2015-03-01

Vision Based Multiple Target Tracking Using Recursive RANSAC

Kyle Ingersoll
Brigham Young University - Provo

Vision Based Multiple Target Tracking Using Recursive RANSAC

James Kyle Ingersoll

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

March 2015

ABSTRACT

Vision Based Multiple Target Tracking Using Recursive RANSAC

James Kyle Ingersoll
Department of Mechanical Engineering, BYU
Master of Science

In this thesis, the Recursive-Random Sample Consensus (R-RANSAC) multiple target tracking (MTT) algorithm is further developed and applied to video taken from static platforms. Development of R-RANSAC is primarily focused in three areas: data association, the ability to track maneuvering objects, and track management. The probabilistic data association (PDA) filter performs very well in the R-RANSAC framework and adds minimal computation cost over less sophisticated methods. The interacting multiple models (IMM) filter as well as higher-order linear models are incorporated into R-RANSAC to improve tracking of highly maneuverable targets. An effective track labeling system, a more intuitive track merging criteria, and other improvements were made to the track management system of R-RANSAC. R-RANSAC is shown to be a modular algorithm capable of incorporating the best features of competing MTT algorithms. A comprehensive comparison with the Gaussian mixture probability hypothesis density (GM-PHD) filter was conducted using pseudo-aerial videos of vehicles and pedestrians. R-RANSAC maintains superior track continuity, especially in cases of interacting and occluded targets, and has fewer missed detections when compared with the GM-PHD filter. The two algorithms perform similarly in terms of the number of false positives and tracking precision.

The concept of a feedback loop between the tracker and sensor processing modules is extensively explored; the output tracks from R-RANSAC are used to inform how video processing is performed. We are able to indefinitely detect stationary objects by zeroing out the background update rate of target-associated pixels in a Gaussian mixture models (GMM) foreground detector. False positive foreground detections are eliminated with a minimum blob area threshold, a ghost suppression algorithm, and judicious tuning of the R-RANSAC parameters. The ability to detect stationary targets also allows R-RANSAC to be applied to a class of problems known as stationary object detection. Additionally, moving camera foreground detection techniques are applied to the static camera case in order to produce measurements with a velocity component; this is accomplished by using sequential-RANSAC to cluster optical flow vectors of FAST feature pairs. This further improves R-RANSAC's track continuity, especially with interacting targets.

Finally, a hybrid algorithm composed of R-RANSAC and the Sequence Model (SM), a machine learner, is presented. The SM learns sequences of target locations and is able to assist in data association once properly trained. In simulation, we demonstrate the SM's ability to significantly improve tracking performance in situations with infrequent measurement updates and a high proportion of clutter measurements.

Keywords: multiple target tracking, computer vision, foreground detection, machine learning

ACKNOWLEDGMENTS

I would like to acknowledge several people who have provided me with invaluable assistance as I have worked to complete this thesis. First and foremost, I would like to thank my wife, Sage, and my son, James, for their endless support, patience, and understanding throughout my entire time here at BYU.

I would like to thank my advisor, Dr. Beard, for his mentorship during my graduate work. As I am sure most of his graduate students would attest to, Dr. Beard believes in his students' ideas and shows that he is confident that they will work. His confidence in my ideas and research direction was encouraging and motivating, especially early on in my research. I would also like to thank Dr. Beard for his continual emphasis on writing about my research. This counsel made writing my thesis much easier and more enjoyable.

I would also like to thank Dr. McLain. Dr. McLain originally recruited me to work in the MAGICC Lab and portrayed the lab in the best light possible during my first visit here. Throughout my first year here, Dr. McLain provided very useful assistance in helping me establish a research direction and in completing the graduation requirements.

I also had the opportunity to work in collaboration with several excellent graduate students in the MAGICC Lab. I would like to acknowledge Peter Niedfeldt for his work in developing R-RANSAC and his help in getting me up to speed on the algorithm. Peter and I also had the opportunity to work on and submit several papers together. Peter was an excellent example for me of a model graduate student and of how research should be conducted. I would also like to acknowledge Patrick DeFranco. Patrick and I worked on numerous class projects together and were able to provide support to each other throughout the research and thesis writing process.

