

Air Force Institute of Technology

AFIT Scholar

Theses and Dissertations

Student Graduate Works

3-2008

Hyperspectral-Augmented Target Tracking

Neil A. Soliman

Follow this and additional works at: <https://scholar.afit.edu/etd>



Part of the [Signal Processing Commons](#)

Recommended Citation

Soliman, Neil A., "Hyperspectral-Augmented Target Tracking" (2008). *Theses and Dissertations*. 2780.
<https://scholar.afit.edu/etd/2780>

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact AFIT.ENWL.Repository@us.af.mil.



HYPERSPECTRAL-AUGMENTED TARGET TRACKING

THESIS

Neil A. Soliman, Captain, USAF

AFIT/GE/ENG/08-28

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GE/ENG/08-28

HYPERSPECTRAL-AUGMENTED TARGET TRACKING

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the
Degree of Master of Science in Electrical Engineering

Neil A. Soliman, B.S.E.E., M.B.A.

Captain, USAF


March 2008

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.


HYPERSPECTRAL-AUGMENTED TARGET TRACKING

Neil A. Soliman, B.S.E.E., M.B.A.
Captain, USAF

Approved:



Maj Michael J. Mendenhall, Ph.D. (Chair) 25-FEB-08
date



Dr. Juan R. Vasquez (Member) 25 Feb 08
date



Dr. Peter S. Maybeck (Member) 25 Feb 08
date

Abstract

With the global war on terrorism, the nature of military warfare has changed significantly. The United States Air Force is at the forefront of research and development in the field of intelligence, surveillance, and reconnaissance that provides American forces on the ground and in the air with the capability to seek, monitor, and destroy mobile terrorist targets in hostile territory. One such capability recognizes and persistently tracks multiple moving vehicles in complex, highly ambiguous urban environments.

This thesis investigates the feasibility of augmenting a multiple-target tracking (MTT) system with hyperspectral imagery. Feature-aided tracking methods have used features obtained from other sources such as panchromatic video, infrared, and radar imagery, but relatively few have examined hyperspectral data for tracking small targets. This research effort evaluates the usefulness of hyperspectral data as a feature set for the purpose of disambiguating targets in ambiguous situations. Classification of hyperspectral data is performed using fuzzy c-means and the self-organizing map clustering algorithms for remote identification of moving vehicles.

Results demonstrate a resounding 29.33% gain in performance from the baseline kinematic-only tracking to the hyperspectral-augmented tracking. Through a novel methodology, the hyperspectral observations are integrated in the MTT paradigm. Furthermore, several novel ideas are developed and implemented—spectral gating of hyperspectral observations, a cost function for hyperspectral observation-to-track association, and a self-organizing map filtering method. It appears that relatively little work in the target tracking and hyperspectral image classification literature exists that addresses these areas. Finally, two hyperspectral sensor modes are evaluated—Pushbroom and Region-of-Interest. Both modes are based on realistic technologies,

and investigating their performance is the goal of performance-driven sensing. Performance comparison of the two modes can drive future design of hyperspectral sensors.

Acknowledgements

First and foremost, I would like to offer praises and thanksgiving to my Lord and God for his guidance, motivation, and leading during this time. His presence in my life sustains me, and with Him on my side, I am able to overcome my weaknesses and take advantage of my strengths.

Second, I extend my love to my wife, son, and mother-in-law for their unconditional support and understanding. I also recognize the encouragement of my parents, who have raised me well. They are truly gifts from God.

Kudos to my faculty advisor, Maj Mendenhall, for his patience and guidance. The quality of this thesis document and the successful outcome of this research are due to your valuable inputs. I also extend my thanks to the other committee members, Lt Col Vasquez and Dr. Maybeck. They have been instrumental as well to the overall success of this research.

Finally, I would like to thank my fellow students at TechEdge, namely, Paul, Scott, Tom F., Tom L., and Andy for making this time together less stressful and enjoyable. They will surely be missed.

Neil A. Soliman

Table of Contents

	Page
Abstract	iv
Acknowledgements	vi
List of Figures	x
List of Tables	xiv
List of Abbreviations	xv
I. Introduction	1-1
1.1 Problems with Kinematic-Only Target Tracking using Panchromatic Video Data	1-2
1.2 Hyperspectral-Augmented Target Tracking	1-3
1.2.1 Hyperspectral and Panchromatic Video Data Processing	1-5
1.2.2 Hyperspectral Pixel Classification	1-6
1.2.3 Target Tracking	1-6
1.3 Research Scope	1-7
1.4 Organization	1-8
II. Background Information	2-1
2.1 Related Works in Classification and Tracking	2-1
2.2 Hyperspectral Data Analysis	2-6
2.2.1 Hyperspectral Sensor Data Calibration and At- mospheric Correction	2-6
2.2.2 Spectral Feature Extraction	2-8
2.2.3 Spectral Matching and Spectral Identification (ID)	2-8
2.2.3.1 Fuzzy C-Means (FCM) Clustering	2-11
2.2.3.2 Self-Organizing Map (SOM) Clustering	2-14
2.2.4 Spectral Library	2-22
2.3 Target Tracking	2-23
2.3.1 Panchromatic Video Sensor Data Processing and Measurement Formation	2-23
2.3.2 Observation-to-Track Association	2-25
2.3.3 Track Maintenance	2-29
2.3.4 Filtering and Prediction	2-30
2.3.5 Gating Computations	2-34

	Page	
2.4	Distance Measures	2-36
2.5	Distances between Data Distributions	2-39
2.5.1	Methods using Functional Form	2-40
2.5.2	Linkage Distances	2-40
2.6	Performance Analysis	2-41
2.6.1	Classification Performance	2-41
2.6.2	Hyperspectral-Augmented Tracking Performance	2-43
2.7	Summary	2-44
III.	Methodology	3-1
3.1	Hyperspectral and Panchromatic Video Measurement Formation and Data Processing	3-3
3.1.1	Scene Synthesis	3-3
3.1.2	Simulation of Sensor Data Measurement Formation	3-5
3.1.2.1	Hyperspectral Sensor	3-5
3.1.2.2	Panchromatic Video Sensor and Target Trajectory Generation	3-6
3.1.3	Data Processing	3-7
3.2	Spectral Library	3-9
3.3	Spectral Matching and Identification (ID)	3-15
3.3.1	Fuzzy C-Means (FCM) Clustering	3-15
3.3.2	Self-Organizing Map (SOM) Clustering	3-15
3.4	Observation-to-Track Association	3-22
3.4.1	Panchromatic Video Observations	3-22
3.4.2	Hyperspectral Observations	3-23
3.5	Track Maintenance	3-26
3.6	Filtering and Prediction	3-28
3.7	Nearest Neighbor Computations	3-28
3.8	Gating Computations	3-30
3.8.1	Kinematic Gating	3-31
3.8.2	Spectral Gating	3-31
3.9	Performance Measures	3-34
3.10	Summary	3-37
IV.	Design of Experiments and Simulation Results	4-1
4.1	Design of Experiments	4-1
4.1.1	Matlab	4-1
4.1.2	Simulated Measurement Noise	4-2
4.1.3	Scenarios	4-2
4.1.4	Configurations	4-4

	Page
4.1.4.1 Kinematic-Only Tracking	4-5
4.1.4.2 Hyperspectral-Augmented Tracking	4-8
4.1.5 Simulations	4-12
4.2 Kinematic-Only Tracking	4-15
4.2.1 Configuration 1	4-15
4.2.2 Configuration 2	4-19
4.3 Hyperspectral-Augmented Tracking	4-21
4.3.1 Fuzzy C-Means Configuration 1	4-21
4.3.2 Self-Organizing Map Configuration 1	4-25
4.3.3 Fuzzy C-Means Configuration 2	4-29
4.3.4 Self-Organizing Map Configuration 2	4-31
4.4 Quantitative Results	4-33
4.4.1 Fuzzy C-Means	4-35
4.4.2 Self-Organizing Map	4-38
4.5 Summary	4-50
V. Conclusions	5-1
5.1 Future Work	5-2
5.2 Concluding Remarks	5-4
Bibliography	BIB-1
Vita	VITA-1

List of Figures

Figure		Page
1.1.	Hyperspectral image cube.	1-4
1.2.	System elements of a hyperspectral-augmented target tracker. .	1-5
2.1.	System elements of a hyperspectral-augmented target tracker in- troducing system sub-components.	2-5
2.2.	Components of the hyperspectral sensor data calibration and at- mospheric correction.	2-6
2.3.	Solar illumination, atmospheric path absorption, and scattering.	2-7
2.4.	Mixed and pure hyperspectral signatures for vehicle class 8. . .	2-10
2.5.	Crisp vs. fuzzy set.	2-12
2.6.	Fuzzy c-means clustering algorithm.	2-13
2.7.	Topology of the self-organizing map (SOM).	2-16
2.8.	Example of the weight updates of the SOM for an input vec- tor \mathbf{s} , its best-matching neuron, and the best-matching neuron's neighbors.	2-18
2.9.	Example of the Gaussian neighborhood function overlaid on the SOM lattice.	2-19
2.10.	Example of the learning of the SOM using two-dimensional input data.	2-21
2.11.	Components of the panchromatic video sensor data processing and measurement formation [3].	2-24
2.12.	Data association example.	2-26
2.13.	An example of sample-wise distances between classes.	2-37
2.14.	Error matrix for a hypothetical 3-class problem.	2-43
3.1.	System elements and flow of a hyperspectral-augmented target tracker.	3-2

Figure		Page
3.2.	A synthetic background scene representing an idealized urban setting.	3-4
3.3.	Atmospheric water absorption bands in the near infrared spectrum.	3-7
3.4.	Example of a hyperspectral image (HSI) chip.	3-8
3.5.	Example of spatial smearing due to sensor dynamics.	3-10
3.6.	Target trajectory used to build the spectral library.	3-14
3.7.	Pseudo-code for the fuzzy c-means (FCM) clustering algorithm.	3-16
3.8.	Pseudo-code for the SOM clustering algorithm.	3-17
3.9.	The 30×30 SOM before and after performing morphological operations.	3-18
3.10.	Density and \mathbf{U} matrix representation of the SOM in Fig. 3.9(b).	3-19
3.11.	8-connected component of the SOM neuron.	3-20
3.12.	The neurons of the upper-right corner of Fig. 3.9(a) describing the SOM format.	3-21
3.13.	An HSI chip with four hyperspectral observations.	3-25
3.14.	The one-sided Bertsekas auction algorithm.	3-27
3.15.	The track confirmation and deletion logic.	3-29
3.16.	Introducing the concept of spectral gating.	3-33
3.17.	Neighbors of class ID 17 plotted in ascending order of distance.	3-34
4.1.	Two ambiguous tracking scenarios.	4-3
4.2.	Comparison of the reflectance plots for the two truth trajectory combinations.	4-6
4.3.	Display of the hyperspectral-augmented tracking simulation environment.	4-14
4.4.	Kinematic-only tracking for configuration 1 (Part A).	4-17
4.5.	Kinematic-only tracking for configuration 1 (Part B).	4-18
4.6.	Kinematic-only tracking for configuration 2.	4-20

Figure		Page
4.7.	Hyperspectral-augmented tracking version of Fig. 4.4 for FCM (Part A).	4-23
4.8.	Hyperspectral-augmented tracking version of Fig. 4.5 for FCM (Part B).	4-24
4.9.	Hyperspectral-augmented tracking version of Fig. 4.4 for the SOM (Part A).	4-27
4.10.	Hyperspectral-augmented tracking version of Fig. 4.5 for the SOM (Part B).	4-28
4.11.	Hyperspectral-augmented tracking version of Fig. 4.6 for FCM.	4-30
4.12.	Hyperspectral-augmented tracking version of Fig. 4.6 for the SOM.	4-32
4.13.	Ensemble statistics for SOM configuration set A, B, C, D, and E using scenario 1 and truth trajectories for class ID 17 and 31. .	4-41
4.14.	Percent $+/-$ from kinematic-only configuration 1 to SOM hyperspectral-augmented tracking configuration sets A, B, C, D, and E using scenario 1 and truth trajectories for class ID 17 and 31.	4-42
4.15.	Ensemble statistics for SOM configuration set A, B, C, D, and E using scenario 2 and truth trajectories for class ID 17 and 31. .	4-43
4.16.	Percent $+/-$ from kinematic-only configuration 2 to SOM hyperspectral-augmented tracking configuration sets A, B, C, D, and E using scenario 2 and truth trajectories for class ID 17 and 31.	4-44
4.17.	Ensemble statistics for SOM configuration set A, B, C, D, and E using scenario 1 and truth trajectories for class ID 17 and 32. .	4-45
4.18.	Percent $+/-$ from kinematic-only configuration 3 to SOM hyperspectral-augmented tracking configuration sets A, B, C, D, and E using scenario 1 and truth trajectories for class ID 17 and 32.	4-46
4.19.	Ensemble statistics for SOM configuration set A, B, C, D, and E using scenario 2 and truth trajectories for class ID 17 and 32. .	4-47

Figure		Page
4.20.	Percent \pm from kinematic-only configuration 4 to SOM hyperspectral-augmented tracking configuration sets A, B, C, D, and E using scenario 2 and truth trajectories for class ID 17 and 32.	4-48

List of Tables

Table		Page
2.1.	Gate thresholds and the probability mass P_G in the gate [3]. . .	2-36
3.1	Description of each of the 41 vehicle classes in the spectral library.	3-11
3.2.	Description of the 5 background classes and an unknown class in the spectral library.	3-13
3.3.	Gate thresholds and the probability mass P_G in the gate [3]. . .	3-31
4.1.	Configurations for kinematic-only tracking.	4-7
4.2.	Configurations for FCM.	4-9
4.3	Configurations for the SOM.	4-10
4.4.	Ensemble statistics for the kinematic-only configurations. . . .	4-34
4.5.	Ensemble statistics for FCM configurations.	4-37
4.6.	Percent $+/-$ from kinematic-only tracking baseline to FCM hyperspectral-augmented tracking.	4-37
4.7.	Ensemble statistics and percent $+/-$ for all kinematic-only and SOM configurations.	4-49
4.8.	Ensemble statistics and percent $+/-$ for original SOM and filtered SOM configurations.	4-49
4.9.	Ensemble statistics for original SOM, FCM, and filtered SOM configurations.	4-49

List of Abbreviations

Abbreviation		Page
AFDD-1	Air Force Doctrine Document 1	1-1
ISR	Intelligence, Surveillance, and Reconnaissance	1-1
USAF	United States Air Force	1-1
IED	Improvised Explosive Device	1-1
UAV	Unmanned Aerial Vehicle	1-1
DARPA	Defense Advanced Research Projects Agency	1-2
EO	Electro-Optical	1-2
IR	Infrared	1-2
HSI	Hyperspectral Image	1-5
MTT	Multiple-Target Tracking	1-6
SOM	Self-Organizing Map	2-1
H-PMHT	Histogram Probabilistic Multi-Hypothesis Tracking	2-3
MF	Matched Filter	2-4
ANN	Artificial Neural Network	2-14
MHT	Multiple Hypothesis Tracking	2-29
PDF	Probability Density Function	2-31
MOE	Measure Of Effectiveness	2-43
MOF	Measure Of Fit	2-43
USGS	United States Geological Survey	3-11
NASA	National Aeronautics and Space Administration	3-11
JPL	Jet Propulsion Laboratory	3-11
AVIRIS	Airborne Visible / Infrared Imaging Spectrometer	3-11
ML	Maximum-Likelihood	3-21
MC	Monte Carlo	4-12

HYPERSPECTRAL-AUGMENTED TARGET TRACKING

I. Introduction

The 2003 publication of Air Force Doctrine Document 1 (AFDD-1) states that “...the Air Force is the major operator of sophisticated air- and space-based intelligence, surveillance, and reconnaissance (ISR) systems and is the Service most able to quickly respond to the information they provide...” [1]. Because the United States Air Force (USAF) is committed to be the leader in the military application of ISR technology, it is continually pursuing new research and development in this area. When the USAF has access to vital, timely, and accurate information, the United States military has an unparalleled operational advantage over its enemies.

In the late 20th Century, the nature of military warfare changed significantly. Adversaries resorted to terrorism and urban warfare [1]. Urban combat became less conventional at both the operational and tactical levels. Terrorists began escalating the use of civilian vehicles instead of the camouflage military vehicles for the transportation of combatants, as a platform to launch weapons, and as improvised explosive devices (IED). ISR technology that recognizes and monitors these vehicles have become essential elements of urban warfare.

According to the Strategic Appraisal, United States Air and Space Power in the 21st Century, a rapid-reaction information force can quickly establish video surveillance of potentially hostile territory [21]. Essential features of such a force are Unmanned Aerial Vehicles (UAVs) equipped with video equipment, coupled with command planes capable of gathering, editing, and instantaneously disseminating that coverage [32]. Mobile targets remain elusive, but if researchers and engineers develop adequate surveillance capability to locate them reliably, military forces can destroy them easily with nuclear or conventional weapons [9]. Several research works that address this ISR need include the following:

- In the United States Army’s 2007 Small Business Technology Transfer Program, the Army funded the development of algorithms for UAV ISR systems for the purpose of tracking several types of targets in urban environments, including human, civilian vehicles, and military targets that could exhibit highly nonlinear motions [42].
- “In 2005, graduate students at the Air Force Institute of Technology in Ohio and scientists at Los Alamos National Laboratories developed project Angel Fire, a persistent city-sized surveillance program. By providing real-time imaging capabilities, IEDs and other threats to ground forces can be detected, prevented, and/or negated. Angel Fire is particularly well suited to provide enhanced situational awareness to forces operating in an urban environment, convoy operations, or other ground operations” [14].
- The Defense Advanced Research Projects Agency (DARPA) developed an automated video-based ground targeting system for UAVs through high performance electro-optical (EO) and infrared (IR) sensors that provide high quality data for target identification and engagement [2].

These are only some of the numerous works that address the use of video surveillance systems for urban warfare. Clearly, the Department of Defense (DoD) is capitalizing on video technology to enhance its capability to “track, record, and analyze the movement of every vehicle in a foreign city” [41].

1.1 Problems with Kinematic-Only Target Tracking using Panchromatic Video Data

This thesis evaluates a tracking system that processes digital imagery, digitized from an analog panchromatic video camera. After the camera presents the video data of the imaged scene to the tracker as frames of two-dimensional imagery, the tracker segments the imaged scene into regions of change using pixel intensity frame differencing (based on an empirically or statistically chosen threshold). The output

is an angular displacement between the measured target line-of-sight and the optical axis of the sensor [8]. The tracker only uses this kinematic information on the resolved targets, which manifests in either or both dimensions of the imaged scene, to initiate or update existing target tracks. The track algorithm, therefore, must be robust enough to respond effectively when multiple targets go through ambiguous situations. Otherwise, track swaps and losses occur, and track purity¹ becomes degraded.² Ambiguous situations are characterized by changes in the target’s velocity (e.g., move-stop-move), obscurations in the scene, or other objects spatially close to the target of interest. Other tracking methods have been developed to address track swaps and losses caused by these ambiguous situations. One such method is feature-aided tracking that utilizes extraction techniques to build a feature set for each target of interest. Tracks are no longer just comprised of position and velocity estimates, but they also consist of unique feature estimates. One such feature set is a track’s *hyperspectral signature*.

1.2 Hyperspectral-Augmented Target Tracking

The main objective of this thesis is to determine the feasibility of using spectral features to disambiguate targets in ambiguous situations. More specifically, this research answers the following question: *When augmented by hyperspectral data, is the performance of the kinematic-only tracker in ambiguous situations improved?* The experimental results indeed show that the hyperspectral-augmented tracker outperforms the kinematic-only tracker. Instead of using features from video images (e.g., color, shape, and gradient), the hyperspectral-augmented tracker exploits the rich hyperspectral data of each target track. Through the target’s potentially unique spectral response, the tracker is more successful in disambiguating closely spaced targets in ambiguous situations.

¹Track purity is the average percentage of correctly associated observations for each track.

²If track swaps occur due to close proximity, then the tracker associates each track with the wrong target. If track losses occur (e.g., due to obscurations or closely spaced targets), then track continuity becomes an issue.

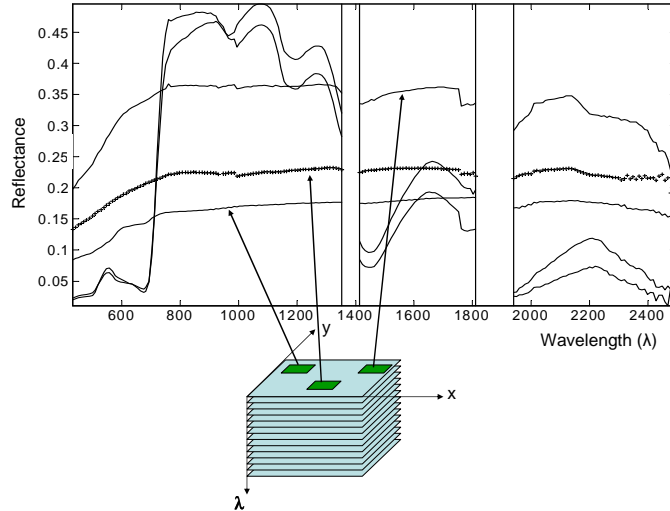


Figure 1.1: In a hyperspectral image cube, each pixel consists of three dimensions—spatial dimensions x and y and hyperspectral dimension λ . The plot shows the spectral responses, which correspond to the third dimension λ , for various background materials in the hyperspectral image. Because of water in the atmosphere, absorption wavelength exist (gaps in the curve) around 1400 and $1900nm$.

The hyperspectral data consist of nearly 200 contiguous spectral bands, forming a three-dimensional image cube (Fig. 1.1). Each pixel consists of two spatial dimensions (x, y) and one spectral dimension (λ). The spectral dimension λ is a feature vector consisting of a spectrum of the imaged pixel area, most notably in the visible (VIS) and infrared (IR) areas of the electromagnetic spectrum. The detailed spectral response of a pixel assists in providing precise target identification [45]. Furthermore, the high spectral resolution preserves important aspects of the spectrum (e.g., shape of narrow absorption bands) and makes differentiation of objects possible [36].

The hyperspectral-augmented target tracking system consists of three main components: hyperspectral and panchromatic video data processing, hyperspectral pixel classification, and kinematic-only target tracking, as shown in Fig. 1.2.

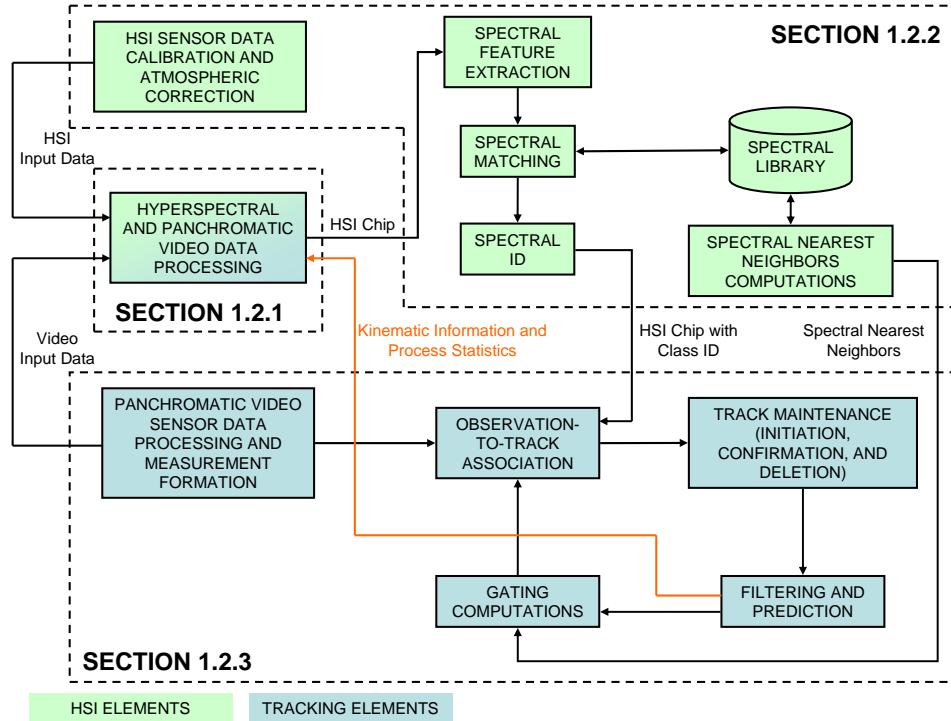


Figure 1.2: The system elements of a hyperspectral-augmented target tracker consist of three main components: hyperspectral and panchromatic video data processing (Sec. 1.2.1), hyperspectral pixel classification (Sec. 1.2.2), and kinematic-only target tracking (Sec. 1.2.3).

1.2.1 *Hyperspectral and Panchromatic Video Data Processing.* The *hyperspectral and panchromatic video data processing*³ element performs several crucial functions. After the tracker initiates a target track (based on change detections in the imaged scene), the data processor extracts the hyperspectral pixels likely to contain the target of interest by commanding the hyperspectral sensor to scan the track region bounded by the track’s propagated position and uncertainty. This bounded region is called a hyperspectral image (HSI) chip. Finally, the data processor populates a *spectral library* using hyperspectral measurements collected from various vehicle types and background materials and computes the prototype vectors representing each vehicle and background.

³This thesis document italicizes the system elements provided in Fig. 1.2 and treats each element as a singular noun. Throughout the rest of this document, the *hyperspectral and panchromatic video data processing* element will be referred to simply as a “data processor.”

1.2.2 *Hyperspectral Pixel Classification.* The hyperspectral pixel classification component consists of five elements and the *spectral library*. Before the data processor extracts the HSI chip for each target track, the *HSI sensor data calibration and atmospheric correction* element preprocesses the hyperspectral imagery to remove unwanted effects due to sensor noise and the atmosphere. *Spectral feature extraction*⁴ determines the relevant dimensions of the hyperspectral data. This element is not currently implemented in this thesis and serves as a placeholder for future efforts. This research provides baseline work, in which all dimensions are assumed relevant. *Spectral matching and identification (ID)* performs a supervised classification on the remaining spectral features (assuming *spectral feature extraction* is accomplished) by comparing the hyperspectral data of each pixel (λ) with samples for each class in the *spectral library*. The class ID of the nearest sample to the hyperspectral pixel is the identifier that disambiguates target tracks in ambiguous situations. Finally, the *spectral nearest neighbor computations* element determines the nearest neighbors for each class in order to gate hyperspectral observations.

1.2.3 *Target Tracking.* This research implements a target tracking approach based on a conventional multiple-target tracking (MTT) system. The MTT system can be divided into five functions [8]: *sensor data processing and measurement formation, gating computations, observation-to-track association, track maintenance, and filtering and prediction*. There is considerable overlap of the functions of these elements, but this representation provides a convenient way to describe the functions required for an MTT system. *Sensor data processing and measurement formation* identifies target detections or observations versus returns from extraneous sources, such as potential false alarms produced by sensor noise and background clutter. The system considers the incoming target observations for the update of existing tracks. *Gating computations* is a screening mechanism that determines which observations are valid candidates. The *observation-to-track association* takes the observation-to-track

⁴Feature extraction is a form of dimensionality reduction.

pairings that satisfy gating requirements and determines which observation-to-track assignments will actually be made. *Track maintenance* refers to the functions of track initiation, confirmation, and deletion. *Filtering and prediction* incorporates the assigned observations and predicts the tracks ahead to the arrival time for the next set of observations. *Gating computations* places gates around these predicted positions. The tracking processing cycle starts over again when the tracker receives new kinematic or hyperspectral observations.

1.3 Research Scope

Extensive work has been accomplished in many of the functions identified in Fig. 1.2, but not as one complete system. This research demonstrates that, by augmenting the kinematic-only tracking system with the targets' hyperspectral signature, the probability of correct identification increases significantly. So long as the classifier correctly identifies the target, even if the kinematic-only tracker swaps target tracks, this work achieves the urban environment tracking goal effectively. This thesis further presents three novel ideas—spectral gating, hyperspectral observation-to-track association, and filtering of the self-organizing map.

First, the spectral gating work develops a method for calculating the nearest neighbors of a target class. Since relatively little work on hyperspectral-augmented target tracking exists in the literature (see Sec. 2.1), this hyperspectral gating work is the first of its kind. Second, the cost function used in the hyperspectral observation-to-track association consists of a sum of weighted kinematic and spectral distances. The weighting affects the influence of an observation's spectral signature on the data association. The weighting combinations are 99-1%, 50-50%, and 1-99%, in which the first value in the combination provides the weight or level of confidence placed on the spectral distance. Finally, a filtered version of the self-organizing map is implemented to remove vehicle samples that are highly influenced by background spectra. These samples are considered noisy and could adversely affect classification accuracy.

1.4 *Organization*

This thesis consists of five chapters. Fig. 1.2 provides a reference for the first three chapters. This first chapter presents the problem statement, the importance and material needed to understand this research, and the scope and limitations of this research effort. The hyperspectral-augmented target tracking system consists of numerous functions, but this thesis only evaluates several key aspects of the system. The research scope defines the boundaries that cover these aspects.

The second chapter provides more in-depth background information. It discusses classification and target tracking in more detail, provides the rationale behind the choice of hyperspectral features (over panchromatic video image features), and describes several approaches for the system elements. Several literature reviews provide insight to other relevant research in feature-aided tracking, including target tracking applications using hyperspectral data.

The third chapter presents the methodology by following the signal flow through the system, from sensor data formation to data association. In this presentation, the data starts as separate entities—hyperspectral sensor data and panchromatic video sensor data. From the information provided by both data sets, an HSI chip is formed. The HSI chip is processed by the hyperspectral elements for pixel classification and by the tracking elements for track initiation or track update. This chapter also describes the procedures and algorithms performed by each system element and provides the parameters used in the hyperspectral data analysis and target tracking functions (Fig. 2.1, Sec. 2.2 and 2.3, respectively).

The fourth chapter describes the design of experiments and provides the quantitative results. The design of experiments discusses various system configurations and ambiguous scenarios, which are used for determining the feasibility of the hyperspectral-augmented tracker. This chapter also analyzes and compares the different configurations in order to identify the configuration that provides the most

meaningful results. The analysis compares the performance of a target track before and after an ambiguous situation is encountered.

Finally, the fifth chapter summarizes the findings, provides a final analysis of the results, and discusses the ramifications of those results on the overall ISR need. The analysis also addresses the shortcomings and assumptions in the methodology and experimental design. Based on goals achieved, this chapter provides a future perspective on this research effort and outlines recommendations for future research.

II. Background Information

This chapter provides a comprehensive problem background, examines what others have done to address the problem area, and presents a literature review on related works in the areas of hyperspectral data classification, target tracking, target tracking using self-organizing maps (SOM), and target tracking with hyperspectral images (HSI). More importantly, this chapter discusses background information on relevant concepts, processes, algorithms, and procedures in hyperspectral data analysis and target tracking.

2.1 Related Works in Classification and Tracking

As previously discussed in Sec. 1.1, kinematic-only target tracking is highly susceptible to track swaps and track losses in ambiguous situations. Track swaps and track losses cause the tracker to perform ineffectively. An alternative is feature-aided target tracking using features derived from panchromatic video images. When going through ambiguous situations, the feature-aided tracker disambiguates closely spaced tracks based on their feature classification. After the tracker extracts features, it identifies a track according to a learned feature model. One such model is a two-dimensional histogram in which the dimensions consist of a pixel's intensity and its radial distance from the centroid of the track. Feature-aided tracking methods often successfully address the deficiencies of kinematic-only tracking and can be effective in disambiguating targets in multi-target scenarios. However, they have significant drawbacks, as illustrated by the following works:

- In [48], the authors point out that the mean-shift algorithm, which is a gradient-based method, is often chosen for object tracking because it is very efficient and easy to implement and performs well in tracking objects with partial occlusions, background clutter, and large motions. However, despite many of its good properties, it can only find a local optimal object position instead of a global one (a common problem exhibited by gradient-based methods). Furthermore,

it does not perform well for scale variations and is sensitive to illumination variations and severe partial occlusions.

- In [49], feature information comes from all moving objects within view, and detected objects are tracked by feature. The tracking system integrates spatial position, motion, shape, and color, which makes the tracker insensitive to changes in background, interruption of motion, and object orientation. The shape and color features, however, are potentially unreliable. An object with complex shape is more likely to change its compactness¹ than an object with a simple shape. Furthermore, its color is likely to change with lighting conditions.
- The authors in [16] present a scheme for vehicle fingerprinting for tracking and reacquisition in video. They claim that, for many man-made objects such as buildings and vehicles, edges are the most dominant features. Ideally, the geometry and appearance of the vehicle can be fully described if (1) all the edges can be detected and their 3D locations and orientations reconstructed, and (2) the color/texture information for all the regions delineated by the edges can be extracted. However, it is difficult to discriminate objects that have the same geometric structure and differ only in color, and edge matches in a cluttered background are impractical for object alignment.

These works are just a small fraction of the extensive research in classification and tracking. From these three, however, two important conclusions can be reached. First, when feature extraction algorithms are applied to video images, they are highly dependent on image resolution, object orientation, intensity changes, object similarity and segmentation, and reliable invariant feature representations [16]. Clearly, one has to overcome these challenges to succeed in classifying and tracking targets. Instead of using video-based features, this research proposes to identify and track targets of interest using their unique hyperspectral signatures. Since hyperspectral data are obtained by a separate sensor, they are not dependent on the features of the tar-

¹Compactness is defined as $\frac{4\pi * Area}{Perimeter^2}$, where *Perimeter* is the number of boundary pixels.

get measurements in the video images. Second, if the classification component of the feature-aided target tracking works well in disambiguating objects, the tracker maintains tracks of moving objects effectively. Thus, tracking performance is directly affected by the classifier's ability to distinguish target classes of interest.

Related works have also been published on the use of hyperspectral data, either as a means to track targets or as an aid to existing tracking methods:

- The authors in [39] extend the histogram model used in previous presentations of the Histogram Probabilistic Multi-Hypothesis Tracking (H-PMHT) algorithm² to treat hyperspectral data. Hyperspectral data is interpreted as a spatial-spectral histogram.³ The objective of the spectral H-PMHT algorithm is to disambiguate tracks when they cross paths, where a track is defined as the physical depression or footprint left behind by a target source on the ground. Spectral H-PMHT assumes that the spectral characteristics of the sources are known and available in simple nonparametric forms.
- The authors in [20] analyze the ability to associate a vehicle uniquely in one image with the same vehicle in a subsequent image by extracting its spectral characteristics and using the information to find its location. Image analysis is performed by converting the hyperspectral data into surface reflectance through an atmospheric compensation process and applying a matched filter to locate vehicles of interest. When contextual information providing building and other fixed object locations is used, initial results show that, for an image-derived spectrum of a blue vehicle, the algorithm finds the same vehicle in a subsequent image with a false alarm rate of approximately six out of 200,000 pixels. However, when the authors attempt this type of analysis for other vehicles in the

²The Histogram Probabilistic Multi-Hypothesis Tracking (H-PMHT) algorithm applies the Expectation-Maximization (EM) method [38] to target tracking using sensor level data. It interprets the received power levels in all of the sensor cells as a synthetic histogram in order to generate point measurements. For images, power level is pixel intensity, and a cell is a pixel.

³Each hyperspectral scan line is a multidimensional array, where each spatial cell has an associated vector of (possibly disjoint) frequency bin amplitudes. The intensity data in this array is interpreted as a spatial-spectral histogram of a synthetic sensor process.

scene (other blue cars, red cars, etc.), they observe false alarm rates significantly higher. Thus, their methodology only works for limited cases.

- The work in [44] addresses the problem of tracking a moving point target in a time sequence of hyperspectral images.⁴ The use of hyperspectral images should be superior to current technologies, due to the benefit of simultaneously exploiting two target-specific properties: the spectral target characteristics and the time-dependent target behavior. The approach consists of two steps. The first step transforms each of the hyperspectral images forming the sequence into a two-dimensional image⁵ using a known point-target detection acquisition algorithm. The hyperspectral image transformation uses a matched filter (MF) detector⁶ based on a linear mixing model. The second step performs target detection and tracking using time-domain processing. Time-domain processing uses an algorithm based on temporal processing of the frame sequence using the spatial location of the hyperspectral detection. Temporal processing exploits the change in the IR pixel's intensity as a moving point target traverses it. The temporal profile of a target-affected pixel rises and falls as the target enters and exits the pixel, while clutter-affected temporal profiles show more monotonic changes over an equal time period. A variance-filter algorithm [37] detects the presence of targets from the temporal profile of each pixel while suppressing clutter-specific influences.

The approach in [44] provides an insightful way of dealing with tracking problems using hyperspectral imagery, specifically the benefit of simultaneously exploiting the spectral target characteristics and the time-dependent target behavior. This research also exploits these two target-specific properties. Hyperspectral data analysis

⁴Since no camera is currently capable of taking a hyperspectral movie, the authors developed a simple algorithm that creates a hyperspectral movie based on a real-world infrared (IR) image sequence. The algorithm also implants a synthetic moving target, which travels in backgrounds that are influenced by both evolving clutter and noise.

⁵The two-dimensional image consists of the spatial dimensions of the target of interest over the IR scene.

⁶MF detectors assume that the spectral characteristics of a target of interest are known. The detectors differ in the way they model mixed pixels and spectral variability.

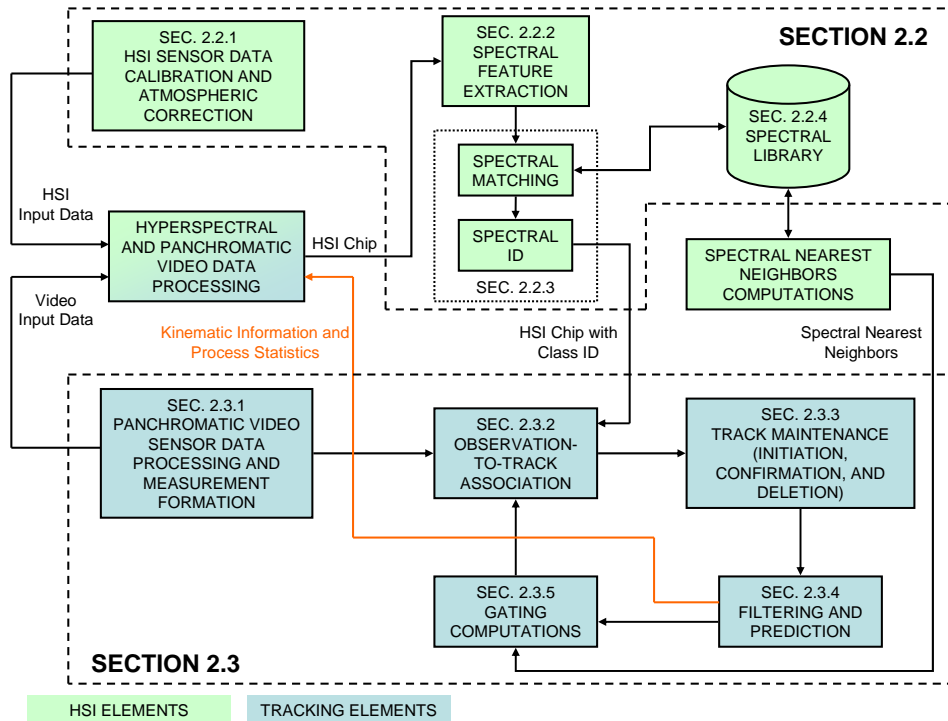


Figure 2.1: The system elements of a hyperspectral-augmented target tracker are described in two main sections: hyperspectral data analysis (Sec. 2.2) and target tracking (Sec. 2.3). Each subsection (identified in each sub-block in the diagram) discusses the algorithms, theories, and concepts performed by each element.

uses hyperspectral imagery to construct the feature model for various targets and applies this feature model to classify hyperspectral pixels, and target tracking uses panchromatic video images to generate and update existing tracks. The hyperspectral-augmented tracker further utilizes hyperspectral line scanner dynamics to provide kinematic updates. This chapter describes these processes in the hyperspectral data analysis and target tracking sections.

Fig. 2.1 summarizes the organization of the next two sections. These sections describe the background concepts and definitions of various system elements and provide a comparative study of several relevant approaches. Hyperspectral data analysis expands on the hyperspectral pixel classification briefly described in Sec. 1.2.2, whereas target tracking expands on the multiple target tracking (MTT) system discussed in Sec. 1.2.3.

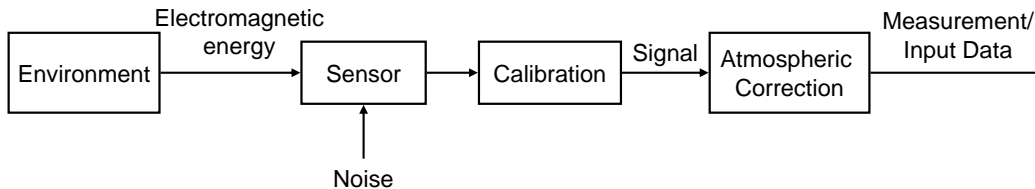


Figure 2.2: The electromagnetic energy that a sensor receives is not purely characteristic of the reflectance of the material. The received signal is corrupted by sensor noise, viewing and illumination geometry, atmospheric scattering, reflected light from adjacent objects, absorption of electromagnetic energy by the earth’s atmosphere, and sensor aberrations.

