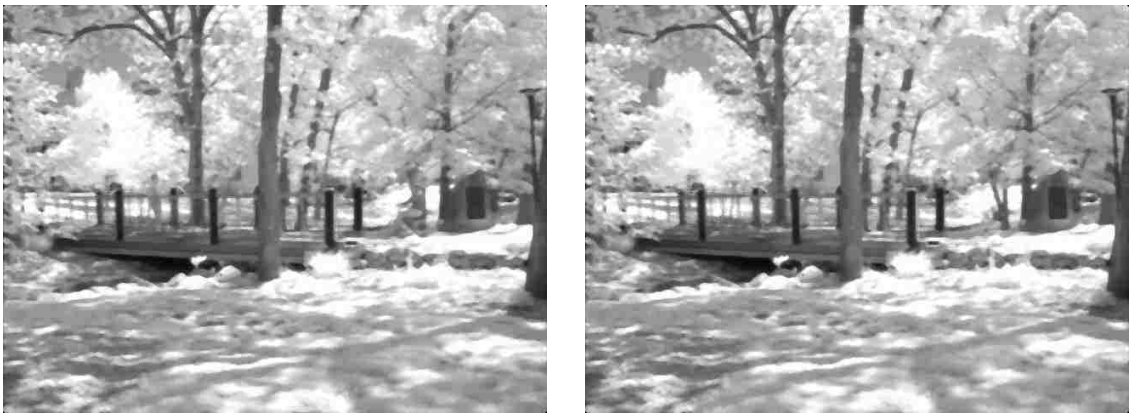

Figure A.59. Noise-Filtered Red Channel Near-Infrared Image. (a) Observed Image. (b) Reference Image.

Figure A.60. Red Channel Near-Infrared Difference Image.

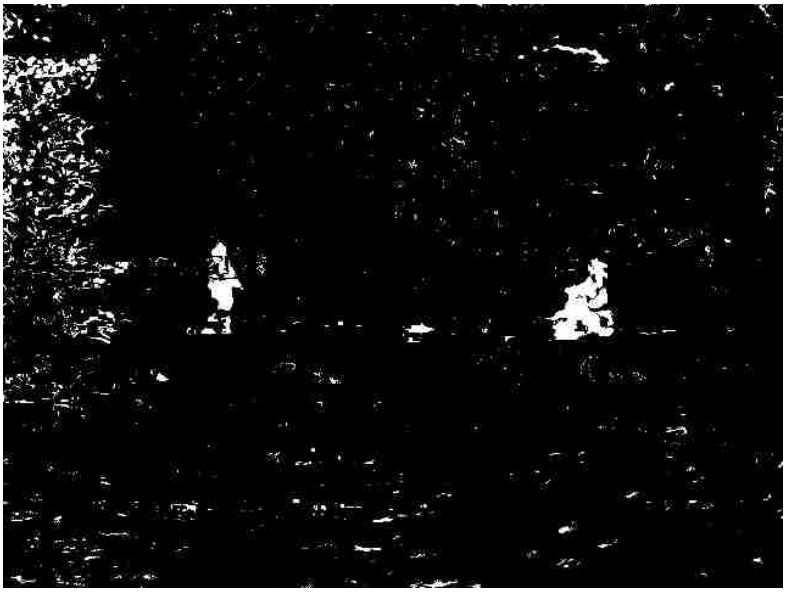

Figure A.61. Red Channel Near-Infrared Threshold Image.

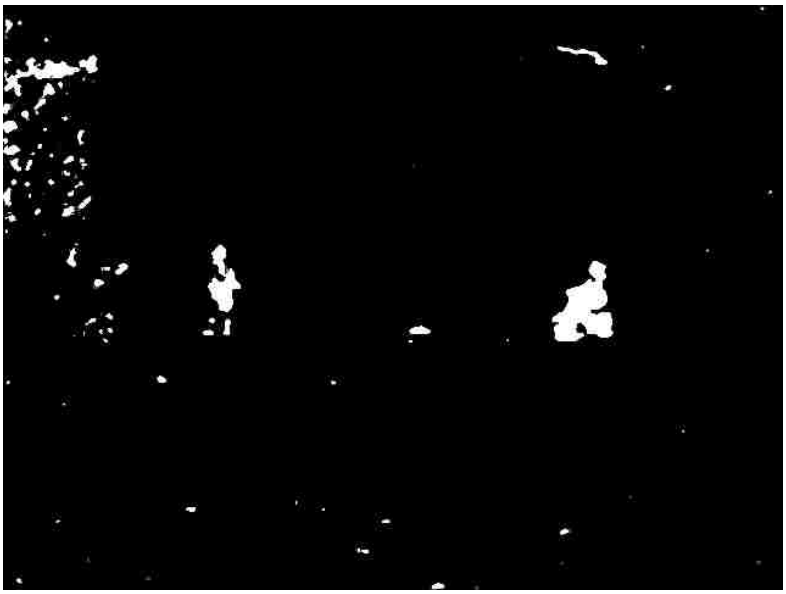

Figure A.62. Red Channel Near-Infrared Object Segmentation Image.

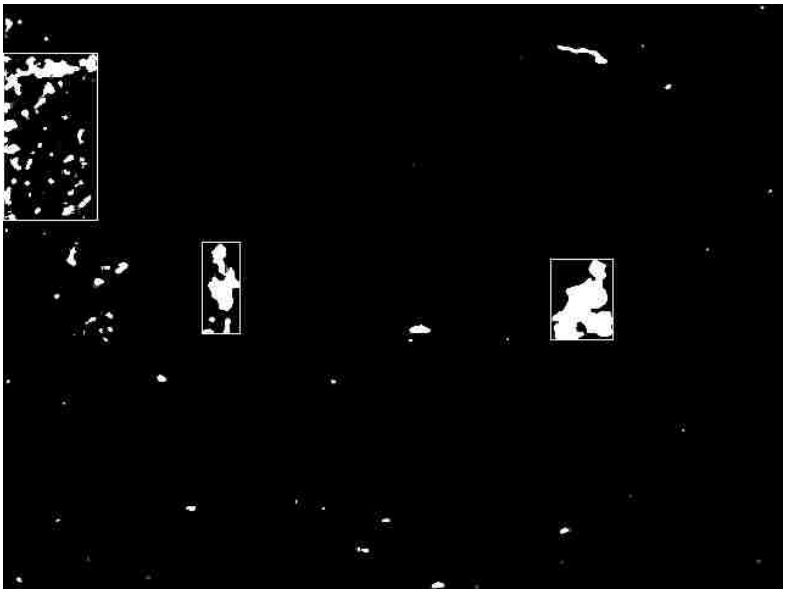

Figure A.63. Red Channel Near-Infrared Final Segmented Image.