This project was funded by the Air Force under the Small Business Innovation Research contract no. FA8650-12-M-3217.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

R-RANSAC Parameters & Abbreviations

$\mathbf{x}$	State estimate
$\mathbf{P}$	Error covariance
χ	Consensus set
A	State transition matrix
Q	Covariance of the process noise
τ_{R}	Inlier threshold, radius of inlier region
z	Measurement
C	Measurement observation matrix
I	Identity matrix
K	Kalman gain
R	Covariance of the sensor or measurement noise
$\mathrm{N_w}$	Length of the measurement history window
ℓ	Number of RANSAC iterations during track initialization
ρ	Inlier ratio
τ_ρ	Good model threshold
τ_{T}	Lifetime threshold
τ_{CMD}	Threshold on allowable number of missed detections
m	Size of model set
γ	RANSAC termination criteria
I_{R}	Inlier region

Metric Parameters & Abbreviations

$d(\hat{\mathbf{x}}^i, \mathbf{x}^j)$	Euclidean distance between a track and truth data
$\mathscr{M}_t^*$	Set of valid R-RANSAC tracks
M_t	Index on valid R-RANSAC tracks
c	Cutoff threshold
p	Metric order
$\alpha_{\text{OSPA-T}}$	Penalty for track label switches
$\mathscr{A}_t$	Track-to-target assignments
a_t	Number of track-to-target pairs
MD	Ratio of missed detections
FP	Ratio of false positives
MM	Ratio of mismatches
$\bar{\delta}$	Complement of the Kronecker delta

Merging Parameters & Abbreviations

τ_v	Threshold on percent difference in speed
τ_θ	Threshold on absolute difference in heading
τ_x, τ_y	Thresholds on absolute difference in x and y positions

$\tau_{x_{\min}}, \tau_{y_{\min}}$	Minimum thresholds on absolute difference in x and y positions
GMN	Good model number

Data Association Parameters & Abbreviations

AN	All neighbors data association
NN	Nearest neighbor data association
PDA	Probabilistic data association
JPDA	Joint probabilistic data association
β	Measurement association probability
P_D	Probability of detection
P_G	Gate probability
L	Likelihood ratio
λ	Spatial density of Poisson distribution
S	Covariance of the innovation
$\hat{z}$	Predicted measurement
N	Gaussian probability density function
ν	Measurement residual

Dynamic Model Parameters & Abbreviations

CV	Nearly constant velocity model
CA	Nearly constant acceleration model
CJ	Nearly constant jerk model
dt	Length of time step
MM	Multiple models
IMM	Interacting multiple models
μ	Posterior IMM probability
$\hat{\mathbf{x}}$	Mixed state estimate for IMM
$\hat{\mathbf{P}}$	Mixed error covariance for IMM
π	Markov transition matrix
m	Number of filters in a multiple model algorithm

Video Processing Parameters & Abbreviations

GMM	Gaussian mixture models
RGB	Red, green, blue
α	Background update rate
μ_{RGB}	Mean array of RGB values
σ_{RGB}	Variance array of RGB values
w	Gaussian mode weighting
s_k	Sortkey
Δ	Difference in pixel values
$\tau_{\text{similarity}}$	Threshold to determine pixel similarity

σ_{min}	Minimum variance of a Gaussian mode
B	Gaussian modes that comprise the background model
T	Background model threshold
$\tau_{\lvert v \rvert}$	Similarity threshold on absolute difference in speed, used in sequential-RANSAC
$\tau_{x,y_{\text{cluster}}}$	Position similarity threshold used in sequential-RANSAC
$\ell_{\text{sequential}}$	Number of sequential-RANSAC iterations

Other Parameters & Abbreviations

SM	Sequence Model
$P(\text{SM})_{\text{max}}$	Upper bound on Sequence Model weight
ℓ_{SM}	Learning confidence period
$\mathscr{J}_k$	Number of models stored between time steps for GM-PHD filter
i	Index
j	Index