2.2 Hyperspectral Data Analysis

Hyperspectral data analysis begins with acquisition and preprocessing to remove known system errors, assure accurate calibration, and correct for atmospheric effects [10] (Sec. 2.2.1). *Spectral feature extraction* determines the relevant dimensions of the data and extracts them for analysis⁷ (Sec. 2.2.2). *Spectral matching and identification (ID)* (Sec. 2.2.3) classifies each pixel of a hyperspectral image (HSI) chip as one of the classes stored in the *spectral library* (Sec. 2.2.4). The tracker uses the hyperspectral observations formed from the chip (if any) to initiate or update existing target tracks.⁸

2.2.1 Hyperspectral Sensor Data Calibration and Atmospheric Correction.

As shown in Fig. 2.2, the measurements of interest are not the electromagnetic energy received by the sensor, but the output of complex radiometric preprocessing and sensor noise. Radiometric preprocessing influences the brightness values of an image to correct for sensor malfunctions and inconsistencies or to compensate for atmospheric degradation. Many important additional effects exist for which the system may need to account. As illustrated in Fig. 2.3, these effects include [36]:

- the angle of the sun,

⁷As previously discussed, this research does not perform *spectral feature extraction*. This system element is a placeholder for relevant future work.

⁸Formation of hyperspectral observations is discussed in Sec. 3.4

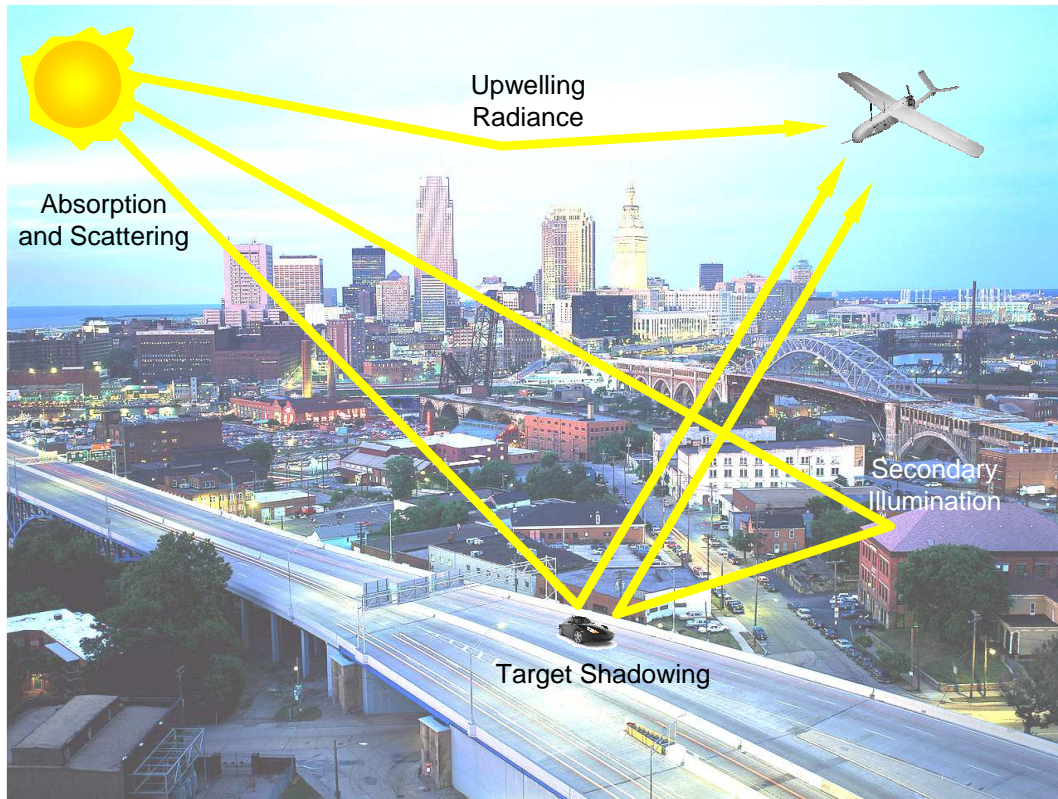


Figure 2.3: Solar illumination and atmospheric path absorption and scattering modulate the direct path radiance signal observed at the sensor.

- the viewing angle of the sensor,
- the upwelling solar radiance from atmospheric scattering,
- the secondary illumination of the material by light reflected from adjacent objects in the scene,
- shadowing,
- the scattering and absorption of the reflected radiance by the atmosphere, and
- spatial and spectral aberrations in the sensor.

Radiant flux or radiance is recorded by sensors that observe the earth's surface using visible or near-visible radiation. For a given ground pixel, the radiance that a sensor observes at any particular wavelength is determined (to first order) by the solar illumination and the reflectivity (reflectance) of the material at that wavelength [10]. Reflectance is the brightness that is of interest for remote sensing. When measure-

ments are based solely on the reflectance property of the material, it is feasible to compare measurements made by different sensors over different locations at different times. Therefore, a key preprocessing step in the exploitation of hyperspectral data characterizes and compensates for the environmental and atmospheric effects. Sophisticated methods have been developed for atmospheric correction and sensor calibration. However, to maintain the focus of this research, this thesis considers these methods a preprocessing step with no further elaboration.

2.2.2 Spectral Feature Extraction. “Features” are not geographical features visible in an image, but are rather “statistical” characteristics of image data—individual bands or combinations of band values that carry information concerning systematic variation within the scene. Thus, *spectral feature extraction* isolates components (or dimensions) within the hyperspectral data that are most useful in portraying essential elements of an image⁹ [10]. High-dimensional spaces are mostly empty; therefore, it is invaluable to find the most appropriate subspace that contains the significant structure for a given classification problem. This is accomplished by feature extraction algorithms. Examples are principal component analysis [10], discriminate analysis [13], decision boundary feature extraction [23], and joint classification and feature extraction methods [17, 28, 33].

2.2.3 Spectral Matching and Spectral Identification (ID). The spectral matching and identification (ID) functions serve as the crux of the supervised classification process. In supervised classification, there exists a feature model or idealized feature representation for different classes. This research uses the knowledge of the hyperspectral signatures of multiple vehicles to classify a vehicle observed in an urban scene. In the classification literature [43], this approach is called “supervised learning,” which means that the phenomena of interest has been divided into a number of *a priori* groups, from each of which a number of samples have been observed and

⁹In the literature, this is also referred to as dimensionality reduction.

have been characterized in terms of a number of discriminating features. The sample set is called training data and stored in a *spectral library*. Each sample in the *spectral library* consists of a multidimensional pattern vector. Given each spectral band is represented by one axis of a multidimensional space (the feature space), a sample is a point in that space. If all the samples in the library are well-defined, they should fall into clearly defined groups. Hyperplanes and hypersurfaces, which represent decision boundaries, can be used to separate the distinct classes in the multidimensional feature space. A typical remote sensing application employs an algorithm that classifies or labels the individual pixels forming the hyperpectral cube (see Fig. 1.1), in which each pixel consists of a multidimensional vector. The algorithm is called a decision rule or a classifier, and it determines the position of the pixel's spectral response with respect to the decision boundaries, and thus allocates a specific label to that pixel [43]. Classifiers can be developed using parametric or nonparametric approaches.

As any pattern recognition book [13,43,45] will show, there are many algorithms from which to choose. When faced with such a range of algorithms, how does one know which algorithm is “best?” According to the *No Free Lunch Theorem* [13], there is no such thing as an overall superior classifier. If one algorithm seems to outperform another in a particular situation, it is a consequence of its fit to the particular pattern recognition problem, not the general superiority of the algorithm. The decision should be based on the aspects that matter most—prior information, data distribution, amount of training data, and cost or reward functions [13].

A number of algorithms that have been developed is grounded to a significant degree in statistical decision theory and regarded as parametric classifiers (e.g., maximum likelihood or minimum distance procedures) [45]. With this type of classifier, serious problems arise when the number of available samples tend to be small and the dimensionality of the feature vector is large [13]. To represent the class distribution accurately, a rule of thumb is that the number of training data samples per class should be at least thirty times the number of features [43]. Hyperspectral data often have more than 200 dimensions, thus requiring approximately 6,000 training samples

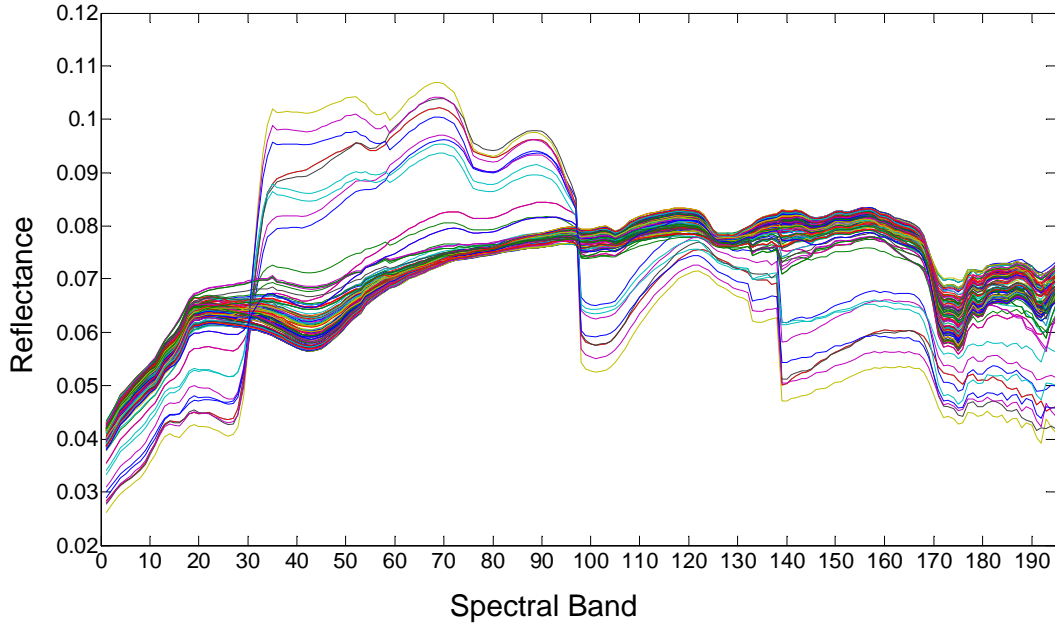


Figure 2.4: This plot shows the 150 spectral responses for vehicle class 8. The variation is due to the different mixtures of class 8 spectrum with background spectra. In some samples, the class 8 spectrum is highly influenced by the effect of one or more background classes, specifically vegetation spectra.

for each class. This requirement is almost never met since many objects of interest only occupy on the order of a 100 pixels. Furthermore, the computational complexity required to develop the classifier does not allow for real-time processing. Therefore, nonparametric algorithms are evaluated in this work. They do not make assumptions about the statistical distribution of the data and permit real-time processing of very large data sets [43].

Because of the spatial resolution of remote sensors, mixed pixels frequently contaminate the hyperspectral data obtained from urban environments. These are the pixels that do not represent a single homogeneous class; instead, two or more classes are present in a single pixel area [45]. In fact, the *spectral library* is made up of such pixels. An example is shown in Fig. 2.4. Traditional classification methods do not provide a good mechanism for coping with such uncertainty and imprecision. This research initially evaluated two traditional classifiers, namely, the minimum Euclidean distance (MED) [13] and k-means classifiers [13]. The problem with these classifiers

is that, once a pixel has been assigned to one class, its effect on other classes is negligible. The hard classification assignment forces the mixed pixels to be allocated to one and only one class, thereby resulting in erroneous classifications [45]. Since this research deals with complex mixtures, it is a a major consideration in the choice of nonparametric algorithms. Two such algorithms that perform well with mixed pixels are evaluated in this research—fuzzy c-means [13] and self-organizing map [22]. Both cluster or partition data into subsets, so that the data in each subset share some common trait, which is often proximity based on a defined distance measure. Since these are clustering algorithms, a classification method will have to be implemented after clustering is performed.

2.2.3.1 Fuzzy C-Means (FCM) Clustering. Fuzzy set theory was motivated by considerations discussed previously and provides a conceptual framework for solving knowledge representation and classification problems in an ambiguous environment [43]. The authors in [31] use *a priori* knowledge of spectral information for certain land cover classes in order to classify SPOT¹⁰ images using fuzzy logic. Hence, it is possible to classify the remotely sensed image (as well as any other digital imagery) in such a way that certain land cover classes are clearly represented in the resulting classified image. Based on the results, fuzzy logic can be satisfactorily used for hyperspectral data classification.

A fuzzy-based classifier assigns class membership to pixels based on a membership function [43]. The membership function characterizes the difference between crisp (or hard) and fuzzy sets. In a crisp set, the membership function can only output two choices: 0 for non-membership and 1 for full membership. Fig. 2.5(a) is the traditional crisp set concept. The membership grade of cluster c_1 or c_2 is either 0 or 1. The concept of a fuzzy set softens this constraint and allows for partial membership. Aside from 0 and 1, each sample can hold intermediate membership grades, signifying partial membership in two or more classes. Fig. 2.5(b) shows overlap between the two

¹⁰SPOT is a high-resolution, optical imaging earth observation satellite system.

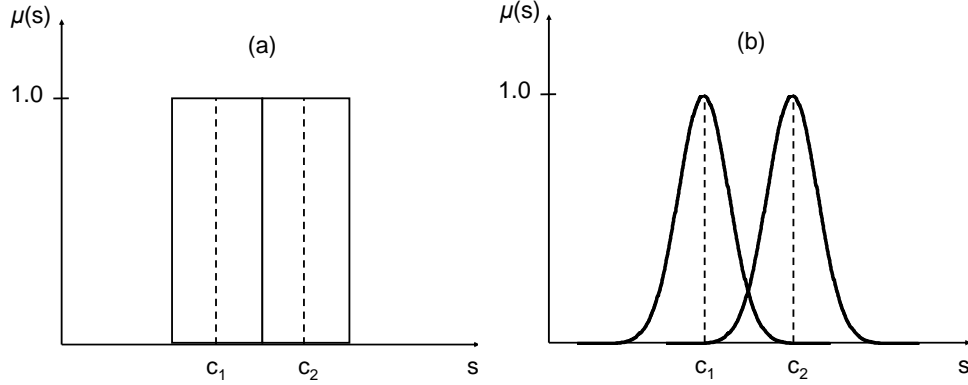


Figure 2.5: (a) In the traditional crisp set concept, the membership grade of cluster c_1 or c_2 is either 0 or 1. (b) In fuzzy set theory, overlap between the two clusters is allowed [43].

clusters. Cluster c_1 and c_2 may share samples, and the membership grade of a sample will generally decrease as the distance between the sample and a given cluster center increases.

This research implements the fuzzy c-means clustering algorithm. It is an iterative algorithm that separates data clusters with fuzzy means and fuzzy boundaries and is less dependent on the initial state of the cluster centers than the traditional k-means. Fig. 2.6 illustrates the FCM clustering algorithm. At each iteration, FCM adjusts the probability of cluster memberships for each point. For illustration purpose, the cluster centers are not randomly initialized, but are located approximately in the center of the data set. Starting from the green cluster centers, the algorithm converges to the blue cluster centers after seven iterations. In general, the performance of fuzzy clustering methods is superior to that of corresponding hard versions (e.g., k-means clustering), and they are less likely to get stuck in a local minima [43]. The FCM algorithm is described as follows [43].

Let $\mathbf{S} = \mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_M$ be a finite subset of \mathbf{R}^N , the N -dimensional real number vector space. Let integer $K, M \geq K > 2$, denote the number of fuzzy subsets. Thus, a fuzzy K partition of \mathbf{S} can be represented by a $K \times M$ matrix \mathbf{U} in which each entry of \mathbf{U} , denoted by u_{km} , satisfies the following constraints:

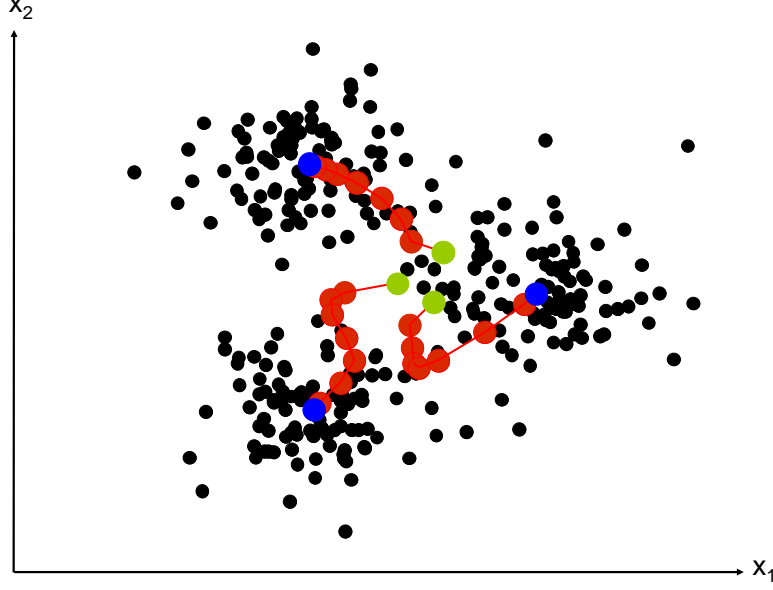


Figure 2.6: At each iteration of the fuzzy c-means clustering algorithm, the probability of cluster membership for each prototype vector is adjusted according to Eqs. (2.3) and (2.4) (here $b = 2$). After seven iterations, the algorithm has converged to the blue cluster centers.

$$u_{km} \in [0, 1] \text{ and } \sum_{k=1}^K u_{km} = 1, \forall m. \quad (2.1)$$

In the case of image classification, M is the number of pixels, and K is the number of classes. The membership values of one pixel must sum to 1, as specified in Eq. (2.1).

The FCM algorithm uses a clustering criterion based on minimizing the generalized within-groups sum of square error function $J_b(\cdot)$:

$$J_b(\mathbf{U}, \mathbf{M}) = \sum_{m=1}^M \sum_{k=1}^K (u_{km}) \cdot \|\mathbf{s}_m - \boldsymbol{\mu}_k\|^2. \quad (2.2)$$

$\mathbf{M} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K)$ is the vector of cluster centers (i.e., the means of the clusters), with $\boldsymbol{\mu}_k \in \mathbf{R}^N$, and b is the membership weighting exponent, $1 \leq b < \infty$. For $b > 1$ and $\mathbf{s}_m \neq \boldsymbol{\mu}_k$, a local minimum of J_b is achieved if:

$$u_{km} = \frac{1}{\sum_{j=1}^K \left(\frac{\|\mathbf{s}_m - \boldsymbol{\mu}_k\|}{\|\mathbf{s}_m - \boldsymbol{\mu}_j\|} \right)^{2/(b-1)}} \quad \forall m, \quad (2.3)$$

and the k^{th} cluster's mean is calculated as:

$$\boldsymbol{\mu}_k = \frac{\sum_m (u_{km})^b \cdot \mathbf{s}_m}{\sum_m (u_{km})^b} \quad \forall k. \quad (2.4)$$

The FCM clustering is performed by iteratively applying Eqs. (2.3) and (2.4). As $b \rightarrow 1$, a pixel's membership grades become closer to 1 or 0. The greater the value of b (e.g., 2 or more), the membership grades move from the crisp '0' or '1' membership assignment to a fuzzy assignment, which allows each of the M -items to belong partially to each of the K clusters.

2.2.3.2 Self-Organizing Map (SOM) Clustering. Because of the efficiency of the human eye and brain combination in solving pattern recognition problems, researchers in this field considered whether computer systems based on a simplified model of the brain can be more effective than standard statistical classification methods [43]. Such research led to the adoption of artificial neural networks (ANN). ANNs are information-processing devices based on heuristically conceived and biologically inspired simple components. Neural-network computing methods are highly effective and economical when [22]:

- Data are not always describable by low-order (first and second order) statistical parameters.
- Their distributions are non-Gaussian.
- Their statistics are nonstationary.
- The functional relations between data elements are nonlinear.

An advantage of neural networks lies in the high computation rate achieved by their massive parallelism, resulting from a dense arrangement of interconnections (weights) and simple processors (neurons) [43]. The high computation rate allows for real-time processing of very large data sets. The performance of a neural network depends to a significant extent on how well it has been trained, and not on the adequacy of assumptions concerning the statistical distribution of the data, as is the case with the maximum likelihood classifier, the minimum classification error classifier (i.e., minimum risk), Linear Discriminant Function (LDF)/Quadratic Discriminant Function (QDF), and others. During the training phase, the neural network “learns” about regularities present in the training data, and based on these regularities, constructs the rules that can be extended to the unknown data.

One kind of fundamental neural network architecture is Kohonen’s self-organizing (feature) map [22]. The SOM, regarded as a simple competitive-learning network, learns to recognize groups of similar input vectors in such a way that neurons physically near each other in the output layer respond to similar input vectors [12]. In the pure form, the SOM defines an “elastic net” of points that are fitted to the input signal space to approximate its density function in an ordered fashion. It converts the nonlinear statistical relationships between high-dimensional data in the input data space \mathbf{R}^N into simple geometric relationships of their image points on a low-dimensional display, usually a regular two-dimensional grid of nodes or neurons [22].

Fig. 2.7 is an example of the topology of a SOM. It consists of three layers [43]: the input layer (sensory cortex) with two neurons, the linking weights (topological feature space), and the output layer (mapping cortex) made up of a grid of 5×5 neurons equally spaced on the grid. The number of input neurons is equal to the number of features or dimensions. As for the output neurons, there are no clear rules about their number. Generally, the output layer of a SOM is a two-dimensional layer made up of $m \times n$ ($n, m > 1$) neurons, spaced apart at a Euclidean distance of unity (rectangular lattice), with each neuron relating to a fixed position in the two-dimensional output space. Synaptic weights link the neurons in the input layer

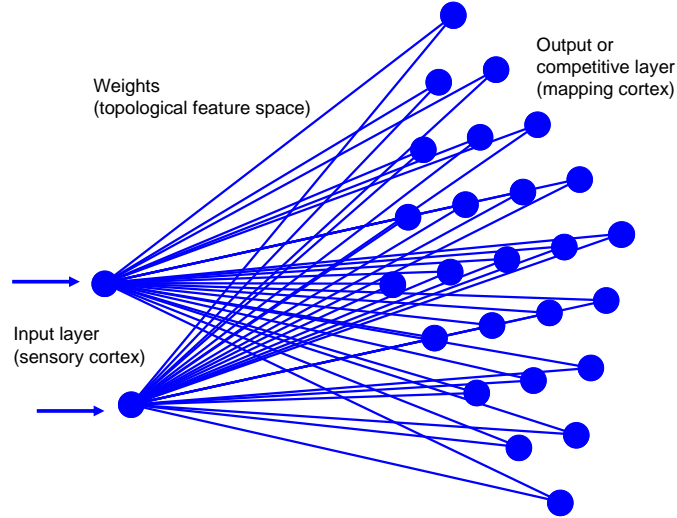


Figure 2.7: The SOM topology consists of the input layer, linking weights, and output layer. The number of neurons in the input layer is equal to the number of features or dimensions. The output layer is generally a two-dimensional $n \times m$ ($n, m > 1$) neurons. Weight vectors link the neurons in the input layer and output layer.

and the output layer. They are initialized randomly and are then continually updated during training in order to organize the relationships among the input patterns. Once the training is complete, the final weights that are close together will have similar magnitude.

The SOM algorithm defines a special recursive regression process, in which only a subset of models is processed at every step [22]. It associates a parametric model vector, also called a weight, reference, or prototypical vector $\mathbf{m}^\ell = [\zeta_1^\ell, \zeta_2^\ell, \dots, \zeta_N^\ell]^T \in \mathbf{R}^N$, with every neuron ℓ . Before recursive processing, the algorithm randomly initializes each weight vector \mathbf{m}^ℓ . In the long run, the \mathbf{m}^ℓ will attain two-dimensionally ordered values. This is the basic effect of self-organization.

The lattice type of the array can be defined to be rectangular, hexagonal, or even irregular. An input vector $\mathbf{s} = [\xi_1, \xi_2, \dots, \xi_N]^T \in \mathbf{R}^N$ is connected to all neurons in parallel via variable scalar weights ζ_j^ℓ , which are in general different for different neurons. In an abstract scheme, it may be imagined that the input \mathbf{s} , by means of some parallel computing mechanisms, is compared with all the \mathbf{m}^ℓ , and the *location*

of *best match* in some metric is defined as the location of the “response.” The exact magnitude of the response need not be determined: the input is simply mapped onto this location. In many practical applications, the smallest of the Euclidean distances $\|\mathbf{s} - \mathbf{m}^\ell\|$ can be made to define the best-matching neuron, denoted by w :

$$w = \arg \min_{\ell} \left(\sqrt{\sum_{n=1}^N (\mathbf{s}_n - \mathbf{m}_n^\ell)^2} \right), \quad (2.5)$$

where ℓ is the index of the SOM lattice neurons. Each dimension can have an associated weighting factor, where the weights of all dimensions are real-valued on the interval $[0, 1]$. This weighting is often used as a binary mask for excluding certain spectral dimensions from the best-matching neuron-finding process (1 for include, 0 for exclude). The distance metric becomes:

$$w = \arg \min_{\ell} \left(\sqrt{\sum_{n=1}^N \beta_n (\mathbf{s}_n - \mathbf{m}_n^\ell)^2} \right), \quad (2.6)$$

where β_n is the mask value of dimension n . This research does not perform spectral extraction or dimensionality reduction, and all dimensions are assumed equally important; therefore, $\beta = 1$ for all N dimensions.

During learning, the weight vectors that are topographically close in the array up to a certain geometric distance will activate each other to learn something from the same input \mathbf{s} (Figure 2.8). This adaptation procedure will result in a local relaxation or smoothing effect on the weight vectors of neurons in this neighborhood, which in continued learning leads to global ordering.

The SOM update rule for the weight vector of neuron ℓ is:

$$\mathbf{m}^\ell(t+1) = \mathbf{m}^\ell(t) + h^{w^\ell}(t)[\mathbf{s}(t) - \mathbf{m}^\ell(t)], \quad (2.7)$$

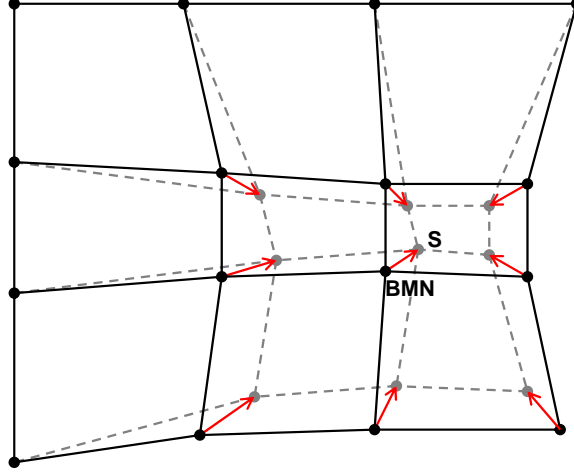


Figure 2.8: The black dots represent the weights of the output neurons. The best matching neuron (BMN) and some of the surrounding weights of the input vector \mathbf{s} are updated according to Eq. (2.7). In the figure, the BMN corresponds to the nearest prototype vector \mathbf{m}^ℓ to the input vector \mathbf{s} . The input vector \mathbf{s} is physically located at ‘s.’ The solid and dashed lines correspond to the weight vectors before and after updating, respectively.

where $t = 0, 1, 2, \dots$ is an integer, discrete-time coordinate. The input vector $\mathbf{s}(t)$ is taken in the order in which it appears in the data set at time t . In the relaxation process, the function $h^{w\ell}(t)$ has a very central role: it serves as the neighborhood function, a smoothing kernel defined over the lattice points. For convergence, it is necessary that $h^{w\ell}(t) \rightarrow 0$ when $t \rightarrow \infty$. Usually

$$h^{w\ell}(t) = h(\|\boldsymbol{\eta}^w - \boldsymbol{\eta}^\ell\|, t), \quad (2.8)$$

where $\boldsymbol{\eta}^w \in \mathbf{R}^2$ and $\boldsymbol{\eta}^\ell \in \mathbf{R}^2$ are the location vectors of neurons w and ℓ , respectively, in the array. With increasing $\|\boldsymbol{\eta}^w - \boldsymbol{\eta}^\ell\|$, $h^{w\ell} \rightarrow 0$. The average width and form of $h^{w\ell}$ define the “stiffness” of the “elastic surface” to be fitted to the data points.

In the literature, a widely applied neighborhood kernel can be written in terms of the Gaussian function,

$$h_{w\ell}(t) = \alpha(t) \cdot \exp\left(-\frac{\|\boldsymbol{\eta}^w - \boldsymbol{\eta}^\ell\|^2}{2\sigma^2(t)}\right), \quad (2.9)$$

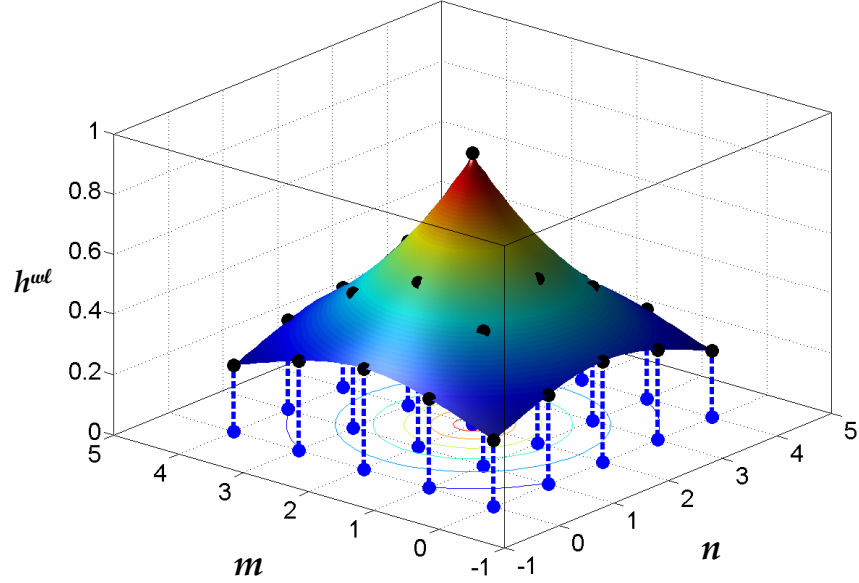


Figure 2.9: In this example, the blue ‘•’ on the m - n plane are the neurons of the two-dimensional 5×5 self-organizing map. The surface is the Gaussian function where $\alpha = 0.9$, $\sigma = 1$, and winning neuron w is located at $[2,2]$. The multiplier $h^{w\ell}$ in Eq. (2.7) is the intersection (represented by the black ‘•’) between the vertical dashed line and the surface of the Gaussian function for each neuron ℓ .

where $\alpha(t)$ is a scalar-valued learning-rate factor, and the parameter $\sigma(t)$ defines the width of the kernel. The latter corresponds to the radius of the neighborhood array points around node w . A reasonable choice for the learning-rate is given in Eq. (2.10),

$$\alpha(t) = \alpha_0 \left(1 - \frac{t}{T}\right), \quad (2.10)$$

where T is the training length and α_0 is the initial learning rate. Both $\alpha(t)$ and $\sigma(t)$ are monotonically decreasing functions of time. Fig. 2.9 shows an example of a Gaussian neighborhood function.¹¹ The Gaussian function $h^{w\ell}$ is at its maximum when $w = \ell$ and decreases radially as one gets farther away from the winning neuron w on the grid.

¹¹Note that since α is large ($\alpha = 0.9$), σ would also have to be large, at least half the diameter ($\sigma \approx 3$). This would cause the “bell” shape to be almost flat. To emphasize the shape of the bell curve in the figure, σ is small ($\sigma = 1$).

The self-organization process consists of two phases. The first phase, also called the ordering phase, uses relatively large initial learning rate α_0 and neighborhood radius σ_0 (at least half the diameter of the network) for fast initial convergence. Throughout this phase, the learning rate and neighborhood radius decrease to the values defined in the second phase. The neurons order themselves in the input space with the same topology in which they are ordered physically [12]. The second phase, also called the tuning phase, lasts for the rest of training or adaptation. Throughout this phase, the neighborhood radius stays at the tuning neighborhood radius, which should include only close neighbors. The learning rate continues to decrease from the tuning phase learning rate, but very slowly. The small neighborhood and slowly decreasing learning rate fine-tunes the map, while keeping the ordering learned in the previous phase stable. The number of training steps for tuning should be much larger than the number of training steps in the ordering phase as the tuning phase takes much longer [46].

Fig. 2.10 is an example of the SOM's weight distribution output. It consists of 1,000 uniformly distributed samples (denoted by a '+' symbol) on a two-dimensional space within the range [0,1] on each coordinate axis. The SOM algorithm constructs the map by using two neurons in the input layer (x and y dimensions) and 5×5 output layer. The red '•' symbol indicates the location of the output neurons in terms of the associated weights in the topological feature space. The red lines show the links between spatially adjacent neurons. The algorithm initializes the weights randomly before training begins (Fig. 2.10(a)) and continually adjusts them during training. After 200 iterations, the weights start to order themselves (Fig. 2.10(b)). After 600 iterations, the weights start to expand (Fig. 2.10(c)). After 1,000 iterations, the weights are approximately uniformly distributed (Fig. 2.10(d)). The iterative process organizes the relationships among the input patterns. This is the nature of self-organization. Eventually, the neurons that are close to each other will have a similar weight magnitude.

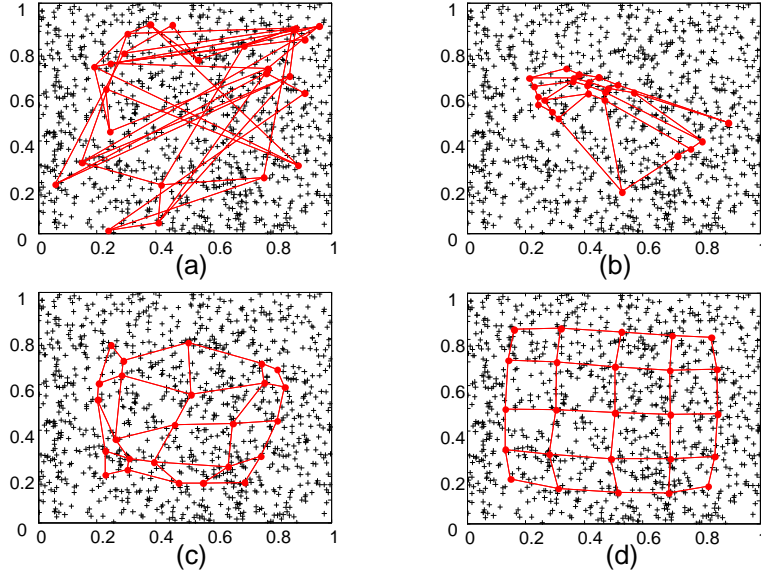


Figure 2.10: (a) Random initial weight vectors. (b) Weight vectors after 200 iterations. (c) Weights after 600 iterations. (d) Weights after 1,000 iterations.

Several works in target tracking have used the SOM, particularly for its self-organizing characteristic. In [5], the authors present a vehicle tracking algorithm based on the Karhunen-Loeve Transform (KLT) feature tracker [40],¹² which exploits a SOM to drastically reduce tracking errors arising from occlusions. Each occluded object uses a 3-neuron SOM to cluster features by speed. Tracked features are correctly separated from those undergoing a tracking error by using the SOM's ability to match input topology. In [35], the authors explore novel automatic target recognition (ATR) techniques to preprocess, track, and classify objects in sequences of IR images. Using a SOM-based classifier, objects are classified using different types of features, such as statistical (based on intensity) and shape (based on edge information) of the object. The tracking results have proven to be better than many proposed methods for IR data [35].

Since the SOM compresses information while preserving the most important topological and/or metric relationships of the primary data elements on the map

¹²The Karhunen-Loeve Transform, also known as principal component analysis, is a representation of a stochastic process as an infinite linear combination of orthogonal functions [19].

(this may be thought to produce some kind of abstractions) [22], it works well for clustering and organizing hyperspectral data obtained from pixels with complex and noisy spectral mixtures. The physical relationships among different material types and their distributions are visualized in a two-dimensional map. The measure of dissimilarity between material types can be expressed not only by the difference between their corresponding weight vectors, but also by the difference between their corresponding lattice locations.

2.2.4 Spectral Library. When creating the spectral library, several aspects of the data should be considered:

- The problem of optimally training a classifier comes down to how completely and precisely the data set is modeled. The rule for establishing the list of classes is that the classes must be [23]:
 - Of informational value. The list must contain all of the classes of interest to the information consumer.
 - Exhaustive. In addition to those desired by the user, it must contain enough additional classes so that there is a logical class to which to assign each pixel in the data set.
 - Separable. The classes must be separable in terms of available spectral features. When classes are separable, one is able to discriminate among them. A typical way of determining separability between two classes of materials is through a distance measure.
- It is well established that the geometry of a vector space changes continually as the dimensionality of the space increases. For example, a line in one-dimension turns to a square in two-dimensions to a box in three-dimensions, and so on. Furthermore, it usually requires a dimensionality of the order of 30 or more to accomplish many practical classification tasks satisfactorily.

- The detailed spectral data can be acquired in the laboratory or in the field. In the laboratory, hyperspectral data are typically measured using a spectrometer, an optical instrument used to measure properties of light over a specific portion of the electromagnetic spectrum. In the field, measurements can be obtained using either a spectrometer or a hyperspectral remote sensor.

2.3 Target Tracking

The target tracker collects sensor data from a field of view (FOV) containing one or more potential targets of interest and partitions the sensor data into sets of observations, or tracks, that are produced by the same sources [8]. Image-based target tracking systems process digital imagery and deal with imagery in units of frames, expressed as two-dimensional matrices of image pixels collected by the sensor at approximately the same time. Once the tracker forms and confirms a track, the number of targets can be estimated and quantities, such as target velocity, future predicted position, and target classification characteristics, can be computed for each track. The elements of the tracking system are *sensor data processing and measurement formation* (Sec. 2.3.1), *observation-to-track association* (Sec. 2.3.2), *track maintenance* (Sec. 2.3.3), *filtering and prediction* (Sec. 2.3.4), and *gating computations* (Sec. 2.3.5). The interaction between these system elements is shown in Fig. 2.1.

2.3.1 Panchromatic Video Sensor Data Processing and Measurement Formation. Similar to the hyperspectral input data, the measurements of interest are not the electromagnetic energy received by the panchromatic video camera. The received signal is preprocessed to remove unwanted effects due to sensor noise, thermal energy from equipment, sensor quantization, and airborne motion (jitter) (Fig. 2.11). Furthermore, an important sensor design consideration is the decision rule on the received signal intensity, so as to discriminate between returns from targets of interest and returns from extraneous sources. Two probabilities that are important parameters for the choice of decision rule are probability of false alarm and probability of

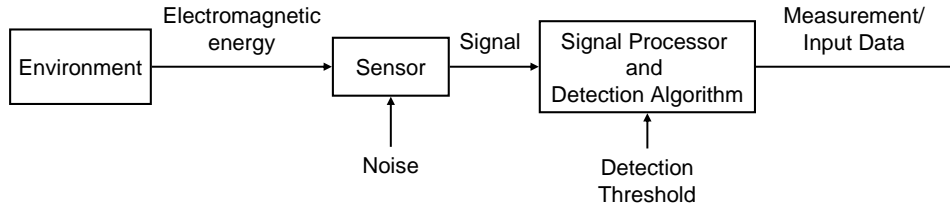


Figure 2.11: The measurements of interest are not raw data points but usually the outputs of complex signal processing and detection subsystems.

detection. The probability of false alarm is the percentage of detections that does not match the true target. Conversely, the probability of detection is the percentage of detections that match the true target. The simplest approach to the decision process is to compare the incoming signal power or intensity to a set threshold so that the probability of false alarm (P_{FA}) remains constant (called a constant false alarm rate detector). For a given threshold setting, the probability of detection (P_D) will generally be a complicated function of the sensor capabilities, the target size and distance from the sensor, and the environment (atmospheric attenuation, and so forth). There are two commonly used target measurement algorithms:

1. Centroid and edge tracking determine a point on the target by segregating target pixels from background pixels via a segmentation or gradient process [8].
 - If the size of the target image obtained from the imaging sensor appears as a “blob” (about 2-10 video pixels), centroid type trackers use the measured pixel intensities to calculate the centroid of the target. The tracker processes this information further to estimate the target motion (e.g, pixel intensity frame differencing). If the size is around 100 pixels, it is too small for feature extraction, but too large for the pixel intensities to be meaningful for centroid calculation. Feature extraction techniques require the pixel region to have enough resolved internal details, which usually occur in higher resolution imagery. Furthermore, it is too large for the pixel intensities to be meaningful for centroid calculation since the pixel intensities vary as image size becomes larger. Instead, the centroid tracker performs

image segmentation by applying a threshold to segment the scene into target and background regions [3, 8], calculates the centroid of the target in the segmented image, and uses the centroid as a point measurement [3].

- Edge tracking is similar to centroid tracking. Instead of using the centroid of the target image, it detects the leading edge (or other edges of interest). Changes in the position of the edge are input to the tracking filter.

2. Larger targets often begin to develop internal details and are more suitably tracked using correlation methods [8]. Correlation trackers encompass a large collection of tracking algorithms. These are either matched filters or an approximation of a matched filter. Once a target has been acquired, the tracker creates a target reference, or template window, from the target area of interest. At each tracking cycle, the tracker matches the reference template to the target in the incoming video image. In some cases, the reference template is updated to allow for variations in the imagery (e.g., lighting variations).