# APPENDIX B

# ALGORITHMIC AND EXPERIMENTAL CODE

## IMAGE PROCESSING ALGORITHM OUTLINE

The image manipulation algorithm proposed for "A Comparison of Near-Infrared and Visible Image Filtering for Surveillance Applications" processes image sequences collected with visible and near-infrared light cameras. This algorithm is intended to locate valuable objects in the images and gather feature information from each object. These vectors are passed to the MLP neural network and ssEAM architectures for intelligent processing. The image manipulation algorithmic process is Figure B.1.

*Import sequence reference image*

*For every other sequence image:*

- *Import non-reference image*
- *Obtain gray-scale or targeted matrix from reference and non-reference image*
- *Perform median filtering to eliminate noise*
- *Calculate absolute difference image between reference and non-reference image*
- *Threshold difference image to eliminate insignificant differences*
- *Perform median filtering to eliminate pixels not by a large patch of similar pixels*
- *Fill holes in image*
- *Find object blobs in image*
- *Merge blobs that are close in proximity*
- *Ignore blobs that are insignificantly small*
- *Compute features for each blob*

*End For*

Figure B.1. Image Processing Algorithm Outline.

## IMAGE PROCESSING ALGORITHM CODE

The MATLAB function shown in Figure B.2 encompasses the algorithm outlined in Figure B.1. This code locates stored image sequences based on user variable input. The first image accessed in a sequence is stored as the reference image. Each of the remaining images in the sequence is processed according to the developed algorithm. The code in Figure B.1 outputs a comma separated value excel file that indicates calculated object feature vectors in a row by row basis.

The functions illustrated in Figures B.3, B.4 and B.5 handle locating important objects in non-reference-images. The code as shown in Figure B.2 locates 'blobs' by finding pixels that are connected to neighboring pixels. Bounding boxes of these neighboring pixels are stored and passed to the function shown in Figure B.3. The code in Figure B.3 merges overlapping and close bounding boxes to create a reduced set of bounding boxes indicative of interesting objects found in the image.

The MATLAB function shown in Figure B.6 receives a reduced set of object bounding boxes as an input from the code outlined in Figure B.2. This code calculates twenty-four photometric and geometric features for each bounding box, which it returns in vector form to the code in Figure B.2.

```
%% Function to process images and calculate object feature vectors
function featStats = features() %declare function name

%% Initialize needed variables
areaLimit = 750; %minimum area of blob for feature calculation
percent = 5/100; %threshold percentage
imageCount = 2; %number of images in sequence
seqNum = 2; %designate sequence number
seqLetter = 'G'; %designate sequence type
seqFolder = 'GE_G_I_2'; %designate sequence folder
```

```
row = 480; %declare size of image
col = 640; %declare size of image
allFeatStats = zeros(250, 27); %declare first row of array to hold all features
allFeatCount = 1; %count to know how many features

%% Operate on entire sequence
for curImage =1:imageCount

   %Open reference image
   bkgImg = imread(sprintf('/%c/%s/%d0000.jpeg',seqLetter, seqFolder, seqNum));

   %Open current image
   if curImage < 10
      inputImg = imread(sprintf('/%c/%s/%d000%d.jpeg',seqLetter, seqFolder,…
          seqNum,curImage));
   elseif curImage < 100
      inputImg = imread(sprintf('/%c/%s/%d00%d.jpeg',seqLetter, seqFolder,…
          seqNum,curImage));
   end

   %Convert inputted images to gray scale
   %Img = rgb2gray(inputImg);
   %BImg = rgb2gray(bkgImg);

   %Convert infrared images to only look at red image
   Img = inputImg(:,:,1);
   BImg = bkgImg(:,:,1);

   %Perform median filtering operation to reduce noise
   medImg = medfilt2(Img, [5 5]);
   medBImg = medfilt2(BImg, [5 5]);
   %imwrite(medImg,'currentimage1.tiff');
   %imwrite(medBImg,'referenceimage2.tiff');

   %Compare image to reference image
   subImg = medImg;
   for i = 1:row
      for j = 1:col
         subImg(i,j) = uint8(abs(double(medImg(i,j)) - double(medBImg(i,j))));
      end
   end
   %imwrite(subImg,'subImg.tiff');

   %Calculate histogram of difference image
   [counts, x] = imhist(subImg);
   for i = 1:256
      if x(i) < 1
         limit = i;
         break;
      end
   end
```

```
%Threshold difference image
upper = (1 - (limit/255))*(percent);
if (upper > (1/255))
   lower = upper - (1/255);
else
   lower = 0;
end
hisImg = double(subImg)/255;
thsImg = imadjust(hisImg,[lower,upper],[0, 1]);
thsImg = uint8(thsImg*255);
%imwrite(thsImg, 'thsImg.tiff');

%Median filter image to reduce noise
medImg2 = medfilt2(thsImg, [5 5]);
%imwrite(medImg2, 'medImg2.tiff');

%Fill holes in image
fillImg = imfill(medImg2,'holes');
%imwrite(fillImg, 'fillImg.tiff');

%Find bounding boxes of blobs in image
lblImg = bwconncomp(fillImg);
stats = regionprops(lblImg, 'BoundingBox');

%Merge bounding boxes that overlap
newStats = mergeBoundingBox(stats);
[M N] = size(newStats);

%Eliminate blobs that are too small for feature calculation
for k = 1:M
   if (newStats(k,2) - newStats(k,1))*(newStats(k,4)-newStats(k,3)) < areaLimit
      newStats(k,:) = 0;
   end
end

%Plot new bounding boxes on output image for reference
gg = fillImg;
for k = 1:M
   for i = newStats(k,1):newStats(k,2)
      for j = newStats(k,3):newStats(k,4)
         if newStats(k,4) ~= 0
            x1 = newStats(k,1);
            if j == 0
               j =1;
            end
            y1 = newStats(k,3);
            x2 = newStats(k,2);
            y2 = newStats(k,4);
            if i == 0
               i = 1;
            end
```

```
                if x1 == 0
                    x1 = 1;
                end
                if y1 == 0
                    y1 = 1;
                end
                gg(y1,i) = 1;
                gg(y2,i) = 1;
                gg(j,x2) = 1;
                gg(j,x1) = 1;
            end
        end
    end
    end
    file = 'BB%d.tiff';
    file = sprintf(file,curImage);
    imwrite(gg, file);

    %Complete feature calculations
    [num, featStats] = featureCalc(newStats, fillImg, curImage);

    %Write features to file
    if num ~= 0
        allFeatStats(allFeatCount:(allFeatCount+num-1),:) = featStats;
        allFeatCount = allFeatCount + num;
    end
end

csvwrite('features.xls',allFeatStats);
```