Subscripts, superscripts, and other indicators

$[\ ]^i$	indicates an index, usually a track or measurement index
$[\ ]^j$	indicates an index, usually a track or measurement index
$[\ ][i]$	indicates the i^{th} element of the array [ ]
$[\ ]_t$	indicates the time index
$[\ ]_{t\|t-1}$	indicates the prediction step of the Kalman filter

CHAPTER 1. INTRODUCTION

Target tracking continues to gain increasing relevance in the robotics and automation communities. In fact, many emerging technologies implicitly rely on a robust, reliable, and computationally efficient tracker. Recent examples of this reliance include intelligent, cooperative control of a team of unmanned air vehicles (UAVs) for tracking ground targets in urban environments [1], target tracking and formation control by a team of mobile robots [2], and robot control based on mutual information theory in order to localize an unknown number of targets [3]. In all of the examples noted above, target tracking enables and motivates higher-level cooperative control and motion-planning technologies. Another important example is simultaneous localization and mapping (SLAM), which inherently relies on multiple object estimation. SLAM is the underlying technology behind GPS-denied navigation and is an especially active research area. Recent work includes incorporating the Gaussian mixture [4] and single-cluster probability hypothesis density (GM-PHD and SC-PHD) [5] filters into the SLAM framework.

A multiple target tracking (MTT) algorithm contains four, fundamental blocks (see Figure 1.1). The first block is track initialization. This block is responsible for initializing tracks and for providing state estimates of the targets at these initialization instances. The second block is data association. As measurements are received, this block determines which measurements should be associated with which tracks. In more sophisticated data association schemes, this block assigns association probabilities to each measurement (soft decision) instead of making hard decisions. The third block is responsible for filtering. Incoming measurements to the tracker include noisy true measurements as well as false positive measurements. Filtering should minimize the effect of false positive measurements and smooth out the noisy true measurements. The fourth block is track maintenance. This block has several responsibilities, all of which contribute to maintaining track continuity. This block establishes a track merging criteria and merges tracks when they meet the criteria. Also, this block identifies tracks that have ceased to exist and eliminates these tracks.

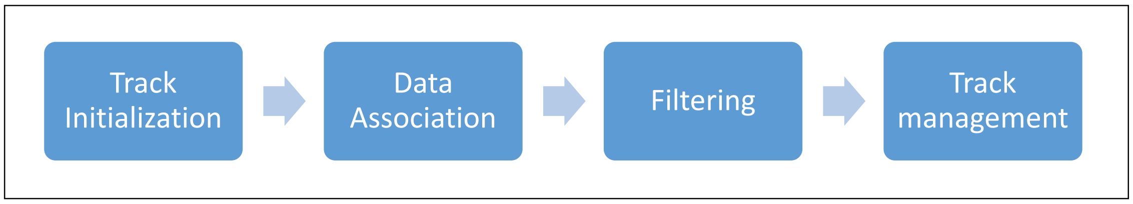


Figure 1.1: The fundamental blocks of a multiple target tracking algorithm.

It is important to clearly differentiate between a multiple object filter and a multiple target tracker. A multiple object filter estimates the *number* of targets and their *states*. A multiple target tracker is an extension of a multiple object filter in that it arranges the state estimates from consecutive time steps into continuous, temporal sequences known as *tracks*. Consequently, multiple target trackers deal with maintaining track continuity, whereas multiple object filters are not concerned with track continuity. Robustly maintaining track continuity in scenarios with closely spaced, interacting or frequently occluded targets is a non-trivial task; this challenge is discussed throughout this thesis.

Multiple target tracking algorithms generally fall into one of three broad categories: track-based algorithms, multiple hypothesis tracking (MHT) algorithms, and random finite set (RFS) based algorithms. Track-based algorithms maintain a set of distinct tracks that are updated recursively, usually with a Bayesian filter. Two of the most widely applied algorithms in this class are the global nearest neighbor (GNN) and the joint probabilistic data association (JPDA) filters [6]. Track-based algorithms perform data association and filtering, but require separate modules to handle track initialization and track management. Other drawbacks suffered by this class of MTT algorithms include poor accuracy in clutter (GNN) and poor performance in low detection environments (GNN and JPDA). Multiple hypothesis tracking [7] propagates many data association, track initiation, and track deletion hypotheses forward and allows future observations decide which hypotheses are valid. Because MHT stores this hypothesis tree, it suffers from high computational complexity. MHT is also subject to jittery or jumping tracks; this is a consequence of the current estimate switching from one branch of the hypothesis tree to another. Attempts have been made to solve this issue and make MHT results more intuitive to a real-time user [8]. The third class of MTT algorithms, those based on RFS, is currently the most active area of research in the tracking community. Significant milestones in this research include the sequential Monte Carlo probability hypothesis density (SMC-PHD) filter [9] and the GM-PHD filter [10]. These algorithms are multiple object filters that, when applied to tracking, have yielded subpar results. The most recently developed algorithm is the labeled multi-Bernoulli filter which is specifically formulated to output tracks [11].