2.3.2 Observation-to-Track Association. The association function takes observation-to-track pairings that satisfy gating constraints and determines the observation-to-track assignments. Gating determines which possible observation-to-track pairings are “reasonable,” and the tracker uses an association algorithm to determine final pairings. If more than one observation exists in the gate, as shown in Fig. 2.12, this leads to association uncertainty [3]. The simplest association approach, denoted global nearest neighbor (GNN), determines a unique assignment so that at most one observation can be used to update a given track, and an observation can be used to update at most a single track. This assignment is typically made such that some cost is minimized (e.g., total summed distance) or the likelihood is maximized [7]. Another simple assignment approach is the “greedy” method [8], whereby the assignments are ranked and the best available assignments are made sequentially. The greedy assignment approach removes observations and tracks as they are assigned and thus does not allow an observation to be used more than once nor a track to be

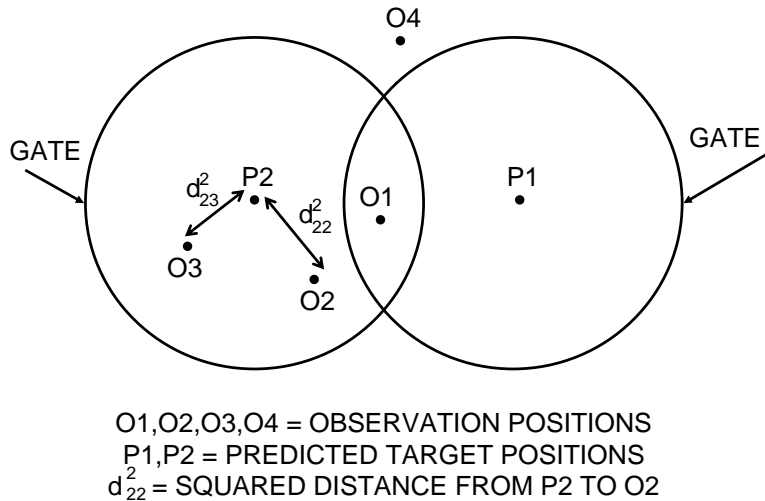


Figure 2.12: This data association example consists of two closely spaced targets (P1 and P2) and four observations (O1..O4). The potential pairings are P1-O1, P2-O1, P2-O2, and P2-O3. Since O1 is the only observation within the gate of track P1, it will likely be associated with P1, whereas P2 will be associated with either O2 or O3, which depends on the results of the association algorithm.

assigned more than one observation. The greedy approach is easy to implement; however, it can lead to a poor overall assignment solution [8].

One solution to the assignment problem originally dealt with problems in economic theory such as assigning personnel to jobs and delivery trucks to locations [4]. The objective in these problems is to minimize cost (or maximize profit) using available resources. One of the faster methods is the Bertsekas' auction algorithm [4]. Based on the efficiency in the time required to understand and program the algorithm and the solution time, the auction algorithm is the most efficient assignment algorithm currently available [8]. The Bertsekas' auction algorithm operates like an auction whereby observations bid simultaneously for tracks, thereby raising their "prices." Once all bids are in, the algorithm awards tracks to the highest bidder. Because of its computational efficiency, the one-sided Bertsekas auction algorithm is the association algorithm implemented in this research. The rest of this section describes the auction algorithm.

Consider B_o observations and B_t tracks. The objective of the assignment process is to divide among the B_o observations the B_t tracks by means of an auction. For each observation i , there is a nonempty subset $\Delta(i)$ of tracks that can be assigned to i . An assignment E is a (possibly empty) set of observation-track pairs (i, j) such that $j \in \Delta(i)$ for all $(i, j) \in E$. For each observation i there is at most one pair $(i, j) \in E$, and for each track j there is at most one pair $(i, j) \in E$. In the context of a given assignment E , observation i is assigned if there exists a track j such that $(i, j) \in E$; otherwise i is unassigned. Similar terminology is used for tracks. The algorithm provides a complete assignment when every observation is assigned to a distinct track. There is a given integer value δ_{ij} that an observation i associates with a track $j \in \Delta(i)$. The goal is to find a complete assignment that maximizes

$$\sum_{(i,j) \in E} \delta_{ij} \tag{2.11}$$

over all complete assignments E . This is called the primal assignment problem [4].

For each track j , the “price” of j is denoted p_j . The vector with coordinates p_j , $j = 1, \dots, B_t$ is called a price vector. For a given price vector \mathbf{p} , the profit margin of observation i corresponding to \mathbf{p} is

$$\pi_i = \max_{j \in \Delta(i)} \{ \delta_{ij} - p_j \}, \tag{2.12}$$

It is helpful to think of p_j as the cost an observation incurs when assigned to track j . Therefore, for a given price vector \mathbf{p} , $\delta_{ij} - p_j$ may be thought of as the benefit that observation i associates with being assigned to track j .

A dual problem to the assignment problem is minimizing

$$\sum_{i=1}^N \pi_i + \sum_{j=1}^N p_j \tag{2.13}$$

subject to $\pi_i + p_j \geq \delta_{ij}$, $\forall i$, and $j \in \Delta(i)$. For a given price vector \mathbf{p} , the cost of this problem is minimized when π_i satisfies Eq. (2.12). The Bertsekas' algorithm allows for observations to be assigned to tracks that come within ϵ of attaining the maximum in Eq. (2.12), given

$$\pi_i - \epsilon \leq \delta_{ij} - p_j \leq \pi_i, \text{ for each } (i, j) \in E, \quad (2.14)$$

where π_i is given by Eq. (2.12), and ϵ is a nonnegative constant.

The algorithm begins with $\epsilon > 0$ and fixed, some assignment E (possibly empty), and price vector \mathbf{p} satisfying Eq. (2.14). It proceeds iteratively and terminates when a complete assignment is obtained. At the end of the iteration, E and \mathbf{p} are updated while maintaining Eq. (2.14). Each iteration consists of a *bidding* and *assignment* phase:

Bidding phase. For each unassigned observation i in the assignment E :

1. Compute the “current value” of each track $j \in \Delta(i)$ given by¹³

$$v_{ij} = \delta_{ij} - p_j. \quad (2.15)$$

2. Find a “best” track j^* having maximum value

$$v_{ij^*} = \max_{j \in \Delta(i)} v_{ij}, \quad (2.16)$$

and find the best value offered by tracks other than j^* .

$$w_{ij^*} = \max_{j \in \Delta(i), j \neq j^*} \delta_{ij} - p_j. \quad (2.17)$$

3. Compute the “bid” of observation i for track j^* given by

¹³The “current value” v_{ij} is the value that comes within ϵ of the profit margin π_i .

$$b_{ij^*} = p_{j^*} + v_{ij^*} - w_{ij^*} + \epsilon = \delta_{ij^*} - w_{ij^*} + \epsilon. \quad (2.18)$$

Assignment phase. For each track j , let $\mathbf{P}(j)$ be the set of observations from which j received a bid in the bidding phase of the iteration. If $\mathbf{P}(j)$ is nonempty, increase p_j to the highest bid by the following:

$$p_j = \max_{i \in \mathbf{P}(j)} b_{ij}. \quad (2.19)$$

Any pair (i, j) (if one exists) is removed from the assignment E , and the pair (i^*, j) is added to E where i^* is some observation in $\mathbf{P}(j)$ attaining the maximum in Eq. (2.19).

2.3.3 Track Maintenance. *Track maintenance* deals with track initiation, confirmation, and deletion. A simple approach to track initiation is to start a new track, also known as a tentative track, for every unassociated observation.¹⁴ A more preferred method, used with multiple hypothesis tracking (MHT), will start tentative tracks on all observations and use subsequent data to determine which of these newly initiated tracks are valid [8]. In Fig. 2.12, O4 will not be associated with any track; therefore, it is a candidate for a new track.

Once a tentative track is formed, confirmation logic is usually required because the probability of a single observation being from an extraneous source is too high for immediate confirmation [8]. A typical simple rule for track confirmation is that B correlating observations should be received within K frames. However, a much better approach is to define a track score function and compare this score with an appropriately chosen track confirmation threshold.

Similarly, deletion logic provides a means to delete a false target track, which is a track that is not updated within some reasonable interval [8]. If a sufficiently long

¹⁴At system initialization, each observation initiates a track.

time elapses without detection, the target will likely no longer be within the frame. A typical simple rule is to delete a track after K_D consecutive frames have produced no updating observations. Again, however, the use of a track score function is more general, and the track score reflects the quality of the update so that updates that barely satisfy the gate may actually decrease the score.

The typical track score function used in modern MTT systems is a log likelihood ratio for use in evaluation the hypothesis H_1 , and H_0 defined as [7]:

H_1 : the observations contained in the track were produced by a single (target) source

H_0 : the observations contained in the track were produced by random false alarms (noise or clutter)

Because the track score is a log likelihood ratio, determining track confirmation (accept H_1) versus track deletion (accept H_0) is an application of the classical sequential probability ratio test (SPRT) [47]. Thus, track confirmation and deletion thresholds and system performance predictions follow directly from SPRT theory.

2.3.4 Filtering and Prediction. The filtering step incorporates the assigned observations into the updated track parameter estimates. For those tracks that are not assigned an observation, the previous predicted estimates become the filtered estimates. Predictions are then made to the time when the next data frame is expected. Hence, prediction quantities are of great importance because they define the center of the gated region. The size of the gate is also directly affected by the prediction uncertainty, which can be determined by the Kalman filter [8].

A Kalman filter is an optimal recursive estimator. It processes all available measurements, regardless of their precision, to estimate the current value of the states.¹⁵ To do this, it uses [27]:

- Knowledge of the system and measurement device dynamics.

¹⁵States are variables of interest, and in tracking, usually position and velocity.

- The statistical description of the system noises, measurement errors, and uncertainty in the dynamics models.
- Any available information about initial conditions of the states.

Furthermore, the filter only needs the estimated states from the previous time step and the current measurement to compute the estimate for the current states. No observation history is needed. Kalman filtering is most applicable if the underlying target dynamics and the measurement processes can be assumed to be linear and jointly Gaussian. In this case, estimation of the mean target state and the associated covariance matrix is all that is required to define the probability density function (PDF) associated with the target position in state space. The Kalman filter has a number of advantages when applied to the MTT problem [8]:

- The Kalman filter provides a general solution to the recursive minimized mean square estimation problem within the class of linear estimators.
- The gain sequence is chosen automatically, based on the assumed target maneuver and measurement noise models. The same filter can be used for varying target and measurement environments by changing a few key parameters.
- The Kalman gain sequence automatically adapts to changing detection histories, including varying sampling interval as well as missed detections.
- The Kalman filter provides a convenient measure of the estimation accuracy through the state covariance matrix. Having a measure of the expected prediction error variance is useful for maneuver detections, in which the Kalman filter model provides a convenient way to adjust for varying target dynamics.
- Through use of the Kalman filter, it is possible at least partially to compensate for effects of misassociation in a dense MTT environment.

The track dynamics process can be modeled in the discrete-time Markov form [8]:

$$\mathbf{x}(t+1) = \Phi \mathbf{x}(t) + \mathbf{q}(t) + \mathbf{f}(t+1|t), \quad (2.20)$$

where \mathbf{x} is the n -dimensional track state vector that includes the quantities to be estimated, Φ is the known state transition matrix, $\mathbf{q}(t)$ is the zero-mean, white Gaussian process noise with known covariance \mathbf{Q} , and $\mathbf{f}(t+1|t)$ is the known deterministic input. The discrete-time Markov process can be defined as a process in which the statistical representation of the process in the future (time $t+1$) is completely determined by the present state (time t).

Measurements are in the form of linear combinations of the system states, corrupted by uncorrelated noise. Thus, the N -dimensional measurement vector, \mathbf{z} , is modeled as [8]

$$\mathbf{z}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{v}(t), \quad (2.21)$$

where \mathbf{H} is the $N \times n$ measurement matrix, and $\mathbf{v}(t)$ is zero-mean, white Gaussian measurement noise with covariance \mathbf{R} .¹⁶

Given the the target model dynamics and measurement models from Eq. (2.20) and Eq. (2.21), the Kalman filter equations become [8]:

$$\begin{aligned} \hat{\mathbf{x}}(t+1|t) &= \Phi \hat{\mathbf{x}}(t|t-1) + \mathbf{f}(t+1|t) + \mathbf{K}_p(t)\mathbf{r}(t), \\ \mathbf{K}_p(t) &= \Phi \mathbf{P}(t|t-1)\mathbf{H}^T \mathbf{S}^{-1}, \\ \mathbf{P}(t+1|t) &= [\Phi - \mathbf{K}_p(t)\mathbf{H}]\mathbf{P}(t|t-1)\Phi^T + \mathbf{Q}, \end{aligned} \quad (2.22)$$

where $\mathbf{K}_p(t)$ is the predicted Kalman gain. The vector difference between measured and predicted quantities

¹⁶Note that in general \mathbf{Q} and \mathbf{H} may also vary with time, and thus could be indexed by t , but for notational convenience \mathbf{Q} and \mathbf{H} will not be indexed.

$$\begin{aligned}
\mathbf{r}(t) &= \mathbf{z}(t) - \mathbf{H}\hat{\mathbf{x}}(t|t-1), \\
&= \mathbf{z}(t) - \hat{\mathbf{z}}(t)
\end{aligned} \tag{2.23}$$

is the residual vector with residual covariance matrix,

$$\mathbf{S} = \mathbf{H}\mathbf{P}(t|t-1)\mathbf{H}^T + \mathbf{R}. \tag{2.24}$$

The state vector $\hat{\mathbf{x}}$ for this research is defined as

$$\hat{\mathbf{x}} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & h \end{bmatrix}^T, \tag{2.25}$$

where x and y are the two spatial dimensions, \dot{x} and \dot{y} are the corresponding spatial velocities, and h is the hyperspectral-based track class ID. The state transition matrix is

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & T_s & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.26}$$

where T_s is the sampling interval. It accounts for constant velocity and constant track class ID. The measurement model for a panchromatic observation is

$$\mathbf{H}_{video} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \tag{2.27}$$

and it updates the position states (x, y) . The measurement model for a hyperspectral observation is

$$\mathbf{H}_{HSI} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.28)$$

which updates not only the position states (x, y) , but also the track spectral ID h . The process noise covariance is defined as

$$\mathbf{Q} = \begin{bmatrix} \frac{T_s^3}{3} & 0 & \frac{T_s^2}{2} & 0 \\ 0 & \frac{T_s^3}{3} & 0 & \frac{T_s^2}{2} \\ \frac{T_s^2}{2} & 0 & T_s & 0 \\ 0 & \frac{T_s^2}{2} & 0 & T_s \end{bmatrix} q, \quad (2.29)$$

where the choice of q is considered a tuning process, determined empirically such that the dynamics model accurately represent the truth trajectories. The observation \mathbf{z} and measurement noise covariance \mathbf{R} are used to update and propagate the track states and process statistics using the Kalman filter equations in Eqs. (2.22).

If the Gaussian assumption cannot be justified, such as in the case of nonlinear target dynamics or measurement processes or both, the basic Kalman filter will not suffice. Nonlinear or linearized filters should be used instead [27]. One commonly used approach is the extended Kalman filter. It involves a linearization process whereby the nonlinear function is linearized around the current estimate.

2.3.5 Gating Computations. Most image-based trackers make use of gating to reduce the number of operations that trackers must perform by eliminating observation-to-track associations that are unlikely. The tracker processes only those observations within a specified gate. The predicted measurement for a track is the center of the kinematic gate, and observations within the gate are candidates for track update. Fig. 2.12 illustrates gating for two closely spaced targets and four observa-

tions. Note that the gates may overlap for closely spaced targets. *Gating computations* establishes kinematic gates and performs gating in the following general manner.¹⁷

Gating computations implements kinematic gating on both panchromatic video observations and the kinematic information of hyperspectral observations¹⁸ (more on this in Sec. 3.4). The gate threshold G determines the “probability that the true measurement will lie within the gated region” [3]. The true measurement conditioned on the past is normally (Gaussian) distributed with PDF given by¹⁹

$$p[\mathbf{z}(t+1)|Z^t] = N[\mathbf{z}(t+1); \hat{\mathbf{z}}(t+1|t), \mathbf{S}(t+1)]. \quad (2.30)$$

Then the true measurement will be in the following region

$$V(t+1, G) = \{ \mathbf{z} : [\mathbf{z} - \hat{\mathbf{z}}(t+1|t)]^T \mathbf{S}(t+1)^{-1} [\mathbf{z} - \hat{\mathbf{z}}(t+1|t)] \leq G \}, \quad (2.31)$$

with probability determined by the gate threshold G . The region V defined by Eq. (2.31) is the gate or association region. It is also known as the ellipse (or ellipsoid) of probability concentration—the region of minimum volume that contains a given probability mass. The left hand side of the inequality in Eq. (2.31) is the Mahalanobis distance between measurement \mathbf{z} and the best estimate of this measurement $\hat{\mathbf{z}}$ [3].

Gating computations obtains the gate threshold from a table of the Chi-square distribution (Table 2.1), since the quadratic form in Eq. (2.31) that defines the gate region is Chi-square distributed with number of degrees of freedom equal to the mea-

¹⁷The system also performs spectral gating. Sec. 3.8 discusses the spectral gating implementation in detail.

¹⁸In this context, hyperspectral measurements are not synonymous with hyperspectral observations. Hyperspectral measurements refer to the hyperspectral input data in Fig. 2.1. Hyperspectral observations refer to the components within in the hyperspectral image chip (more on this in Sec. 3.4). For panchromatic video, measurement and observation are synonymous.

¹⁹The symbol $N(\mathbf{x}; \boldsymbol{\mu}, \mathbf{S})$ stands for the normal (Gaussian) PDF with argument random variable \mathbf{x} , mean $\boldsymbol{\mu}$, and covariance matrix \mathbf{S} .

	G	1	4	6.6	9	9.2	11.4	16	25
	g	1	2	2.57	3	3.03	3.38	4	5
N									
1		0.683	0.954	0.99	0.997			0.99994	1
2		0.393	0.865		0.989	0.99		0.9997	1
3		0.199	0.739		0.971		0.99	0.9989	0.99998

Table 2.1: The table provides values of the probability mass P_G given the gate threshold G and measurement dimension N .

surement dimension N . The gate probability P_G , or the probability that the (true) measurement will fall in the gate, is described by the following equation [3]:

$$P_G = P \{ \mathbf{z}(t+1) \in V(t+1, G) \}. \quad (2.32)$$

The equation $G = g^2$ refers to the number of standard deviations of the gate. The volume V of the gate region V from Eq. (2.31) corresponding to the threshold $G = g^2$ (“g-sigma” gate) is

$$V(t+1) = c_N |GS(t+1)|^{1/2} = c_N g^N |S(t+1)|^{1/2}, \quad (2.33)$$

where c_N is the volume of the unit hypersphere for dimension N ($c_1 = 2, c_2 = \pi, c_3 = \frac{3\pi}{4}$, etc.). In general,

$$c_N = \frac{\pi^{N/2}}{\Gamma(N/2 + 1)}, \quad (2.34)$$

where $\Gamma(\cdot)$ is the gamma function.

2.4 Distance Measures

A distance measure (also known as similarity measure) is a function that gives a generalized scalar distance between two points. The main idea is to use a distance measure to define the distance between class samples in order to determine a class’ nearest neighbors. *Nearest neighbor computations* determines the sample-wise dis-

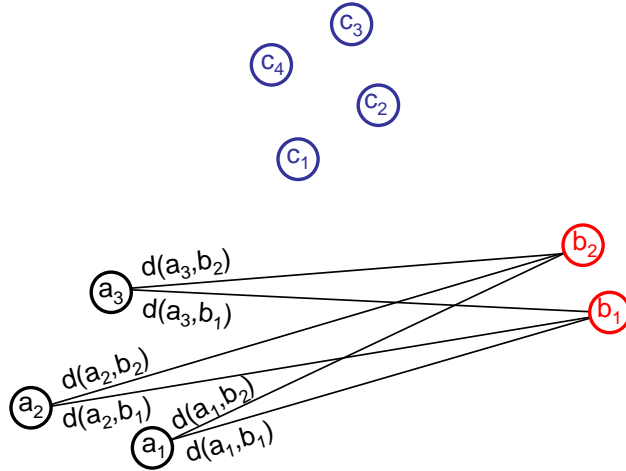


Figure 2.13: In this example, there are three classes, labeled a, b, and c. Distances between samples of class a and b, which are represented by solid lines, are computed using the distance measures described in this section.

tances for all class pairs (Fig. 2.13) and summarizes these distances in a meaningful way to provide a scalar distance for each class pair.

One distinction worth noting is that not all distance measures are metrics. Given two points A and B and a distance function d , a metric must have the following properties:

- Nonnegative: $d(A, B) \geq 0$
- Reflexive: $d(A, B) = 0$ if and only if $A = B$
- Symmetric: $d(A, B) = d(B, A)$
- Triangle Inequality: Given a third feature C , $d(A, B) + d(B, C) \geq d(A, C)$

Since these properties are restrictive, the *nearest neighbor computations* element uses other “non-metric” distance measures. This research evaluates distance metrics (Euclidean, Manhattan/cityblock, Chebyshev, and Canberra) and other non-metric distance measures (correlation and squared chord). Given an $M \times N$ matrix \mathbf{S} , where the rows are N -dimensional sample vectors $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_M$ of a class, the following list

defines the different distance measures between sample vector \mathbf{s}_{mc_1} of class c_1 and sample vector \mathbf{s}_{mc_2} of class c_2 ²⁰ [26]:

1. The Euclidean metric is the straight-line distance between two samples. An often underlying assumption is that the samples are in Euclidean space, which makes the Euclidean distance an appropriate distance metric. Although commonly used in practice, the Euclidean distance has some shortcomings. First, it assumes the Euclidean assumption is valid. Second, in the case of noisy data, it accumulates the noise power. This is particularly troublesome for high-dimensional noisy data, as is the case with hyperspectral data (especially in the visible blue and short-wave infrared regions). The Euclidean distance is a special case of the more generic l_p -norm, where $p = 2$ and the squared form is defined as

$$dist(\mathbf{s}_{c_1}, \mathbf{s}_{c_2})^2 = (\mathbf{s}_{c_1} - \mathbf{s}_{c_2})(\mathbf{s}_{c_1} - \mathbf{s}_{c_2})^T. \quad (2.35)$$

2. Manhattan distance, also known as the ‘‘Taxicab’’ distance, means to take a ‘city walk’ to get from one point to another. One can only move along the direction of an orthogonal axis from the starting point to the final destination. The Manhattan distance is also a metric and is defined as

$$dist(\mathbf{s}_{c_1}, \mathbf{s}_{c_2}) = \sum_{k=1}^N |x_{c_1k} - x_{c_2k}|, \quad (2.36)$$

where $|\cdot|$ is the absolute value.

3. The correlation distance is one minus the sample correlation between points (treated as sequences of values). It measures the disagreement between s_{c_1} and s_{c_2} and is defined as

²⁰For notational convenience, m is dropped from the expressions.

$$dist(\mathbf{s}_{c_1}, \mathbf{s}_{c_2}) = 1 - \frac{(\mathbf{s}_{c_1} - \tilde{\mathbf{s}}_{c_1})(\mathbf{s}_{c_2} - \tilde{\mathbf{s}}_{c_2})^T}{[(\mathbf{s}_{c_1} - \tilde{\mathbf{s}}_{c_1})(\mathbf{s}_{c_1} - \tilde{\mathbf{s}}_{c_1})^T]^{1/2}[(\mathbf{s}_{c_2} - \tilde{\mathbf{s}}_{c_2})(\mathbf{s}_{c_2} - \tilde{\mathbf{s}}_{c_2})^T]^{1/2}}, \quad (2.37)$$

where $\tilde{s}_{c_1} = \frac{1}{N} \sum_k s_{c_1 k}$ and $\tilde{s}_{c_2} = \frac{1}{N} \sum_k s_{c_2 k}$.

4. The Chebyshev distance, also known as the l_∞ norm, is a metric where the distance between two vectors is the largest difference (i.e., made into positive values) along any coordinate dimension. It is defined as

$$dist(\mathbf{s}_{c_1}, \mathbf{s}_{c_2}) = \lim_{p \rightarrow \infty} \left(\sum_{k=1}^N |s_{c_1 k} - s_{c_2 k}|^p \right)^{1/p}. \quad (2.38)$$

5. The Canberra metric makes a summation of a series of ratios between corresponding planar values. It considers not only the distance between two points, but also its relation to the ‘origin.’ It is defined as

$$dist(\mathbf{s}_{c_1}, \mathbf{s}_{c_2}) = \sum_{k=1}^N \left[\frac{|s_{c_1 k} - s_{c_2 k}|}{|s_{c_1 k} + s_{c_2 k}|} \right]. \quad (2.39)$$

6. The squared chord is a signal-to-noise dissimilarity measure, defined as

$$dist(\mathbf{s}_{c_1}, \mathbf{s}_{c_2}) = \sum_{k=1}^N (\sqrt{s_{c_1 k}} - \sqrt{s_{c_2 k}})^2 \quad (2.40)$$

2.5 Distances between Data Distributions

The previous section provides several ways to represent the hyperspectral data directly in terms of distance (similarity) measures between pairs of samples. This section discusses two categories that summarize these distances. The *nearest neighbor computations* element of Fig. 2.1 uses the “summary” distances to determine the nearest neighbors for each class.

2.5.1 Methods using Functional Form. Two common methods that measure the similarity of two probability distributions are Kullback-Leibler divergence and the Bhattacharyya distance. Both are statistical distance measures and require a functional form of the probability distributions. Since the distribution of the hyperspectral data is not well-modeled as Gaussian, accurate parametric modeling of the 195-dimensional joint PDF is difficult to achieve, especially since a large number of samples is required. Hence, these methods are not appropriate for this research.

2.5.2 Linkage Distances. Clustering algorithms makes no assumptions about the statistical distribution of the data. Instead, they require a similarity measure between (disjoint) classes of samples, based on the pairwise similarities among the samples of class pairs (provided in Sec. 2.4). The similarity measure between disjoint classes are called linkage distances. Three common linkage distances are [18]:

1. Single linkage takes the interclass similarity to be that of the closest pair:

$$d(c_1, c_2) = \min(\text{dist}(\mathbf{s}_{c_1}, \mathbf{s}_{c_2})), c_1 \in (1, \dots, C), c_2 \in (1, \dots, C). \quad (2.41)$$

2. Average linkage uses the average similarity between classes:

$$d(c_1, c_2) = \frac{1}{M_{c_1} M_{c_2}} \sum_{u=1}^{M_{c_1}} \sum_{v=1}^{M_{c_2}} \text{dist}(s_{uc_1}, s_{vc_2}). \quad (2.42)$$

3. Centroid linkage uses the Euclidean distance between centroids of each class pair:

$$d(c_1, c_2) = \|\bar{\mathbf{s}}_{c_1} - \bar{\mathbf{s}}_{c_2}\| \quad (2.43)$$

where $\bar{\mathbf{s}}_{c_1} = \frac{1}{M_{c_1}} \sum_{u=1}^{M_{c_1}} \mathbf{s}_{uc_1}$, $\hat{\mathbf{s}}_{c_2}$ is defined similarly.

2.6 Performance Analysis

As discussed in Sec. 1.1, the main goal of this research is to answer the question: *When augmented by hyperspectral data, is the performance of the kinematic-only tracker improved?* Performance analysis provides a quantitative comparison between the performance of the kinematic-only and the hyperspectral-augmented trackers. The analysis can be divided into classification and hyperspectral-augmented tracking performance:

2.6.1 Classification Performance. The performance of the tracker, i.e. how well the tracker is tracking the true target of interest, is affected by the accuracy of the classification. In the context of image classification [10], accuracy defines “correctness” and measures the agreement between truth and prediction. If the classified image corresponds closely with truth, it is said to be “accurate.” The usefulness of a classified image is related not only to its correctness, but also to the precision with which the user can make statements about specific points depicted on the image. For example, this research categorizes vehicles based on color, make, and model, as opposed to blue, red, or black vehicle classes. Clearly, it is more difficult to assign detailed classes correctly than to assign general classes correctly.

Classification error is the assignment of a pixel belonging to one class²¹ to another class during the classification process [10]. There are two common ways of reporting classifier performance [24]:

1. Non-equal weighted or traditional/biased classification accuracy (NEWA)

$$\text{NEWA} = \sum_{m=1}^M \sum_{c=1}^C \frac{1(\text{ if } x_m^\ell \in X_c \text{ and } \ell = c)}{M_c} \quad (2.44)$$

where c is the class under evaluation, M_c is the number of samples in class c , and 1 is an indicator function that evaluates to 1 if x_m^ℓ is in class X_c and the

²¹Correct pixel class assignment is determined by truth.

predicted class label $\ell = c$. NEWA is easy to compute; however, it does not consider the importance of each class. The results are therefore biased towards large classes.

2. Equal-weighted classification accuracy (EWA)

$$\text{EWA} = \frac{1}{C} \sum_{m=1}^M \sum_{c=1}^C \frac{(1 \text{ if } x_m^\ell \in X_c \text{ and } \ell = c)}{M_c}. \quad (2.45)$$

EWA allows each class to participate equally in the evaluation of the classifier, regardless of its size. Though the weighting may not be the most appropriate for each class, EWA is widely used in the literature and provides better evaluation than NEWA [24].

The standard form for reporting the classification error is the error matrix (also referred to as the confusion matrix or contingency table) because it identifies not only errors for each class, but also misclassifications²² by class [10]. To construct the error matrix, the true and predicted class memberships are evaluated on a pixel-by-pixel basis. The error matrix consists of a $C \times C$ array, where C represents the number of classes. The column of sums on the right-hand edge of the matrix, known as row marginals, gives total numbers of pixels in each class in the true image; the row of sums at the bottom, known as column marginals, shows total pixels in each class in the classified image. The sequence of values that extends from the upper left to the lower right corner is referred to as a diagonal and shows the number of correctly classified pixels on a class-by-class basis.²³ The sum of the diagonals is given in Eq. (2.46),

$$\text{Diagonal Sum} = \sum_{c=1}^C X_{c,c}. \quad (2.46)$$

An example is shown in Fig. 2.14. From the error matrix, the NEWA and EWA can be calculated:

²²Misclassifications are due to confusion between classes.

²³Nondiagonal values provide the distribution of classification errors.

		Classified As			Row Marginals
		A	B	C	
Truth	A	16	3	2	21
	B	4	11	0	15
	C	1	5	18	24
Column Marginals		21	19	20	Sum of Diagonals = 45

Figure 2.14: In this problem, there are 3 classes, labeled A, B, and C. The table tabulates the hypothetical classification results. Column marginals sum the total pixels classified as each class in the image. Row marginals sum the true total pixels for each class. The lower right number is the sum of the diagonal entries, given by Eq. (2.46).

$$NEWA = \frac{45}{21 + 15 + 24} \times 100 = 75\% \quad (2.47)$$

$$EWA = \frac{\frac{16}{21} + \frac{11}{15} + \frac{18}{24}}{3} \times 100 = 74.8\% \quad (2.48)$$

As seen from Eqs. (2.47) and (2.48), EWA often judges harder than NEWA. The higher the accuracy, the more samples belonging to a given class are actually classified correctly. Furthermore, high accuracy means that bias is low²⁴ and that the variability of estimates is low.

2.6.2 Hyperspectral-Augmented Tracking Performance. Evaluation of MTT systems uses measures of effectiveness (MOEs) [8]. This research performs a Monte Carlo evaluation in which truth target trajectories are known and are used to generate observation data so that the correct (target-generated) observations are known. A track-to-truth assignment method is the first step in the evaluation of a tracking system. The auction algorithm performs track-to-truth assignment for the confirmed tracks. The assignment solution produces track-to-truth pairings as well as tracks and targets that are not paired. Then, a measure of fit (MOF) between these tracks

²⁴The estimated values are consistently close to an accepted reference value.

and truth is computed. Those pairings that satisfy an acceptable criterion for the MOF, such as a kinematic gate on the Mahalanobis distance, populate a global track-to-truth assignment matrix. The track-to-truth assignments are used by difference computations between true target and track estimated quantities. Difference computations lead to computation of metrics and error statistics. An example of a track metric is a relatively simple numerical score that can be given to each Monte Carlo run using the track-to-truth assignment matrix. An appropriate time interval (Δt) is defined and a point is awarded at each interval time to each target for which the currently assigned track is the same as the assigned track Δt before. No points are awarded if there is no assigned track at either time or if there was a switch over the time interval. Thus, rapid track confirmation is rewarded, and track switching is penalized.

Other MOEs that use observation data are also useful. They are required for systems in which observation attributes (e.g., hyperspectral data) are also used for target type identification/discrimination [8]. One such MOE is the probability of correct target identification, which determines the percentage of frames that a track is correctly identified.

2.7 Summary

This chapter not only reviews several related literature works, but also describes the various system elements and their functions. Furthermore, it discusses several commonly used methods and the rationale behind their use. The next chapter discusses the methodology in the context of these methods, provides various innovative procedures and rules, and expounds on the novel ideas of this research. The methodology is presented by following the flow of the input data through the system. The chapter concludes with a discussion on the various performance measures.

III. Methodology

This chapter provides the methodology that analyzes the principles of image classification and tracking methods, rules, and postulates employed in the experimental design. It includes a collection of theories, concepts, and ideas as they relate to the proposed solution. Most importantly, it addresses the rationale and philosophical assumptions that underlie this particular research.

Aside from implementing a tracking system augmented with hyperspectral data, this thesis presents several novel ideas. First, the spectral gating work develops a method for calculating the nearest neighbors of a target class. The observation-to-track association gates a hyperspectral observation using the nearest neighbors of the track's class ID. Second, the observation-to-track association uses the sum of weighted kinematic and spectral distances as a cost function—the cost of assigning a hyperspectral observation to a track. Third, a filtering method is applied to the self-organizing map (SOM) to remove noisy samples for each vehicle class that are highly influenced by background spectra.

This chapter traces the flow of the input signals through the system, from the simulation of the synthetic sensor data to classification to data association. Fig. 3.1 provides a breakdown of the system elements. Sec. 3.1 describes *hyperspectral and panchromatic video data processing*, *HSI sensor data calibration and atmospheric correction*, and *panchromatic video sensor data processing and measurement formation*. This section discusses measurement formation, processing of hyperspectral and video input data, generation of the HSI chip, and removal of irrelevant features or dimensions (i.e., atmospheric water absorption bands). Since the system utilizes all the dimensions of the hyperspectral data, *spectral feature extraction* is a placeholder for future hyperspectral exploitation work. *Spectral matching and identification (ID)* performs two pixel classification methods on the HSI chip (Sec. 3.3) based on a predefined feature model stored in a *spectral library* (Sec. 3.2). *Observation-to-track association* determines one or more regions of contiguous hyperspectral pixels (using the class ID) known as hyperspectral observations. Furthermore, *observation-to-track associ-*

ation performs observation-to-track assignments for both hyperspectral observations and kinematic observations obtained from panchromatic video data (Sec. 3.4). *Track maintenance* uses both types of observations to initiate, confirm, and delete tracks (Sec. 3.5). In turn, *filtering and prediction* uses the observations to update existing tracks (Sec. 3.6). When needed by *data processing* and *observation-to-track association*, *filtering and prediction* provides kinematic information and process statistics for a specified track. *Gating computations* determines the gate thresholds for each existing track, which is then used by the *observation-to-track association* to eliminate observations that are unlikely for that track (Sec. 3.8). For spectral gating of hyperspectral observations, the *spectral nearest neighbors computations* provides the nearest neighbors of each class to *gating computations* (Sec. 3.7). The entire process

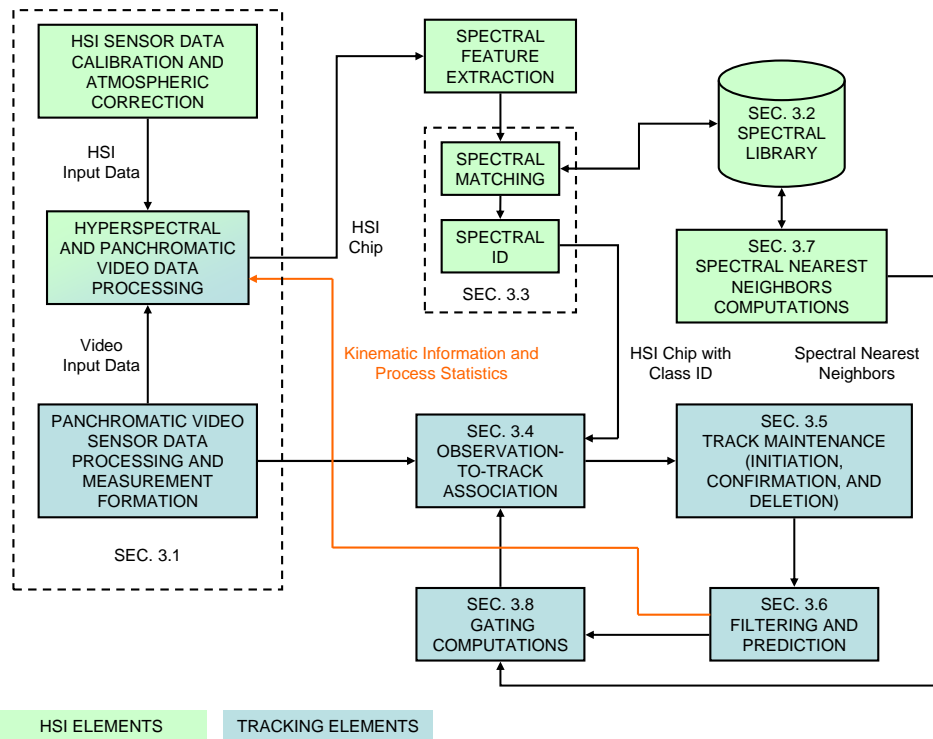


Figure 3.1: This chapter organizes the system elements based on the flow of the input signals through the entire system. Panchromatic video observations only flow through the tracking elements, whereas hyperspectral observations first flow through the hyperspectral image (HSI) elements (for classification), then through the tracking elements.

starts over again after formation of a new HSI chip or receipt of a new kinematic observation.

3.1 Hyperspectral and Panchromatic Video Measurement Formation and Data Processing

Because of signal preprocessing challenges involved with real data (e.g., difference in the frame rates of the hyperspectral and panchromatic video sensors and atmospheric correction of hyperspectral data), the system uses synthetic images of an urban environment. To create a realistic representation of the sensor data and sensor dynamics, the *measurement formation and data processing* element emulates what would be experienced in a real remote sensing situation. Two main representations that significantly impact the level of realism are the materials on the Earth's surface (Sec. 3.1.1) and the sensor dynamics (Secs. 3.1.2 and 3.1.3).

3.1.1 Scene Synthesis. As shown in Fig. 3.2, the background scene is an idealized urban setting, consisting of two main intersections, roadways with concrete and grassy medians, buildings, overhanging trees, and parking lots. The scene generator utilizes five background material types typically found in urban settings (concrete, roadway asphalt, grass, trees, and roofing tar) and generates two different images of the scene, one representing panchromatic video and the other representing hyperspectral imagery. The background panchromatic image consists of a stationary 600×600 pixel region, where the scene generator assigns one of the five background material types to each pixel. This fine grid represents a $0.2m$ spatial resolution for the video camera. The generator down-samples the panchromatic image by 5 in both spatial dimensions, creating a 120×120 hyperspectral image. Each resulting hyperspectral pixel has a $1m$ spatial resolution¹ and consists of a linear mixture of the material from

¹This resolution is typical of hyperspectral imagery for medium altitude sensors.

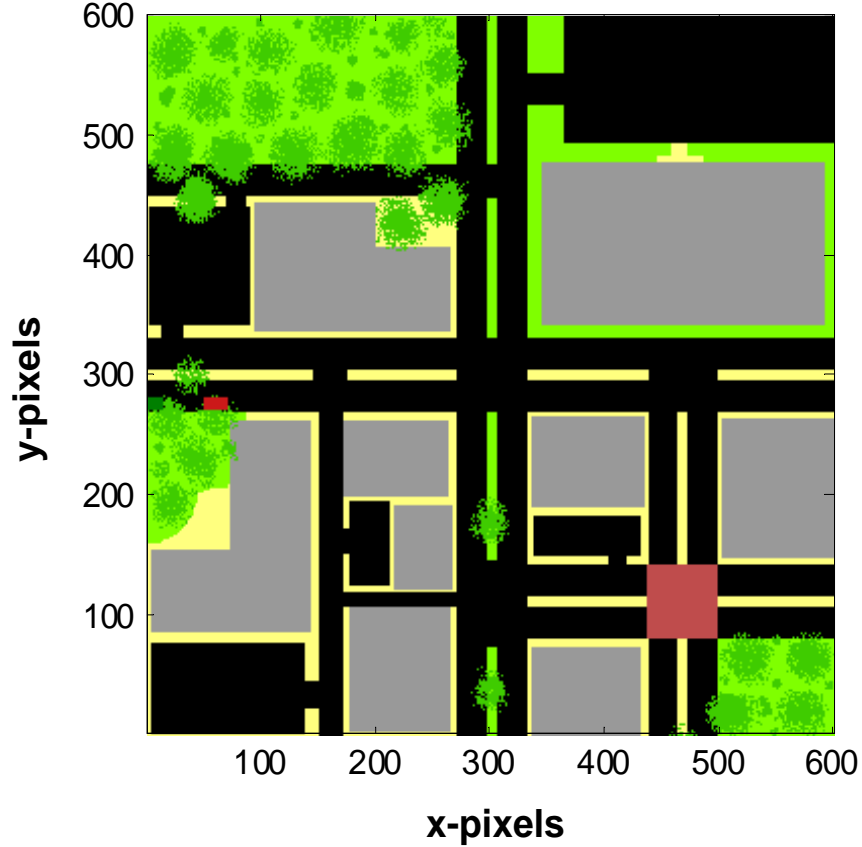


Figure 3.2: The background scene consists of a stationary 600×600 pixel region. There are two intersections; the view of the lower right intersection is occluded from above by a tar-covered roof. A gray rectangular region, representing either a parking lot or building, consists of concrete spectrum; a green region consists of grass or tree spectrum; and the roadways consist of asphalt spectrum.

the 25 panchromatic pixels within its 1×1 pixel region.² The following computes the linear mixture:

$$\mathbf{s}_{mp} = \sum_{c=1}^C \mathbf{s}_c a_c, \quad (3.1)$$

where \mathbf{s}_{mp} is the spectrum of the mixed pixel, \mathbf{s}_c is a vector of known pure spectrum, and a_c is the abundance or proportion of class c in the 5×5 panchromatic pixel region.