Figure B.2. Code to Manage Object Detection and Feature Calculation.

```
%% Function to merge related bounding boxes
function newStats = mergeBoundingBox(stats)

%Initialize variables
restart = 1; %designate when restart is needed
exRate = 3; %extension rate of boxes
[M N] = size(stats);
number = M; %current blob count
boundingBoxes = zeros(M,4); %holds vounding box locations

%Store bounding box values
for k = 1:M
    boundingBoxes(k,1) = floor(stats(k).BoundingBox(1));
    boundingBoxes(k,2) = floor(stats(k).BoundingBox(1) + stats(k).BoundingBox(3));
```

```
   boundingBoxes(k,3) = floor(stats(k).BoundingBox(2));
   boundingBoxes(k,4) = floor(stats(k).BoundingBox(2) + stats(k).BoundingBox(4));
end

%Join close bounding boxes
while (restart == 1)
   restart = 0;
   for k = 1:number
      for j = (k+1):number
        [exBox1 exBox2] = exBoundingBox(boundingBoxes(k,:), boundingBoxes(j,:), exRate);
        [restart newBox] = ckBoundingBox(exBox1, exBox2, boundingBoxes(k,:),…
                boundingBoxes(j,:) );
        if (restart == 1)
           boundingBoxes(k,:) = newBox;
           boundingBoxes(j,:) = boundingBoxes(number,:);
           number = number - 1;
           break;
         end
       end
       if restart == 1
          break;
       end
   end
end

%Return new bounding boxes
newStats = boundingBoxes(1:number,:);
```

Figure B.3. Function to Merge Bounding Boxes.

```
%% Extend bounding box
function [oldBox1 oldBox2] = exBoundingBox(oldBox1, oldBox2, rate)

%Extend bounding box by rate
oldBox1(1,1) = oldBox1(1,1) - rate;
oldBox2(1,1) = oldBox2(1,1) - rate;
oldBox1(1,3) = oldBox1(1,3) - rate;
oldBox2(1,3) = oldBox2(1,3) - rate;
oldBox1(1,2) = oldBox1(1,2) + rate;
oldBox2(1,2) = oldBox2(1,2) + rate;
oldBox1(1,4) = oldBox1(1,4) + rate;
oldBox2(1,4) = oldBox2(1,4) + rate;
```

Figure B.4. Function to Extend Boundaries of Object Boxes.

```
%% Function to determine if bounding boxes overlap
function [restart newBox] = ckBoundingBox(exBox1, exBox2, oldBox1, oldBox2)

%Initialize variable
temp1 = oldBox1; %store old box information
extemp1 = exBox1; %store old box information
newBox = zeros(1,4); %hold new box informaton
restart = 0; %indicate if new box was found

%Determine overlap and merge boxes
for k = 1:2
   if k == 2
      oldBox1 = oldBox2;
      oldBox2 = temp1;
      exBox1 = exBox2;
      exBox2 = extemp1;
   end
   if((((exBox1(1,1) > exBox2(1,1)) && (exBox1(1,1) < exBox2(1,2))) || ((exBox1(1,2)…
               > exBox2(1,1)) && (exBox1(1,2) < exBox2(1,2))))...
         && (((exBox1(1,3) > exBox2(1,3)) && (exBox1(1,3) < exBox2(1,4))) || ((exBox1(1,4…
               > exBox2(1,3)) && (exBox1(1,4) < exBox2(1,4)))))
         restart = 1;
         if(oldBox1(1,1) < oldBox2(1,1))
            newBox(1,1) = oldBox1(1,1);
         else
            newBox(1,1) = oldBox2(1,1);
         end
         if(oldBox1(1,2) > oldBox2(1,2))
            newBox(1,2) = oldBox1(1,2);
         else
            newBox(1,2) = oldBox2(1,2);
         end
         if(oldBox1(1,3) < oldBox2(1,3))
            newBox(1,3) = oldBox1(1,3);
         else
            newBox(1,3) = oldBox2(1,3);
         end
         if(oldBox1(1,4) > oldBox2(1,4))
            newBox(1,4) = oldBox1(1,4);
         else
            newBox(1,4) = oldBox2(1,4);
         end
         k = 2;
      end
   end
end
```

Figure B.5. Function to Check for Bounding Box Overlap.

```
%% Function to calculate 24 features on object bounding boxes
function [num, featStats] = featureCalc(stats, image, frame)

%Initialize variables
[M,N] = size(stats);
count = 0;

%Calculate features for all bounding boxes
for k = 1:M
   if stats(k,4) ~= 0
      count = count + 1;
      iLim = stats(k,2) - stats(k,1);
      jLim = stats(k,4) - stats(k,3);
      for  j= 1:(jLim);
         for i = 1:(iLim);
            h(j,i) = image((stats(k,3)+ (j)),(stats(k,1)+ (i)));
         end
      end
      imwrite(h,'tempImg.tiff');

      %Feature calculations
      stat = regionprops(h,'All');
      dh = im2double(h*255);

      %Frame number
      f1 = frame;

      %X start location
      f2 = stats(k,1);

      %Y start location
      f3 = stats(k,3);

      %Height
      f4 = jLim;

      %Width
      f5 = iLim;

      %Aspect ratio
      f6 = f4/f5;

      %Area
      f7 = stat.Area;

      %Perimeter
      f8 = stat.Perimeter;

      %Convex hull area
      f9 = stat.ConvexArea;
```

```
%Solidity
f10 = f7/f9;

%Compactness
f11 = f7/(f8*f8);

%Horizontal Centroid Offset
f12 = stat.Centroid(1,1);

%Vertical Centroid Offset
f13  = stat.Centroid(1,2);

%Euler Number
f14 = bweuler(h);

%Skewness
tempf15 = skewness(dh);
i = ~isnan(tempf15);
tempf15 = tempf15(i);
f15 = svd(tempf15);

%Kurtosis
tempf16 = kurtosis(dh);
i = ~isnan(tempf16);
tempf16 = tempf16(i);
f16 = svd(tempf16);

%2nd Order Moment
tempf17 = moment(dh,2);
f17 = svd(tempf17);

%3rd Order Moment
tempf18 = moment(dh,3);
f18 = svd(tempf18);

%4th Order Moment
tempf19 = moment(dh,4);
f19 = svd(tempf19);

%Ellipse Major Axis Length
f20 = stat.MajorAxisLength;

%Ellipse Minor Axis Length
f21 = stat.MinorAxisLength;

%Ellipse Eccentricity
f22 = stat.Eccentricity;

%Ellipse Orientation
f23 = stat.Orientation;
```

```
      %Calculate Hu Moments
      huMom = calcHuMoment(h);

      %1st Hu Moment
      f24 = huMom(1);

      %2nd Hu Moment
      f25 =huMom(2);

      %3rd Hu Moment
      f26 =huMom(3);

      %4th Hu Moment
      f27 =huMom(4);

      %Store feature calculations
      featStats(count,:) = [f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16,...
        f17 , f18, f19, f20, f21, f22, f23, f24, f25, f26, f27];
      delete('tempImg.tiff');
   end
end

num = count;
if num == 0
   featStats = 0;
end
```

Figure B.6. Function to Calculate Object Features.