The recently developed Recursive-RANSAC (R-RANSAC) algorithm [12] overcomes many of the deficiencies of currently available MTT algorithms in the following ways:

1. It successfully operates in scenarios with a high degree of clutter measurements and/or low probability of detection.
2. It does not require prior information about the birth and death times or locations of the targets.
3. It is easy to implement and computationally efficient.
4. It maintains exceptional track continuity [12].
5. It autonomously initiates and deletes tracks.

The standard RANSAC algorithm [13] has spawned an enormous number of variations with applicability to a wide range of problems since it was first introduced. Here we cite four RANSAC variations that share similarities with R-RANSAC and then emphasize R-RANSAC's uniqueness and novelty. Reference [14] introduces sequential-RANSAC, a method of estimating multiple *static* signals from a single batch of data. R-RANSAC estimates multiple *dynamic* signals, recursively, from incoming batches of data. Reference [15] introduces multiple-model RANSAC, a technique for estimating ego-motion in dynamic environments. The premise of this technique is that by segmenting both the moving and stationary parts of an image, a better estimate of ego-motion can be obtained. Instead of directly sampling measurements like R-RANSAC, this technique coarsely segments the image into distinct objects and samples these objects. This technique creates virtual scans by estimating the velocity of moving parts of the image with a backwards finite differencing approach. These virtual scans are primarily used to classify objects as either moving or non-moving, but presumably, are used to improve data association as well. However, the authors do not specify how data association is performed or how the multiple models are initialized. R-RANSAC, on the other hand, maintains full state estimates of the targets and has explicit methods for data association and track initialization. Reference [16] introduces incremental-RANSAC, an algorithm designed to estimate the tranformation between local and global maps in SLAM applications. Incremental-RANSAC is similar to R-RANSAC in that hypotheses are generated at each time step with a set of randomly selected features that must contain at least one newly arrived feature. However, incoming features are not classified as inliers or outliers to existing models. Also, incremental-RANSAC only seeks to estimate one model rather than multiple models. Lastly we cite [17] which introduces KALMANSAC. KALMANSAC seeks to find the best state estimate $\mathbf{x}_t$ and set of inliers χ_t for a single signal. This is accomplished by performing several iterations in which a new state estimate is calculated via RANSAC using a refined set of inliers and the set of inliers is then re-optimized given the new state estimate. R-RANSAC differs in that the state estimate is not estimated via RANSAC at every time step, rather, the estimation is performed using a dynamic filter.

The research presented in this thesis occupies an important position in a continuing line of research taking place in Brigham Young University's Multiple Agent Intelligent Coordination and Control (MAGICC) Lab. The primary contribution of this thesis is to complete the development of R-RANSAC; this includes further improvements to the data association, filtering, and track management blocks of R-RANSAC. As a result of this research, future avenues of research in the MAGICC Lab will be able to use R-RANSAC as a plug-and-play algorithm that robustly tracks a wide range of targets. Concurrent research projects focusing on the development of a moving camera foreground detector and on the application of R-RANSAC to decentralized, multiple agent tracking scenarios have already been able to use the research presented here in this manner. Another major contribution of this thesis is the application of R-RANSAC to video taken from static platforms. Several novel stationary foreground detection techniques were developed that allow R-RANSAC to track stationary objects and maintain track continuity in the case of crossing objects. These techniques could be further developed to make them compatible with video taken from moving platforms or could be implemented without modification on stationary surveillance systems or on UAVs with automated landing zone detection software that primarily perform tracking while landed. The final contribution of this thesis is the application of machine learning to the MTT problem.