²This is equivalent to a 5×5 panchromatic pixel region.

The expressions $a_c \geq 0$ for $c = 1, \dots, C$ and $\sum_c a_c = 1$ constrain the abundance of Eqn. (3.1).

3.1.2 Simulation of Sensor Data Measurement Formation. The system simulates two input sources—an HSI sensor and a panchromatic video camera [6]. Both are co-boresighted and view the urban scene from a nadir³ perspective, a common practice in remote sensing. The system assumes the airborne platform that houses both sensors is hovering or loitering over the urban scene.

3.1.2.1 Hyperspectral Sensor. The hyperspectral sensor model is a traditional dispersive slit scanning instrument, using a pushbroom⁴ approach for image acquisition. The focusing slit reduces the image height to the equivalent of one pixel and the image width to a row of pixels along the slit. A scan line consists of the $1 \times N$ group of pixels. The sensor mechanically steers a mirror in a fixed direction, thus sweeping a line through the scene and exposing the imager once at each time step. With each successive hyperspectral scan line, the captured image grows from the bottom up; hence, the sensor “recycles” the mirror to a home position in order to start the next scan. This temporal disparity throughout a scene contributes to the complexity of tracking moving targets. The simulation uses parameters conservatively derived from commercially available instruments with precision mirror servo and encoder subsystems. Latencies due to moving the mirror to an adjacent scan line and exposing the imager yield 100 lines acquired per second. Commanding the mirror to travel to an arbitrary position without image acquisition employs a velocity equivalent to 240 lines traveled per second plus a $10ms$ settling latency.

This research evaluates two sensor modes, which differ in mirror control methodology. The *pushbroom* implementation simulates the continuous scan mode generally employed by this family of imagers. The hyperspectral scan mirror sweeps the entire

³Nadir is the point on any given observer’s celestial sphere diametrically opposite of one’s zenith.

⁴Pushbroom gets its name from the sweeping movement of the camera over an area.

scene in one direction, and recycles to the bottom of the scene to begin the next scan immediately. Notably, the pushbroom model does not use information from the tracker to steer the mirror to locations of existing tracks; hence, target revisit rates are sub-optimal. The *region-of-interest* (ROI) implementation maximizes target revisit rates by exploiting track state information. An ROI contains adjacent scan lines bounded by the target position and uncertainty in the state information. The simulated sensor applies realistic mirror scanning parameters to achieve realistic latencies when traveling between ROIs and when recycling to the home position. Performance comparison of the two implementations can drive future design of hyperspectral sensors. This is the goal of “performance-driven sensing.”

3.1.2.1.1 Atmospheric Correction. In this research, the extent of atmospheric correction involves only the removal of the water absorption bands from the collected data. Water in the atmosphere is the main absorber of sunlight and responsible for about 70% of all atmospheric absorption of radiation, mainly in the infrared region where water shows strong absorption. Water vapor is an important factor in hyperspectral imaging for remote sensing because it absorbs radiation differently in different spectral bands. During atmospheric correction, data are irrecoverably lost around the atmospheric water absorption bands located at 1400 and 1900nm (Fig. 3.3).

3.1.2.2 Panchromatic Video Sensor and Target Trajectory Generation.

A parametric measurement generator uses a straightforward motion segmentation algorithm. As discussed previously in Sec. 2.3.1, a centroid tracker works effectively when a target is about 2-10 video pixels. If a target is on the order of hundreds of pixels, it often has internal details or varying pixel intensity, and a more suitable tracker is a correlation tracker. Since the system simulates a target with approximately 200 pixels, it generates detections consisting of 200 panchromatic pixels. But for simplicity in the target tracking implementation (i.e., implementing a centroid tracker versus

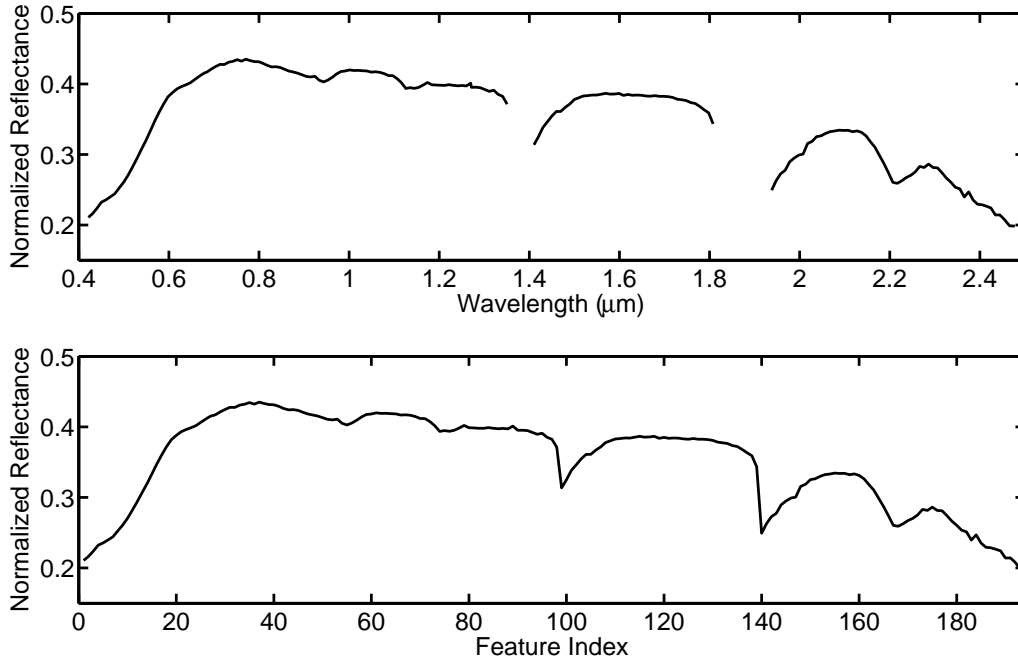


Figure 3.3: Water in the atmosphere is responsible for 70% of the known absorption of incoming sunlight, particularly in the infrared region. The top plot shows the absorption wavelength (where gaps in the curve exist) around 1400 and 1900nm. The bottom plot shows the feature index (or spectral band) after the absorption bands are removed. Each unit in the feature index is 10nm wide.

a correlation tracker), the detections are generated with uniform intensity. Since a centroid tracker works well for such detections, it is employed in this research.

The measurement generator resamples the true target trajectories at 10Hz with additive normally distributed noise. Measurement dropouts occur due to target occlusion from vegetation and structures. A user-defined setting for a minimal detectable velocity simulate a change detection algorithm. However, the system neglects other common effects, such as false measurements due to parallax and merged observations of closely spaced targets.

3.1.3 Data Processing. The system implements an innovative and novel approach when it generates hyperspectral observations from the imaged scene. When the tracker confirms a track, the data processor prompts the hyperspectral sensor to scan the track region. Note that after a track is updated by both video *and* hyper-

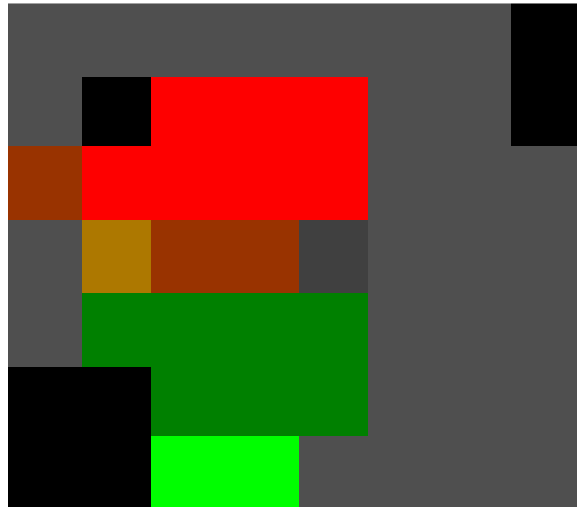


Figure 3.4: The data processor forms an HSI chip using a track’s predicted position and uncertainty. The chip shown here consists of two vehicles (one red and one green) and is an example of a fused chip, where the bounded regions for the two tracks overlap. The black rectangular regions (lower left and upper right) are outside of the region of interest and no data are associated with them. The chip, therefore, can be non-rectangular in shape.

spectral observations, the data processor uses predicted track states for subsequent scanning of the track region. Hence, the sensor continually scans the track region throughout the life of the track. If there is more than one track region in the image, the hyperspectral sensor scans the regions one at a time. The data processor obtains the track’s position and uncertainty (propagated to the time step of each collected hyperspectral scan line) from the tracker and keeps all the scan lines that intersect the track. The region that forms from the intersected lines consists of the track’s centroid, bounded by the standard deviation (square root of the covariance) of the predicted position states in both spatial dimensions.⁵ The group of hyperspectral pixels within the bounded region is called an HSI chip⁶ (Fig. 3.4). Formation of the chip consists of several rules:

⁵In the pushbroom implementation, the sensor scans all lines in the scene, whereas in the ROI implementation, the data processor commands the hyperspectral sensor to steer its mirror to the region prior to line scanning.

⁶An HSI chip is a group of hyperspectral pixels considered likely to contain one or more targets.

- A chip spans two spatial dimensions by including multiple adjacent lines (and therefore multiple discrete time steps). The data processor fuses singleton chips if they are adjacent or overlapped at any time step (see Fig. 3.4).
- When the current line scan is outside or no longer intersecting an existing chip, the data processor forms the chip.
- The system immediately exploits a chip with one target track as soon as the data processor forms the chip. For a fused chip, the system cannot process the chip until the data processor forms all overlapping target tracks.

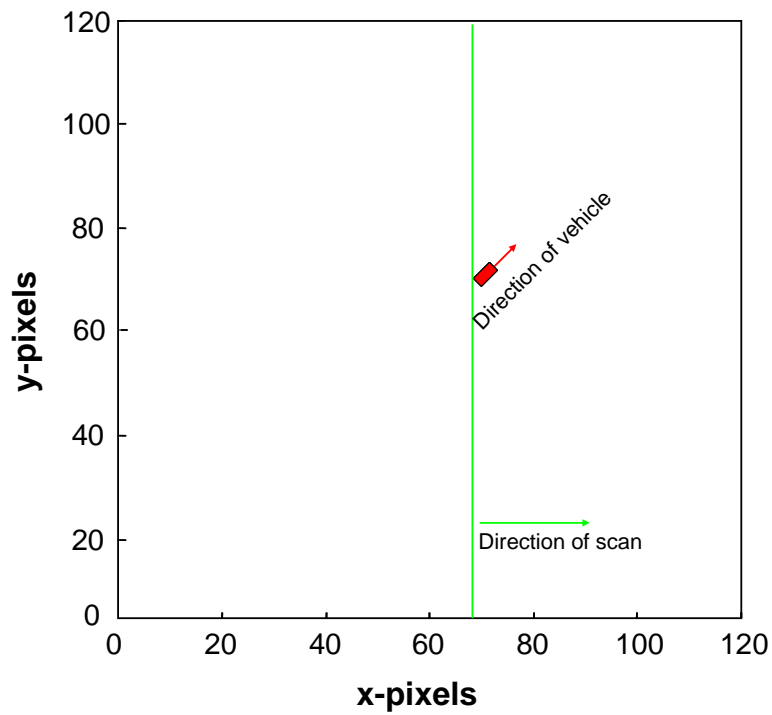
After chip formation, the data processor sends the chip to the classifier for pixel classification.

It is worth noting that the simulated hyperspectral sensor captures a scan line instantaneously, i.e., even if a target is in motion, it is considered stationary during integration time.⁷ If a scan line is not instantaneous, temporal spectral smearing occurs, where the reflectance value for each spectral band changes during integration time. One way to account for this smearing effect consists of capturing a scan line at the beginning and another scan line at the end of integration time and taking the average of both lines. However, since integration time is approximately $10ms$, it is reasonable to assume that each scan line is acquired instantaneously. Spatial smearing also occurs with hyperspectral data. As the hyperspectral sensor captures a scan line at each discrete time step while a target is in motion, the sensor stretches or smears the track across the image, as shown in Fig. 3.5. The *observation-to-track association* accounts for this effect in Sec. 3.4.

3.2 Spectral Library

The spectral library is a synthesized version of real data, consisting of 41 vehicle (Table 3.1) and 5 background classes (Table 3.2). Vehicle spectra are courtesy of Sensors Directorate, Air Force Research Laboratory (AFRL/RV), which obtained

⁷Integration time is the time during which the sensor observes a target.



(a) Hyperspectral line scanning.



(b) HSI chip of vehicle.

Figure 3.5: In this example, the hyperspectral sensor scans lines from west to east (a). The red vehicle is moving in the north easterly direction (b). As the sensor scans lines over the vehicle, the chip appear stretched or smeared. The size of the chip depends on sensor dynamics and the velocity of the vehicle.

the data using an Analytical Spectral Devices Inc. (ASDI) FieldSpec[®] Spectrometer. Background spectra are obtained from the United States Geological Survey (USGS) Digital Spectral Library [11] and from previous hyperspectral field analysis [29]. The FieldSpec[®] and USGS spectral library data have a spectral range of 350 – 2500nm with a spectral resolution of 1nm, yielding 2,151 spectral bands. To simulate more typical hyperspectral data (e.g., National Aeronautics and Space Administration/Jet Propulsion Laboratory’s (NASA/JPL) Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor [15]), the data processor resamples all class spectra to 10nm spectral resolution, which produces 224 spectral bands for each hyperspectral measurement. Additionally, the system removes the water absorption bands, previously discussed in Sec. 3.1.2.1. Each measurement, therefore, ends up with 195 spectral bands.

Table 3.1: The vehicle classes consists of 41 various vehicle types. AFRL/RV obtained the vehicle spectra from volunteers using an ASDI FieldSpec[®] Spectrometer. No methodology was implemented in the vehicle selection process.

Class Label	Color	Make	Model
1	White	Volvo	740 GL
2	Black	Mitsubishi	Montero
3	White	Honda	Accord
4	Black	Jeep	Grand Cherokee
5	Maroon	Toyota	Camry
6	Dark Blue	Landrover	Discovery
7	Blue	Chevy	Colorado
8	Gold	Honda	Odyssey
9	Silver	Nissan	Altima
10	Gold	Honda	Accord
11	Red	VW	Beetle

Continued on next page

Table 3.1 – continued from previous page

Class Label	Color	Make	Model
12	Silver	Buick	LaSabre
13	Sterling	Buick	Park Avenue
14	Silver	Honda	CRV
15	Light Gold	Saturn	SL2
16	Gold	Nissan	Maxima
17	Red	Saturn	SL2
18	Black	Pontiac	Grand Am
19	Silver	Ford	Focus
20	Gray with Black top	Chrysler	Sebring Convertible
21	Silver	Cadillac	DeVille
22	Black	Chevy	Equinox
23	Green	Geo	Prism
24	Black	Ford	Ranger
25	Brown	Nissan	Altima
26	Black	Hyundai	Sonata
27	Black	Chevy	Colorado
28	Maroon	Chevy	Malibu
29	Silver	Jeep	CJ7
30	Black	Mazda	Protege
31	Red	Pontiac	Vibe
32	Green	Chevy	Silverado
33	Blue with Black top	Chrysler	Sebring Convertible
34	Silver	Mitsubishi	Diamonte
35	Black	Toyota	Camry
36	Cranberry	Ford	Expedition
37	Blue	Dodge	Dakota

Continued on next page

Table 3.1 – continued from previous page

Class Label	Color	Make	Model
38	Red/Silver	Dodge	Ram
39	Gold	Chevy	Impala
40	Charcoal Gray	Mazda	Tribute
41	Gray	Buick	Regal

Class Label	Material Type
42	Asphalt
43	Concrete
44	Grass
45	Tree
46	Roof
47	Unknown

Table 3.2: Background spectra are courtesy of USGS. The background spectra are typical material types seen in an urban environment. The classifier uses the unknown class for any pixel that does not fall into one of the 46 classes.

The FieldSpec[®]-collected vehicle spectra are from six location points consistent with a nadir view (i.e., left and right hood, left and right roof, and left and right trunk). Each location point consists of four samples, for a total of 24 spectral samples per vehicle.⁸ Because of the purity of the point measurements from the FieldSpec[®] and USGS spectral library data, the samples do not provide realistic training for the classifier. Using the ROI implementation described in the previous section, the data processor generates mixed pixels by running each vehicle individually through an arbitrary trajectory⁹ (Fig. 3.6) and calculating the linear mixture (using Eq. (3.1)) of each hyperspectral pixel in the chip for every chip formed. The data processor assigns the hyperspectral pixel to the class with the maximum number of panchromatic

⁸The exception is the set of two convertible vehicles (class 20 and 33). The FieldSpec[®] collected only their hood and trunk samples, for a total of 16 spectral samples per vehicle.

⁹Because of the randomness in the sensor and process dynamics, each vehicle’s trajectory is slightly different.

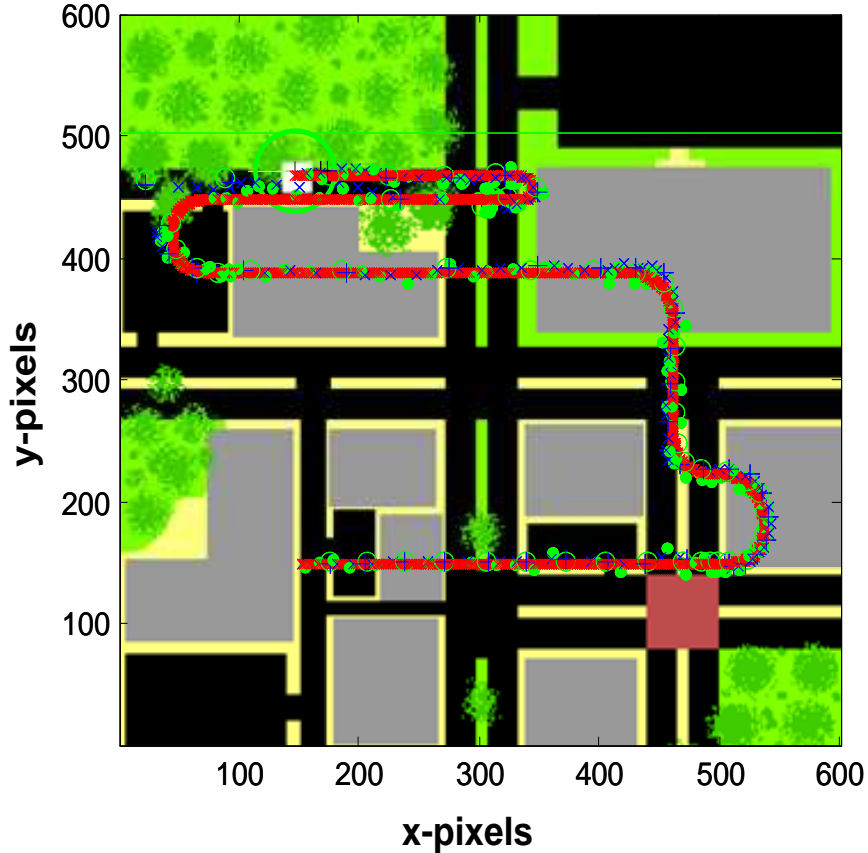


Figure 3.6: For each vehicle, the data processor generates mixed pixels or noisy data using an arbitrary trajectory. The mixed pixels for each vehicle are the samples used in the spectral library.

pixels.¹⁰ The mixed pixels for each vehicle are the samples assigned to the vehicle’s class in the spectral library. Finally, the data processor normalizes each hyperspectral observation with its l_2 -norm to remove albedo effects for each pixel [30]. Albedo is defined as the ratio of diffusely reflected to incident electromagnetic radiation. It is a more specific form of the term reflectivity. Urban areas in particular have very unnatural values for albedo because of the many human-built structures which absorb light before the light can reach the surface.

¹⁰If two classes equally have the maximum number of panchromatic pixels in the 5×5 panchromatic pixel grid, then the data processor does not assign the hyperspectral pixel to a class.

3.3 Spectral Matching and Identification (ID)

The *spectral matching and ID* element matches each hyperspectral pixel in the HSI chip with prototype vectors that represent each class in the spectral library. It assigns the unclassified pixel $\boldsymbol{\lambda}$ (see Fig. 1.1) with the class label w of the nearest prototype vector in the Euclidean sense, as described in the following equation:

$$w = \arg \min_c \sqrt{\sum_{n=1}^N (\lambda_n - m_n^c)^2}, \quad (3.2)$$

where \mathbf{m}^c is a prototype vector with class label $c \in 1, \dots, C$, and N is the number of dimensions.

This research evaluates two classification approaches—fuzzy c-means (FCM) and the self-organizing map (SOM). *Spectral matching and ID* implements the FCM as a supervised classifier, whereas it implements the SOM in two steps: (1) it performs unsupervised learning or clustering, then (2) it uses the SOM's prototype vectors and known class labels of target and background spectra to classify pixels.

3.3.1 Fuzzy C-Means (FCM) Clustering. As a supervised classifier, *spectral matching and ID* performs the FCM algorithm (Fig. 3.7) on the *samples for each class*. At the completion of the iterative procedure, each class has K cluster centers or prototype vectors.¹¹ For each unclassified pixel $\boldsymbol{\lambda}$, the classifier assigns the class label c of the nearest prototype vector $\boldsymbol{\mu}_k^c$ in a Euclidean sense. This computation follows that of Eq. (3.2).

3.3.2 Self-Organizing Map (SOM) Clustering. *Spectral matching and ID* implements the SOM classification approach in two stages. First, it performs unsupervised learning or clustering on *all samples* based on the algorithm described in Fig. 3.8, with no knowledge of the samples' class label. The SOM algorithm assigns a

¹¹Throughout the rest of this thesis, *cluster centers* or *means* will be referred to as *FCM prototype vectors*.

```

BEGIN
  Choose the number of clusters  $K$ , the weighting exponent  $b$ ,
  and the tolerance  $\epsilon$ ;
  Randomly initialize clustering membership matrix  $\mathbf{U}$ ;
  DO
    Compute prototype vector  $\boldsymbol{\mu}_k^c$ , for all  $k$ , using Eq. (2.4);
    Update matrix  $\mathbf{U}$  using Eq. (2.3);
  UNTIL ( $\|\mathbf{U}^{new} - \mathbf{U}^{old}\| < \epsilon$ )
END

```

Figure 3.7: *Spectral matching and ID* performs the fuzzy c-means iterative procedure on the *samples of each class* in the spectral library. It pre-computes the FCM prototype vectors $\boldsymbol{\mu}_k^c$ using the following parameters: $K = 3$ (i.e., 3 prototype vectors per class), $b = 2$, and $\epsilon = 1 \times 10^{-8}$. The K value is chosen through experimentation, based on the number of distinct FCM values observed when K is varied from 1-10. The values for b and ϵ are based on typical values used in the literature.

weight or prototype vector¹² \mathbf{m}^ℓ to each output neuron ℓ and updates each prototype vector using all samples in the spectral library. Furthermore, the algorithm maps each sample \mathbf{s} to an output neuron ℓ on the 30×30 rectangular lattice, as shown in Fig. 3.9. Another way to visualize the SOM is shown in Fig. 3.10. There is no restriction on the number of classes that can be mapped to an output neuron. Each output neuron can represent more than one class because the two-dimensional class distributions can overlap.

Because of the self-organizing property of the SOM (discussed previously in Sec. 2.2.3), the algorithm maps class samples highly influenced by background spectra to the neurons located in the “tail” of the class distribution. These samples have the least information about the class; hence, removing or filtering that part of the distribution should reduce misclassification errors. The SOM filtering procedure described below is based on the SOM’s ability to preserve the geometric relationships among class samples in the 195-dimensional input space on the 2-dimensional rectangular lattice. Neurons that are physically near each other are based on the similarity

¹²Throughout the rest of this thesis, *weight vectors* will be referred to as *SOM prototype vectors*.

```

BEGIN Ordering Phase
  Randomly initialize weight vectors  $\mathbf{m}_i(0)$ ;
  Choose relatively large initial learning rate  $\alpha_0$  and
  neighborhood radius  $\sigma_0$ ;
  DO
    Determine winner  $w$  using Eq. (2.5) and Eq. (2.6);
    Update  $\mathbf{m}^\ell$  using Eq. (2.7);
    Decrease learning rate  $\alpha(t)$  and neighborhood radius  $\sigma(t)$ ;
  UNTIL Number of ordering steps
END
BEGIN Tuning Phase
  Keep neighborhood radius  $\sigma$  fixed;
  DO
    Determine winner  $w$  using Eq. (2.5) and Eq. (2.6);
    Update  $\mathbf{m}^\ell$  using Eq. (2.7);
    Decrease learning rate  $\alpha(t)$  very slowly;
  UNTIL Number of tuning steps
END

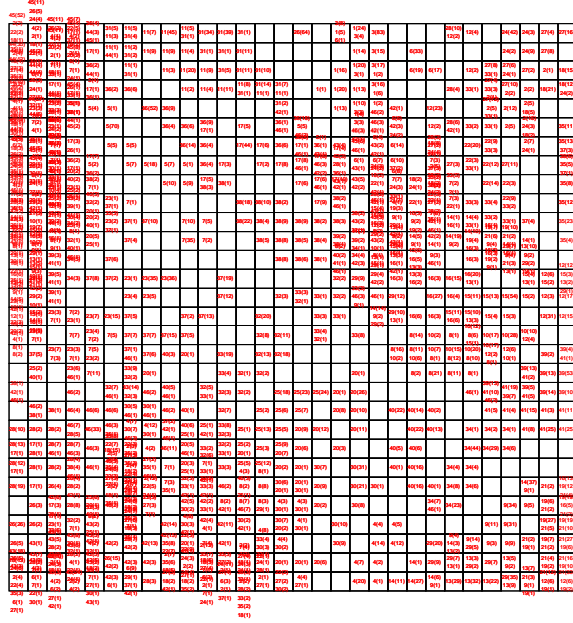
```

Figure 3.8: *Spectral matching and ID* performs clustering on *all samples* in the spectral library. It pre-computes the SOM prototype vectors \mathbf{m}^ℓ using the following parameters: The ordering phase takes 13,800 steps, where the learning rate $\alpha(t)$ and neighborhood radius $\sigma(t)$ decrease linearly from 0.9 to 0.02 and from 15 to 1, respectively. The tuning phase takes 41,400 steps, where $\alpha(t)$ decreases linearly from 0.02 to 0, and $\sigma(t)$ stays at 1. These parameters were chosen using the approach described in Sec. 2.2.3.2 and in [22].

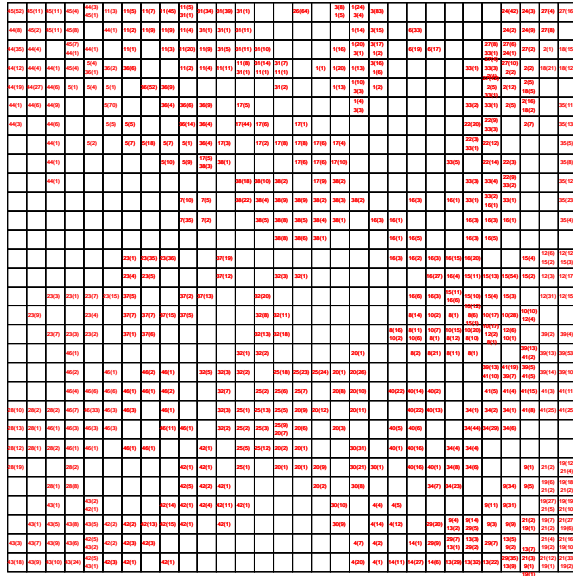
of the hyperspectral samples mapped to the neurons. The SOM filtering procedure is a novel concept, and very little work in the classification literature deal with the SOM in this manner.

Spectral matching and ID filters the samples located at the tails of the distribution by performing morphological operations on the map, as described in the following procedure:

1. Treat the “original” map (Fig. 3.9(a)) as an image, where each output neuron is a pixel.



(a) Original map.



(b) Filtered map.

Figure 3.9: The SOM consists of output neurons organized on a rectangular lattice. The number of neurons is 900, or equivalently, the grid consists of 30×30 neurons. The number of neurons may vary from a few dozen up to several thousands. For fast processing, however, fewer than a thousand nodes is a reasonable number. Each sample is mapped to a neuron, represented by a square in the two-dimensional grid. (a) The 30×30 map prior to performing morphological operations. (b) The 30×30 map after performing morphological operations.

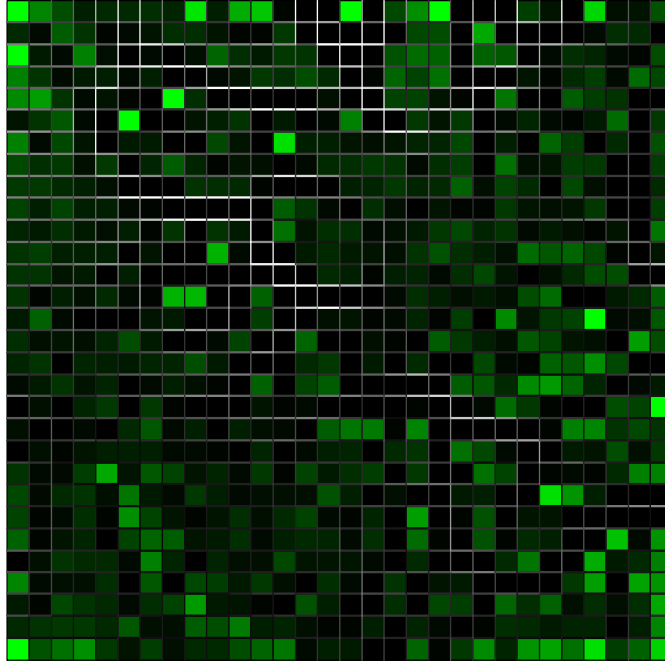


Figure 3.10: In this map, a unit is also represented by a square, and the lighter the color, the more samples are assigned to it. If the average distance of neighborhood \mathbf{m}^ℓ is small, a whiter shade is used. Dark shades represent large distances.

2. Create a separate two-dimensional map for each class, where the neurons are only those containing the class samples. The pixel “intensity” is the number of samples mapped to the corresponding neuron.
3. Determine the 8-connected components (Fig. 3.11) for each map of the background class. The connected component with the most number of samples (or the highest density) represents the background class. The other connected components (if any) are removed because they have less information on the class under consideration. The procedure converts the modified map, which only consists of the connected component that represents the class, into a binary mask. It replaces all nonzero neurons with ‘0,’ and all zero neurons with ‘1.’
4. Apply each background binary mask to the separate map for each vehicle class, i.e., perform a neuron-by-neuron multiplication between each background mask and vehicle map. Doing so, the procedure removes the neurons common between

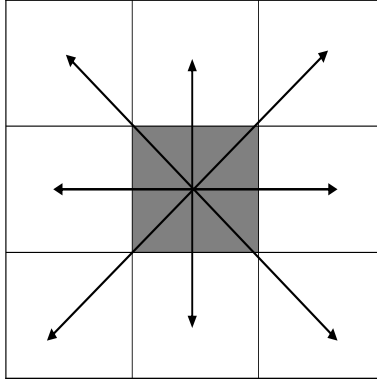


Figure 3.11: Neurons are connected if their edges or corners touch. This means that if two adjoining neurons are nonzero, they are part of the same object, regardless of whether they are connected along the horizontal, vertical, or diagonal direction.

the vehicle and all background maps. These neurons have the least information about the vehicle and the most influence from the background.

5. Determine the 8-connected components of the resulting vehicle map. The connected component with the highest density represents the vehicle class. Any other connected component is most likely due to outliers or background neurons removed in step 3.
6. Combine all the maps into one map by concatenating the classes for each neuron. Fig. 3.9(b) shows the combined image or “filtered” map (compare it to the original map in Fig. 3.9(a)).

The second stage of the SOM approach classifies each pixel λ by assigning it with the label of the pixel’s best-matching neuron.¹³ Pixels are classified by the following rules:

1. If all of the samples mapped to the best-matching neuron belong to only one class, then the classifier assigns the pixel with the class label. For example, in Fig. 3.12, class 27 is the only class mapped to the top rightmost neuron (encircled

¹³In Sec. 2.2.3.2, the best-matching neuron of an unclassified pixel λ is the neuron with the nearest prototype vector in a Euclidean sense.

		28(10) 12(2)	12(4)		24(42)	24(3)	27(4)	27(16)
33)					24(2)	24(9)	27(8)	
19)	6(17)		12(2)	27(8) 33(1) 27(4)	27(6) 24(1)	27(2)	2(1)	18(15)
		28(4)	33(1)	33(3) 27(12) 27(12)	27(10) 2(2)	2(2)	18(21)	18(12) 24(2)
	12(23)			2(5) 33(1)	2(12)	2(5) 18(5) 2(16)		
	12(2)	28(6) 42(1)	33(2)	33(1)	2(5)	24(3)		35(11)
	28(3) 18(8) 24(4)		22(20)	22(9) 33(3)		2(7) 24(1)		35(13) 37(3) 38(6)
	7(3) 2(5) 37(3) 38(2)	27(3)	22(3) 33(1)	22(12)	27(11)			35(5) 37(1)
(2) ...	18(2)	33(5) 7(2)		22(14)	22(3)			35(8)

Figure 3.12: The number outside the parenthesis is the class label, and the number inside the parenthesis is the number of samples assigned to the neuron.

- in blue). If this is the pixel’s best-matching neuron, then the classifier labels the pixel as class 27.
2. If the SOM algorithm maps samples from different classes to the best-matching neuron, then the classifier assigns the pixel with the class label that has the most samples. This implementation is a hard assignment and is a maximum-likelihood (ML) classification. In Fig. 3.12, the SOM algorithm maps the two classes (classes 18 and 24) to the fourth unit of the last column (encircled in green). Since the maximum number of samples mapped to the unit is 12, which belongs to class 18, the classifier labels the pixel as class 18.¹⁴
 3. If the SOM algorithm does not map any sample to the best-matching neuron, such as the second unit in the last column of Fig. 3.12 (encircled in black), then the pixel is assigned an “unknown” label. This label does not necessarily

¹⁴If the two or more classes have the same maximum number of samples mapped to the unit, the classifier labels the pixel with the first class on the list. This choice is arbitrary. A better approach is to implement a multiple hypothesis tracker (MHT) [8], in which the tracker forms a hypothesis for each potential class.

imply that the unclassified pixel actually belongs to a vehicle class that is not identified in the spectral library. At the very least, it is an indication that the best-matching neuron is physically far from any class; therefore, the SOM algorithm did not assign any sample to the neuron.

After classifying all the pixels in the chip, *spectral matching and ID* provides the *HSI chip with class ID* to the *observation-to-track association* element.

3.4 *Observation-to-Track Association*

The *observation-to-track association*¹⁵ element takes observation-to-track pairings that satisfy gating constraints and determines the observation-to-track assignments. The observations are either hyperspectral or panchromatic.¹⁶ The system generates panchromatic observations 10 times per second, whereas it generates hyperspectral observations several seconds apart depending on track location and line scanner dynamics. In the assignment process, the associator uses a distance measure to assign observations to existing tracks. It uses a different distance measure for each type of observation.

3.4.1 Panchromatic Video Observations. The associator propagates the predicted position and uncertainty for each track to all of the possible times of the observations. It calculates the Mahalanobis distance between observation i and track j propagated to the time of observation i . The Mahalanobis distance¹⁷ is a function

¹⁵The *observation-to-track association* element will also be referred to as an “associator.”

¹⁶In this context, hyperspectral measurements are not synonymous with hyperspectral observations. Hyperspectral measurements refer to the hyperspectral input data of Fig. 3.1. On the other hand, hyperspectral observations refer to the components in the HSI chip. For panchromatic video, measurement and observation are synonymous.

¹⁷The quantity d_{ij}^2 is actually a squared distance but, for convenience, it will be referred to simply as a *distance*.

of the measurement residual \mathbf{r}_{ij} , and its associated covariance matrix \mathbf{S}_i :

$$\mathbf{r}_{ij} = \mathbf{z}_i - \mathbf{H}\hat{\mathbf{x}}_j, \quad (3.3)$$

$$\mathbf{S}_j = \mathbf{H}\mathbf{P}_j\mathbf{H}^T + \mathbf{R}, \quad (3.4)$$

$$d_{ij,kin}^2 = \mathbf{r}_{ij}^T \mathbf{S}^{-1} \mathbf{r}_{ij}, \quad (3.5)$$

where $d_{ij,kin}^2$ is the observation-to-track kinematic distance. The associator normalizes this distance using the l_1 -norm. The l_1 -norm is the sum of all the observation-to-track distances. By normalizing with the l_1 -norm, a “percentage” of the total distance is represented by each observation. The *filtering and prediction* element in Sec. 2.3.4 discusses Eqs. (3.3) and (3.4) in more detail. Eq. (3.5) is the distance measure used by the auction algorithm.

3.4.2 Hyperspectral Observations. The associator determines the distance measure for the hyperspectral observation by performing the following procedure:

1. Generate the hyperspectral observations. Since it appears that no previous work on hyperspectral observations (based on predicted track state information) has been developed, the steps describing the generation of the hyperspectral observations are formulated heuristically. The approach is based on target size and computational complexity. Each pixel in the HSI chip can be treated as a hyperspectral observation, but because a target consists of approximately six hyperspectral pixels, the morphological operation performed reduces computational complexity.¹⁸
 - (a) Determine the 8-connected components for each class ID present in the HSI chip. Each connected component consists of contiguous pixels with the same class ID. The associator ignores components with fewer than 3 pixels because it considers them noise or background clutter.

¹⁸If the target size is at the sub-pixel level, then the morphological operation does not provide additional information and could potentially merge targets.

- (b) Find the spatial centroid for each component.
 - (c) Obtain the unique time stamp for each hyperspectral scan line contained within each component.
 - (d) Determine the measurement time for each spatial centroid by weighting each time stamp with the number of pixels in the scan line and averaging all the time stamps contained within the component. Weighting each time stamp with the number of pixels in the scan line accounts for the spatial smearing effect described in Sec. 3.1.3. Each resulting spatial/temporal centroid is a hyperspectral observation. Fig. 3.13 shows an example of an HSI chip with four connected components.
2. Determine the spectral distance¹⁹ of each observation-to-track pairing from a pre-computed $C \times C$ matrix. The spectral distance between class pairs is calculated using the following equation:

$$d_{ij,spec}^2 = \left\| \frac{\boldsymbol{\mu}_i^{c_1} - \boldsymbol{\mu}_j^{c_2}}{\sigma_j^{c_2}} \right\|^2, \quad (3.6)$$

where $d_{ij,spec}^2$ is the observation-to-track spectral distance and $\|\cdot\|$ is the Euclidean distance. Eq. (3.6) uses a different $\boldsymbol{\mu}$ for each type of classification algorithm used by *spectral matching and ID* in Sec. 3.3:

Fuzzy C-Means. Take the mean of the fuzzy c-means for each class $\boldsymbol{\mu}_k^c$, $k = 1, \dots, K$.

Self-Organizing Map. Take the mean of all the samples for each class $\boldsymbol{\mu}^c$.

The associator computes the class-wise Euclidean distance (Eq. (3.6)) using each class mean and stores the distances in the $C \times C$ matrix, where each row index corresponds to the class ID of observation i and each column index corresponds

¹⁹The *nearest neighbors computations* element also computes this distance in Sec. 3.7. The associator uses the spectral distance as part of the cost in the observation-to-track assignment, whereas the *nearest neighbor computations* element uses the distance to determine the nearest neighbors for each class.