## INTELLIGENT PROCESSING ALGORITHM OUTLINE

Two intelligent processing architectures were proposed in this research to classify persons, bicycles, vehicles and clutter. Calculated features from an image processing algorithm are inputted to these architectures. The intelligent architectures use the vectors to train themselves in supervised and semi-supervised methods in order to develop a classification system. Accuracies of developed systems are characterized by precision and recall percentages. The basic intelligent processing algorithmic process is shown Figure B.7.

```
Import feature vectors
Divide feature vector into train set and test set
For all architecture variations to be tested:
        Set architecture parameters
        Train architecture with train set
        Test architecture on train set
        Test architecture on test set
        Calculate recall percentage of train set test
        Calculate recall percentage of test set test
        Calculate precision percentage of test set test
End For
```

Figure B.7. Intelligent Processing Algorithm Outline.

**MLP NEURAL NETWORK EXPERIMENTAL CODE**

The two MATLAB functions featured in this section train and test a MLP neural network with the feature vectors previously calculated. Prior to using this code, the feature vectors were divided into ten equal sized groups to enable ten-fold cross-validation.

The code shown in Figure B.7 accesses the stored feature vector groups and sorts them into a train and test set. This code then passes a variety of learning rate and momentum parameters as well as the train and test sets to the function shown in Figure B.8. The function in Figure B.8 trains a backpropagation neural network and returns the classified labels to the function in Figure B.8. At the end of the code in Figure B.8,

precision and recall of the test group are calculated. At code completion, the following outputs are stored in a comma separated value file: 1) input parameters (learning rate and momentum), 2) recall percentage of testing the train set, 3) recall percentage of testing the test set, 4) precision percentage of testing the test set, 5) total train time, 6) total test time of test set.

```
%% Function to organize and initiate variations of training neural network
function [results] = main(imageType, testNumber)

%% Initialize needed variables
if imageType == 1
   dataFolder = 'Visible'; %folder holding feature data
   totNumber = 1489; %total number of features in the sequence set
   expNumbers = [137,188,174,124,104,132,159,132,149,190]; %feature count per experiment
elseif imageType == 2
   dataFolder = 'Gray'; %folder holding feature data
   totNumber = 1458; %total number of features in the sequence set
   expNumbers = [161,150,148,117,127,169,108,127,160,191]; %feature count per experiment
elseif imageType == 3
   dataFolder = 'Red'; %folder holding feature data
   totNumber = 1448; %total number of features in the sequence set
   expNumbers = [164,108,217,111,157,176,118,131,142,124]; %feature count per experiment
else
   return;
end

trainNumber = totNumber - expNumbers(testNumber); %number of features in training set

trainClassNum = zeros(trainNumber,1);  %designate array to hold training class information
trainFeatures = zeros(trainNumber,24); %designate array to hold training feature information
trainIndex = 1;

index = 0; %index designating neural network experiment
varCount = 10*9; %designating the number of variations of trained neural networks
results = zeros(varCount,27); %create arary to hold results

%% Read in feature vectors
for i = 1:10
   i
   if i == testNumber
      testClassNum = dlmread(sprintf('/%s/%d.csv',dataFolder,testNumber),',',…
            sprintf('B1..B%d',expNumbers(i)));
```

```
      testFeatures = dlmread(sprintf(/%s/%d.csv',dataFolder, testNumber),',',…
            sprintf('F1..AC%d',expNumbers(i)));
   else
      trainClassNum(trainIndex:trainIndex+expNumbers(i)-1,:) = dlmread(sprintf…
            ('/%s/%d.csv',…dataFolder, i),',',…sprintf('B1..B%d',expNumbers(i)));
      trainFeatures(trainIndex:trainIndex+expNumbers(i)-1,:) = dlmread(sprintf…
            (‘/%s/%d.csv',dataFolder, i),',',sprintf('F1..AC%d',expNumbers(i)));
      trainIndex = trainIndex + expNumbers(i);
   end
end

%% Define array for class vectors
trainClass = zeros(4,trainNumber);
testClass = zeros(4,expNumbers(testNumber));
for i = 1:trainNumber
   temp = trainClassNum(i,1);
   trainClass(temp,i) = 1;
end
for i = 1:expNumbers(testNumber)
   temp = testClassNum(i,1);
   testClass(temp,i) = 1;
end

%% Train and Test Neural Network
%for cc = 1:5  %vary the epochs from 10 to 30 in steps of 5
   for cc2 = 1:10 %vary the learning rate from 0.04 to 0.16 in steps of 0.04
      for cc3 = 1:9 %vary the momentum from 0.76 to 0.88 in steps of 0.04

         index = index + 1 %specify the index of the experiment
         learnRate = 0.04 * cc2; %calculate learning rate
         momRate = 0.60 + (0.04 * cc3); %calculate momentum
         %epoch = cc * 5 + 5; %calculate epochs
          epoch = 30;
         [netOutput, trainTime, testTime1, testTime2] = netTrain(epoch, learnRate, momRate,…
               trainClass, trainFeatures, testClass, testFeatures);  %train and test neural nets

         %calculate training accuracy
         trainCorrect = zeros(4,2);
         for k = 1:trainNumber
            [x,temp] = max(netOutput(:,k));
            [x,temp2] = max(trainClass(:,k));
            if temp == temp2
               if temp == 1
                  trainCorrect(1,1) = trainCorrect(1,1) + 1;
               elseif temp == 2
                  trainCorrect(2,1) = trainCorrect(2,1) + 1;
               elseif temp == 3
                  trainCorrect(3,1) = trainCorrect(3,1) + 1;
               elseif temp == 4
                  trainCorrect(4,1) = trainCorrect(4,1) + 1;
               end
```

```
    else
        if temp2 == 1
            trainCorrect(1,2) = trainCorrect(1,2) + 1;
        elseif temp2 == 2
            trainCorrect(2,2) = trainCorrect(2,2) + 1;
        elseif temp2 == 3
            trainCorrect(3,2) = trainCorrect(3,2) + 1;
        elseif temp2 == 4
            trainCorrect(4,2) = trainCorrect(4,2) + 1;
        end
    end
end
ptrainPercent = (trainCorrect(1,1)/(trainCorrect(1,1) + trainCorrect(1,2)))*100;
btrainPercent = (trainCorrect(2,1)/(trainCorrect(2,1) + trainCorrect(2,2)))*100;
vtrainPercent = (trainCorrect(3,1)/(trainCorrect(3,1) + trainCorrect(3,2)))*100;
ctrainPercent = (trainCorrect(4,1)/(trainCorrect(4,1) + trainCorrect(4,2)))*100;

%calculate testing accuracy
testCorrect = zeros(4,3);
for k = 1:expNumbers(testNumber)
    [x, temp] = max(netOutput(:,k+trainNumber));
    [x, temp2] = max(testClass(:,k));
    if temp == temp2
        if temp == 1
            testCorrect(1,1) = testCorrect(1,1) + 1;
        elseif temp == 2
            testCorrect(2,1) = testCorrect(2,1) + 1;
        elseif temp == 3
            testCorrect(3,1) = testCorrect(3,1) + 1;
        elseif temp == 4
            testCorrect(4,1) = testCorrect(4,1) + 1;
        end
    else
        if temp2 == 1 %calculate false negative classifications
            testCorrect(1,2) = testCorrect(1,2) + 1;
        elseif temp2 == 2
            testCorrect(2,2) = testCorrect(2,2) + 1;
        elseif temp2 == 3
            testCorrect(3,2) = testCorrect(3,2) + 1;
        elseif temp2 == 4
            testCorrect(4,2) = testCorrect(4,2) + 1;
        end
        if temp == 1 %calculate false positive classifications
            testCorrect(1,3) = testCorrect(1,3) + 1;
        elseif temp == 2
            testCorrect(2,3) = testCorrect(2,3) + 1;
        elseif temp == 3
            testCorrect(3,3) = testCorrect(3,3) + 1;
        elseif temp == 4
            testCorrect(4,3) = testCorrect(4,3) + 1;
        end
```

```
            end
        end

        ptestRecall = (testCorrect(1,1)/(testCorrect(1,1) + testCorrect(1,2)))*100;
        btestRecall = (testCorrect(2,1)/(testCorrect(2,1) + testCorrect(2,2)))*100;
        vtestRecall = (testCorrect(3,1)/(testCorrect(3,1) + testCorrect(3,2)))*100;
        ctestRecall = (testCorrect(4,1)/(testCorrect(4,1) + testCorrect(4,2)))*100;
        ptestPrecision = (testCorrect(1,1)/(testCorrect(1,1) + testCorrect(1,3)))*100;
        btestPrecision = (testCorrect(2,1)/(testCorrect(2,1) + testCorrect(2,3)))*100;
        vtestPrecision = (testCorrect(3,1)/(testCorrect(3,1) + testCorrect(3,3)))*100;
        ctestPrecision = (testCorrect(4,1)/(testCorrect(4,1) + testCorrect(4,3)))*100;

        trainAvg = (ptrainPercent + btrainPercent + vtrainPercent + ctrainPercent)/4;
        testAvg = (ptestRecall + btestRecall + vtestRecall + ctestRecall)/4;
        allAvg = (trainAvg + testAvg)/2;
        results(index,:) = [0, epoch, learnRate, momRate, 1, ptrainPercent, btrainPercent,…
                vtrainPercent, ctrainPercent,2, ptestRecall, btestRecall, vtestRecall, ctestRecall,…
               3, ptestPrecision, btestPrecision, vtestPrecision, ctestPrecision, 4, trainTime,…
               testTime1, testTime2, 5, trainAvg, testAvg, allAvg ];
    end
  end
%end

csvwrite(sprintf('%s/NN/All/%d.csv',dataFolder,testNumber),results);
```