It should be noted that R-RANSAC is not a computer vision tracking algorithm. Rather, it is an MTT algorithm that can process measurements received by a variety of sensors, including cameras. Traditional computer vision approaches to tracking include feature matching, template matching, and mean shift [18]. Feature matching approaches are often based on the Lucas-Kanade tracker (for the pyramidal implementation found in OpenCV, see [19]). Reference [20] describes a feature matching method that, similar to R-RANSAC, uses a Kalman filter to estimate the features' future locations and to track objects through mild occlusions. Template matching approaches vary widely, but common approahces include a course-to-fine searching method or using a sparser representation of the image for improved efficiency [21], [22]. In general, computer vision tracking approaches require the user to manually initialize tracks and would be prohibitively computationally expensive to run on a small UAV. By applying R-RANSAC to measurements extracted from video data, we hope to be able to efficiently and autonomously track objects in video.

This thesis is outlined as follows. Chapter 2 reviews the RANSAC algorithm and different aspects of the original R-RANSAC algorithm. Chapter 3 describes the simulation environment used to test R-RANSAC and explains the metrics used to measure tracking performance. Chapter 4 discusses improvements to the track management block of R-RANSAC, including the conversion of R-RANSAC from a multiple object filter into a multiple target tracker. Chapter 5 describes the incorporation of several established data association techniques into the R-RANSAC framework and presents simulation results showing improved tracking performance. Chapter 6 describes modifications to R-RANSAC designed to increase its ability to track highly maneuverable targets. Chapter 7 presents a comprehensive comparison between R-RANSAC and the GM-PHD filter. Chapter 8 gives background information on the video processing necessary to track objects in video. Chapter 9 discusses several foreground detection techniques that create a feedback loop between the tracker module (R-RANSAC) and the video processing module of a video tracking system. Simulation results are presented that demonstrate the improvement in tracking performance enabled by these techniques. Chapter 10 discusses how machine learning can be leveraged to improve tracking performance and describes the incorporation of the Sequence Model into the R-RANSAC framework. Chapter 11 provides conclusing remarks about the research presented here.

CHAPTER 2. THE ORIGINAL RECURSIVE-RANSAC ALGORITHM

In this chapter, the original R-RANSAC algorithm is reviewed. This discussion is preceded by a review of RANSAC in Section 2.1. The filtering, track initialization, and track management blocks of R-RANSAC are explained in Sections 2.2.1, 2.2.2, and 2.2.3 respectively. Section 2.2.4 discusses alternatives ways to view R-RANSAC in the context of other MTT algorithms. This chapter contains the background information necessary to understand the developments made to R-RANSAC presented in later chapters.

2.1 Random Sample Consensus

The Random Sample Consensus (RANSAC) algorithm [13] was designed to estimate the parameters of a signal in the presence of gross errors. When gross errors are present, traditional methods like least-squares regression often poorly model the signal of interest. RANSAC has been successfully applied to a wide range of computer vision problems. One of the most well-known applications of RANSAC is the computation of the homography, or geometrical transformation, between two images. When computing a homography, feature points along with their accompanying descriptors are found in each image. The feature points are then matched across the images by comparing their descriptors. While many features are correctly matched, incorrect matches are inevitable. Figure 2.1 demonstrates this situation[1]. Given this set of feature matches, RANSAC is used to compute the true homography; the resulting panoramic image can be seen in Figure 2.2.

The RANSAC algorithm proceeds as follows. First, an assumed signal model is selected. In the homography example, the chosen model is a 4 by 4 matrix encoding translation, rotation, and scaling information. In the example presented in Figure 2.3, the chosen model is a line characterized by its slope and vertical intercept. Second, a random subset of data points is selected.