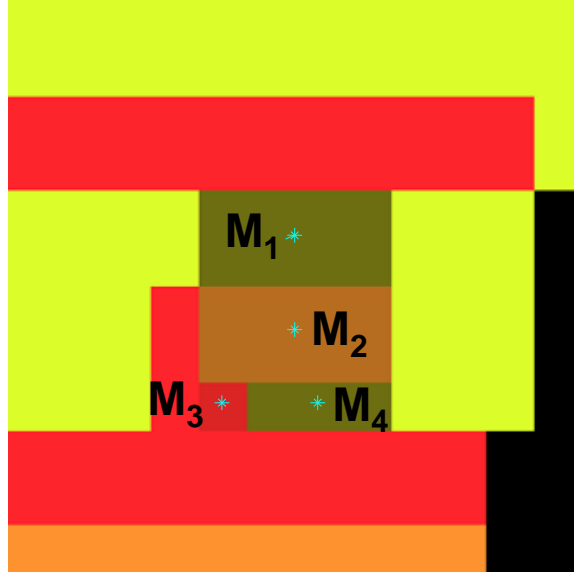


Figure 3.13: In this example, M_1, \dots, M_4 are the four hyperspectral observations formed. Color of the region has no meaning and only provides a means to distinguish among different regions. The symbol ‘*’ represents the centroid of each component.

to the class ID of track j . Note that Eq. (3.6) assumes the hyperspectral image bands are uncorrelated. This is known to be false. However, because each class has relatively few representative samples, its covariance matrix is not invertible (i.e., the matrix is singular). The spectral Mahalanobis distance (Eq. (3.6)) is a unitless distance that fits within the current tracking paradigm. One way to mitigate this false assumption is simply to use the kinematic distance as the distance measure. This is accomplished by setting the weighting factor γ to zero, as described in the next step.

3. Obtain the position and uncertainty for each track (propagated to the measurement time of each centroid) from the tracker. Using Eqs. (3.3)-(3.5), the associator computes the Mahalanobis distance $d_{ij,kin}^2$ for each hyperspectral observation.
4. Weight the spectral distance $d_{ij,spec}^2$ and kinematic distance $d_{ij,kin}^2$ and calculate the overall distance using the following equation:

$$d_{ij}^2 = (\gamma)d_{ij,spec}^2 + (1 - \gamma)d_{ij,kin}^2, \quad (3.7)$$

where γ is the weighting factor for the spectral distance $d_{ij,spec}^2$. By having different values of γ , the associator can vary the influence of the spectral and kinematic information on the data association. Eq. (3.7) is the distance measure used by the auction algorithm.

In the auction algorithm (Fig. 3.14), the associator uses the calculated distance (corresponding to the type of observation being processed) as the cost of assigning an observation to a track. The auction algorithm is an iterative process, in which each iteration consists of the bidding and assignment phase. The process terminates when the algorithm obtains a complete assignment. At the completion of the auction algorithm, one of three outcomes occurs:

1. Each observation is assigned to a track.
2. More observations are available than tracks; therefore, those observations not assigned to existing tracks are unassociated.
3. Fewer observations are available than tracks; therefore, some tracks will have missed detections.

The outcome is evaluated by the *track maintenance* element.

3.5 Track Maintenance

In *track maintenance*, three functions evaluate the results of the assignment process:

Track Initiation. For each unassociated observation, either panchromatic or hyperspectral, the tracker initiates new tracks called *tentative* or *preliminary* tracks (until confirmed). Since hyperspectral line scanning is cued when a tracker confirms a track, the tracker initiates all tracks on moving objects primarily using panchromatic video observations. For kinematic-only tracking simulations,


```

REPEAT
  BEGIN Bidding Phase
    FOR each observation  $i$  that is unassigned in  $E$ 
      Compute the value of each track  $j \in \Delta(i)$ ;
      Find the “best” track  $j^*$  with the maximum value;
      Find the best value offered by tracks other than  $j^*$ ;
      Compute the “bid” of observation  $i$  for track  $j^*$ 
    END
  END
  BEGIN Assignment Phase
    FOR each track  $j$ 
      IF  $\mathbf{P}(j)$  is nonempty
        Increase  $p_j$  to the highest bid;
        Find track  $i^*$  with maximum  $p_j$ ;
        Remove from  $E$  any pair  $(i, j)$ ;
        Add to  $E$  the pair  $(i^*, j)$ 
      END
    END
  END
UNTIL assignment complete

```

Figure 3.14: In the Bertsekas auction algorithm, the cost p_j corresponds to the kinematic distance $d_{ij,kin}^2$ (if the observation is panchromatic) or the overall distance d_{ij} (if the observation is hyperspectral), where the distance represents the cost observation i incurs for being assigned to track j .

the tracker initiates a track using the class ID of the nearest vehicle. This is a design choice that provides a way to discriminate between tracks. For the hyperspectral-augmented tracking simulations, the tracker initially assigns the “unknown” class ID to each track. After the hyperspectral sensor scans the track region, the tracker updates the class ID state variable with the class ID of the hyperspectral observation that it assigns to the track.

Track Confirmation. The tracker confirms a track if it receives K associated observations within B frames.

Track Deletion. The tracker deletes a track if it does not receive associated observations after K_D frames.

Fig. 3.15 provides the pseudo-code for the confirmation and deletion logic. A special function called track stitching stitches a newly confirmed track to the nearest dead track within the confirmed track’s gate. *Track maintenance* performs this function solely for the kinematic-only simulations in order for swapped tracks to maintain track continuity after they are deleted (due to no updating information). Thus, the tracks remain swapped after going through ambiguous situations. The *design of experiments* in Sec. 4.1 provides additional information on this implementation.

3.6 Filtering and Prediction

The *filtering and prediction* element uses both panchromatic and hyperspectral observations to update track kinematic states. The kinematic-only tracker employs a constant velocity Kalman filter (previously discussed in Sec. 2.3.4). Similarly, the hyperspectral-augmented tracker employs a constant velocity Kalman filter, for which the state vector for track j , $\mathbf{x}_j = [x \ y \ \dot{x} \ \dot{y} \ h]^T$, includes an additional state variable h . The variable h represents the class ID of the track. During updates, the modeled measurement noise covariance \mathbf{R} accounts for uncertainty in position, regardless of the type of observation. On the other hand, the class ID for each track, which is updated by hyperspectral observations only, has zero uncertainty. The class ID of the hyperspectral observation (in the observation-to-track assignment) replaces the current class ID of the track. Admittedly, this is a naive assumption, especially in the absence of an MHT [8]. Statistical methods can be used to address the uncertainty of the class ID state variable, namely, Bayesian or *a posteriori* inference, maximum *a posteriori* (MAP), likelihood processing, and maximum likelihood, and evidential reasoning [8]. For simplicity, this research assumes perfect class ID and defers the use of statistical methods to future work.

3.7 Nearest Neighbor Computations

The *nearest neighbor computations* element pre-computes the nearest neighbors for each class in the spectral library. *Gating computations* in Sec. 3.8.2 uses the

```

BEGIN at track initiation
  Create  $B_{video}$  and  $B_{HSI}$  bit variables (initialized to 5 and
  3 bits, respectively) and  $k_{video}$  and  $k_{HSI}$  iteration counters
  for each track (both initialized to zero);
  REPEAT for every observation-to-track pair
  (whenever not specified, replace “ $k$ ,” “ $K$ ,” and “ $B$ ”
  based on observation type; e.g., if observation is video,
  use  $k_{video}$ ,  $K_{video}$ , and  $B_{video}$ , respectively)
    WHILE  $k$  is not equal to multiple of  $K$ 
      Increment  $k$  by 1;
      IF the associator assigns an observation to a track
        Assign 1 to the appropriate  $B$  bit corresponding to
         $k$  (if  $k \leq B$ ) or remainder of  $(k/B)$  (if  $k > B$ );
      END
      IF sum of  $B$  bits  $\geq K$ 
        Confirm track;
      END
      IF  $k_{video} > B_{video}$  and  $k_{HSI} > B_{HSI}$  and (sum of
       $K_{video}$  bits  $< K_{D,video}$  and sum of  $K_{HSI}$  bits  $< K_{D,HSI}$ 
      unless sum of  $K_{video}$  bits  $< K_{D,video}$  occurs before
      a hyperspectral sensor scans a track)
        Delete track;
      END
    UNTIL track is deleted
  END

```

Figure 3.15: The tracker confirms tracks based on either $K_{video} = 3/B_{video} = 5$ or $K_{HSI} = 2/B_{HSI} = 3$ observations. The tracker deletes tracks based on $K_{D,video} = 3/K_{video} = 5$ and $K_{D,HSI} = 2/K_{HSI} = 3$, unless the video deletion criteria is met before the hyperspectral sensor scans the track. These values were determined experimentally by identifying the number of frames (or length of time) before the target is no longer within the kinematic gate of the track or the scan region of the hyperspectral sensor.

nearest neighbors to gate hyperspectral observations. *Nearest neighbor computations* calculates the nearest neighbors differently for each classification algorithm used by *spectral matching and ID* in Sec. 3.3:

Fuzzy C-Means. *Nearest neighbor computations* determines the nearest neighbors for each class using the following procedure:

1. Compute sample-wise spectral distances between class \mathbf{s}_{mc_1} and each of the other classes \mathbf{s}_{mc_2} , where $m = 1, \dots, M$, $c_1 = 1, \dots, C$, $c_2 = 1, \dots, C$, and $c_1 \neq c_2$. The spectral distance measures are the Euclidean, Manhattan, correlation, Chebyshev, Canberra, and squared chord defined in Sec. 2.4.
2. Summarize the sample-wise distances for each computed distance (from step 1) by taking their average (Eq. (2.42) of Sec. 2.5.2).
3. Sort the linkage for each distance measure in ascending order.

Self-Organizing Map. *Nearest neighbor computations* performs a similar, but more intricate procedure to determine the nearest neighbors for each class:

1. Compute the sample-wise Euclidean distances (Eq. (2.35) of Sec. 2.4) between a class \mathbf{s}_{mc_1} and each of the other classes \mathbf{s}_{mc_2} , where $m = 1, \dots, M$, $c_1 = 1, \dots, C$, $c_2 = 1, \dots, C$, and $c_1 \neq c_2$. This procedure evaluates two sample distributions:

SOM lattice. Use the lattice location points ($\boldsymbol{\eta}^\ell \in \mathbf{R}^2$) of the class samples.

Prototype vectors. Use the prototype vectors ($\mathbf{m}^\ell \in \mathbf{R}^N$) associated with the SOM lattice points of the class samples.

2. Summarize the sample-wise distances using the average linkage (Eq. (2.42) of Sec. 2.5.2).
3. Sort the linkage distances in ascending order.

At the completion of each procedure, the class labels of the sorted distances provide the neighbors in ascending order of distance for the class under consideration.

3.8 Gating Computations

The *gating computations* element performs kinematic and spectral gating as described in the following:

3.8.1 *Kinematic Gating.* *Gating computations* obtains the gate threshold for the Mahalanobis distance calculated in Sec. 3.4 from tables of the Chi-square distribution, since the quadratic form in Eq. (2.31) that defines the gate region is Chi-square distributed with number of degrees of freedom equal to measurement dimension N . Table 3.3 gives the gate probability P_G [3]

$$P_G = P \{ \mathbf{z}(t+1) \in V(t+1, G) \}, \quad (3.8)$$

or the “probability that the (true) measurement will fall in the gate” for various values of G and measurement dimension N . The square root $g = \sqrt{G}$ is the number of standard deviations of the gate.

3.8.2 *Spectral Gating.* Since kinematic gating is not suitable for hyperspectral data, *gating computations* gates hyperspectral observations using a different approach called spectral gating. Gating hyperspectral data is a novel concept, and it appears that very little research has been done in this area. The main idea is that a hyperspectral observation is outside the gate of a track if the observation’s class ID is spectrally distinct from the track’s class ID. Spectral gating reduces the number of operations performed by the *observation-to-track association*. It is not unlikely for an HSI chip to have more than ten hyperspectral observations, and if three target tracks exist, the number of operations amounts to around 1,000 (as compared to 27 operations for three observations and three tracks).

	G	1	4	6.6	9	9.2	11.4	16	25
	g	1	2	2.57	3	3.03	3.38	4	5
N									
1		0.683	0.954	0.99	0.997			0.99994	1
2		0.393	0.865		0.989	0.99		0.9997	1
3		0.199	0.739		0.971		0.99	0.9989	0.99998

Table 3.3: The table provides values of the probability mass P_G given the gate threshold G and measurement dimension N . The desired probability P_G is 0.99. Since the measurement vector has two dimensions, the gate threshold G is 9.2.

Spectral gating using a single hypothesis tracker, however, has potentially significant drawbacks, especially for more complex ambiguous situations. For example, if one vehicle goes into an occlusion (e.g., under a bridge) and a second vehicle goes out of the occlusion (heading in the same direction as the first vehicle), the tracker will likely “latch” onto the second vehicle. Assuming the second vehicle’s hyperspectral signature is outside of the gate of the existing track, the associator ignores the hyperspectral observation generated by the second vehicle. Hence, the hyperspectral-augmented tracker does not update the class ID of the track. In a single hypothesis tracker, one way to address this is by performing a track deletion based on the track’s class ID. If the associator consistently associates an observation with another class ID (e.g., 3 out of 5 frames), the tracker should delete the current track and initiate a new track with this class ID. A more robust approach is to implement an MHT [8].

Gating computations performs spectral gating using the nearest neighbors described in Sec. 3.7. It implements a different spectral gating computation for each type of classification algorithm used by *spectral matching and ID* in Sec. 3.3:

Fuzzy C-Means. *Gating computations* takes the most frequent first, second, and third nearest neighbor class for all six distance measures. These are the most likely confusers; therefore, they are the classes considered within the gate of the class under consideration. The choice of three is arbitrary. Thresholding the average linkage can also be used to limit the number of nearest neighbors. Fig. 3.16 is a plot of a (hypothetical) data set, where samples with the same color belong to the same class. Using the Euclidean distance and average linkage, represented by the red arrows, the three nearest neighbors of the yellow class are the black, light blue, and dark blue classes.

Self-Organizing Map. *Gating computations* thresholds the sorted linkage distances. For each distance m , starting with the smallest distance, compute Eq. 3.9 - 3.11 until $\Psi > 0$

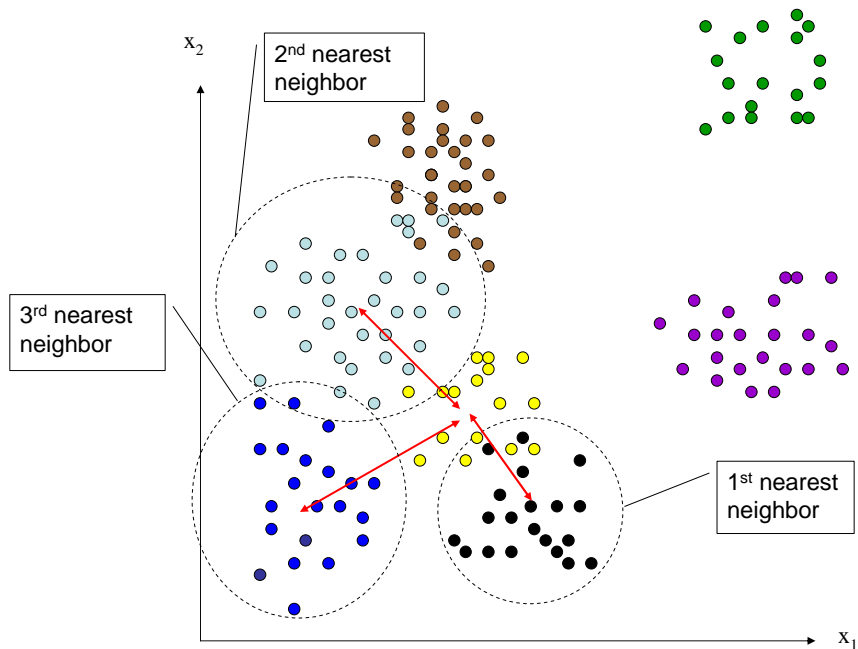


Figure 3.16: In this example, *nearest neighbor computations* computes the sample-wise Euclidean distances between the yellow class and the other classes in the data set. The average linkage, represented by the red arrows, summarizes the Euclidean distances. The first, second, and third nearest neighbor are the black, light blue, and dark blue classes.

$$\psi_1 = d(c_1, c_{m+1}) - d(c_1, c_m), \quad (3.9)$$

$$\psi_2 = d(c_1, c_{m+2}) - d(c_1, c_{m+1}), \quad (3.10)$$

$$\Psi = \psi_2 - \psi_1. \quad (3.11)$$

The class labels of the first m linkage distances are the classes within the gate for the class of the track under consideration. This algorithm determines the “knee in the curve,” and each class with distance that is less than the distance at the knee is considered a nearest neighbor (Fig. 3.17).

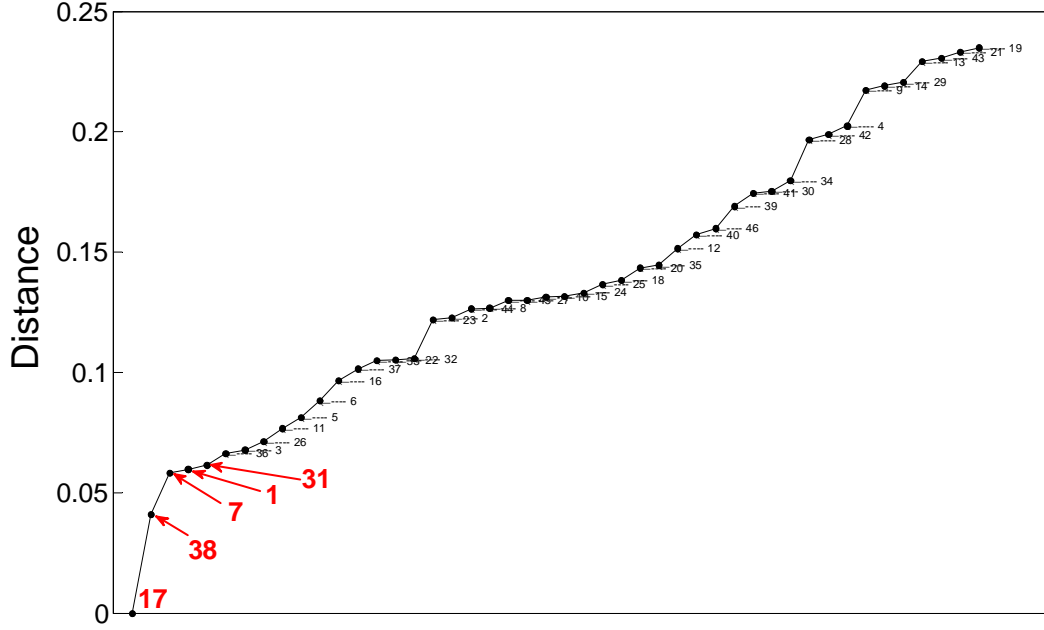


Figure 3.17: The curve provides the neighbors of class ID 17, as described in Sec. 3.7. The knee in the curve is calculated using Eqs. (3.9) - (3.10). Class ID 38, 7, 1, and 31 (in red) are the nearest neighbors of class ID 17 in ascending order of distance.

3.9 Performance Measures

Performance analysis uses seven measures to compare the performance of the tracker (before and after augmenting the system with hyperspectral data) and to assess the performance of the classifier [6]:

Probability of Detection (P_D) and Probability of False Alarm (P_{FA}).

These first two measures are based on Bayesian decision theory, where H_0 is the hypothesis that an unclassified pixel is any background type, and $H_{c_v}, 1 \leq c_v \leq 41$, is the hypothesis that a pixel is target type c_v :

$$\begin{aligned}
P_D &= P(H_i|H_i)_{i \neq 0} \\
&= \frac{N_D}{N_D + N_{MISS}},
\end{aligned} \tag{3.12}$$

$$\begin{aligned}
P_{FA} &= P(H_i|H_j)_{0 < c_v \leq 41, c_v \neq j} \\
&= \frac{N_{FA}}{N_{FA} + N_{REJECT}}.
\end{aligned} \tag{3.13}$$

where N_D is the number of correct detections, N_{MISS} is the number of missed detections, N_{FA} is the number of false alarms, and N_{REJECT} is the number of rejections. The experimental design of Sec. 4.1 defines detection, false alarm, miss, and rejection at the *pixel level* in the following manner:

1. A detection occurs when the classifier identifies the target pixel correctly.
2. A false alarm occurs when the classifier identifies a background pixel as any target or the target of interest as a different target type.
3. A miss occurs when the classifier misidentifies a target as a background class.
4. A rejection occurs when the classifier identifies background as any of the background types.

For each pixel in the HSI chip, a counter increments one of these four parameters, depending on the result of the comparison between the pixel's class ID and pixel truth.

Equal-Weighted Classification Accuracy (EWA). This third measure, which is also at the pixel level, measures the percentage of correctly identified pixels and computes the accuracy of each HSI chip as the average of the individual class accuracy for the classes contained within the chip:²⁰

²⁰Sec. 2.6 describes EWA in more detail.

$$\text{EWA} = \frac{\sum_{c=1}^C \left(\frac{\#correct}{total\#} \right)_c 1_c}{\sum_{c=1}^C 1_c}, \quad (3.14)$$

where 1_c is an indicator function that evaluates to true if class c exists in the current image chip. This measure obtains the $\#correct$ and $total\#$ from the error matrix discussed in Sec. 2.6.

Measures of effectiveness. The remaining four metrics, which are at the frame level, provide measures of effectiveness using track-to-truth assignments.²¹

1. For each truth trajectory associated with a track (without a class ID),²² the probability of association is

$$P_{assoc,j} = \frac{N_{assoc,j}}{N_f}, \quad (3.15)$$

where $N_{assoc,j}$ is the number of track associations for track j at frame f and N_f is the number of frames.

2. For each truth trajectory associated with a track (with any class ID),²³ the probability of declaration given association is

$$P_{declare,j|assoc,j} = \frac{N_{declare,j}}{N_{assoc,j}}, \quad (3.16)$$

where $N_{declare,j}$ is the number of track declarations for track j .

3. For each truth trajectory associated with a track (matching the true class ID), the probability of correct ID given declaration is

²¹The associator performs the auction algorithm on track-to-truth pairings to determine the track-to-truth assignment.

²²A track without class ID (track association) is only based on kinematic observations. The hyperspectral sensor has not scanned the track region; therefore, the track has not received any hyperspectral observation updates.

²³A track with any class ID (track declaration) has received updates from hyperspectral observations. The assigned class ID is not necessarily the true class ID of the target.

$$P_{ID|declare,j} = \frac{1}{N_{declare}} \sum_{f=1}^{N_f} ID_{j,f}^{correct}, \quad (3.17)$$

$ID_{j,f}^{correct}$ is a correct target identification for track j at frame f .

4. The last measure is the performance check for the hyperspectral-augmented tracking system. It is based on the probability of a correct target identification,

$$P_{ID,j} = \frac{1}{N_f} \sum_{f=1}^{N_f} ID_{j,f}^{correct}. \quad (3.18)$$

Correct target identification occurs whenever a current identification is the same as truth for a particular target's position independent of the track number. In other words, if the kinematic tracker swaps tracks, but still correctly identifies the target, even though it is now assigned to the "wrong" track number, it is still deemed correct.

3.10 Summary

This chapter provides the different algorithms, procedures, and rules employed by the hyperspectral-augmented tracker. The next chapter describes the implementation of the methodology through experiments. By using different parameters for the weighting factor γ , implementing two different classification algorithms, simulating a change detection algorithm, gating hyperspectral observations, and evaluating different ambiguous situations, the results offers meaningful insights on what effects these factors have (if any), and more importantly, provides patterns that show predictable cause-and-effect relationships between the system parameters and the results themselves.

IV. Design of Experiments and Simulation Results

This chapter discusses the design of experiments (Sec. 4.1), analyzes the various simulations (Secs. 4.2 and 4.3), and summarizes their respective results (Sec. 4.4). Sec. 4.2 describes a single kinematic-only tracking baseline simulation run through a sequence of images taken over time. It highlights the ambiguous situations and the different configurations employed by the system. For comparison, Sec. 4.3 provides the same sequence of images, but this time, augmented with hyperspectral data. Furthermore, it discusses the various configurations used by both classification approaches (fuzzy c-means and two-stage self-organizing map). Finally, since the main objective of this research is to compare the performance of kinematic-only tracking with hyperspectral-augmented tracking, Sec. 4.4 summarizes the quantitative results for both tracking methods using the performance measures described in Sec. 3.9.

4.1 *Design of Experiments*

The design of experiments is a strategy for setting up several sets of experiments to determine ultimately the feasibility of the overall system. In order to provide a complete analysis, several key system parameters are varied in a systematic manner for the purpose of determining the correlation between parameters and results. Sec. 4.1.1 discusses briefly the software package used in the implementation of the experimental design. Sec. 4.1.2 summarizes the different effects that are accounted for in the simulated measurement noise. Sec. 4.1.3 examines the two ambiguous scenarios used in the experiments, followed by the description of the various parameter configurations (Sec. 4.1.4). Finally, Sec. 4.1.5 discusses the simulation process and describes the visualization display that provides the state of the system at a point in time.

4.1.1 Matlab. This research implements the kinematic-only and hyperspectral-augmented tracking systems using the **Matlab**[®] software package. **Matlab**[®] is a high-performance language for technical computing [25] that integrates computation, visualization, and programming in an easy-to-use environment where

problems and solutions are expressed in familiar mathematical notation. It features a family of add-on application-specific solutions called toolboxes. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the `Matlab`[®] environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others [25]. For these reasons, `Matlab`[®] is an excellent choice for developing the simulations for this research.¹

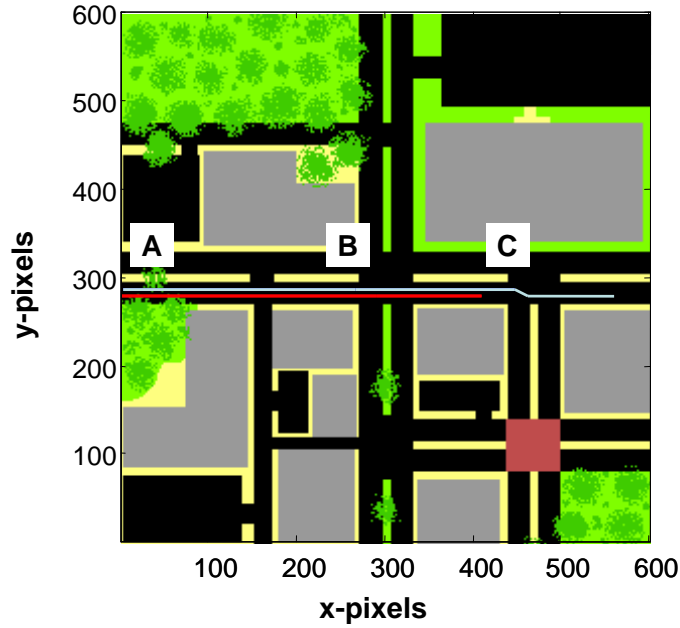
4.1.2 Simulated Measurement Noise. Although the simulations use synthetic data, the hyperspectral data that produced the imagery originate from real vehicle and background measurements. The tracking system also emulates the real environment using a simulated change detector and a simulated measurement generator, allowing for missed detections due to minimum detectable velocity and measurement dropouts due to occlusions. Furthermore, measurement noise is added to account for external influences that affect the generation of video detections (e.g., atmospheric effect, sensor noise, and jitter). Although these provide a certain level of realism, other effects are ignored such as false alarms due to clutter or background noise, parallax, and image registration errors.

4.1.3 Scenarios. The simulations evaluate two scenarios using two vehicles,² as shown in Fig. 4.1. Both scenarios, which are 20 seconds long, provide one or more vehicle maneuvers that generally cause the failure of a kinematic-only tracker. The following describes these two scenarios:

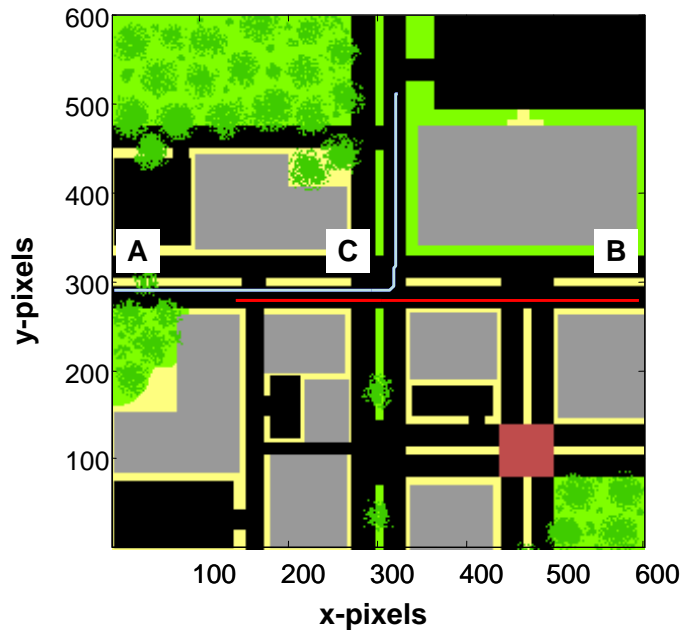
Scenario 1. Both vehicles depart from the west (Fig. 4.1(a)(A)), heading in an easterly direction. They travel side by side with the same speed. As the vehicles come to a complete stop at the intersection (Fig. 4.1(a)(B)), the tracker “loses”

¹Although the processing speed of `Matlab`[®] can be a significant drawback (compared to other languages such as C++), it is not a major consideration for this research effort. The easy-to-use environment and the availability of toolboxes outweigh the need for fast processing time.

²This research limits the number of targets to two vehicles to minimize computational complexity.



(a) Scenario 1.



(b) Scenario 2.

Figure 4.1: Both plots show two truth tracks (one light blue and one red) representing the trajectories of two vehicles. In scenario 1, both vehicles depart from the west (Fig. 4.1(a)(A)) and perform a move-stop-move at the intersection (Fig. 4.1(a)(B)). The top vehicle speeds up and overtakes the bottom vehicle (Fig. 4.1(a)(C)). In scenario 2, one vehicle departs from the west (Fig. 4.1(b)(A)), while the other departs from the east (Fig. 4.1(b)(B)). At the intersection (Fig. 4.1(b)(C)), both vehicles stop, and the vehicle from the east continues heading west, while the other vehicle turns and heads north.

track of both vehicles. Since the tracker simulates a change detection algorithm, the measurement generator stops providing kinematic observations once the speed of the vehicles goes below the minimum detectable velocity (MDV) of $1.5m/s$. After several seconds, the vehicles depart heading in the same direction, but this time, the top vehicle speeds up and passes the bottom vehicle (Fig. 4.1(a)(C)). This scenario has two ambiguous situations: (1) The spacing of both vehicles causes track swaps to occur. (2) At the intersection, both vehicles perform a move-stop-move, leading to track losses.

Scenario 2. One vehicle departs from the west (Fig. 4.1(b)(A)). The other vehicle departs from the east (Fig. 4.1(b)(B)). Both vehicles head toward each other at the same speed. At the intersection (Fig. 4.1(b)(C)), both vehicles come to a complete stop. Similar to scenario 1, the tracker loses track of both vehicles. After a few seconds, the vehicle coming from the east continues heading west, while the other vehicle turns left and heads north. The ambiguous situation in this scenario occurs at the intersection, where both vehicles perform a move-stop-move.

4.1.4 Configurations. Each configuration consists of user-defined parameters. The following list the different parameters and settings used in the simulations.

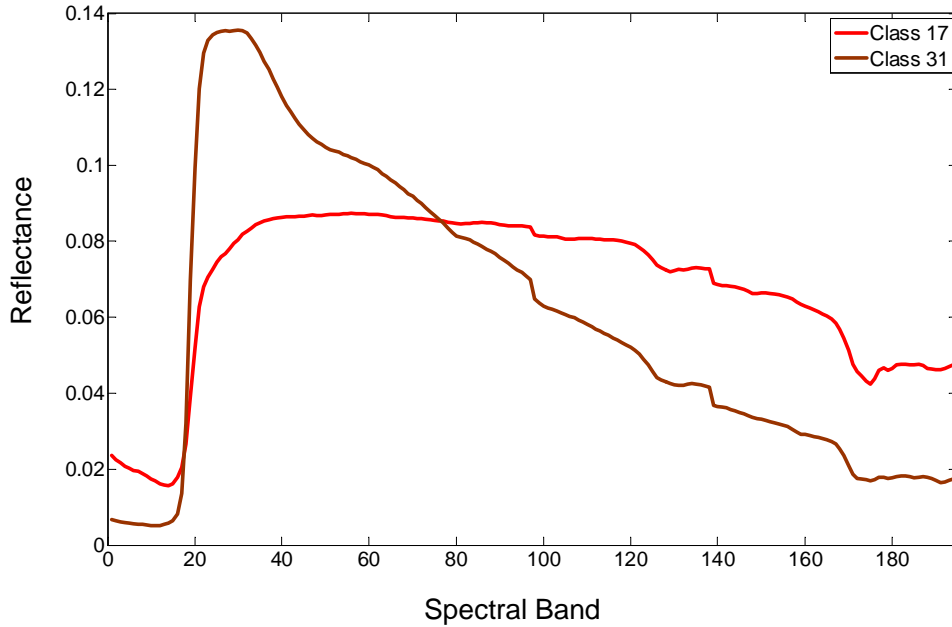
1. Truth trajectory – $\{[17,31],[17,32]\}$
2. Weight γ (Sec. 3.4) – $\{0.99,0.5,0.01\}$
3. Hyperspectral line scanner (Sec. 3.1.2) – $\{\text{Pushbroom, ROI}\}$
4. Class model (Sec. 3.7) – $\{\text{SOM lattice points, SOM weight/prototype vectors, FCM vectors}\}$
5. Linkage (Sec. 3.7) – $\{\text{Average}\}$
6. Map type (Sec. 3.3.2) – $\{\text{Original, Filtered}\}$
7. Scenario (Sec. 4.1.3) – $\{1, 2\}$

The truth trajectories consist of two combinations—two vehicles with class ID 17 (red₁) and class ID 31 (red₂) and two vehicles with class ID 17 (red₁) and class ID 32 (green). From Table 3.1, class ID 17 (red₁), class ID 31 (red₂), and class ID 32 (green) are red Saturn, red Pontiac, and green Chevy, respectively. The first combination corresponds to two similarly colored vehicles, whereas the second corresponds to two distinctly colored vehicles. Similarly colored truth trajectories allow for “ambiguous” hyperspectral features, since similarly colored vehicles potentially have similar hyperspectral signatures. However, Fig. 4.2 shows that, although class ID 17 (red₁) and class ID 31 (red₂) are described as red vehicles, both have different hyperspectral signatures. This is the power of hyperspectral data. One can easily discriminate between two similarly colored vehicles through their detailed spectral signatures. This can be attributed to different shades of color, the material composition of the vehicles’ exterior, and other materials on the vehicles’ surface such as dirt.

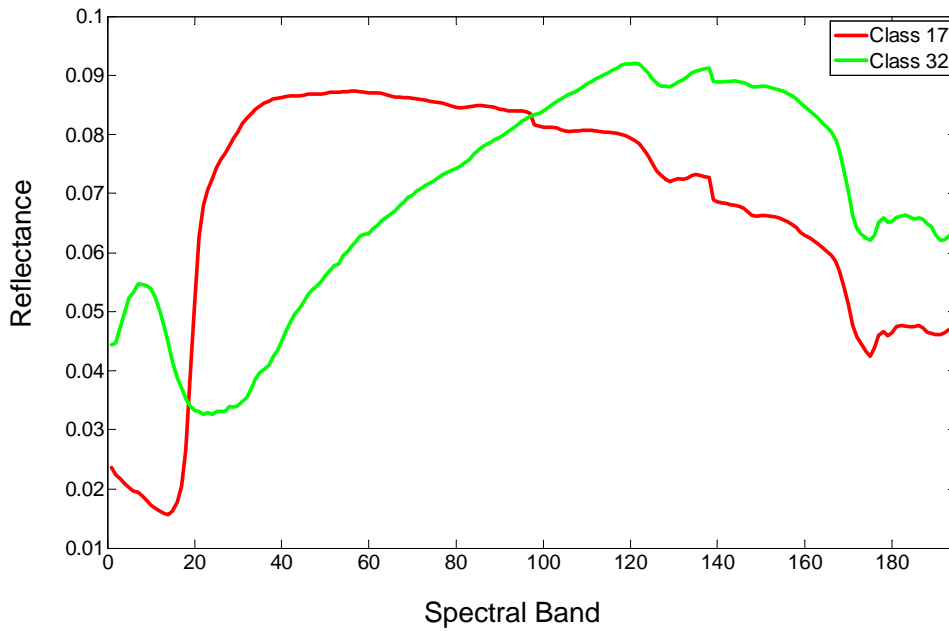
Since the main objective of the simulations is to compare the performance of the kinematic-only versus the hyperspectral-augmented tracker, the first set of configurations performs a baseline simulation using the kinematic-only tracker. The hyperspectral-augmented tracker performs similar simulations using different configurations of the parameters listed above.

4.1.4.1 Kinematic-Only Tracking. The kinematic-only tracking baseline simulation consists of four configurations (Table 4.1), which correspond to unique combinations of truth trajectories and scenarios. The kinematic-only tracker initiates each track using the class ID of the nearest vehicle (even if a track is already assigned to the vehicle). This is a design choice that provides a means to discriminate between two tracks in the scenario.³ The tracker does not update this ID for the life of a track. Because the tracker initiates tracks this way, the kinematic-only simulations have an advantage over the hyperspectral-augmented tracking simulations. Further-

³An unintended consequence of this design is that both tracks can be initialized using the class ID of the same vehicle.



(a) Reflectance plot of class 17 (red_1) and 31 (red_2).



(b) Reflectance plot of class 17 (red_1) and 32 (green).

Figure 4.2: The reflectance plots for class 17 (red_1) and 31 (red_2) of two similarly colored vehicles (a). It turns out that the hyperspectral signatures for both red vehicles are different. As expected, the reflectance plots for class 17 (red_1) and 32 (green) are distinct (b).

Configuration	Truth Trajectories	Scanner	Scenario
1	[17,31]	off	1
2	[17,31]	off	2
3	[17,32]	off	1
4	[17,32]	off	2

Table 4.1: The kinematic-only tracking baseline simulation (without hyperspectral augmentation) consists of four combinations of the truth trajectories and scenarios.

more, when a vehicle stops in a move-stop-move maneuver or is occluded from view (e.g., tree occlusions), the tracking logic subsequently deletes the track assigned to the vehicle (due to no updating information). If a track swap occurs prior to track deletion, i.e., the class ID of the track and the class ID of the vehicle to which the track is assigned do not match, the track is likely to be un-swapped upon track initiation. Thus, the class ID of the track and the class ID of the vehicle match once again after the vehicles start moving.

For example, in scenario 2, the tracks are always correctly identified until they reach the intersection. At the intersection, the tracker always swaps each track to the other vehicle because of the vehicles' close proximity. Assume that one of the tracks is assigned with class ID 17 (red₁) and the other with class ID 32 (green). The class ID 17 (red₁) track is tracking the class ID 17 (red₁) vehicle. A track swap occurs when the tracker assigns or swaps the track to the class ID 32 (green) vehicle, i.e., the observations for the class ID 32 (green) vehicle is now associated with the class ID 17 (red₁) track. A similar swap event also occurs for class ID 32 (green) track. After failing to receive update information, the tracker eventually deletes both tracks. As the vehicles begin to speed up ($MDV > 1.5m/s$), the tracker once again initiates each track using the class ID of the nearest vehicle, effectively swapping the tracks back; therefore, the class ID 17 (red₁) track, which was assigned to the class ID 32 (green) vehicle prior to track deletion, is now assigned back to class ID 17 (red₁) vehicle.

Alternative tracking logic or different Kalman filter tuning parameters can address the track deletions that occur in the move-stop-move maneuver or measurement dropouts due to tree occlusions. For the move-stop-move maneuver, if the alterna-

tive tracking logic allows tracks to persist while the vehicles are at rest rather than delete tracks (due to no updating information), the tracks can either remain correctly identified (class ID of track and vehicle match) or misidentified (if the class ID of track and vehicle do not match due to close proximity). One way to address the track deletion, which causes track discontinuity, with the current tracking logic is through track stitching, in which a new track is “connected” to the nearest deleted track. Doing so, the tracker updates the track’s class ID (based initially on the nearest vehicle) with the class ID of the nearest deleted track. Due to measurement noise, however, track continuity is not always maintained since a track is not always stitched to the correct deleted track. This actually works well because if the track persisted in the move-stop-move maneuver, it allows for the possibility that the kinematic-only tracker swaps each track back to the correct vehicle. The kinematic-only simulation of Sec. 4.2 illustrates this process.

4.1.4.2 Hyperspectral-Augmented Tracking. The hyperspectral-augmented tracking simulation consists of the FCM and two-stage SOM classification approaches. The results of the two classification approaches, which should perform well with mixed spectra, are compared in Sec. 4.4.

4.1.4.2.1 Fuzzy C-Means. The hyperspectral-augmented tracker integrates the FCM classification algorithm (described in Sec. 3.3.1) into the tracking process. Since the SOM evaluates the other values for γ , the FCM simulation uses $\gamma = 0.5$ only for all combinations of truth trajectories and scenarios, for a total of four configurations. The four FCM configurations are summarized in Table 4.2. Note that the simulations only implement the ROI hyperspectral line scanning. In [6], the results of the FCM simulations show that the hyperspectral sensor mode is not a critical performance driver. Therefore, the pushbroom implementation is not investigated for the FCM.

Configuration	Truth Trajectories	Scanner	Weight γ	Scenario
1	[17,31]	ROI	0.5	1
2	[17,31]	ROI	0.5	2
3	[17,32]	ROI	0.5	1
4	[17,32]	ROI	0.5	2

Table 4.2: The FCM simulation only uses $\gamma = 0.5$, ROI, and the four combinations of truth trajectories and scenarios.