Figure B.8. Code to Manage Testing MLP Neural Networks.

```
%% Function to train and test neural networks
function [netOutput, trainTime, testTime1, testTime2] = netTrain(epoch, learnRate, momRate,…
      trainClass, trainFeatures, testClass, testFeatures)

%% Initialize needed variables
outputCount = 4;  %number of output targets
[rowTrain,colTrain] = size(trainClass);
[rowTest, colTest] = size(testClass);
totalNum = colTrain + colTest; %calculate total number of features
featureCount = 24;  %number of features fed into network
netOutput = zeros(4,totalNum); %initialize variable to hold results


%% Normalize and format training data
trainFeatures = trainFeatures';  %transpose train feature set
trainFeatures = mapminmax(trainFeatures,-1,1);
```

```
%% Train Neural Network
net = newff(trainFeatures,trainClass,[featureCount,outputCount],{'logsig', 'logsig'}, 'trainlm');
net.trainParam.epochs = epoch;  %set max number of epochs
net.trainParam.lr = learnRate; %set learning rate
net.trainParam.mc = momRate;%set momentum
net.trainParam.goal = 0.001; %set error goal
tStart=tic;
net = train(net,trainFeatures,trainClass);  %train network
trainTime=toc(tStart); %calculate train time

%% Get and record training data results
tStart=tic;
trainResult = sim(net,trainFeatures); %test train set
testTime1=toc(tStart); %calculate test train set time
netOutput(:,1:colTrain) = trainResult;

%% Normalize and format testing data
testFeatures = testFeatures';  %transpose test feature set
testFeatures = mapminmax(testFeatures,-1,1);

%% Get and record testing data results
tStart=tic;
testResult = sim(net,testFeatures); %test test set
testTime2=toc(tStart); %calculate test test set time
netOutput(:,colTrain+1:colTrain+colTest) = testResult;
```

Figure B.9. Function to Train and Test MLP Neural Network.

**ssEAM ARCHITECTURE EXPERIMENTAL CODE**

The two MATLAB functions featured in this section train and test a semi-supervised Ellipsoid ARTMAP (ssEAM) architecture with the feature vectors collected previously. Prior to using this code, the feature vectors were divided into ten equal sized groups to enable ten-fold cross-validation.

The code shown in Figure B.10 accesses the stored feature vector groups and sorts them into a train and test set. This code then passes the function shown in Figure B.11 a variety of input parameters such as eccentricity limits of hyper-ellipsoids (mu), prediction error tolerance parameter (tolerance) and a baseline vigilance parameter

(vigilance). Because ssEAM architectures performance is dependent on the order in which it reads input vectors, a variety of input vector orders are also used as an input parameter to the function in Figure B.11.