[1]Images are from the windows dataset and can be accessed at `https://canvas.instructure.com/courses/743674/assignments/1929377`

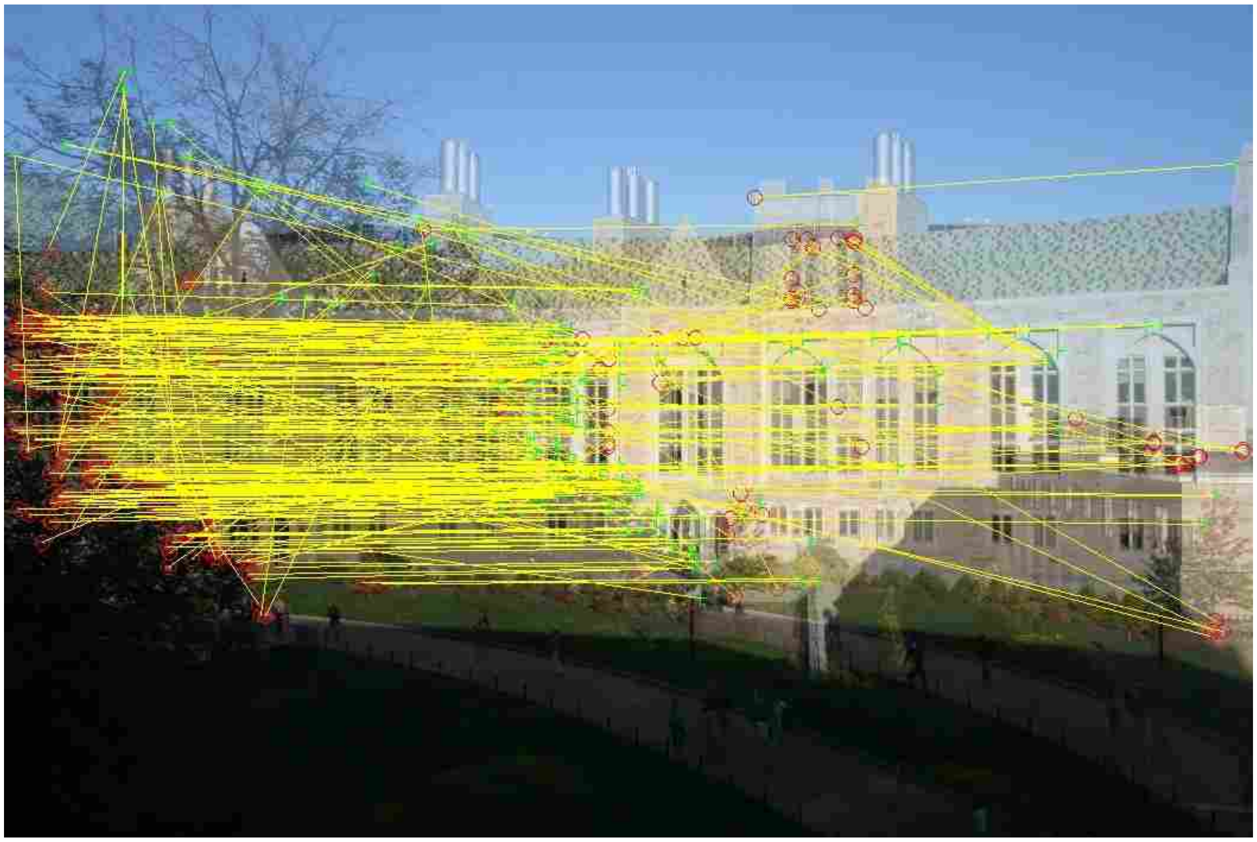


Figure 2.1: Two overlapping images of a building are overlaid on top of each other. The transformation between these images is almost purely translational in the horizontal direction. Features in one image are labeled in red and features in the other image are labeled in green. Matching features are indicated by yellow lines. The majority of features are correctly matched, though some incorrect matches do exist.

The number of data points selected is the minimum number needed to estimate the model's parameters. In the homography case, four points are needed; in the case of Figure 2.3, only two points are needed. Third, a model is constructed with this random subset of data points. Fourth, all remaining data points are classified as either inliers or outliers to this new model. This inlier/outlier classification is performed by setting an inlier threshold; all points that fall within this threshold are denoted as inliers and are stored as the new model's consensus set. Steps two through four are performed iteratively. During the iterations, the model with the largest consensus set, or in other words, the model with the most support, is stored. At the end of the iterations, a smoothed model is produced by performing a least-squares regression on the consensus set of the model with the most support.