4.1.4.2.2 Self-Organizing Map.

The hyperspectral-augmented tracker integrates the SOM classification algorithm (described in Sec. 3.3.2) using various parameter combinations. The following list summarizes the parameter combinations into six configuration sets, which are outlined in Table 4.3:

- A.** Configurations 1-8 – $\{\{[17,31],[17,32]\}, 0.5, \text{ROI}, \text{SOM lattice points}, \{\text{Original}, \text{Filtered}\}, \{1, 2\}\}$
- B.** Configurations 8-16 – $\{\{[17,31],[17,32]\}, 0.5, \text{Pushbroom}, \text{SOM lattice points}, \{\text{Original}, \text{Filtered}\}, \{1, 2\}\}$
- C.** Configurations 17-24 – $\{\{[17,31],[17,32]\}, 0.5, \text{ROI}, \text{SOM Weight vectors}, \{\text{Original}, \text{Filtered}\}, \{1, 2\}\}$
- D.** Configurations 25-32 – $\{\{[17,31],[17,32]\}, 0.5, \text{Pushbroom}, \text{SOM Weight vectors}, \{\text{Original}, \text{Filtered}\}, \{1, 2\}\}$
- E.** Configurations 33-40 – $\{\{[17,31]\}, \{0.9,0.01\}, \text{ROI}, \text{SOM lattice points}, \{\text{Original}, \text{Filtered}\}, \{1, 2\}\}$
- F.** Configurations 41-48 – $\{\{[17,32]\}, \{0.9,0.01\}, \text{ROI}, \text{SOM lattice points}, \{\text{Original}, \text{Filtered}\}, \{1, 2\}\}$

Each configuration set consists of eight configurations because there are eight unique combinations for the two truth trajectories, two SOM maps, and two scenarios common to all configuration sets. For configuration sets A and B, the difference is in the sensor mode. By comparing the two configuration sets, the difference in performance is shown between pushbroom and ROI while using the SOM lattice

points. For configuration sets C and D, the sensor modes are also compared, but this time, using the SOM prototype vectors. Configuration sets A and C evaluate the performance between the SOM lattice points and weight vectors. For configuration set E, the other weighting factors (0.9 and 0.01) are implemented for the first truth trajectory, and for configuration set F, the other weighting factors are implemented for the second truth trajectory. Both configuration sets E and F use the scanner and class model settings used in configuration set A.

Table 4.3: The hyperspectral-augmented tracker performs a SOM classification approach using various settings (Sec. 4.1.4) for the truth trajectories, weight γ , scanner type, class model, original versus filtered map, and scenario parameters.

Index	Truth	Weight	Scanner	Class model	Map	Scenario
1	[17,31]	0.5	ROI	SOM Lattice	Original	1
2						2
3					Filtered	1
4						2
5	[17,32]				Original	1
6						2
7					Filtered	1
8						2
9	[17,31]	0.5	Pushbroom	SOM Lattice	Original	1
10						2
11					Filtered	1
12						2
13	[17,32]				Original	1
14						2
15					Filtered	1
16						2
Continued on next page						

Table 4.3 – continued from previous page

Index	Truth	Weight	Scanner	Class model	Map	Scenario			
17	[17,31]	0.5	ROI	Weight Vectors	Original	1			
18						2			
19					Filtered	1			
20						2			
21	[17,32]				0.5	ROI	Weight Vectors	Original	1
22									2
23								Filtered	1
24									2
25	[17,31]	0.5	Pushbroom	Weight Vectors				Original	1
26									2
27								Filtered	1
28									2
29	[17,32]				0.5	Pushbroom	Weight Vectors	Original	1
30									2
31								Filtered	1
32									2
33	[17,31]	0.99	ROI	SOM Lattice				Original	1
34									2
35								Filtered	1
36									2
37		0.01			0.01	ROI	SOM Lattice	Original	1
38									2
39								Filtered	1
40									2

Continued on next page

Table 4.3 – continued from previous page

Index	Truth	Weight	Scanner	Class model	Map	Scenario
41	[17,32]	0.99	ROI	SOM Lattice	Original	1
42						2
43					Filtered	1
44						2
45		0.01			Original	1
46						2
47					Filtered	1
48						2

4.1.5 *Simulations.* A Monte Carlo (MC) evaluation, which is the approach used to evaluate the system, consists of 100 simulation runs per configuration. Ensemble statistics for the performance measures of Sec. 3.9 are calculated for each MC evaluation. Results are generated and compiled for each run.

One useful and informative feature of the simulation is the visualization of the simulated panchromatic video images. Fig. 4.3 is an example of the state of the system at a particular time step for a SOM-based simulation run. The left plot indicates the true target trajectories, track states with confidence bounds, and the current hyperspectral scan line (horizontal line located at line 61). The center top plot is a spectral representation of the finalized hyperspectral image (HSI) chip. The center left plot shows the pixel classification of the HSI chip in red, green, blue format (RGB), and for comparison, the center right plot shows the true class label of each pixel. The fused chip exemplifies the non-rectangular nature allowed by the system; in fact, chips occasionally contain holes (black regions in the upper left, upper right, and lower right regions). This instance contains several misclassified pixels. For example, several pixels in the red vehicle’s pixel region are either unknown or classified as either target or background. The classifier also identified several background pixels as green vehicle. The center bottom plot indicates the morphological operations performed by

the associator, where the cyan '*' represents the centroid of each hyperspectral region and the green '*' represents the track predicted position states. A line is drawn for each observation-to-track association. The P_{ID} above the right plot shows that the associator assigned both vehicles incorrectly. The track with class ID 32 (green) is associated with the vehicle with class ID 17 (red₁). Similarly, the track with class ID 17 (red₁) is associated with the vehicle with class ID 32 (green). Because of the misassociation, each track is bounded by a red circle. If a track is correctly identified, it is bounded by a green circle.

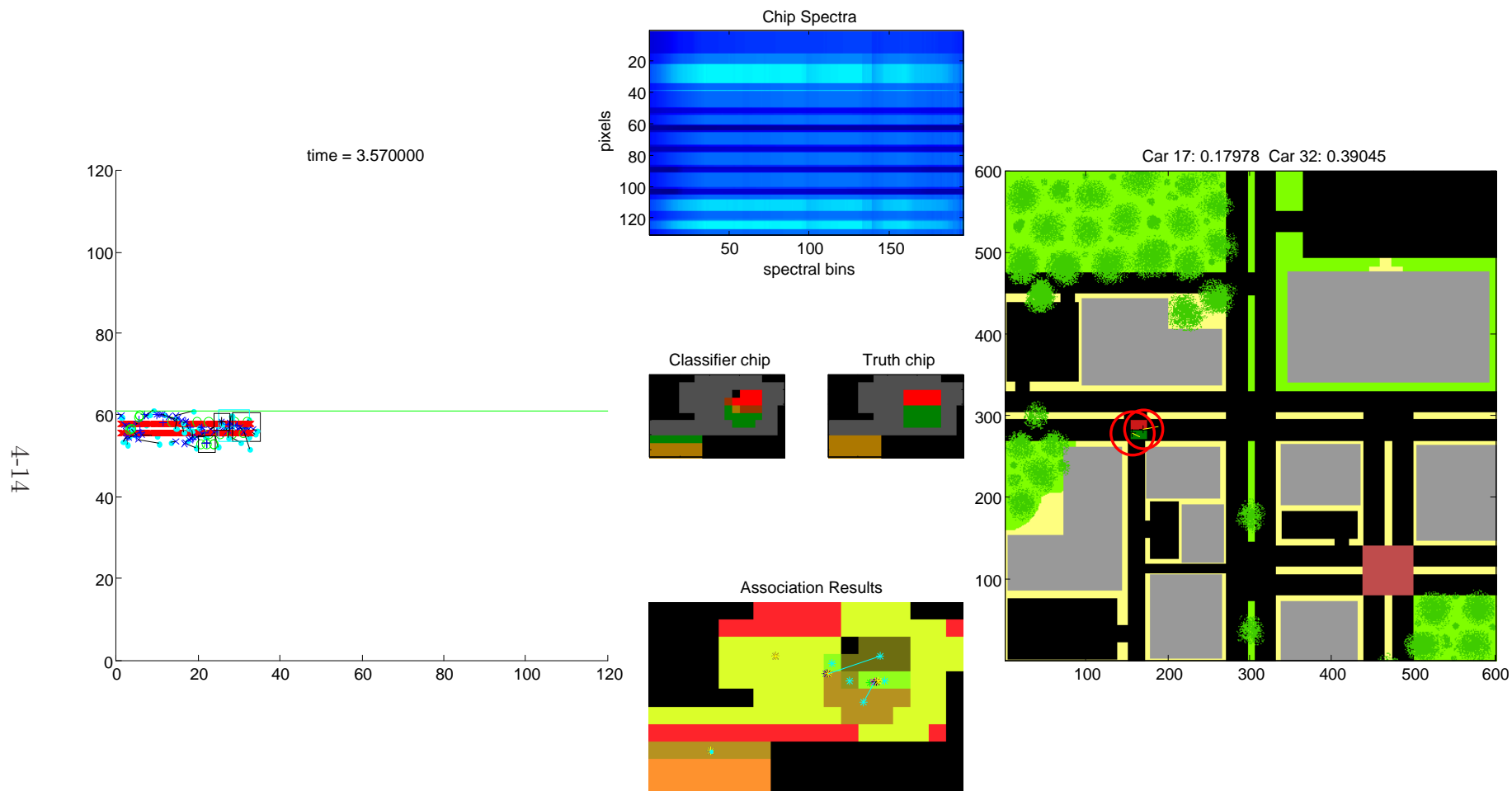


Figure 4.3: The simulation display is a visualization of the state of the system at a particular time. The parameters are $\{[17,32], 0.5, \text{ROI}, \text{SOM lattice points}, \text{Average}, \text{Original}, 2\}$. The values above the right plot provide the P_{ID} performance measure. Since the tracker swapped both tracks, the P_{ID} for each track is low.

4.2 Kinematic-Only Tracking

The kinematic-only baseline simulation uses the configurations summarized in Table 4.1. Each simulation performs an MC evaluation, in which the truth trajectories are resampled at $10Hz$ with additive Gaussian distributed noise (mean = $0m$, standard deviation = $2m$). As discussed in Sec. 2.6, the associator performs a truth-to-track assignment for the computation of track statistics. Percentage of correct identification (P_{ID}) is the primary performance measure. By comparing the P_{ID} between kinematic-only and hyperspectral-augmented tracking, the feasibility of the hyperspectral-augmented tracking can be verified. It is worth noting that the kinematic-only simulations actually have an advantage over the hyperspectral-augmented simulations because the tracker initiates a track using the class ID of the nearest vehicle (see Sec. 4.1.4.1). Hence, the track's P_{ID} starts increasing after track confirmation (assuming subsequent assigned observations belong to the nearest vehicle). For every frame that a correct match occurs between the class ID of the track and the class ID of the nearest vehicle, a green circle bounds the track and the P_{ID} of the track increases. Otherwise, a red circle bounds the track and the P_{ID} of the track decreases. Figs. 4.4 and 4.5 illustrate a single tracking simulation run for configuration 1 at various time steps.

4.2.1 Configuration 1. A single MC run for configuration 1 is described in this section. It consists of a class ID 17 (red₁) and a class ID 32 (green) vehicle. Both vehicles start out heading east. The tracker initiates each track using the class ID of the nearest vehicle (Fig. 4.4(a)). Sometime between $t = 0.8s$ and $t = 1.0s$, the class ID 17 (red₁) track experiences measurement dropouts due to tree occlusions, causing the tracker to delete it⁴ (Fig. 4.4(b)). Between $t = 1.0s$ and $t = 1.2s$, the tracker receives kinematic measurements intermittently from both vehicles, keeping the class ID 32 (green) track alive (Fig. 4.4(c)). At one point, however, the class ID 32 (green)

⁴Note that measurement dropouts do not necessarily result in track deletion. They can be addressed using alternative tracking logic similar to the move-stop-move maneuver described in Sec. 4.1.4.1. Tracks can be allowed to persist unless contrary evidence comes in from an observation.

vehicle becomes completely occluded, and the tracker consistently updates the class ID 32 (green) track using the kinematic measurements of class ID 17 (red₁) vehicle. This event is called a track swap. Since their respective class IDs do not match, the track is bounded by a red circle and its P_{ID} decreases. Sometime between $t = 1.2s$ and $t = 2s$, the tracker initiates a track using the kinematic measurements of class ID 32 (green) vehicle, but because the nearest vehicle is the class ID 17 (red₁) vehicle, the tracker assigns the track with class ID 17 (red₁). After track confirmation, the tracker stitches the track to the nearest deleted track;⁵ hence, it updates the class ID with the previous deleted class ID 17 (red₁) track (the class IDs happen to be the same). The tracker updates the class ID 17 (red₁) track using the kinematic measurements of class ID 32 (green) vehicle (Fig. 4.4(d)).

In Fig. 4.5(a), both tracks remain swapped until they reach the intersection where they come to a complete stop. Both tracks meet the deletion criteria and are deleted by the tracker (Fig. 4.5(b)). After both vehicles speed up to an $MDV > 1.5m/s$, the tracker initiates each track using the class ID of the nearest vehicle, “re-assigning” each previously deleted swapped track back to the correct vehicle. After track confirmation, the tracker stitches each track to the nearest deleted track. The confirmed class ID 32 (green) track is stitched to the deleted class ID 17 (red₁) track, which was previously assigned to the class ID 32 (green) vehicle prior to track deletion (Fig. 4.5(c)). Doing so, the tracker updates the class ID of the confirmed track with the class ID of the deleted track; thus, swapping it from class ID 32 (green) to class ID 17 (red₁). This is the desired effect, since a swapped track needed to remain swapped after track deletion.⁶ The confirmed class ID 17 (red₁) track also undergoes the same process. Both tracks remain swapped throughout the rest of the simulation (Fig. 4.5(d)).

⁵Note that alternative tracking logic can address the tracking difficulties in the move-stop-move maneuver, as described in Sec. 4.1.4.1. For the current tracking logic, in order for swapped tracks to remain in the swapped state after track deletion, track stitching is performed.

⁶Due to measurement noise, however, track continuity is not always maintained since not all new tracks are stitched to the correct deleted track.

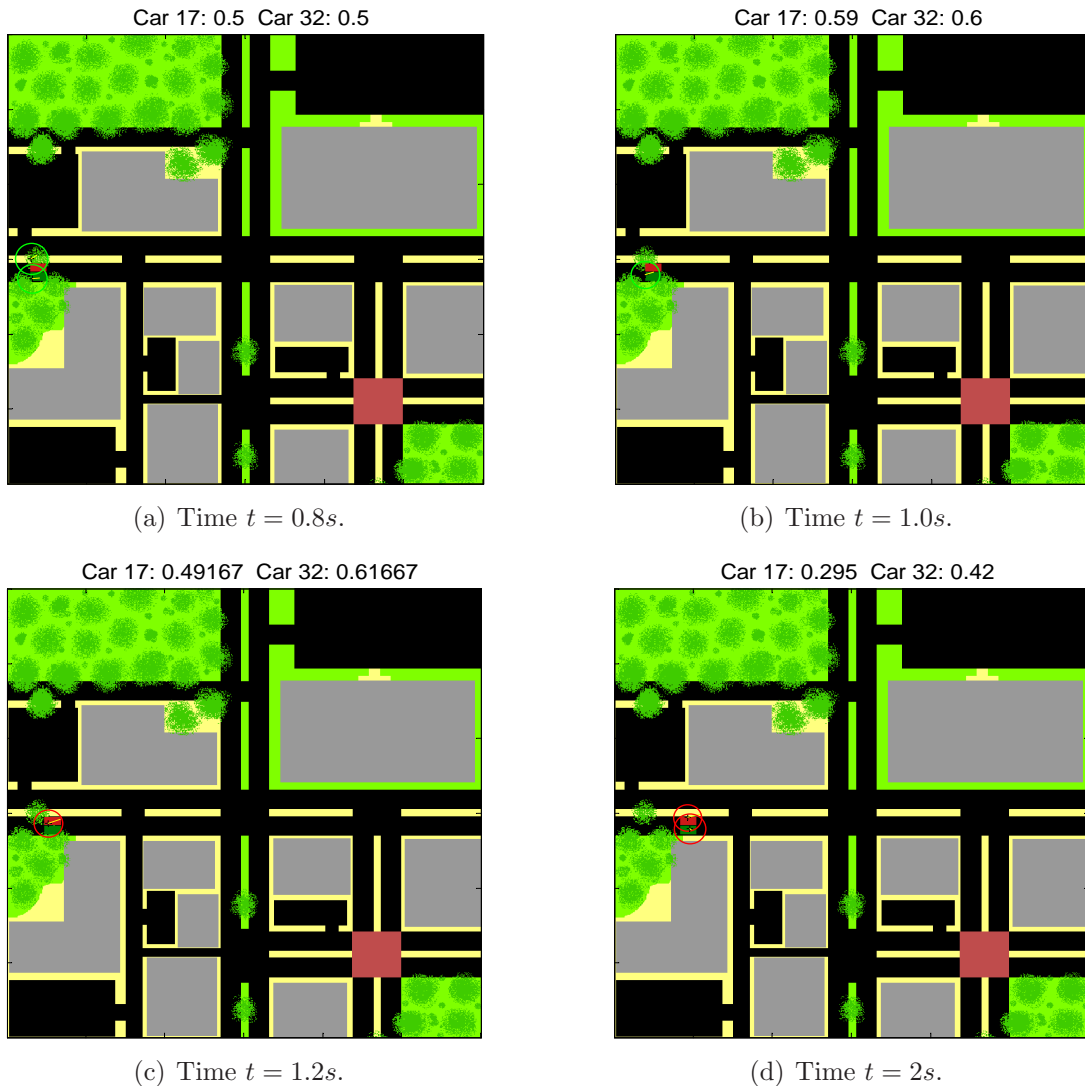


Figure 4.4: **Kinematic-only tracking for configuration 1 (Part A)**. This configuration consists of a class ID 17 (red_1) and a class ID 32 (green) vehicle. Both vehicles start out heading east. The tracker initiates each track using the class ID state of the nearest vehicle (a). Because of tree occlusions, the tracker deletes the class ID 17 (red_1) track (b) and swaps the class ID 32 (green) track to the class ID 17 (red_1) vehicle (c). After the vehicles pass the trees, the tracker initiates a track and assigns it the class ID 17 (red_1), which is the nearest vehicle. The tracker updates the class ID 17 (red_1) track using the kinematic measurements of class ID 32 (green) vehicle.

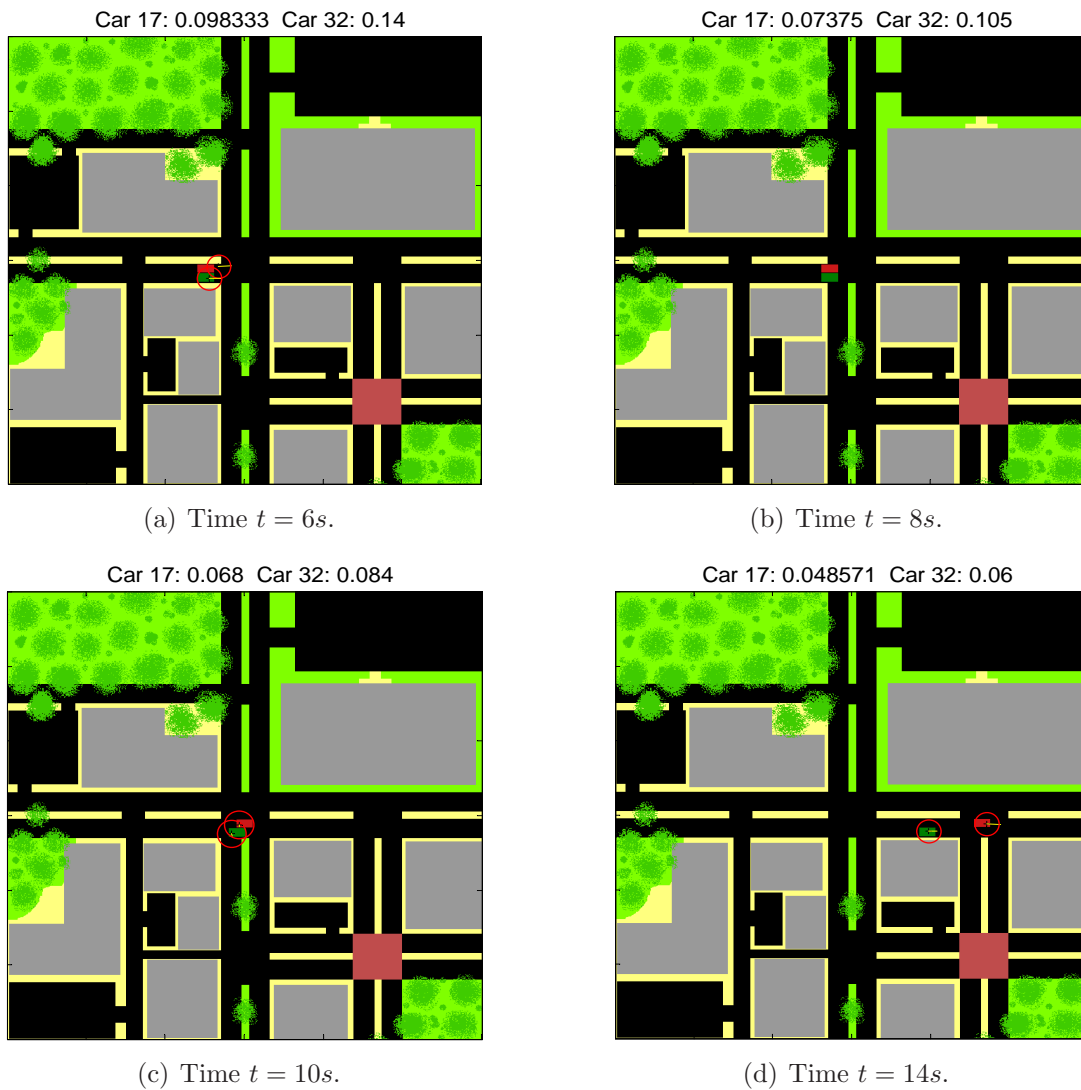


Figure 4.5: **Kinematic-only tracking for configuration 1 (Part B)**. The tracks remain swapped until they reach the intersection where both vehicles stop (a). The tracker fails to receive update information and deletes both tracks (b). After both vehicles start moving again, the tracker initiates a track for each and stitches each confirmed track to the nearest deleted track (c). They remain swapped throughout the rest of the simulation (d).

4.2.2 *Configuration 2.* Fig. 4.6 illustrates a single tracking simulation run for configuration 2 at various time steps. Prior to reaching the intersection, the tracker correctly identifies both tracks for $\approx 93.5\%$ of the frames (Figs.4.6 (a) and (b)). Because of the close proximity at the intersection, each track swaps to the other vehicle and becomes bounded by a red circle (Fig. 4.6(c)). As the vehicles come to a complete stop, the tracks continue to propagate and are eventually deleted by the tracker (no updating kinematic observations available) (Fig. 4.6(d)). When the vehicles begin moving ($MDV > 1.5m/s$), the tracker detects the motion and initiates tracks for both vehicles. After track confirmation, the tracker stitches each track to the nearest deleted track.⁷ For this Monte Carlo run, the class ID 17 (red₁) track is stitched to the deleted class ID 32 (green) track, and the class ID 32 (green) track is stitched to the deleted class ID 17 (red₁) track⁸ (Fig. 4.6(e)). The tracks remain swapped throughout the rest of the simulation (Fig. 4.6(f)).

⁷Note that alternative tracking logic or different Kalman filter tuning parameters can address the tracking difficulties in the move-stop-move maneuver, as described in Sec. 4.1.4.1. For the current tracking logic, in order for swapped tracks to remain in the swapped state after track deletion, track stitching is performed.

⁸Due to measurement noise, however, track continuity is not always maintained since not all new tracks are stitched to the correct deleted track.

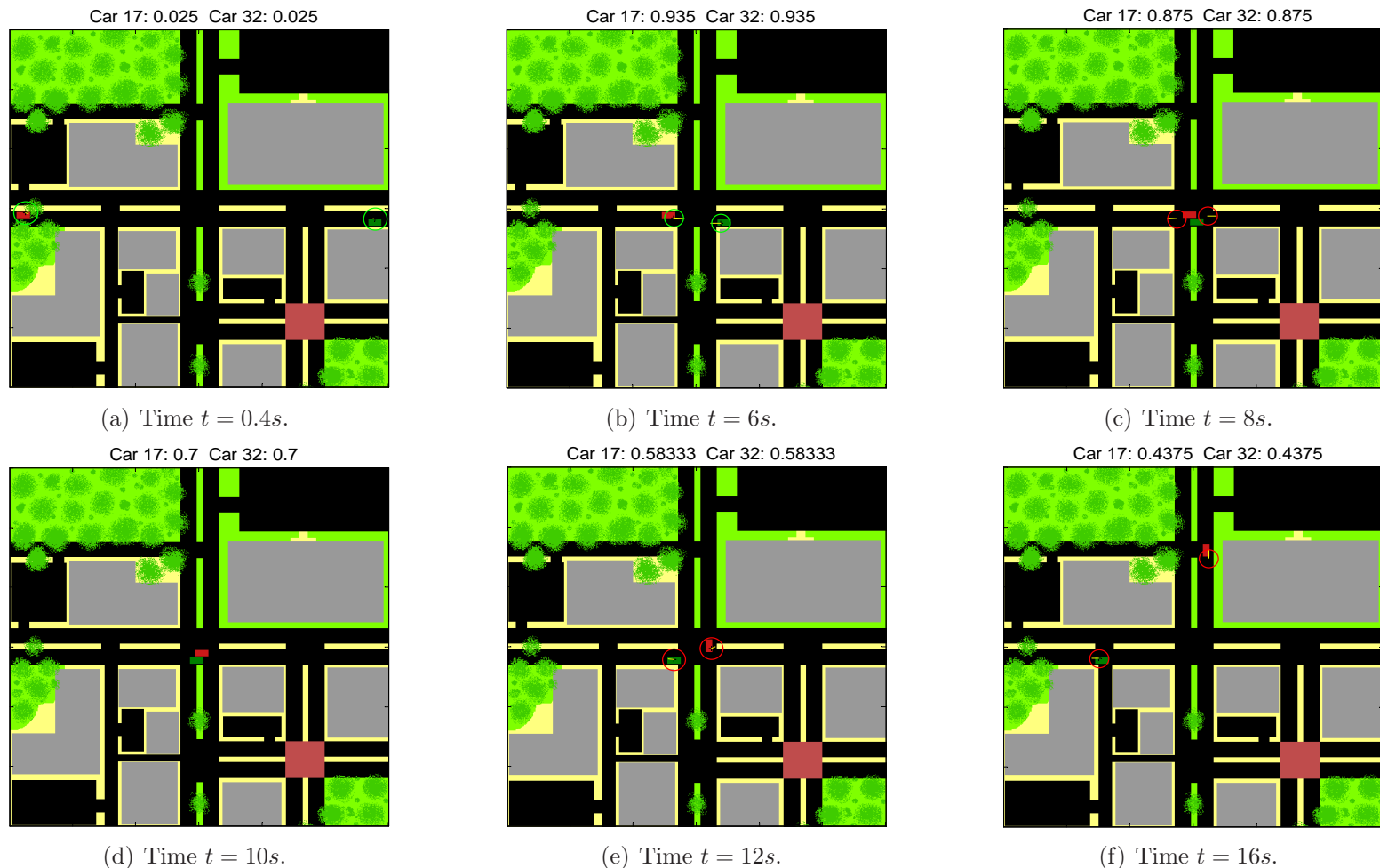


Figure 4.6: **Kinematic-only tracking for configuration 2.** By $t = 0.4s$, the tracker correctly identifies both tracks (a) until the intersection where they go through an ambiguous situation (b). The tracker swaps both tracks, which continue to propagate after the vehicles stop (c). The tracker eventually deletes the tracks after being starved of kinematic observations (d). When the vehicles start moving again ($MDV > 1.5m/s$), the tracker initiates tracks for both vehicles. After track confirmation, the class ID 17 (red₁) track is stitched to the deleted class ID 32 (green) track, and the class ID 32 (green) track is stitched to the deleted class ID 17 (red₁) track (e). The tracks remain swapped throughout the rest of the simulation (f).

4.3 *Hyperspectral-Augmented Tracking*

As the name implies, hyperspectral-augmented tracking augments the kinematic-only tracking with hyperspectral data. Each track starts out with no feature information until after the hyperspectral sensor scans the track region. The hyperspectral line scanning is prompted when the tracker confirms the track. The hyperspectral-augmented tracker updates the class ID of the confirmed track with the class ID of the first hyperspectral observation assigned to the track. This implementation relies heavily on the performance of the classifier; i.e., the first assigned hyperspectral observation must have the hyperspectral signature of the vehicle (or at a minimum, one of its nearest neighbors) with kinematic measurements that previously updated the kinematic states of the track under consideration. Once the class ID is updated, the only way it can change is through assigned hyperspectral observations with class ID that is within the track’s spectral gate. This is especially critical when the vehicles are closely spaced during initial hyperspectral scanning. The simulation run for scenario 2 illustrates this point. For comparison, images are taken at the same time steps as their equivalent kinematic-only simulation run.

4.3.1 Fuzzy C-Means Configuration 1. Fig. 4.7 and 4.8 illustrate an FCM simulation run for configuration 1 of Sec. 4.2. The FCM simulation uses the corresponding FCM configuration from Table 4.2.⁹ In Fig. 4.7, the tracker assigns an unknown class ID to both tracks during track initiation (based on panchromatic video) (Fig. 4.7(a)). Note that when the tracker confirms a track, it cues the hyperspectral sensor to scan the track region. Sometime between $t = 0.8s$ and $t = 1.0s$, measurement dropouts due to tree occlusions cause the tracker to delete the (unknown) track assigned to the class ID 17 (red₁) vehicle¹⁰ (Fig. 4.7(b)). By $t = 1.0s$, the hyperspectral sensor has scanned the (unknown) track region for the class ID 32 (green)

⁹Kinematic-only configuration 1 corresponds to FCM configuration 1, kinematic-only configuration 2 corresponds to FCM configuration 2, and so on.

¹⁰The deletion logic for panchromatic video is met prior to the hyperspectral scanning of the track region (see Sec. 3.5). Note that measurement dropouts do not necessarily result in track deletion. They can be addressed using alternative tracking logic similar to the move-stop-move maneuver

vehicle. Since both vehicles are closely spaced, the sensor scans part of the class ID 17 (red₁) vehicle while the class ID 32 (green) vehicle is completely hidden from view. The classification of the first assigned hyperspectral observation happens to be class ID 17 (red₁). The tracker, therefore, updates the class ID of the (unknown) track from the unknown ID to class ID 17 (red₁). Furthermore, because the class ID 32 (green) vehicle remains occluded, the tracker updates the kinematic states of class ID 17 (red₁) track with the kinematic and hyperspectral observations of class ID 17 (red₁) vehicle (Fig. 4.7(c)). One might conclude that the tracker performs a track swap. From the perspective of track purity, this is true. However, for this research, *so long as the tracker correctly identifies the vehicle ($P_{ID} \uparrow$), even if it swaps the track (track purity \downarrow), the tracking goal is achieved effectively.*

The tracker initiates an (unknown) track for the class ID 32 (green) vehicle after it becomes unoccluded. At $t = 2s$, the tracker confirms the (unknown) track, but the hyperspectral sensor has not scanned the track region. At the same time, the class ID 17 (red₁) track is closer to the class ID 32 (green) vehicle; thus, the truth-to-track association assigns it to the class ID 32 (green) vehicle. Hence, both tracks are bounded by a red circle (Fig.4.7(d)). After $t = 2s$, the sensor scans the (unknown) track region, and the tracker updates it with class ID 32 (green). Once both tracks maintain a steady trajectory, the tracker correctly identifies both tracks consistently, as shown in Fig. 4.8(a). At $t = 8s$, both vehicles come to a complete stop, causing the tracker to delete both tracks¹¹ (Fig. 4.8(b)). By $t = 10s$, the tracker has initiated tracks for both vehicles, the hyperspectral sensor has scanned both track regions, and the tracker has assigned the correct class ID to each track (Fig. 4.8(c)). Finally, the tracker correctly identifies both tracks throughout the rest of the simulation (Fig. 4.8(d)). Recall that the kinematic-only tracker left the tracks swapped for the duration of the simulation.

described in Sec. 4.1.4.1. Tracks can be allowed to persist unless contrary evidence comes in from an observation.

¹¹Note that alternative tracking logic can address the tracking difficulties in the move-stop-move maneuver, as described in Sec. 4.1.4.1.

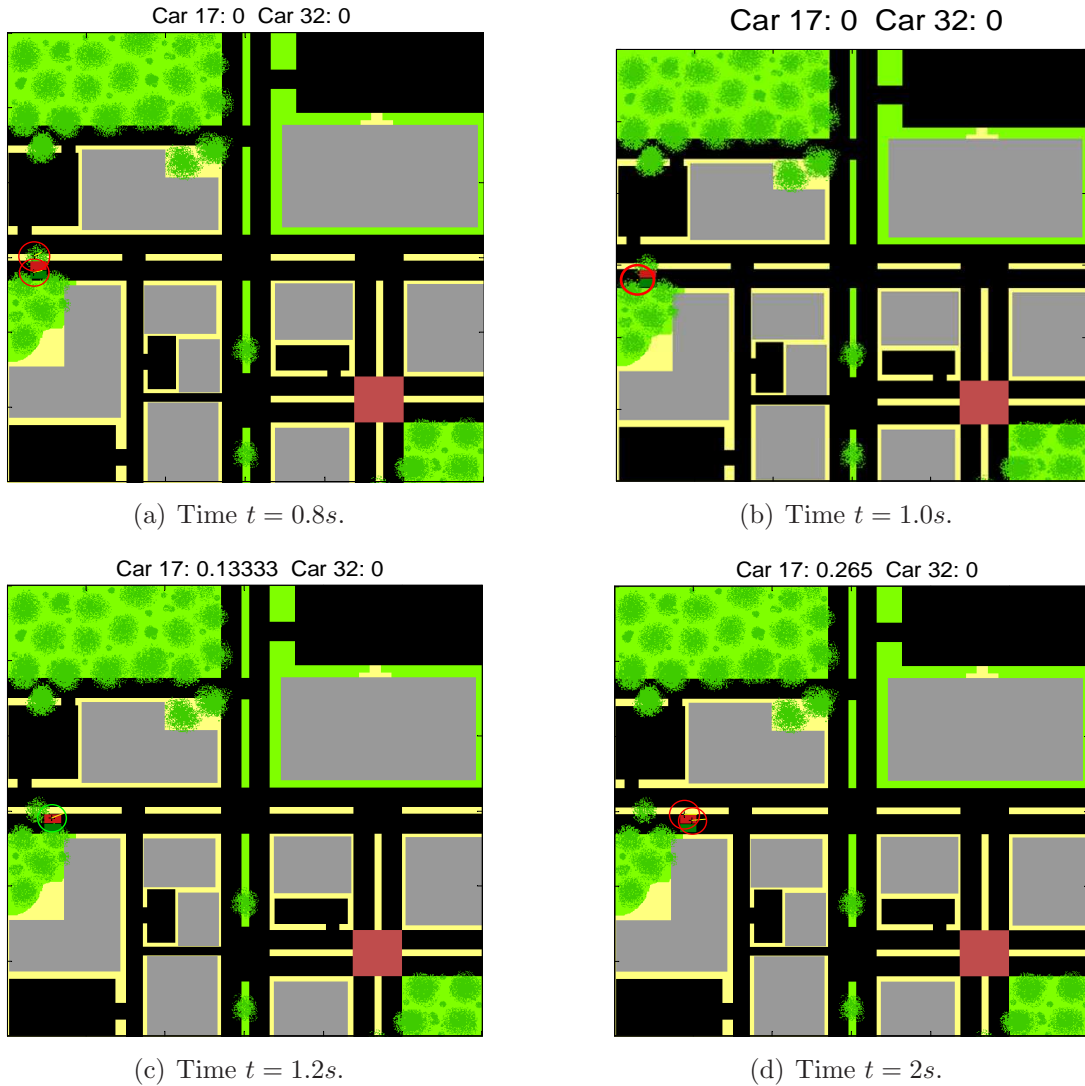


Figure 4.7: **Hyperspectral-augmented tracking version of Fig. 4.4 for fuzzy c-means (Part A).** During track initiation, the tracker assigns an unknown class ID to both tracks (a). Measurement dropouts due to tree occlusions cause the deletion of the track assigned to the class ID 17 (red_1) vehicle (b). The hyperspectral sensor scans the remaining (unknown) track region and updates it with class ID 17 (red_1). Because the class ID 32 (green) vehicle is hidden from view, the tracker updates the kinematic states of the class ID 32 (green) using the observations of class ID 17 (red_1) vehicle (c). At $t = 2s$, the truth-to-track association assigns the class ID 17 (red_1) track to the class ID 32 (green) vehicle. At the same time, the tracker initiates an (unknown) track for the class ID 32 (green) vehicle (d).

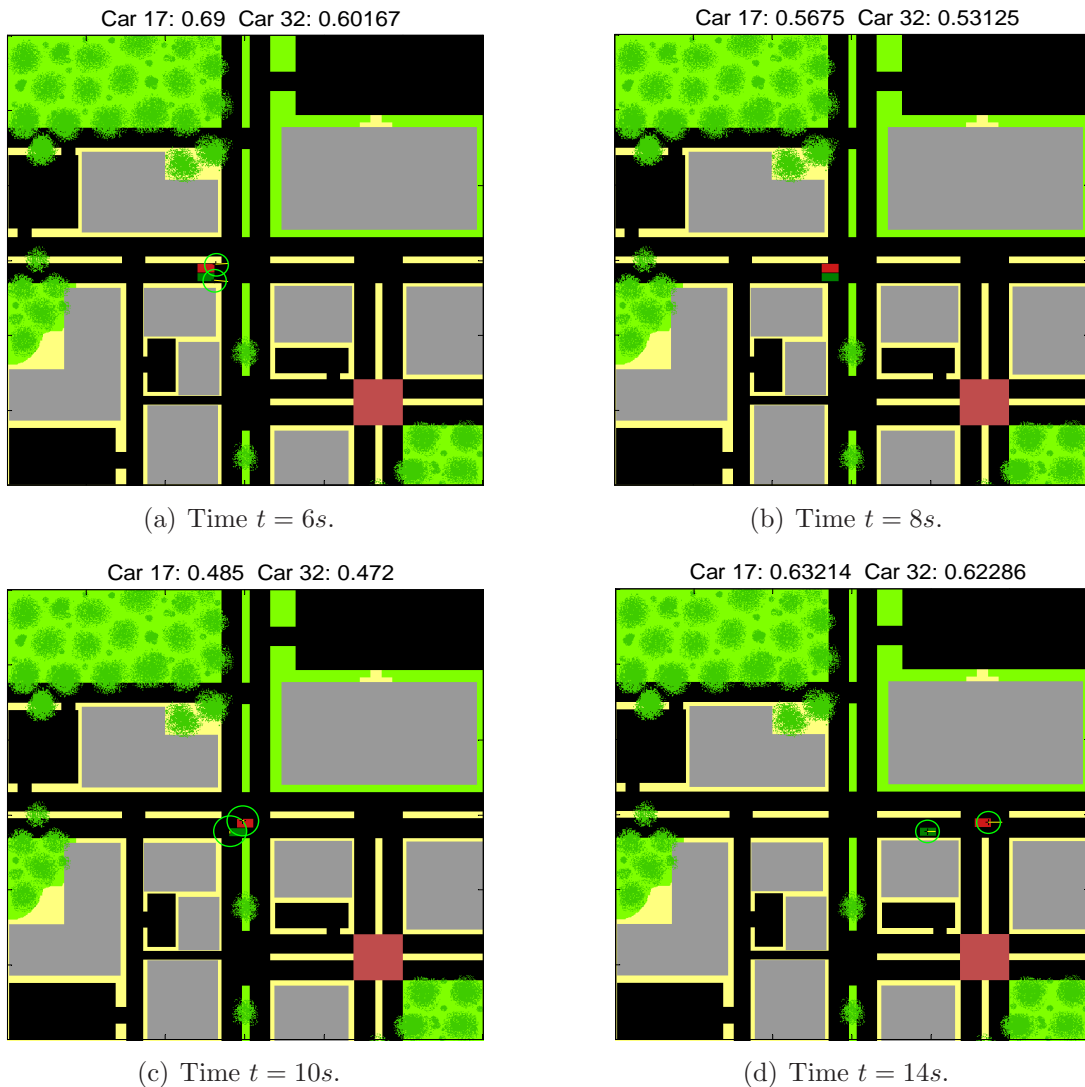


Figure 4.8: **Hyperspectral-augmented tracking version of Fig. 4.4 for fuzzy c-means (Part B).** By $t = 6s$, the tracker updates each track using the observations with matching class ID (a). As the vehicles slows to a stop at the intersection, the tracks are starved of observations, causing the tracker to delete them (b). By $t = 10s$, the tracker has initiated tracks for both vehicles, the hyperspectral sensor has scanned the track regions, and the tracker associates the hyperspectral observations with matching class ID to each track (c). The tracker correctly identifies both tracks throughout the rest of the simulation (d).