The function in Figure B.11 uses the input parameters and feature vectors contained in the train set to train the ssEAM architecture. Both the train and test sets are tested on the developed architecture and classified labels are returned to the main function featured in Figure B.10. At code completion (Figure B.10), the following outputs are stored in a comma separated value file: 1) input parameters (mu, tolerance and vigilance), 2) recall percentage of testing the train set, 3) recall percentage of testing the test set, 4) precision percentage of testing the test set, 5) total train time, 6) total test time of test set.

```
%% Function to organize and initiate variations of training ArtMAP
function [results] = main2(imageType, testNumber)

%% Initialize needed variables
if imageType == 1
   dataFolder = 'Visible'; %folder holding feature data
   totNumber = 1489; %total number of features in the sequence set
   expNumbers = [137,188,174,124,104,132,159,132,149,190]; %feature count per experiment
elseif imageType == 2
   dataFolder = 'Gray'; %folder holding feature data
   totNumber = 1448; %total number of features in the sequence set
   expNumbers = [164,108,217,111,157,176,118,131,142,124]; %feature count per experiment
elseif imageType == 3
   dataFolder = 'Red'; %folder holding feature data
   totNumber = 1458; %total number of features in the sequence set
   expNumbers = [161,150,148,117,127,169,108,127,160,191]; %feature count per experiment
else
   return;
end

trainNumber = totNumber - expNumbers(testNumber); %number of features in training set

orders = zeros(100,trainNumber); %variable to hold variety of feature orders
```

```
for j = 1:100
        orders(j,:) = randperm(trainNumber);
end

trainClass = zeros(trainNumber,1);  %designate array to hold training class information
trainFeatures = zeros(trainNumber,24); %designate array to hold training feature information
ordTrainClass = zeros(trainNumber,1); %designate array to hold ordered training classes
ordTrainFeatures = zeros(trainNumber,24); %designate array to hold ordered training features
trainIndex = 1;

index = 0; %index designating neural network experiment
varCount = 100*5*5*6; %designating the number of variations of trained ssEAMs
results = zeros(varCount,27); %create arary to hold results

%% Read in feature vectors
for i = 1:10
   if i == testNumber
      testClass = dlmread(sprintf('/%s/%d.csv',dataFolder,testNumber),',',…
            sprintf('B1..B%d',expNumbers(i)));
      testFeatures = dlmread(sprintf('/%s/%d.csv',dataFolder,testNumber),',',…
            sprintf('F1..AC%d',expNumbers(i)));
   else
      trainClass(trainIndex:trainIndex+expNumbers(i)-1,:) = dlmread(sprintf…
            ('/%s/%d.csv',dataFolder, i),',',sprintf('B1..B%d',expNumbers(i)));
      trainFeatures(trainIndex:trainIndex+expNumbers(i)-1,:) = dlmread(sprintf…
            ('/%s/%d.csv',dataFolder, i),',',sprintf('F1..AC%d',expNumbers(i)));
      trainIndex = trainIndex + expNumbers(i);
   end
end

%% Train and Test ArtMAP
for cc = 1:50 %vary the order
   for cc2 = 1:5 %vary the ellipsoid axis length from 0.2 to 1 in steps of 0.2
      for cc3 = 1:5 %vary the tolerance from 0.1 to 0.9 in steps of 0.2
         for cc4 = 1:6 %vary vigilance size

            vigilance = (cc4-1)*.2; %calculate vigilance
            index = index + 1 %specify the index of the experiment
            learnRate = 1; %calculate learning rate
            tol = (cc3 - 1) * 0.2 + 0.1; %calculate tolerance
            mu = 0.2*cc2; %calculate mu
            alpha = .001;

            for i = 1:trainNumber
               ordTrainFeatures(i,:) = trainFeatures(orders(cc,i),:);
               ordTrainClass(i,:) = trainClass(orders(cc,i),:);
            end

            %normalize and format testing data
            trainFeatures = mapminmax(trainFeatures,0,1000);
            testFeatures = mapminmax(testFeatures,0,1000);
```

```
epoch = 5;
[netOutput, trainTime, testTime1, testTime2] = eamTrain(vigilance, epoch,…
      learnRate,mu, tol, alpha, ordTrainClass, ordTrainFeatures, testClass, testFeatures);
      %train and test neural nets

%calculate training accuracy
trainCorrect = zeros(4,2);
for k = 1:trainNumber
   temp = netOutput(k,1);
   temp2 = trainClass(k,1);
   if temp == temp2
      if temp == 1
         trainCorrect(1,1) = trainCorrect(1,1) + 1;
      elseif temp == 2
         trainCorrect(2,1) = trainCorrect(2,1) + 1;
      elseif temp == 3
         trainCorrect(3,1) = trainCorrect(3,1) + 1;
      elseif temp == 4
         trainCorrect(4,1) = trainCorrect(4,1) + 1;
      end
   else
      if temp2 == 1
         trainCorrect(1,2) = trainCorrect(1,2) + 1;
      elseif temp2 == 2
        trainCorrect(2,2) = trainCorrect(2,2) + 1;
      elseif temp2 == 3
         trainCorrect(3,2) = trainCorrect(3,2) + 1;
      elseif temp2 == 4
         trainCorrect(4,2) = trainCorrect(4,2) + 1;
      end
   end
end
ptrainPercent = (trainCorrect(1,1)/(trainCorrect(1,1) + trainCorrect(1,2)))*100;
btrainPercent = (trainCorrect(2,1)/(trainCorrect(2,1) + trainCorrect(2,2)))*100;
vtrainPercent = (trainCorrect(3,1)/(trainCorrect(3,1) + trainCorrect(3,2)))*100;
ctrainPercent = (trainCorrect(4,1)/(trainCorrect(4,1) + trainCorrect(4,2)))*100;

%calculate testing accuracy
testCorrect = zeros(4,3);
for k = 1:expNumbers(testNumber)
   temp = netOutput(k+trainNumber,1);
   temp2 = testClass(k,1);
   if temp == temp2 %calculate true positive classifications
      if temp == 1
         testCorrect(1,1) = testCorrect(1,1) + 1;
      elseif temp == 2
         testCorrect(2,1) = testCorrect(2,1) + 1;
      elseif temp == 3
         testCorrect(3,1) = testCorrect(3,1) + 1;
      elseif temp == 4
         testCorrect(4,1) = testCorrect(4,1) + 1;
```

```
                end
            else

                if temp2 == 1 %calculate false negative classifications
                    testCorrect(1,2) = testCorrect(1,2) + 1;
                elseif temp2 == 2
                    testCorrect(2,2) = testCorrect(2,2) + 1;
                elseif temp2 == 3
                    testCorrect(3,2) = testCorrect(3,2) + 1;
                elseif temp2 == 4
                    testCorrect(4,2) = testCorrect(4,2) + 1;
                end

                if temp == 1 %calculate false positive classifications
                    testCorrect(1,3) = testCorrect(1,3) + 1;
                elseif temp == 2
                    testCorrect(2,3) = testCorrect(2,3) + 1;
                elseif temp == 3
                    testCorrect(3,3) = testCorrect(3,3) + 1;
                elseif temp == 4
                    testCorrect(4,3) = testCorrect(4,3) + 1;
                end
            end
        end

        ptestRecall = (testCorrect(1,1)/(testCorrect(1,1) + testCorrect(1,2)))*100;
        btestRecall = (testCorrect(2,1)/(testCorrect(2,1) + testCorrect(2,2)))*100;
        vtestRecall = (testCorrect(3,1)/(testCorrect(3,1) + testCorrect(3,2)))*100;
        ctestRecall = (testCorrect(4,1)/(testCorrect(4,1) + testCorrect(4,2)))*100;

        ptestPrecision = (testCorrect(1,1)/(testCorrect(1,1) + testCorrect(1,3)))*100;
        btestPrecision = (testCorrect(2,1)/(testCorrect(2,1) + testCorrect(2,3)))*100;
        vtestPrecision = (testCorrect(3,1)/(testCorrect(3,1) + testCorrect(3,3)))*100;
        ctestPrecision = (testCorrect(4,1)/(testCorrect(4,1) + testCorrect(4,3)))*100;

        trainAvg = (ptrainPercent + btrainPercent + vtrainPercent + ctrainPercent)/4;
        testAvg = (ptestRecall + btestRecall + vtestRecall + ctestRecall)/4;
        allAvg = (trainAvg + testAvg)/2;

        results(index,:) = [0, mu, tol, vigilance, 1, ptrainPercent, btrainPercent, vtrainPercent,…
              ctrainPercent,2,ptestRecall, btestRecall, vtestRecall, ctestRecall, 3, ptestPrecision,…
               btestPrecision, vtestPrecision, ctestPrecision, 4, trainTime, testTime1, testTime2,…
              5,trainAvg, testAvg, allAvg];
        end
    end
  end
end
```

```
csvwrite(sprintf('%s/EAM/All/%d.xls',dataFolder,testNumber),results);
```

Figure B.10. Code to Manage Testing ssEAM Architecture.

```
%% Function to train and test elliptical ArtMAP
function [netOutput, trainTime, testTime1, testTime2] = eamTrain(vigilance, epoch, learnRate,…
        mu, tol, alpha, trainClass, trainFeatures, testClass, testFeatures)

%% Initialize needed variables
trim_option = 1; %1 to trim unneeded nodes, 0 to not trim
unknown_label = 0; %integer to fill no known label
omega = inf;
oflag = 0; %output control
ignore_option = 0; %does nothing
[rowTrain,colTrain] = size(trainClass); %calculate size of train set
[rowTest, colTest] = size(testClass); %calculate size of test set
totalNum = rowTrain + rowTest; %calculate total number of features
netOutput = zeros(totalNum,1); %initialize variable to hold results
D = sqrt(colTest)/mu;  %calculate D input

%% Normalize and format training data
trainFeatures = mapminmax(trainFeatures,0,1);

%% Train ArtMAP
tStart=tic;
[Templates,NLabels,list_presentations] = sseam_train(trainFeatures,trainClass',mu,D,…
        vigilance,alpha,omega,tol,learnRate, epoch,trim_option,unknown_label,oflag);
trainTime = toc(tStart);  %calculate train set train time

%% Get and record training data results
oflag = 0; %output control
vigilance = 0; %set variable for mandatory classification
omega = inf; %set variable for mandatory classification
tStart=tic;
[PLabels, CCFvalues, CMFvalues] = sseam_perf(Templates,NLabels,mu,D,vigilance,…
        alpha,omega,trainFeatures,unknown_label,ignore_option,oflag); %test train set
testTime1=toc(tStart); %calculate train set test time
netOutput(1:rowTrain,1) = PLabels;

%% Normalize and format testing data
testFeatures = mapminmax(testFeatures,0,1);

%% Get and record testing data results
tStart=tic;
vigilance = 0; %set variable for mandatory classification
omega = inf; %set variable for mandatory classification
```

```
[PLabels, CCFvalues, CMFvalues] = sseam_perf(Templates,NLabels,mu,D,vigilance,…
        alpha,omega,testFeatures,unknown_label,ignore_option,oflag); %test test set
testTime2=toc(tStart); %calculate test set test time
netOutput(rowTrain+1:totalNum,1) = PLabels;
```

Figure B.11. Function to Train and Test ssEAM Architectures.

# BIBILOGRAPHY

Anagnostopoulos, G. C., Georgiopoulos, M. "Ellipsoid ART and ARTMAP for incremental clustering and classification," *Proceedings of the International Joint Conference on Neural Networks,* IEEE, 2, pp. 1221-1226, 2001.

Anagnostopoulos, G. C., Georgiopoulos, M., Verzi, S., Heileman, G. "Reducing generalization error and category proliferation in Ellipsoid ARTMAP via tunable misclassification error tolerance: Boosted Ellipsoid ARTMAP," *Proceedings of the International Joint Conference on Neural Networks*, IEEE, 3, pp. 2650-2655, 2002.

Arrue B. C., Ollero A., Matinez de Dios J. R. "An intelligent system for false alarm reduction in infrared forest-fire detection," *Intelligent Systems and their Applications*, IEEE, 15(3), pp 64-73, 2000.

Aytac, T., Barsham, B. "Recognizing targets from infrared intensity scan patterns using artificial neural networks," *Journal of Optical Engineering*, SPIE, 48(1), pp. 017203-1 - 017203-13, 2009.

Bankman, D. J., Neighoff, T. M. "Pattern recognition for detection of human heads in infrared images," *Journal of Optical Engineering*, SPIE, 47(4), pp. 046404-1 - 046404-7, 2008.

Bar-Ilan, J. "On the overlap, the precision and estimated recall of search engines: A case study of the query 'Erdos'," *Scientometrics*, 42(2), pp. 207-208, 1998.

Carpenter, G. A. "Unifying multiple knowledge domains using the ARTMAP information fusion system," *Proceedings of the 11th International Conference on Information Fusion*, IEEE, 11, 2008.

Carpenter, G. A., Grossberg, S. "Adaptive Resonance Theory," Technical Report, Department of Cognitive and Neural Systems Center for Adaptive Systems and Center of Excellence for Learning in Education, Science, and Technology, Boston University, 2009.