4.3.2 *Self-Organizing Map Configuration 1.* Figs. 4.4 and 4.5 illustrate a SOM simulation run for configuration 1 of Sec. 4.2 using SOM configuration 7 from Table 4.3. The parameters are [17,32], $\gamma = 0.5$, ROI line scanner, SOM lattice points, filtered map, and scenario 1. The tracker initiates tracks for both vehicles and assigns the unknown class ID to both tracks (Fig. 4.9(a)). When the vehicles are partly occluded by the trees, the tracker deletes the (unknown) track for the class ID 17 (red₁) vehicle due to measurement dropouts.¹² The hyperspectral sensor scans the (unknown) track region for the class ID 32 (green) vehicle; however, the tracker fails to assign a class ID to the track (no hyperspectral observation with any vehicle class ID exists in the HSI chip). During this time, the class ID 32 (green) vehicle becomes completely hidden from view. The tracker updates the (unknown) track intermittently using the kinematic measurements of class ID 17 (red₁) vehicle (Fig. 4.9(b) and (c)). After $t = 1.2s$, the hyperspectral sensor scans the (unknown) track region again, and the tracker updates the (unknown) track with class ID 17 (red₁). Within the same time period, the green vehicle becomes unoccluded. The tracker initiates an (unknown) track for the class ID 32 (green) vehicle. After the hyperspectral sensor scans the (unknown) track, the tracker assigns it with class ID 32 (green). (Fig. 4.9(d) and Fig. 4.10(a)).

At the intersection, both vehicles come to a complete stop, but the tracks continue to propagate.¹³ In Fig. 4.10(b), the class ID 17 (red₁) track is still alive, and since the class ID 17 (red₁) vehicle is outside the kinematic gate of class ID 17 (red₁) track, the track is bounded by a red circle and its P_{ID} decreases. The class ID 17 (red₁) red vehicle speeds up to overtake the class ID 32 (green) vehicle. The tracker detects its motion and performs track initiation, which cues the hyperspectral sensor to scan the track region. Based solely on the hyperspectral line scanner, the tracker initiates and

¹²Note that measurement dropouts do not necessarily result in track deletion. They can be addressed using alternative tracking logic similar to the move-stop-move maneuver described in Sec. 4.1.4.1. Tracks can be allowed to persist unless contrary evidence comes in from an observation.

¹³Note that alternative tracking logic can address the tracking difficulties in the move-stop-move maneuver, as described in Sec. 4.1.4.1.

confirms two “tracks” (red circles in Fig. 4.10(c)) due to the classification results of the HSI chip. Two hyperspectral observations are formed with vehicle class ID other than the two vehicles and their nearest neighbors. The class ID of both observations are outside the vehicles’ spectral gate, and therefore, unassociated. The tracker initiates a track for each hyperspectral observation. After $t = 10s$, the two “tracks” eventually meet the deletion criteria. The tracker initiates an (unknown) track for the class ID 32 (green) vehicle. After hyperspectral scanning, the tracker assigns the hyperspectral observation with correct class ID to the track. The tracker correctly identifies both target tracks throughout the rest of the simulation (Fig. 4.10(d)).

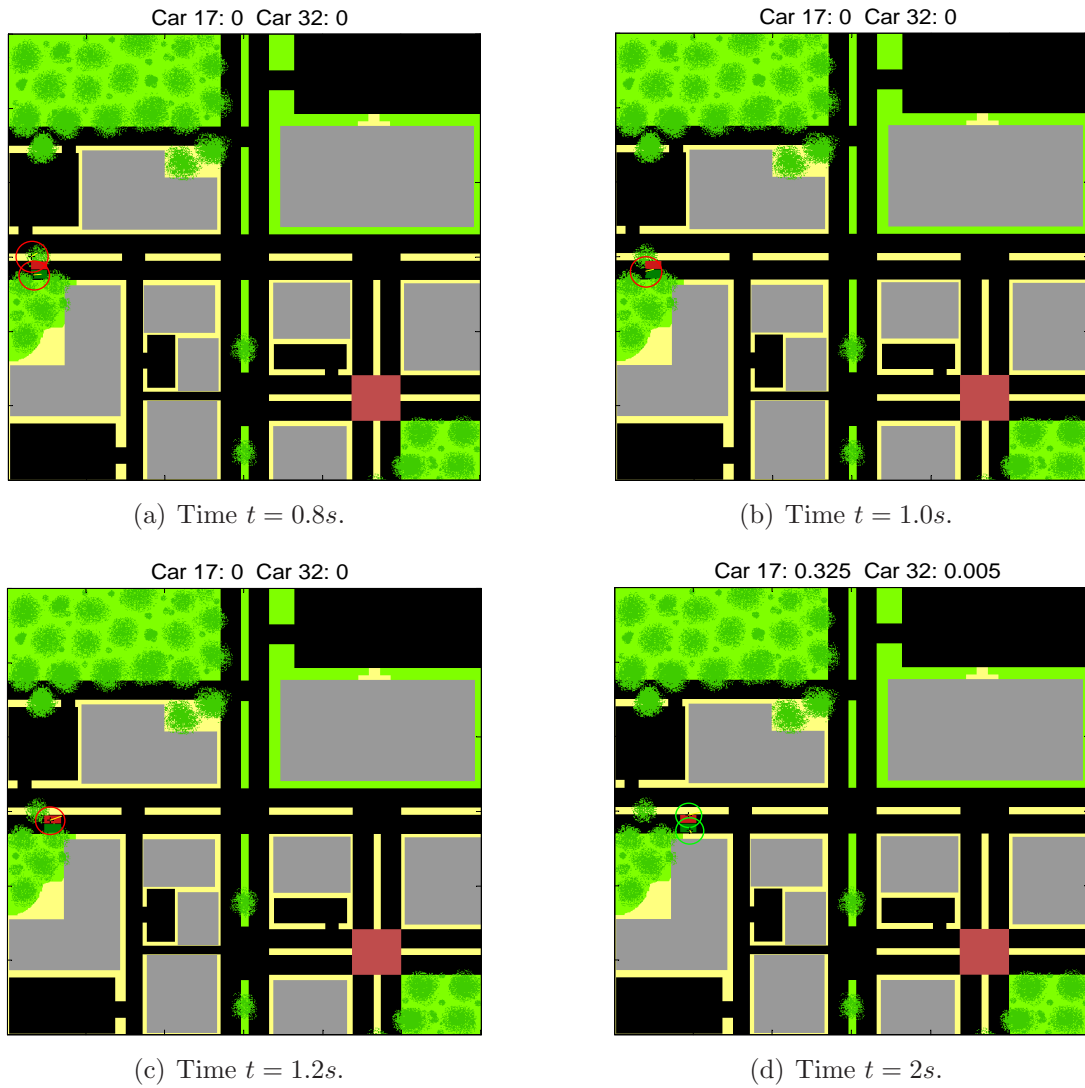


Figure 4.9: **Hyperspectral-augmented tracking version of Fig. 4.4 for the self-organizing map (Part A).** The tracker initiates tracks for both vehicles with an unknown class ID (a). Prior to hyperspectral scanning, the tracker deletes the track for the class ID 32 (green) vehicle due to measurement dropouts (b). The track for the class ID 17 (red₁) vehicle, however, receives enough measurements to stay alive (c). The tracker scans both track regions and correctly identifies both tracks (d).

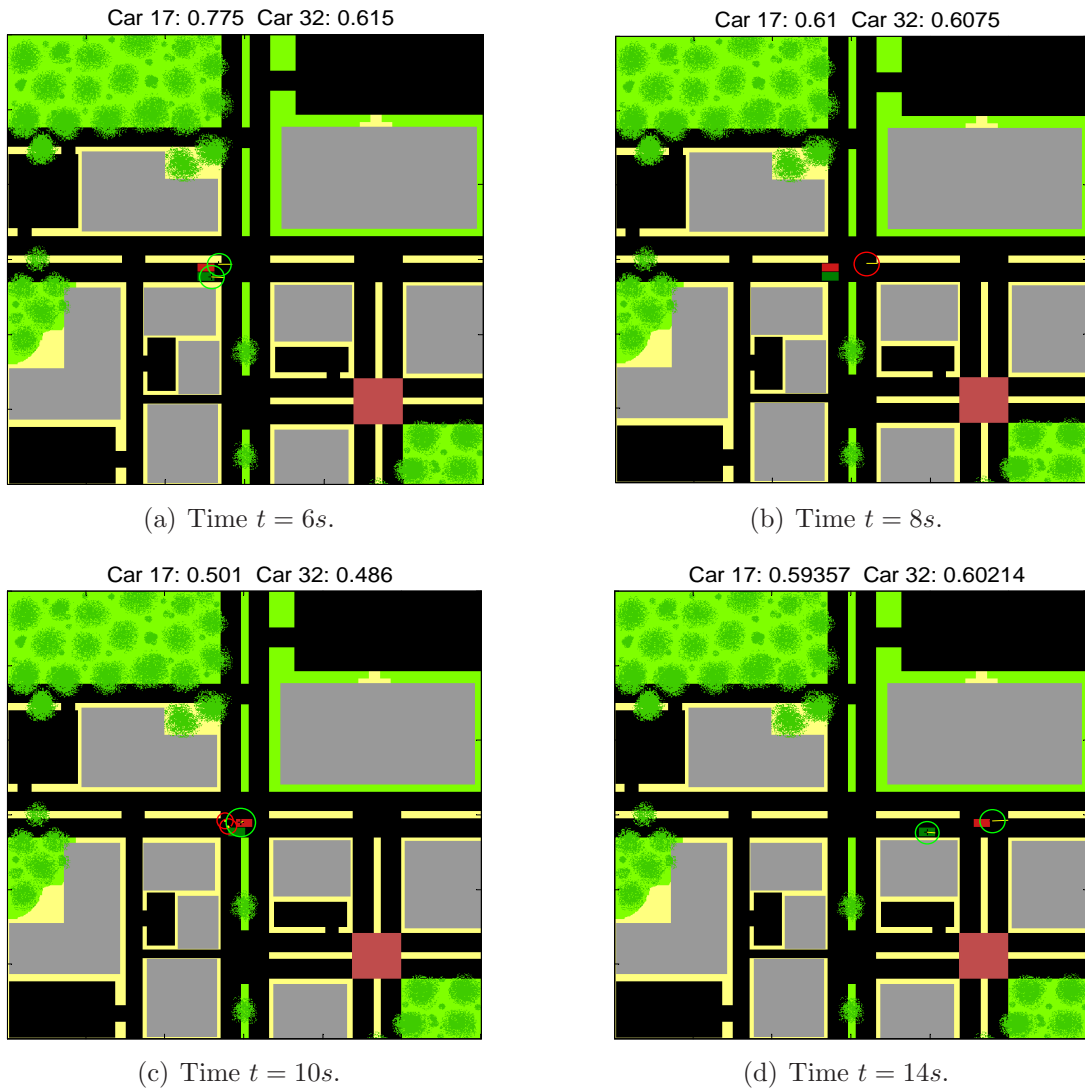


Figure 4.10: **Hyperspectral-augmented tracking version of Fig. 4.4 for the self-organizing map (Part B).** The tracker correctly identifies both tracks until they reach the intersection (a). At the intersection, both vehicles come to a complete stop, and their tracks continue to propagate. The tracker deletes the class ID 32 (green) track for the class ID 32 (green) vehicle first (b). Shortly after $t = 8s$, the tracker deletes class ID 17 (track). As the vehicles start moving again, the class ID 32 (green) vehicle is too slow for the tracker to detect its motion. The class ID 17 (red₁) vehicle speeds up first, and the tracker initiates and correctly identifies its track (c). The tracker correctly identifies both tracks throughout the rest of the simulation (d).

4.3.3 *Fuzzy C-Means Configuration 2.* Fig. 4.11 provides an FCM simulation run for configuration 2 of Sec. 4.2. At $t = 0.4s$, the tracker has already initiated tracks for both vehicles, but the hyperspectral sensor has not scanned both tracks (Fig. 4.11(a)). Sometime between $t = 0.4s$ and $t = 6s$, the sensor scans both tracks and the tracker assigns the hyperspectral observation with the correct class ID to each track. The tracker initiates a “track” using an unassociated hyperspectral observation with a vehicle class ID other than the two vehicles and their nearest neighbors (red circle in Fig. 4.11(b)). This is due to the classification results in the HSI chip, where the classifier incorrectly classifies several contiguous pixels.¹⁴ The hyperspectral sensor keeps scanning the track region and the classifier continues to misclassify the same pixel region. Hence, the false alarm persists throughout the rest of the simulation. As the vehicles come to a complete stop at the intersection, the tracks propagate and swap before track deletion¹⁵ (Fig. 4.11(c)). Since the tracks fail to receive updates, the tracker deletes them. Once the vehicle starts moving again, the tracker initiates tracks for both vehicles ((Fig. 4.11(d)). The hyperspectral sensor scans both vehicles and the tracker assigns the hyperspectral observation with the correct class ID to each track ((Fig. 4.11(e)). The tracks are correctly identified throughout the rest of the simulation ((Fig. 4.11(f)).

¹⁴The misclassification is due to the spectrum of certain mixtures of background. The spectrum is similar in a Euclidean sense to one of the vehicle classes.

¹⁵Note that alternative tracking logic can address the tracking difficulties in the move-stop-move maneuver, as described in Sec. 4.1.4.1.

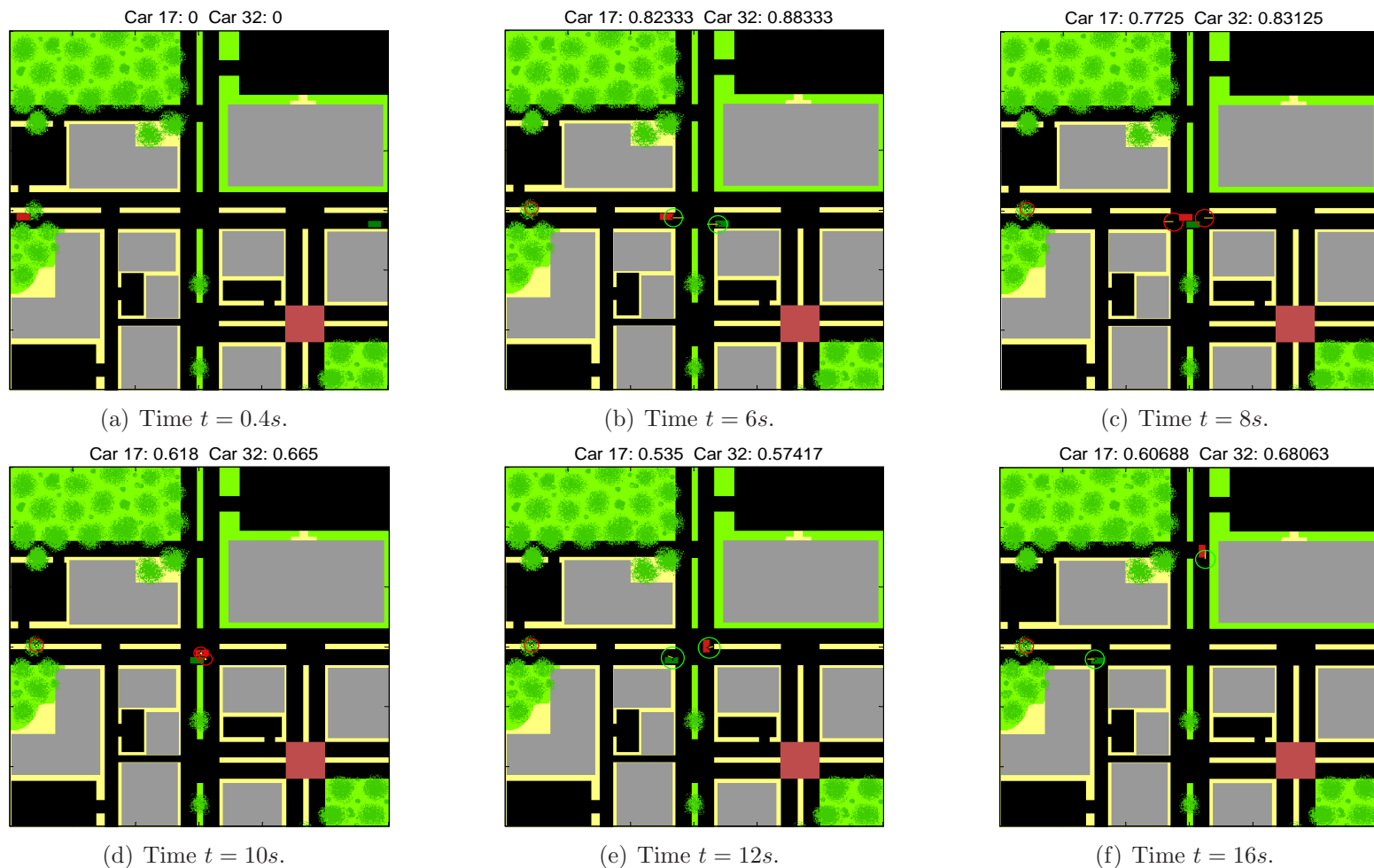


Figure 4.11: **Hyperspectral-augmented tracking version of Fig. 4.6 for fuzzy c-means.** The tracker initiates a track for each vehicle (a). After hyperspectral scanning, the tracker correctly identifies both tracks (b). At the intersection, the tracks propagate and swap before track deletion (c). After the vehicles start moving again, the tracker initiates tracks on both vehicles, the hyperspectral sensor scans both vehicles, and the tracker assigns the hyperspectral observation with the correct class ID to each track (e). The tracks are correctly identified throughout the rest of the simulation (f).

4.3.4 *Self-Organizing Map Configuration 2.* Fig.4.12 shows a simulation run of the SOM implementation for configuration 2 using SOM configuration 8 from Table 4.3. The parameters are [17,32], $\gamma = 0.5$, ROI line scanner, SOM lattice points, filtered map, and scenario 2. Figs. 4.12(a)-(c) are similar to Figs. 4.11(a)-(c). The main differences are: (1) Although the P_{ID} for the SOM increased at a slower rate, by $t = 8s$, it is higher than the P_{ID} for the FCM. (2) The SOM does not generate a false alarm, which implies that the SOM-based classifier handles background mixtures better than the FCM classifier. These differences are likely due to the effectiveness of the filtered SOM lattice, for which the samples that are highly influenced by background spectra were removed from the SOM. At $t = 10s$, the class ID 17 (red₁) track is still alive, while the tracker initiates an (unknown) track for the class ID 32 (green) vehicle (Fig. 4.12(d)). The hyperspectral sensor scans both track regions and the tracker assigns the hyperspectral observation with the correct class ID to each track (Fig. 4.12(e)). The tracks are correctly identified throughout the rest of the simulation (Fig. 4.12(f)).

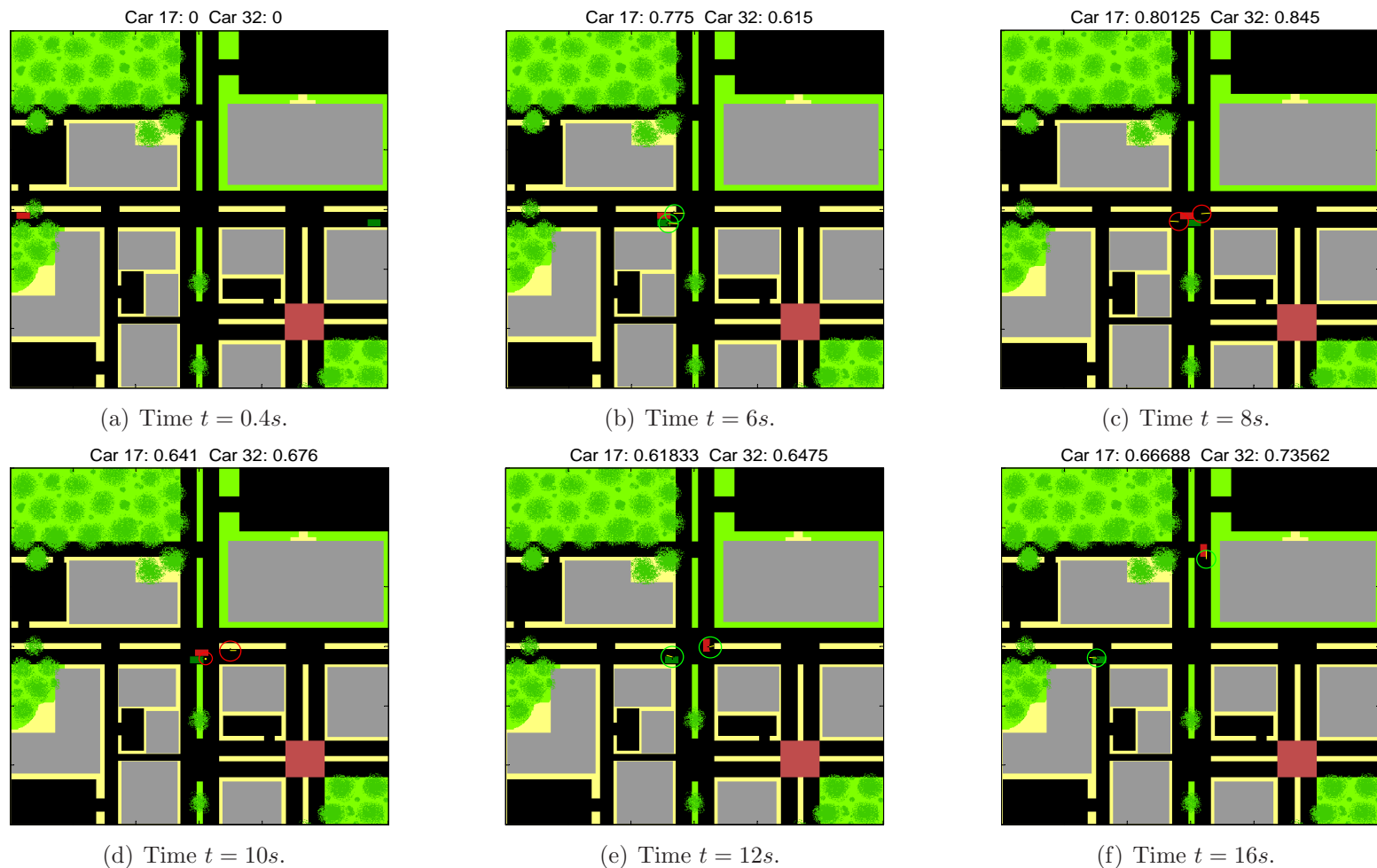


Figure 4.12: **Hyperspectral-augmented tracking version of Fig. 4.6 for the self-organizing map.** By $t = 0.4s$, the tracker has not performed track initiation for both vehicles (a). The tracker eventually performs track initiation and correctly identifies both tracks (b). At the intersection, the tracks continue to propagate prior to deletion (c). The tracker initiates a track for the green vehicle, while the track for the red vehicle is still alive (d). The tracker eventually deletes the track for the red vehicle, initiates a new track, and assigns the hyperspectral observation with the correct class ID to each track (e). The tracker correctly identifies both tracks throughout the rest of the simulation (f).

4.4 Quantitative Results

The quantitative results are based on the ensemble statistics using the performance measures described in Sec. 3.9. The tables and figures presented in this section summarize the results for the two classification methods of the hyperspectral-augmented tracker. The change in performance is represented by the percent $+/-$, which is calculated as follows:

$$\text{Percent } +/- = \frac{meas_{HSI} - meas_{kin}}{meas_{kin}} \times 100 \quad (4.1)$$

where $meas$ is the performance measure. The symbol ‘+’ and ‘-’ indicate a gain and a loss in performance, respectively, when tracking is augmented with hyperspectral data.

The experimental results indeed show that the hyperspectral-augmented tracking significantly outperformed the kinematic-only tracking, regardless of the classification algorithm used. Clearly, this novel work is very promising since it incorporates hyperspectral data in a feature-aided tracker, producing positive performance results. However, the results suggest that a tradeoff exists between performance and robustness.

In terms of performance, the hyperspectral-augmented tracking concept is not only feasible given the current technology, it is also very effective at increasing the probability of correct track identification (P_{ID}) in ambiguous situations. Furthermore, the overall P_{ID} is consistently higher for scenario 1, which further emphasizes the improved tracking performance of the hyperspectral-augmented tracker in highly ambiguous situations (typical in complex urban environments). In terms of robustness, however, the current implementation of the hyperspectral-augmented tracker has a potential weakness. When the kinematic-only tracker “resolves” swapped tracks (i.e., tracks perform a swap, then swap back in an ambiguous situation, which is simulated when track stitching does not maintain track continuity, as described in Sec. 4.2), it performed slightly better than the hyperspectral-augmented tracker. This is due to

two reasons: (1) In the kinematic-only tracking simulations, the tracker initiates a track using the nearest vehicle, so it is not penalized prior to track stitching (which occurs after track confirmation). (2) In the hyperspectral-augmented tracking, the hyperspectral sensor is cued *after* the tracker confirms a track, so the track is penalized from the time of track initiation to the time of chip formation. These reasons, which are based on the design choices described in Sec. 4.2, obscured the P_{ID} -non-swapped track results.

Table 4.4 provides the ensemble statistics for the kinematic-only configurations. The $P_{ID,j}$, $P_{assoc,j}$, $P_{declare,j|assoc,j}$, and $P_{ID|declare,j}$ measures are track-specific, where j is the track index. Each table consists of two rows for each measure: the first and second row refer to the first and second truth trajectory, respectively. For example, the truth trajectories for configuration 1 of the kinematic-only are vehicles [17,31]. The first and second row thus refer to vehicle 17 and 31, respectively. Furthermore, the second row of the table provides the configuration index. The indices for the different configurations are defined in Sec. 4.1.4.

Statistic	Kinematic-Only Configurations			
	1	2	3	4
$P_{ID,j}$ - Swapped Tracks	19.56% 17.73%	34.19% 34.16%	16.95% 17.81%	34.40% 34.49%
$P_{ID,j}$ - Non-swapped Tracks	59.81% 57.22%	68.43% 64.96%	60.35% 56.70%	69.02% 64.72%
$P_{ID,j}$	30.03% 28.39%	62.61% 59.72%	30.40% 31.81%	65.91% 62.00%
$P_{assoc,j}$	60.93% 57.15%	71.86% 68.10%	61.17% 56.90%	72.22% 68.02%
$P_{declare,j assoc,j}$	100.00% 100.00%	100.00% 100.00%	100.00% 100.00%	100.00% 100.00%
$P_{ID,j declare,j}$	49.41% 49.69%	87.22% 87.67%	49.86% 55.77%	91.24% 91.17%

Table 4.4: Ensemble statistics for the kinematic-only configurations.

4.4.1 *Fuzzy C-Means.* Table 4.5 summarizes the results of the FCM configurations. The table layout described previously for Table 4.4 also applies to Table 4.5, with the addition of the following:

- The P_D , P_{FA} , and EWA measures apply to hyperspectral-augmented tracking only; therefore, they are blank for the kinematic-only tracker.
- The term “swapped tracks” refers to simulation runs in which the tracks remain swapped after an ambiguous situation. Conversely, the term “non-swapped tracks” refers to simulations runs in which tracks do not remain swapped (or the kinematic-only tracker is able to “resolve” a track swap or loss event) after an ambiguous situation.

The ensemble statistics summarized in Table 4.5 consistently demonstrate that the P_{ID} of swapped tracks improved significantly when augmented by hyperspectral data. In Table 4.6, the average gain in performance is 124.8% P_{ID} for swapped tracks. Because of the reasons discussed previously, the hyperspectral-augmented tracker appears to perform slightly worse than the kinematic-only tracker for non-swapped tracks, with an average decrease in performance of 11.79% P_{ID} . The 30.55% increase in performance for the overall P_{ID} (which is based on all simulations) further supports the improved performance.

Performance for P_{ID} for swapped tracks is more significant with scenario 1 than scenario 2. This can be attributed to the “level” of ambiguity between the two scenarios. The vehicles in scenario 1 are closely spaced apart (from $t = 0s$ to $t = 15s$), where the likelihood for track swaps is much higher; whereas in scenario 2, the vehicles only experience the ambiguous situation at the intersection. Another performance measure that supports this observation is P_{assoc} , which scores the truth-to-track associations. The P_{assoc} for scenario 1 is smaller compared to scenario 2. The misassociations are clearly due to the high level of ambiguity.

In terms of classifier performance, the equal-weighted classification accuracy (EWA) shows that the FCM classifier performed rather effectively in evaluating the

simulated mixed spectra. A higher EWA is definitely preferred, and the $\approx 75\%$ average accuracy for all four configurations can be attributed to the complex mixtures in the scene that the FCM algorithm fails to address. This is supported by the low P_D and slightly higher P_{FA} .

The remaining track measures also provide interesting insights on the performance of the tracker. Since the kinematic-only tracker always initialized a track's class ID using the nearest vehicle, it is expected that $P_{declare|assoc}$ is 100%. Recall that this measure penalized a track when its class ID was not valid or "unknown." For the hyperspectral-augmented tracker, each track was assigned the unknown class ID $\approx 5\%$ of the frames. This is not surprising since initial hyperspectral scanning of a track region is cued by track confirmation. Depending on sensor dynamics, the hyperspectral sensor can take up to several seconds to steer its mirrors to the track region. In addition to the initial assignment of the unknown class ID, the $P_{ID|declare}$ measure also penalized the track when its class ID did not match with the class ID of the vehicle that it tracked. Generally speaking, for the highly ambiguous scenario 1, the hyperspectral-augmented tracker performed very well.

Statistic	FCM Configurations				Average
	1	2	3	4	
P_{ID,j^-} Swapped Tracks	50.01%	60.67%	44.53%	59.48%	55.18%
	54.20%	65.47%	44.95%	62.10%	
P_{ID,j^-} Non-swapped Tracks	48.66%	60.98%	46.68%	60.60%	55.40%
	54.73%	65.38%	44.39%	61.76%	
$P_{ID,j}$	49.66%	60.93%	45.20%	60.50%	55.32%
	54.35%	65.40%	44.75%	61.79%	
$P_{assoc,j}$	60.13%	71.65%	60.45%	71.42%	66.46%
	62.99%	71.21%	62.34%	71.46%	
$P_{declare,j assoc,j}$	95.18%	94.98%	95.33%	95.50%	95.48%
	95.55%	96.32%	94.97%	95.98%	
$P_{ID,j declare,j}$	86.70%	89.55%	78.52%	88.74%	86.85%
	90.31%	95.34%	75.53%	90.11%	
P_D	73.28%	69.20%	60.49%	55.22%	64.55%
P_{FA}	13.27%	12.21%	12.69%	16.57%	13.68%
EWA	74.27%	74.23%	79.35%	71.87%	74.93%

Table 4.5: Ensemble statistics for FCM configurations.

Statistic	Percent +/- for Configurations				Average
	1	2	3	4	
P_{ID,j^-} Swapped Tracks	155.62%	77.45%	162.71%	72.91%	124.80%
	205.66%	91.67%	152.34%	80.04%	
P_{ID,j^-} Non-swapped Tracks	-18.63%	-10.89%	-22.65%	-12.20%	-11.79%
	-4.35%	0.66%	-21.70%	-4.58%	
$P_{ID,j}$	65.38%	-2.69%	48.65%	-8.20%	30.55%
	91.40%	9.51%	40.67%	-0.34%	
$P_{assoc,j}$	-1.32%	-0.29%	-1.18%	-1.11%	3.19%
	10.22%	4.57%	9.56%	5.06%	
$P_{declare,j assoc,j}$	-4.82%	-5.02%	-4.67%	-4.50%	-4.52%
	-4.45%	-3.68%	-5.03%	-4.02%	
$P_{ID,j declare,j}$	75.45%	2.68%	57.46%	-2.74%	32.20%
	81.75%	8.75%	35.42%	-1.16%	

Table 4.6: Percent +/- from kinematic-only tracking baseline to FCM hyperspectral-augmented tracking.

4.4.2 *Self-Organizing Map.* Figs. 4.13 - 4.20 compare the results of the kinematic-only tracker and the SOM implementation. Clearly, the SOM results are more extensive because the SOM implementation consists of 48 configurations and nine additional measures, namely, $P_{ID,j}$ Average, P_D Average, P_{FA} Average, $P_{ID,j}$ Original, $P_{ID,j}$ Filtered, P_D Original, P_D Filtered, P_{FA} Original, and P_{FA} Filtered. These additional measures are based on the various SOM parameters described in Sec. 4.1.4.2. In addition to the descriptions for the table layout described above, the following provides additional information for the SOM tables:

- The additional nine measures apply to hyperspectral-augmented tracking only; therefore, they are blank for the kinematic-only tracker.
- The SOM configuration set index in the first row is defined in the configuration set list of Sec. 4.1.4.
- Several measures are blank because not all configurations compute these measures. For example, the $P_{ID,j}$ Original is only computed by configurations 1, 2, 5, and 6 of configuration set A. Configurations 3, 4, 7 and 8 do not (refer to Table 4.3 for the configuration settings and Sec. 4.1.4.2 for the configuration descriptions).

Figs. 4.13, 4.15, 4.17, and 4.19 provide the ensemble statistics for the classifier and tracking performance measures. Figs. 4.14, 4.16, 4.18, and 4.20 are the percent gain or loss (+/-) in performance for each performance measure between the kinematic-only tracker and the hyperspectral augmented tracker. Fig. 4.14 corresponds with Fig. 4.13, Fig. 4.16 corresponds with Fig. 4.15, and so on.

The ensemble statistics summarized in Table 4.7 demonstrate that the P_{ID} improved significantly when augmented by hyperspectral data. For example, the overall average performance gain is 28.11% P_{ID} , with an average of 121.48% P_{ID} for swapped tracks. Furthermore, the overall P_{ID} and the P_{ID} for swapped tracks show that for the highly ambiguous scenario 1, the tracker performed very well, as shown in Figs. 4.14 and 4.18. For comparison, the hyperspectral-augmented tracker

experienced a loss in performance, 13.82% P_{ID} , for the simulation runs in which the kinematic-only tracker was able to “resolve” track swaps. The reasons for this are explained previously in Sec. 4.4.

In terms of classifier performance, Table 4.9 shows that the filtered SOM outperformed the original SOM by 6.54% overall P_{ID} . Although the P_D measure shows a loss of performance for the filtered SOM, its P_{FA} is significantly lower than the original SOM. There is also a modest performance improvement in the EWA of the filtered SOM. Furthermore, Table 4.9 shows that the filtered SOM outperformed the FCM FCM classifier. Clearly, the filtered SOM approach is highly effective in dealing with mixed spectra. This is also supported by its significantly low P_{FA} (2.39% versus 13.47% for the original SOM and 13.68% for the FCM).

Additionally, the SOM implementation addressed several key parameters. The remainder of this section provides an analysis of the various parameter settings used in the SOM simulations.

1. The hyperspectral sensor has two sensor modes, ROI (configuration set A) and Pushbroom (configuration set B). With the pushbroom mode, the sensor took a longer time to initially scan the track region because it swept the entire scene. During this time, the P_{ID} of the track was penalized. As expected, the P_{ID} for configuration set B is lower than set A by a few percentage points (4-7%). Hence, ROI scanning generated better performance results.
2. The matching neuron determined by the SOM-based classifier can be modeled either as lattice points (for ROI scanning, configuration set A; for Pushbroom, configuration set B) or their corresponding weight vectors (for ROI scanning, configuration set C; for Pushbroom, configuration set D). Comparing set A with set C and set B with set D, the percent difference in the P_{ID} is statistically insignificant. Hence, the relationship among the weight vectors in the high-dimensional space is preserved in the two-dimensional space (as discussed in the SOM theory of Sec. 2.2.3.2). The consequence is that the two-dimensional

SOM lattice points can be used instead of the 195-dimensional weight vectors; thus, significantly reducing the number of computations.

3. Additional weighting γ values (0.99 and 0.01) are evaluated to vary the level of influence that the hyperspectral signature of a hyperspectral observation can have on the data association. Recall that the cost function for a hyperspectral observation is a sum of weighted kinematic and spectral distances. Configuration set A, E, and F provide the results for different values of $\gamma = \{0.99, 0.5, 0.01\}$ (see Table 4.3 for the configuration settings). The percent difference among the three γ values are statistically insignificant. In general, the weighting value does not matter, since the performance results are essentially the same. This is due to the fact that the kinematic distance and spectral distance of a hyperspectral observation are highly correlated. Since the spectral Mahalanobis distance is based on the assumption that the hyperspectral data is Gaussian, which is known to be false, this distance can be taken out of the cost function, and the cost function can be fully represented by its kinematic distance without any loss in tracking performance.

Statistic	SOM Configurations												Average
	A		B		C		D		E				
	1	3	9	11	17	19	25	27	33	35	37	39	
$P_{ID,j}$ - Swapped Tracks	42.76%	51.97%	45.79%	48.79%	44.38%	52.17%	45.47%	48.79%	43.51%	51.56%	43.54%	52.02%	48.3%
	44.37%	53.66%	46.81%	52.50%	42.14%	51.75%	46.94%	52.50%	44.13%	53.10%	46.62%	53.55%	
$P_{ID,j}$ - Non-swapped Tracks	41.60%	51.74%	45.93%	50.03%	38.96%	52.16%	45.93%	50.03%	40.28%	51.74%	44.50%	50.24%	47.9%
	44.40%	54.65%	44.31%	51.38%	43.21%	53.74%	43.64%	51.38%	44.66%	54.73%	45.78%	53.85%	
$P_{ID,j}$	42.46%	51.91%	45.83%	49.11%	42.97%	52.17%	45.59%	49.11%	42.67%	51.61%	43.79%	51.56%	48.2%
	44.38%	53.93%	46.14%	52.20%	42.43%	52.29%	46.05%	52.20%	44.27%	53.54%	46.39%	53.63%	
$P_{assoc,j}$	59.72%	60.10%	65.35%	65.45%	59.34%	59.96%	65.24%	65.45%	59.61%	59.94%	59.66%	60.10%	61.8%
	61.88%	62.12%	61.76%	62.33%	61.81%	62.29%	61.66%	62.33%	61.76%	61.90%	61.72%	62.20%	
$P_{declare,j assoc,j}$	93.13%	94.66%	88.15%	89.00%	93.08%	94.81%	87.81%	89.00%	93.14%	94.58%	93.16%	94.66%	92.2%
	93.40%	95.53%	87.60%	88.05%	93.28%	95.33%	87.60%	88.05%	93.35%	95.47%	93.52%	95.53%	
$P_{ID,j declare,j}$	76.46%	91.32%	79.56%	84.39%	77.82%	91.96%	79.46%	84.39%	76.90%	91.15%	79.07%	90.65%	84.6%
	76.56%	90.88%	85.28%	95.23%	73.58%	88.08%	85.24%	95.23%	76.53%	90.58%	79.99%	90.30%	
$P_{ID,j}$ Average	42.46%	51.91%	45.83%	49.11%	42.97%	52.17%	45.59%	49.11%	42.67%	51.61%	43.79%	51.56%	48.2%
	44.38%	53.93%	46.14%	52.20%	42.43%	52.29%	46.05%	52.20%	44.27%	53.54%	46.39%	53.63%	
$P_{ID,j}$ Original	42.46%		45.83%		42.97%		45.59%		42.67%		43.79%		44.4%
	44.38%		46.14%		42.43%		46.05%		44.27%		46.39%		
$P_{ID,j}$ Filtered		51.91%		49.11%		52.17%		49.11%		51.61%		51.56%	51.9%
		53.93%		52.20%		52.29%		52.20%		53.54%		53.63%	
P_D	84.56%	85.49%	82.15%	83.03%	84.81%	85.42%	82.19%	83.03%	84.60%	85.48%	84.61%	85.46%	84.2%
P_{FA}	14.40%	2.57%	11.30%	1.91%	14.43%	2.57%	11.41%	1.91%	14.16%	2.57%	14.44%	2.59%	7.9%
EWA	81.90%	92.05%	87.53%	92.58%	82.09%	91.91%	87.48%	92.58%	82.08%	92.06%	81.77%	92.06%	88.0%
P_D Average	84.56%	85.49%	82.15%	83.03%	84.81%	85.42%	82.19%	83.03%	84.60%	85.48%	84.61%	85.46%	84.2%
P_{FA} Average	14.40%	2.57%	11.30%	1.91%	14.43%	2.57%	11.41%	1.91%	14.16%	2.57%	14.44%	2.59%	7.9%
P_D Original	84.56%		82.15%		84.81%		82.19%		84.60%		84.61%		83.8%
P_D Filtered		85.49%		83.03%		85.42%		83.03%		85.48%		85.46%	84.7%
P_{FA} Original	14.40%		11.30%		14.43%		11.41%		14.16%		14.44%		13.4%
P_{FA} Filtered		2.57%		1.91%		2.57%		1.91%		2.57%		2.59%	2.4%

Figure 4.13: Ensemble statistics for SOM configuration set A, B, C, D, and E using scenario 1 and truth trajectories for class ID 17 and 31.

Statistic	Percent +/- for Configurations												Average
	A		B		C		D		E				
	1	3	9	11	17	19	25	27	33	35	37	39	
$P_{ID,j}$ - Swapped Tracks	118.57%	165.64%	134.05%	149.40%	126.87%	166.68%	132.44%	149.40%	122.41%	163.56%	122.59%	165.91%	159.75%
$P_{ID,j}$ - Non-swapped Tracks	-30.44%	-13.49%	-23.20%	-16.34%	-34.85%	-12.78%	-23.20%	-16.34%	-32.65%	-13.49%	-25.60%	-16.00%	-18.11%
$P_{ID,j}$	41.40%	72.88%	52.62%	63.56%	43.12%	73.74%	51.84%	63.56%	42.11%	71.87%	45.84%	71.70%	65.13%
$P_{assoc,j}$	-1.99%	-1.36%	7.24%	7.41%	-2.61%	-1.60%	7.08%	7.41%	-2.17%	-1.63%	-2.09%	-1.37%	4.83%
$P_{declare,j assoc,j}$	-6.87%	-5.34%	-11.85%	-11.00%	-6.92%	-5.19%	-12.19%	-11.00%	-6.86%	-5.42%	-6.84%	-5.34%	-7.84%
$P_{ID,j declare,j}$	54.73%	84.80%	61.00%	70.78%	57.49%	86.11%	60.81%	70.78%	55.62%	84.45%	60.02%	83.45%	70.74%
	54.08%	82.89%	71.63%	91.64%	48.07%	77.25%	71.55%	91.64%	54.01%	82.29%	60.97%	81.73%	

Figure 4.14: Percent +/- from kinematic-only configuration 1 to SOM hyperspectral-augmented tracking configuration sets A, B, C, D, and E using scenario 1 and truth trajectories for class ID 17 and 31.

Statistic	SOM Configurations												Average
	A		B		C		D		E				
	2	4	10	12	18	20	26	28	34	36	38	40	
$P_{ID,j}$ - Swapped Tracks	60.51%	58.68%	56.40%	55.87%	61.11%	60.31%	56.02%	57.20%	59.76%	57.41%	60.71%	57.34%	60.76%
	64.60%	65.72%	57.43%	61.62%	64.05%	65.36%	56.41%	61.56%	64.53%	65.12%	64.59%	65.84%	
$P_{ID,j}$ - Non-swapped Tracks	61.00%	59.30%	54.58%	54.98%	61.52%	59.60%	54.57%	56.06%	60.59%	58.41%	60.91%	59.36%	60.94%
	64.40%	65.57%	59.59%	62.23%	64.28%	65.72%	59.14%	61.99%	62.74%	65.40%	64.91%	65.67%	
$P_{ID,j}$	60.91%	59.19%	54.89%	55.13%	61.45%	59.72%	54.82%	56.26%	60.44%	58.24%	60.88%	59.02%	60.91%
	64.43%	65.59%	59.22%	62.12%	64.24%	65.66%	58.68%	61.91%	63.05%	65.35%	64.86%	65.70%	
$P_{assoc,j}$	72.05%	72.74%	72.12%	72.21%	72.04%	72.34%	71.65%	71.84%	72.08%	72.62%	72.14%	72.64%	71.94%
	71.74%	71.39%	71.99%	72.02%	71.18%	71.13%	71.95%	71.97%	71.87%	71.46%	71.88%	71.41%	
$P_{declare,j assoc,j}$	94.61%	92.41%	87.84%	86.52%	95.18%	93.06%	88.34%	86.94%	94.10%	92.12%	94.25%	92.41%	92.55%
	94.62%	96.26%	88.79%	90.88%	95.65%	96.82%	88.37%	90.60%	94.22%	95.84%	94.97%	96.32%	
$P_{ID,j declare,j}$	89.41%	88.13%	86.24%	88.25%	89.61%	88.80%	86.19%	90.14%	89.18%	87.08%	89.55%	87.92%	91.42%
	94.86%	95.44%	92.55%	94.92%	94.30%	95.32%	92.18%	94.97%	93.06%	95.42%	94.98%	95.52%	
$P_{ID,j}$ Average	60.91%	59.19%	54.89%	55.13%	61.45%	59.72%	54.82%	56.26%	60.44%	58.24%	60.88%	59.02%	60.91%
	64.43%	65.59%	59.22%	62.12%	64.24%	65.66%	58.68%	61.91%	63.05%	65.35%	64.86%	65.70%	
$P_{ID,j}$ Original	60.91%		54.89%		61.45%		54.82%		60.44%		60.88%		60.66%
	64.43%		59.22%		64.24%		58.68%		63.05%		64.86%		
$P_{ID,j}$ Filtered		59.19%		55.13%		59.72%		56.26%		58.24%		59.02%	61.16%
		65.59%		62.12%		65.66%		61.91%		65.35%		65.70%	
P_D	77.92%	79.48%	77.35%	78.92%	78.75%	80.22%	77.47%	78.91%	77.90%	79.57%	77.92%	79.53%	78.66%
P_{FA}	12.70%	2.27%	10.59%	1.71%	12.50%	2.26%	10.54%	1.74%	12.58%	2.27%	12.74%	2.28%	7.02%
EWA	81.74%	91.89%	86.08%	93.28%	81.90%	91.78%	86.14%	93.09%	81.75%	91.92%	81.72%	91.90%	87.77%
P_D Average	77.92%	79.48%	77.35%	78.92%	78.75%	80.22%	77.47%	78.91%	77.90%	79.57%	77.92%	79.53%	78.66%
P_{FA} Average	12.70%	2.27%	10.59%	1.71%	12.50%	2.26%	10.54%	1.74%	12.58%	2.27%	12.74%	2.28%	7.02%
P_D Original	77.92%		77.35%		78.75%		77.47%		77.90%		77.92%		77.89%
P_D Filtered		79.48%		78.92%		80.22%		78.91%		79.57%		79.53%	79.44%
P_{FA} Original	12.70%		10.59%		12.50%		10.54%		12.58%		12.74%		11.94%
P_{FA} Filtered		2.27%		1.71%		2.26%		1.74%		2.27%		2.28%	2.09%

Figure 4.15: Ensemble statistics for SOM configuration set A, B, C, D, and E using scenario 2 and truth trajectories for class ID 17 and 31.

Statistic	Percent +/- for Configurations												Average
	A		B		C		D		E				
	2	4	10	12	18	20	26	28	34	36	38	40	
$P_{ID,j}$ - Swapped Tracks	76.97% 89.12%	71.62% 92.40%	64.95% 68.13%	63.42% 80.38%	78.74% 87.50%	76.38% 91.34%	63.85% 65.13%	67.30% 80.22%	74.77% 88.91%	67.92% 90.65%	77.57% 89.10%	67.70% 92.76%	77.78%
$P_{ID,j}$ - Non-swapped Tracks	-10.86% -0.86%	-13.34% 0.94%	-20.24% -8.27%	-19.66% -4.20%	-10.09% -1.04%	-12.91% 1.17%	-20.25% -8.95%	-18.07% -4.57%	-11.46% -3.41%	-14.64% 0.68%	-10.99% -0.07%	-13.25% 1.10%	-8.47%
$P_{ID,j}$	-2.71% 7.89%	-5.45% 9.83%	-12.33% -0.84%	-11.94% 4.02%	-1.85% 7.57%	-4.62% 9.94%	-12.45% -1.75%	-10.15% 3.67%	-3.46% 5.57%	-6.98% 9.43%	-2.77% 8.60%	-5.74% 10.01%	-0.27%
$P_{assoc,j}$	0.27% 5.35%	1.23% 4.83%	0.36% 5.71%	0.49% 5.76%	0.25% 4.52%	0.67% 4.45%	-0.29% 5.64%	-0.02% 5.67%	0.30% 5.53%	1.06% 4.93%	0.39% 5.55%	1.09% 4.86%	2.86%
$P_{declare,j assoc,j}$	-5.39% -5.38%	-7.59% -3.74%	-12.16% -11.21%	-13.48% -9.12%	-4.82% -4.35%	-6.94% -3.18%	-11.66% -11.63%	-13.06% -9.40%	-5.90% -5.78%	-7.88% -4.16%	-5.75% -5.03%	-7.59% -3.68%	-7.45%
$P_{ID,j declare,j}$	2.52% 8.20%	1.05% 8.87%	-1.12% 5.57%	1.18% 8.27%	2.74% 7.57%	1.81% 8.73%	-1.18% 5.15%	3.35% 8.33%	2.24% 6.15%	-0.16% 8.85%	2.67% 8.34%	0.80% 8.95%	4.54%

Figure 4.16: Percent +/- from kinematic-only configuration 2 to SOM hyperspectral-augmented tracking configuration sets A, B, C, D, and E using scenario 2 and truth trajectories for class ID 17 and 31.

Statistic	SOM Configurations												Average
	A		B		C		D		F				
	5	7	13	15	21	23	29	31	41	43	45	47	
$P_{ID,j}$ - Swapped Tracks	47.21%	52.47%	44.73%	46.48%	47.27%	52.47%	44.68%	46.48%	46.62%	53.21%	45.64%	51.80%	47.39%
	45.49%	49.44%	44.34%	46.64%	45.36%	49.44%	44.25%	46.64%	45.03%	50.08%	42.97%	48.60%	
$P_{ID,j}$ - Non-swapped Tracks	43.25%	47.31%	43.12%	46.04%	43.92%	47.31%	43.43%	46.04%	43.53%	47.70%	44.96%	47.30%	46.16%
	41.64%	50.10%	49.55%	48.21%	42.12%	50.10%	49.55%	48.21%	41.28%	50.45%	42.84%	49.86%	
$P_{ID,j}$	45.99%	50.87%	44.23%	46.34%	46.23%	50.87%	44.29%	46.34%	45.67%	51.50%	45.43%	50.41%	47.02%
	44.10%	49.68%	46.22%	47.20%	44.20%	49.68%	46.16%	47.20%	43.68%	50.21%	42.92%	49.05%	
$P_{assoc,j}$	59.24%	59.09%	67.00%	62.97%	59.21%	59.09%	66.64%	62.97%	59.04%	59.74%	59.45%	58.94%	60.69%
	60.85%	59.90%	59.91%	60.53%	60.85%	59.90%	59.88%	60.53%	60.38%	60.36%	60.21%	59.84%	
$P_{declare,j assoc,j}$	94.39%	93.12%	89.37%	85.69%	94.50%	93.12%	89.29%	85.69%	94.48%	92.89%	94.76%	93.17%	91.34%
	94.93%	91.52%	87.88%	85.20%	94.80%	91.52%	87.89%	85.20%	94.84%	91.57%	94.67%	91.60%	
$P_{ID,j declare,j}$	81.90%	92.43%	74.09%	85.87%	82.35%	92.43%	74.68%	85.87%	81.57%	92.84%	80.70%	91.71%	85.04%
	76.60%	90.66%	87.59%	91.28%	76.85%	90.66%	87.52%	91.28%	76.39%	90.90%	75.30%	89.61%	
$P_{ID,j}$ Average	45.99%	50.87%	44.23%	46.34%	46.23%	50.87%	44.29%	46.34%	45.67%	51.50%	45.43%	50.41%	47.02%
	44.10%	49.68%	46.22%	47.20%	44.20%	49.68%	46.16%	47.20%	43.68%	50.21%	42.92%	49.05%	
$P_{ID,j}$ Original	45.99%		44.23%		46.23%		44.29%		45.67%		45.43%		44.93%
	44.10%		46.22%		44.20%		46.16%		43.68%		42.92%		
$P_{ID,j}$ Filtered		50.87%		46.34%		50.87%		46.34%		51.50%		50.41%	49.11%
		49.68%		47.20%		49.68%		47.20%		50.21%		49.05%	
P_D	72.10%	48.22%	71.41%	50.34%	72.11%	48.22%	71.42%	50.34%	72.09%	47.90%	72.00%	48.31%	60.37%
P_{FA}	14.44%	3.01%	10.77%	2.15%	14.48%	3.01%	11.10%	2.15%	14.28%	2.89%	14.51%	3.06%	7.99%
EWA	81.33%	81.12%	86.63%	87.27%	81.50%	81.12%	86.60%	87.27%	81.90%	81.23%	81.53%	80.97%	83.21%
P_D Average	72.10%	48.22%	71.41%	50.34%	72.11%	48.22%	71.42%	50.34%	72.09%	47.90%	72.00%	48.31%	60.37%
P_{FA} Average	14.44%	3.01%	10.77%	2.15%	14.48%	3.01%	11.10%	2.15%	14.28%	2.89%	14.51%	3.06%	7.99%
P_D Original	72.10%		71.41%		72.11%		71.42%		72.09%		72.00%		71.86%
P_D Filtered		48.22%		50.34%		48.22%		50.34%		47.90%		48.31%	48.89%
P_{FA} Original	14.44%		10.77%		14.48%		11.10%		14.28%		14.51%		13.26%
P_{FA} Filtered		3.01%		2.15%		3.01%		2.15%		2.89%		3.06%	2.71%

Figure 4.17: Ensemble statistics for SOM configuration set A, B, C, D, and E using scenario 1 and truth trajectories for class ID 17 and 32.

Statistic	Percent +/- for Configurations												Average
	A		B		C		D		E				
	5	7	13	15	21	23	29	31	41	43	45	47	
$P_{ID,j}$ - Swapped Tracks	178.55%	209.57%	163.88%	174.20%	178.89%	209.57%	163.60%	174.20%	175.07%	213.91%	169.26%	205.63%	172.94%
	155.38%	177.56%	148.95%	161.83%	154.68%	177.56%	148.41%	161.83%	152.82%	181.15%	141.21%	172.85%	
$P_{ID,j}$ - Non-swapped Tracks	-28.33%	-21.61%	-28.55%	-23.71%	-27.23%	-21.61%	-28.04%	-23.71%	-27.87%	-20.96%	-25.51%	-21.63%	-21.01%
	-26.56%	-11.64%	-12.60%	-14.98%	-25.72%	-11.64%	-12.60%	-14.98%	-27.18%	-11.02%	-24.43%	-12.06%	
$P_{ID,j}$	51.25%	67.32%	45.47%	52.42%	52.06%	67.32%	45.68%	52.42%	50.19%	69.39%	49.41%	65.79%	51.25%
	38.64%	56.16%	45.29%	48.38%	38.93%	56.16%	45.10%	48.38%	37.32%	57.85%	34.93%	54.20%	
$P_{assoc,j}$	-3.16%	-3.41%	9.53%	2.93%	-3.21%	-3.41%	8.93%	2.93%	-3.49%	-2.35%	-2.82%	-3.66%	2.90%
	6.93%	5.27%	5.28%	6.37%	6.93%	5.27%	5.22%	6.37%	6.11%	6.07%	5.80%	5.15%	
$P_{declare,j assoc,j}$	-5.61%	-6.88%	-10.63%	-14.31%	-5.50%	-6.88%	-10.71%	-14.31%	-5.52%	-7.11%	-5.24%	-6.83%	-8.66%
	-5.07%	-8.48%	-12.12%	-14.80%	-5.20%	-8.48%	-12.11%	-14.80%	-5.16%	-8.43%	-5.33%	-8.40%	
$P_{ID,j declare,j}$	64.24%	85.36%	48.58%	72.21%	65.15%	85.36%	49.77%	72.21%	63.58%	86.19%	61.83%	83.92%	61.48%
	37.34%	62.56%	57.05%	63.66%	37.79%	62.56%	56.92%	63.66%	36.96%	62.98%	35.01%	60.67%	

Figure 4.18: Percent +/- from kinematic-only configuration 3 to SOM hyperspectral-augmented tracking configuration sets A, B, C, D, and E using scenario 1 and truth trajectories for class ID 17 and 32.

Statistic	SOM Configurations												Average
	A		B		C		D		F				
	6	8	14	16	22	24	30	32	42	44	46	48	
$P_{ID,j}$ - Swapped Tracks	59.36%	61.32%	54.22%	57.16%	59.36%	61.32%	54.22%	57.16%	59.34%	57.05%	60.14%	60.40%	60.44%
	61.42%	65.76%	59.09%	61.58%	61.42%	65.76%	59.09%	61.58%	61.45%	65.18%	61.65%	65.49%	
$P_{ID,j}$ - Non-swapped Tracks	61.25%	59.22%	57.99%	54.64%	61.25%	59.22%	57.99%	54.64%	60.74%	59.56%	61.30%	58.98%	61.57%
	65.20%	65.63%	61.57%	62.47%	65.20%	65.63%	61.57%	62.47%	64.88%	65.61%	65.00%	65.68%	
$P_{ID,j}$	61.08%	59.41%	57.65%	54.87%	61.08%	59.41%	57.65%	54.87%	60.61%	59.33%	61.19%	59.11%	61.47%
	64.86%	65.65%	61.35%	62.39%	64.86%	65.65%	61.35%	62.39%	64.57%	65.57%	64.70%	65.66%	
$P_{assoc,j}$	71.83%	72.02%	72.17%	72.27%	71.83%	72.02%	72.17%	72.27%	71.98%	71.81%	71.80%	72.08%	71.77%
	71.38%	71.19%	71.88%	71.89%	71.38%	71.19%	71.88%	71.89%	71.54%	71.38%	71.34%	71.23%	
$P_{declare,j assoc,j}$	94.95%	93.06%	89.33%	88.18%	94.95%	93.06%	89.33%	88.18%	94.25%	92.64%	95.18%	93.09%	93.24%
	95.74%	96.75%	89.90%	91.37%	96.75%	89.90%	91.37%	94.92%	96.58%	96.58%	95.64%	96.77%	
$P_{ID,j declare,j}$	89.54%	88.67%	89.07%	86.07%	89.54%	88.67%	89.07%	86.07%	89.31%	89.21%	89.53%	88.14%	91.80%
	94.90%	95.31%	94.91%	94.97%	94.90%	95.31%	94.91%	94.97%	95.02%	95.11%	94.81%	95.26%	
$P_{ID,j}$ Average	61.08%	59.41%	57.65%	54.87%	61.08%	59.41%	57.65%	54.87%	60.61%	59.33%	61.19%	59.11%	61.47%
	64.86%	65.65%	61.35%	62.39%	64.86%	65.65%	61.35%	62.39%	64.57%	65.57%	64.70%	65.66%	
$P_{ID,j}$ Original	61.08%		57.65%		61.08%		57.65%		60.61%		61.19%		61.75%
	64.86%		61.35%		64.86%		61.35%		64.57%		64.70%		
$P_{ID,j}$ Filtered		59.41%		54.87%		59.41%		54.87%		59.33%		59.11%	61.19%
		65.65%		62.39%		65.65%		62.39%		65.57%		65.66%	
P_D	68.80%	40.76%	68.57%	39.19%	68.80%	40.76%	68.57%	39.19%	68.79%	40.69%	68.74%	40.65%	54.46%
P_{FA}	12.32%	1.70%	10.19%	1.16%	12.32%	1.70%	10.19%	1.16%	12.28%	1.68%	12.32%	1.68%	6.56%
EWA	80.64%	81.92%	84.76%	85.48%	80.64%	81.92%	84.76%	85.48%	80.80%	82.10%	80.61%	81.87%	82.58%
P_D Average	68.80%	40.76%	68.57%	39.19%	68.80%	40.76%	68.57%	39.19%	68.79%	40.69%	68.74%	40.65%	54.46%
P_{FA} Average	12.32%	1.70%	10.19%	1.16%	12.32%	1.70%	10.19%	1.16%	12.28%	1.68%	12.32%	1.68%	6.56%
P_D Original	68.80%		68.57%		68.80%		68.57%		68.79%		68.74%		68.71%
P_D Filtered		40.76%		39.19%		40.76%		39.19%		40.69%		40.65%	40.21%
P_{FA} Original	12.32%		10.19%		12.32%		10.19%		12.28%		12.32%		11.61%
P_{FA} Filtered		1.70%		1.16%		1.70%		1.16%		1.68%		1.68%	1.51%

Figure 4.19: Ensemble statistics for SOM configuration set A, B, C, D, and E using scenario 2 and truth trajectories for class ID 17 and 32.

Statistic	Percent +/- for Configurations												Average
	A		B		C		D		F				
	6	8	14	16	22	24	30	32	42	44	46	48	
$P_{ID,j}$ - Swapped Tracks	72.56%	78.26%	57.63%	66.16%	72.56%	78.26%	57.63%	66.16%	72.50%	65.83%	74.83%	75.58%	75.44%
	78.05%	90.63%	71.30%	78.51%	78.05%	90.63%	71.30%	78.51%	78.13%	88.95%	78.73%	89.85%	
$P_{ID,j}$ - Non-swapped Tracks	-11.26%	-14.20%	-15.99%	-20.83%	-11.26%	-14.20%	-15.99%	-20.83%	-12.00%	-13.71%	-11.19%	-14.55%	-7.70%
	0.74%	1.41%	-4.87%	-3.48%	0.74%	1.41%	-4.87%	-3.48%	0.25%	1.37%	0.44%	1.48%	
$P_{ID,j}$	-7.33%	-9.86%	-12.53%	-16.74%	-7.33%	-9.86%	-12.53%	-16.74%	-8.03%	-9.97%	-7.15%	-10.31%	-3.67%
	4.61%	5.88%	-1.05%	0.63%	4.61%	5.88%	-1.05%	0.63%	4.15%	5.76%	4.36%	5.91%	
$P_{assoc,j}$	-0.54%	-0.29%	-0.08%	0.06%	-0.54%	-0.29%	-0.08%	0.06%	-0.34%	-0.57%	-0.58%	-0.20%	2.43%
	4.93%	4.66%	5.68%	5.69%	4.93%	4.66%	5.68%	5.69%	5.18%	4.94%	4.88%	4.72%	
$P_{declare,j assoc,j}$	-5.05%	-6.94%	-10.67%	-11.82%	-5.05%	-6.94%	-10.67%	-11.82%	-5.75%	-7.36%	-4.82%	-6.91%	-6.76%
	-4.26%	-3.25%	-10.10%	-8.63%	-4.26%	-3.25%	-10.10%	-8.63%	-5.08%	-3.42%	-4.36%	-3.23%	
$P_{ID,j declare,j}$	-1.87%	-2.82%	-2.38%	-5.67%	-1.87%	-2.82%	-2.38%	-5.67%	-2.12%	-2.23%	-1.87%	-3.40%	0.66%
	4.09%	4.54%	4.10%	4.17%	4.09%	4.54%	4.10%	4.17%	4.23%	4.32%	3.99%	4.49%	

Figure 4.20: Percent +/- from kinematic-only configuration 4 to SOM hyperspectral-augmented tracking configuration sets A, B, C, D, and E using scenario 2 and truth trajectories for class ID 17 and 32.

Statistic	Average		
	Kinematic-Only	SOM	Percent +/-
P_{ID} - Swapped Tracks	26.16%	54.22%	121.48%
P_{ID} - Non-swapped Tracks	62.65%	54.13%	-13.82%
P_{ID}	46.36%	54.39%	28.11%
P_{assoc}	64.55%	66.55%	3.25%
$P_{declare assoc}$	100.00%	92.32%	-7.68%
$P_{ID declare}$	70.25%	88.22%	34.35%

Table 4.7: Ensemble statistics and percent +/- for all kinematic-only and SOM configurations.

Statistic	Average		
	Original	Filtered	Percent +/-
P_{ID}	53.53%	57.03%	6.54%
P_D	75.84%	63.49%	-16.29%
P_{FA}	13.47%	2.39%	-82.26%
EWA	81.40%	86.75%	6.57%

Table 4.8: Ensemble statistics and percent +/- for original SOM and filtered SOM configurations.

Statistic	Average		
	Original SOM	FCM	Filtered SOM
P_{ID}	53.53%	55.32%	57.03%
P_D	75.84%	64.55%	63.49%
P_{FA}	13.47%	13.68%	2.39%

Table 4.9: Ensemble statistics for original SOM, FCM, and filtered SOM configurations.

4.5 Summary

This chapter provides not only the experimental design used in this research, but also the quantitative results of the simulations. The performance measures of Sec. 3.9 are calculated in order to compare the performance between kinematic-only and hyperspectral-augmented tracking. As the quantitative results show, hyperspectral-augmented tracking outperforms kinematic-only tracking with a resounding success. The next chapter provides the final analysis of the quantitative results and addresses the shortcomings and assumptions in the methodology and experimental design. Furthermore, based on research goals achieved, it provides a future perspective on this research work and outline recommendations for future research.

V. Conclusions

This chapter discusses the research conclusions and major result trends. It answers the question: *When augmented by hyperspectral data, is the performance of the kinematic-only tracker in ambiguous situations improved?* with a resounding yes. As discussed in Sec. 4.4, the hyperspectral-augmented tracker outperformed the kinematic-only tracker with an overall average of 123.14% P_{ID} for swapped tracks. Within the bounds of the methodology and experimental design, the results show great promise, and as long as the effects that were ignored in the experiments (e.g., parallax, false alarm, and image registration errors) are addressed and accounted for, the methods developed in this research can be extended to real data.

The hyperspectral-augmented tracking system presented a novel approach to feature-aided single hypothesis tracking. Spectral gating work developed a novel method for calculating the nearest neighbors of a target class. The observation-to-track association gates a hyperspectral observation using the nearest neighbors of the track's class ID, thus reducing computational complexity. It is common for an HSI chip to have more than ten hyperspectral observations, and if three target tracks exist, the number of operations amounts to 1,000 (as compared to 27 operations for three observations and three tracks).

The hyperspectral observation-to-track association offers an innovative method for representing the cost function. Instead of using the conventional approach used in a typical multi-target tracking system (i.e., the kinematic Mahalanobis distance), this research uses a sum of weighted kinematic and spectral distances. Since the two distances are highly correlated, results show that the weighting factor γ does not make a difference on the outcome of the observation-to-track association. This can be attributed to the spectral gating that occurs prior to the assignment process (i.e., the spectral signature of the observations are spectrally similar to the spectral signature of the track).

The concepts applied in this research are very effective. As compared to the fuzzy c-means (FCM), the use of the self-organizing map (SOM) as a classification ap-

proach is an excellent choice. It has a reduced computational cost, allows for real-time processing of hyperspectral data, is effective at organizing noisy data, and provides convenient processing (i.e., morphological operations) of high-dimensional data in a two-dimensional space. Based on the results, the SOM-based classifier is highly effective in dealing with mixed spectra, specifically the *filtered* map configuration. For each vehicle class, the samples at the tail of the distribution are assumed highly influenced by background spectra. The filtering method removes these samples from the distribution, reducing the misclassification errors.

Analysis of the two hyperspectral sensor modes (Pushbroom versus Region-of-Interest scanning) can drive future design of hyperspectral sensors. An innovative approach is the ROI hyperspectral line scanning, which uses of the track's kinematic information and process statistics to steer the mirror to locations of existing tracks. This implementation increases the track's revisit rates. Results show that the ROI scanner type outperforms the typical Pushbroom mode by an average of 6% P_{ID} .

5.1 Future Work

This thesis effort has shown the feasibility of the hyperspectral-augmented tracking system. The potential for this capability definitely exists, and more work can be performed to refine and improve the methodology and experimental design used in this research.

Future efforts include the addition of more sophisticated track management techniques, such as a multiple hypothesis tracker (MHT). Since the observations used by the tracker in the simulations are devoid of clutter or false alarms, a single hypothesis tracker (SHT) suffices. However, these issues are unavoidable when working with real data. An MHT is more robust since it forms hypotheses of the potential outcomes and defers a decision until after subsequent data allow it to resolve the uncertainty. Hence, an MHT can better deal with spurious observations than an SHT. Spectral gating is another area where MHT is more advantageous. For example, if one vehicle becomes occluded (e.g., under a bridge) and a second vehicle comes out

of the occlusion (heading in the same direction as the first vehicle), the tracker will likely “latch” onto the second vehicle. Assuming the hyperspectral signature of the second vehicle is outside of the gate of the existing track, the associator ignores the hyperspectral observation generated by the second vehicle. Hence, the hyperspectral-augmented tracker does not update the class ID of the track. If MHT is used in this scenario, it will create another hypothesis for the second vehicle and defer the decision on which class ID to update the track until it receives more observations. Another example deals with the SOM. If two or more classes that are mapped to the best matching neuron equally have the highest density, an MHT can be implemented so that the decision on which class to assign to the unclassified pixel can be deferred until additional observations are received and the uncertainty can be resolved. The MHT method forms a hypothesis for each potential class and propagates each hypothesis until subsequent data satisfy a decision logic.

Other classification approaches can also be investigated. Effective methods for decision boundary approximation include those based on Kohonen’s Learning Vector Quantization 2.1 and variants (e.g., Generalized LVQ [34], Generalized Relevance LVQ (GRLVQ) [17], and GRLVQ-Improved [28]). The advantage of these methods is that they learn decision boundaries between the current class and its nearest neighbors in distribution, potentially allowing for a better classification. Note that it might be best to use the filtered SOM as a preprocessing step in order to provide good quality samples for classification. Otherwise, the extreme target and background mixtures will degrade the classification.

The framework for track maintenance of static targets already exists in this research. Track maintenance uses hyperspectral detections to initiate or delete “tracks.” Doing so, targets do not have to be in motion to be monitored. Current implementation only allows for track maintenance on potential targets that happen to be in the scan region of the hyperspectral sensor. This can be modified so that the sensor not only scans existing track regions, but also takes advantage of contextual aspects of

the scene (e.g., parking lots and other roadways) in order to identify and track static targets.

The approach used in calculating the spectral Mahalanobis distance for the hyperspectral observation-to-track association assumes a Gaussian distribution among the image bands of the hyperspectral data. This is a limitation, since the image bands do not actually have a Gaussian distribution. Additional work is needed to better determine an accurate representation of this distance. Based on the results, another option is to simply remove this distance since the kinematic and spectral distances are highly correlated, the kinematic distance is sufficient and can be used as the cost function for the hyperspectral observation.

Unmixing algorithms that extract the constituent spectra comprising a pixel can also be investigated. For example, a linear mixing model provides an estimation of the abundances of materials present in the pixel region. This can be incorporated to better identify the targets of interest in the imaged scene.

5.2 Concluding Remarks

As the nature of military warfare continues to evolve, the United States Air Force is at the leading edge of providing American forces on the ground with vital, timely, and accurate information in order to accomplish their missions. Persistent tracking of enemy forces is a critical intelligence, surveillance, and reconnaissance need in this ever-changing battlefield—urban environments that have no clearly defined demarcations, where the enemy fights with no rules of engagement. This research offers a means for persistent vehicle recognition and monitoring in such a battlefield through the employment of a highly effective feature-based tracking paradigm based on hyperspectral imagery.

Bibliography

1. “United States Air Force Basic Doctrine, Air Force Doctrine Document 1”. November 2003.
2. Arambel, Pablo O., Jeff Silver, Jon Krant, Matthew Antone, and Thomas Strat. “Multiple-Hypothesis Tracking of Multiple Ground Targets from Aerial Video with Dynamic Sensor Control”. *The International Society for Optical Engineering*, 5429:23–32, August 2004.
3. Bar-Shalom, Yaakov and Xiao-Rong Li. *Multitarget-Multisensor Tracking: Principles and Techniques*. Yaakov Bar-Shalom, Storrs, Connecticut, 1995.
4. Bertsekas, Dimitri P. “The Auction Algorithm: A Distributed Relaxation Method for the Assignment Problem”. *Annals of Operations Research*, 14(1):105–123, December 1988.
5. Bevilacqua, Alessandro, Luigi Di Stefano, and Stefano Vaccari. “Occlusion Robust Vehicle Tracking based on SOM (Self-Organizing Map)”. *IEEE Workshop on Motion and Video Computing*, volume 2, 84–89. January 2005.
6. Blackburn, Joshua, Michael Mendenhall, Andrew Rice, Paul Shelnut, Neil Soliman, and Juan Vasquez. “Feature Aided Tracking with Hyperspectral Imagery”. Oliver E. Drummond and Richard D. Teichgraber (editors), *Signal and Data Processing of Small Targets*. September 2007.
7. Blackman, Samuel. *Multiple Target Tracking with Radar Applications*. Artech House, 685 Canton St. Norwood, MA 02062, 1986.
8. Blackman, Samuel and Robert Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House, 685 Canton St. Norwood, MA 02062, 1999.
9. Buchan, Glen. *Nuclear Weapons and U.S. National Security Strategy for a New Century*, chapter 7, 225–282. Khalilzad and Shapiro [21], 2003.
10. Campbell, James B. *Introduction to Remote Sensing*. The Guilford Press, fourth edition, 2007.
11. Clark, R. N., G. A. Swayze, R. Wise, K. E. Livo, T. M. Hoefen, R. F. Kokaly, and S. J. Sutley. “USGS Digital Spectral Library”. http://speclab.cr.usgs.gov/spectral_lib.html, 2003.
12. Demuth, Howard, Mark Beale, and Martin Hagan. *Neural Network Toolbox 5 User’s Guide*. The MathWorks, Inc., 2007.
13. Duda, Richard O., Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley and Sons, Inc., second edition, 2001.
14. “ENV transfer Angel Fire Project to AFRL”. *The Envoy*, 1(1):5, Fall 2006.

15. Green, R. O. “Summaries of the 6th Annual JPL Airborne Geoscience Workshop, 1. AVIRIS Workshop”. Pasadena, CA, Mar 4–6 1996.
16. Guo, Yanlin, Steve Hsu, Ying Shan, Harpreet Sawhney, and Rakesh Kumar. “Vehicle Fingerprinting for Reacquisition and Tracking in Videos”. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 761–768. June 2005.
17. Hammer, B. and T. Vilmann. “Generalized Relevance Learning Vector Quantization”. *Neural Networks*, volume 15, 1059–1068. 2002.
18. Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.
19. Jolliffe, I. T. *Principal Component Analysis*. Springer, second edition, 2002.
20. Kerekes, John, Michael Muldowney, Kristin Strackerjan, Lon Smith, and Brian Leahy. “Vehicle Tracking with Multi-temporal Hyperspectral Imagery”. *Proceedings of the SPIE*, volume 6233. June 2006.
21. Khalilzad, Zalmay and Jeremy Shapiro (editors). *Strategic Appraisal: United States Air and Space Power in the 21st Century*. RAND, 2002.
22. Kohonen, Tuevo. *Self-Organizing Maps*. Springer, third edition, 2001.
23. Landgrebe, David. “Hyperspectral Image Data Analysis”. *IEEE Signal Processing Magazine*, 19(1):17–28, January 2002.
24. Lillesand, Thomas M. and Ralph W. Kiefer. *Remote Sensing and Image Interpretation*. John Wiley & Sons, Inc., 4 edition, 2000.
25. The MathWorks, Inc. *Getting Started with Matlab[®] 7*, September 2007.
26. The MathWorks, Inc. *Statistics Toolbox 6 User’s Guide*, September 2007.
27. Maybeck, Peter S. *Stochastic Models, Estimation, and Control*, volume 1. Navtech Book and Software Store, 1994.
28. Mendenhall, M. J. and E. Merényi. “Relevance-based Feature Extraction for Hyperspectral Images”. *IEEE Trans. on Neural Networks, Inprint*, Mar 2008.
29. Mendenhall, Michael J. *Hyperspectral Field Analysis with SpecTIR Corp. Internal Technical Report*. Technical report, Air Force Research Laboratory, Sensors Directorate, 2004.
30. Merényi, E., R. Singer, and J. Miller. “Mapping of Spectral Variations on the Surface of Mars for High Spectral Resolution Telescopic Images”. *ICARUS*, 124:280–295, 1996.
31. Nedeljkovic, I. “Image Classification based on Fuzzy Logic”. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume 34. 2004.

32. Nichiporuk, Brian. *US Military Opportunities: Information-Warfare Concepts of Operation*, chapter 6, 187–223. Khalilzad and Shapiro [21], 2002.
33. Oehler, Karen L. and Robert M. Gray. “Combining Image Compression and Classification Using Vector Quantization”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):461–473, May 1995.
34. Sato, A. and K. Yamada. “Generalized Learning Vector Quantization”. *Advances in Neural Information Processing Systems 8: Proceedings of the 1995 Conference*, 423–429. 1996.
35. Shaik, Jahangheer S. and Kahn M. Iftekharuddin. “Automated Tracking and Classification of Infrared Images”. *IEEE Int. Conference on Image Processing*, 1201–1206, 2003.
36. Shaw, Gary and Dimitris Manolakis. “Signal Processing for Hyperspectral Image Exploitation”. *IEEE Signal Processing Magazine*, 19(1):12–16, January 2002.
37. Silverman, J., C. E. Caefer, and S. DiSalvo V. E. Vickers. “Temporal Filtering for Point Target Detection in Staring IR Imagery”. *SPIE*, 3373:111–122, 1998.
38. Streit, R. L. *Tracking on Intensity-Modulated Data Streams*. NUWC-NPT Technical Report 11,221, Naval Undersea Warfare Center Division, Newport, RI, May 2000.
39. Streit, Roy L., Marcus L. Graham, and Michael J. Walsh. “Tracking in Hyper-Spectral Data”. *Proceedings of the Fifth International Conference on Information Fusion*, volume 2, 852–859. 2002.
40. Sugaya, Y. and K. Kanatani. “Outlier Removal for Motion Tracking by Subspace Separation”. *8th Symposium on Sensing via Image Information (SSII2002)*, 603–608, July 2002.
41. “U.S. Develops Urban Surveillance System”. http://www.prisonplanet.com/us_develops_urban_surveillance_system.html, July 2003.
42. “Efficient and Robust Algorithms for Real-time Video Tracking of Multiple Moving Targets”. Proposal, United States Army Research Office, 2007.
43. Tso, Brandt and Paul M. Mather. *Classification Methods for Remotely Sensed Data*. Taylor and Francis, Inc., 2001.
44. Varsano, Louisa, Irena Yatskaer, and Stanley R. Rotman. “Temporal target tracking in hyperspectral images”. *Optical Engineering*, 45(12):126–201, 2006.
45. Varshney, Pramod K. and Manoj K. Arora. *Advanced Image Processing Techniques for Remotely Sensed Hyperspectral Data*. Springer, 2004.
46. Vesanto, Juha, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas. *SOM Toolbox for Matlab 5*. Helsinki University of Technology, April 2007.
47. Wald, A. *Sequential Analysis*. Dover Publications, New York, 1973.

48. Zhang, Cha and Yong Rui. “Robust Visual Tracking via Pixel Classification and Integration”. *18th International Conference on Pattern Recognition*, volume 3, 37–42. 2006.
49. Zhou, Quming and J.K. Aggarwal. “Tracking and Classifying Moving Objects from Video”. *2nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*. Kauai, HI, USA, December 2001.

Vita

Neil A. Soliman graduated from University of California in Berkeley in 1999 with a B.S. in Electrical Engineering and Computer Science. His first assignment was at Schriever AFB, CO, where he worked as a Global Positioning System payload system operator, instructor, and navigation tactics and payload mission officer. Following his assignment at Schriever AFB, Captain Soliman transferred to Peterson AFB, CO, where he worked as a systems engineer for the Mobile Consolidated Command Centers and as a project manager for space data exploitation. While in CO, he received his M.B.A. at the University of Colorado in Colorado Springs in 2003. After his stint in CO, he was selected to attend the Air Force Institute of Technology at Wright-Patterson AFB, OH where he received his M.S. in Electrical Engineering in March of 2008. Captain Soliman has a follow on assignment to Eglin AFB, FL. He is assigned to the Munitions Directorate, Air Force Research Laboratory.

Permanent address: 2950 Hobson Way
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 074-0188</i>		
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY) 27-03-2008		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From - To) July 2007 - March 2008	
4. TITLE AND SUBTITLE Hyperspectral-Augmented Target Tracking			5a. CONTRACT NUMBER		
			5b. GRANT NUMBER		
			5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Soliman, Neil A., Captain, USAF			5d. PROJECT NUMBER ENG08-262		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765 DSN: 785-3636			8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GE/ENG/08-28		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr. Devert Wicker Air Force Research Laboratory/Sensors Directorate 2241 Avionics Circle WPAFB OH 45433-7321 (937) 904-9871; email: devert.wicker@wpafb.af.mil			10. SPONSOR/MONITOR'S ACRONYM(S)		
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT With the global war on terrorism, the nature of military warfare has changed significantly. The United States Air Force is at the forefront of research and development in the field of intelligence, surveillance, and reconnaissance that provides American forces on the ground and in the air with the capability to seek, monitor, and destroy mobile terrorist targets in hostile territory. One such capability recognizes and persistently tracks multiple moving vehicles in complex, highly ambiguous urban environments. The thesis investigates the feasibility of augmenting a multiple-target tracking system with hyperspectral imagery. The research effort evaluates hyperspectral data classification using fuzzy c-means and the self-organizing map clustering algorithms for remote identification of moving vehicles. Results demonstrate a resounding 29.33% gain in performance from the baseline kinematic-only tracking to the hyperspectral-augmented tracking. Through a novel methodology, the hyperspectral observations are integrated in the MTT paradigm. Furthermore, several novel ideas are developed and implemented—spectral gating of hyperspectral observations, a cost function for hyperspectral observation-to-track association, and a self-organizing map filtering method. It appears that relatively little work in the target tracking and hyperspectral image classification literature exists that addresses these areas. Finally, two hyperspectral sensor modes are evaluated—Pushbroom and Region-of-Interest. Both modes are based on realistic technologies, and investigating their performance is the goal of performance-driven sensing. Performance comparison of the two modes can drive future design of hyperspectral sensors.					
15. SUBJECT TERMS Target tracking, hyperspectral imagery, classification, self-organizing map (SOM)					
16. SECURITY CLASSIFICATION OF:		17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON	
REPORT	ABSTRACT			c. THIS PAGE	Maj Michael J. Mendenhall, PhD (ENG)
U	U	UU	166	19b. TELEPHONE NUMBER (Include area code) (937) 255-6565; email: Michael.mendenhall@afit.edu	