Carpenter, G. A., Grossberg, S., Markuzon, N.m Reynolds, J.H., Rosen, D.B. "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *IEEE Transactions on Neural Networks*, IEEE, 3(5), pp. 698-713, 1992.

Carpenter, G. A., Martens, S. "Self-Organizing Hierarchical Knowledge Discovery by an ARTMAP Information Fusion System," Technical Report, Department of Cognitive and Neural Systems Center for Adaptive Systems and Center of Excellence for Learning in Education, Science, and Technology, Boston University, 2005.

Correia. B., Nunes R. C. "Grouping multiple neural networks for automatic target recognition in infrared imagery," *Proceedings of SPIE Automatic Target Recognition*, SPIE, 4379, pp. 124-135, 2001.

Dowdall, J. B., Pavlidis, I., Bebis, G. "Face Detection in the Near-IR Spectrum," *Infrared Technology and Applications XXIX*, SPIE, 5074, pp. 745-756, 2003.

Friedrich G., Yeshurun Y. "Seeing People in the Dark : Face Recognition in Infrared Images," *Proceedings of the Second International Workshop on Biologically Motivated Computer Vision*, ACM, 2002.

Gonzalez, R. C., Woods, R. E. *Digital Image Processing, 3rd Edition*, Upper Saddle River, Prentice Hall, 2008.

Grossberg, S. "How does a brain build a cognitive code," *Psychological Review*, American Psychological Association, 87(1), pp. 1-51, 1980.

Hagan, M. T., Demuth, H. B., Beale M. H. *Neural Network Design*, PWS Publishing, Boston, MA, USA, 1996.

Kanzawa Y., Kimura Y., Naito T. "Human Skin Detection by Visible and Near-Infrared Imaging," *Conference on Machine Vision Applications*, IAPR, pp. 503-507, 2011.

Kohavi, R., "A study of cross-validation and bootstrap for accuracy estimation and model selection,", *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 2, pp. 1137-1143, 1995.

Le, Q., Anagnostopoulos, G. C., Georgiopoulous, M., Ports, K. "An Experimental Comparison of Semi-Supervised ARTMAP Architectures, GCS and GNG Classifiers," *Proceedings of International Joint Conference on Neural Networks*, IEEE, 5, pp. 3121-3126, 2005.

Lee, Avery. "VirtualDub," Version 1.9.11, 2012.

Li, S. Z., Chu R., Liao S., Zhang L. "Illumination Invariant Face Recognition Using Near-Infrared Images," *Transactions on Pattern Analysis and Machine Intelligence*, IEEE, 29(4), pp. 627-639, 2007.

MATLAB, "Matlab Help," Version R2012a, 2012.

Miller, G., Fels, S., Oldridge, S. "A Conceptual Structure for Computer Vision," *2011 Canadian Conference on Computer and Robot Vision* (*CRV*), IEEE, pp.168-174, May 2011.

Rockwood, A., McAndless, J. "Through the looking glass: the synthesis of computer graphics and computer vision," *Multimedia*, IEEE, 6(3), pp.8-11, Jul-Sep 1999.

Sentenac T., Maoult Y. L., Orteu J., Boucourt G. "Evaluation of a charge-coupled-device-based video sensor for aircraft cargo surveillance," *Journal of Optical Engineering*, SPIE, 41(4), pp 796-810, 2002.

Sentenac T., Maoult Y. L., Orteu J., Boucourt G. "Overheating, flame, smoke and freight movement detection algorithms based on charge-coupled device camera for aircraft cargo hold surveillance," *Journal of Optical Engineering*, SPIE, 43(12), pp. 2935-2953, 2004.

Shah, M. "Guest Introduction: The Changing Shape of Computer Vision in the Twenty-First Century," *International Journal of Computer Vision*, 50(2), pp.103-110, 2002.

Shirvaikar, M. V., Trivedi, M. M. "A Neural Network Filter to Detect Small Targets in High Clutter Backgrounds,"*IEEE Transactions on Neural Networks*, IEEE, 6(1), pp. 252-257, Jan 1995.

Stanley, R. J., Watkins, S. E., Gopal, A. Moss, R. H., "A web-shareable real-world imaging problem for enhancing an image-processing curriculum," *IEEE Transactions on Education*, IEEE, 47(2) , pp. 211- 219, May 2004.

Stanley, R. J., Watkins, S. E., Moss, R. H., Gopal, A. "Traffic monitoring using a three-dimensional object tracking approach," *International Journal of Engineering Education*, 22(4), pp. 886-895, 2006.

Watkins S. E., Stanley R. J., Gopal A., Moss R. "Surveillance of Pedestrian Bridge Traffic using Neural Networks," *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2009*, SPIE, 7292, 2009.

Xu, R., Anagnostopouslos, G., Wunsch, D. C. "Multi-class Cancer Classification by Semi-supervised Ellipsoid ARTMAP with Gene Expression Data," *IEEE Proceedings of the 26th Annual International Conference of the Engineering in Medicine and Biology Society*, IEEE, pp. 188-191, 2004.

Yang H., Xie S., Hu X., Chen L., Lu Z. "Infrared Spectrum Visualizing Human Acupoints and Meridian-like Structure," *International Symposium on Biophotonic, Nanophotonics and Metamaterials 2006*, Metamaterials, 2006.

Zhang, B. "Computer vision vs. human vision," *2010 9th IEEE International Conference on Cognitive Informatics (ICCI)*, pp.3, July 2010.

# VITA

Kathryn N Rodhouse was born in Danville, Illinois. She attended the Missouri University of Science and Technology and obtained a bachelor's degree in Computer Engineering with minors in Mathematics and Computer Science in May of 2011. She was the first Honors Scholar in Computer Engineering for her research in the Applied Optics Laboratory. She started her master's degree studies in August, 2011 and worked as a graduate teaching assistant. Kathryn received a Master of Science degree in Computer Engineering from the Missouri University of Science and Technology in August, 2012. She accepted a fulltime position with Sandia National Laboratories.

Kathryn is a member of the Institute of Electrical and Electronics Engineers (IEEE), the Society of Women Engineers, and the Zeta Tau Alpha Fraternity. She was inducted into Eta Kappa Nu (HKN) and Phi Kappa Phi. She was selected as the 2011 national winner of the Alton B. Zerby and Carl T. Koerner Outstanding Electrical and Computer Engineering Student Award for HKN, was selected as the 2011 Region 5 winner of the IEEE Larry K. Wilson Regional Student Award, was first place winner in the IEEE Region 5 Student Papers Competition and was recognized as the 2011 Greek Woman of the Year and the 2009 2$^{nd}$ Place Woman Student of the Year by the Missouri University of Science and Technology.