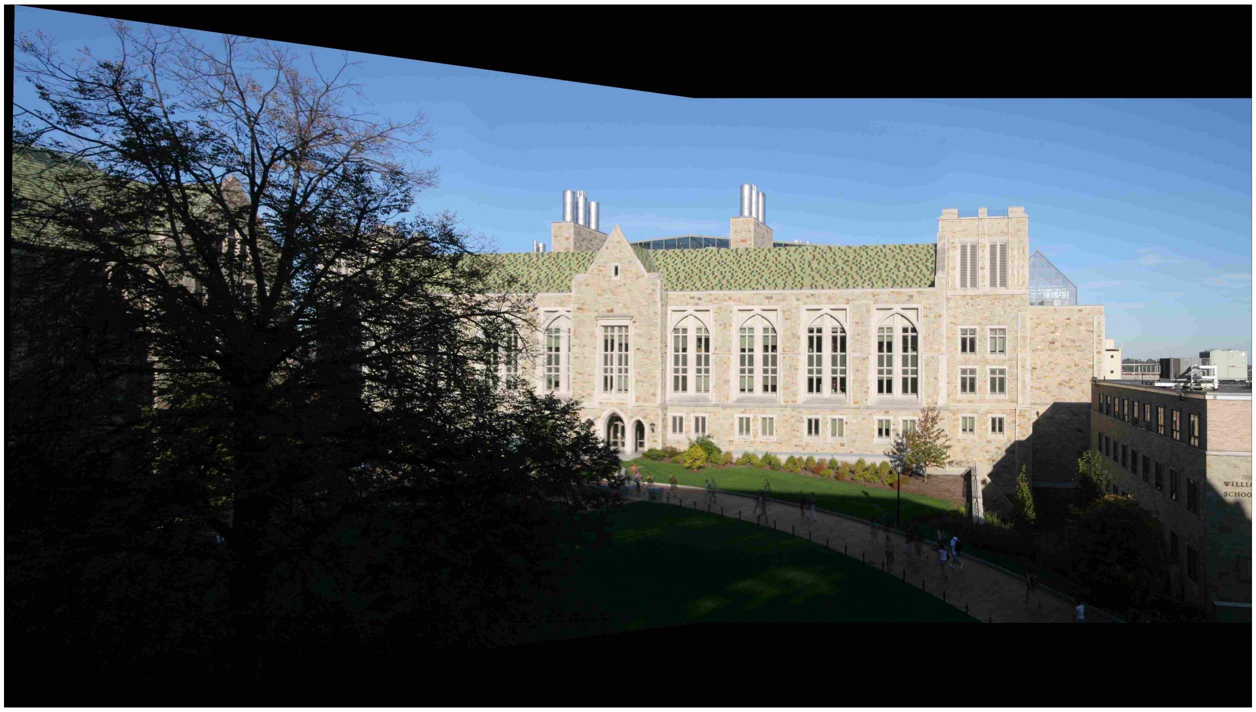


Figure 2.2: The resulting panoramic image produced with the homography calculated from the correctly matched feature pairs identified in Figure 2.1.

As mentioned earlier, the primary strength of the RANSAC algorithm is its robustness to gross errors. Consider the situation in Figure 2.3. In this example, the majority of points follow a monotonically increasing trend, but there do exist two outlier points. The result of a linear least-squares regression is plotted in green. Note that even with just two outliers, the least-squares estimate clearly diverges from the true signal. The RANSAC estimate is plotted in red and is not influenced by the outlier data points (in this example, the RANSAC estimate is actually equivalent to a least squares estimate of the true measurements).

To summarize, RANSAC generates several hypotheses of how to best model the available data and relies on the data to determine which hypothesis has the most support.

2.2 Recursive-RANSAC

Recursive-RANSAC was originally developed as a multiple object filter by Dr. Randal W. Beard and Dr. Peter C. Niedfeldt at Brigham Young University. The motivation behind R-RANSAC was to design a filter that inherited RANSAC's ability to robustly reject gross errors, but

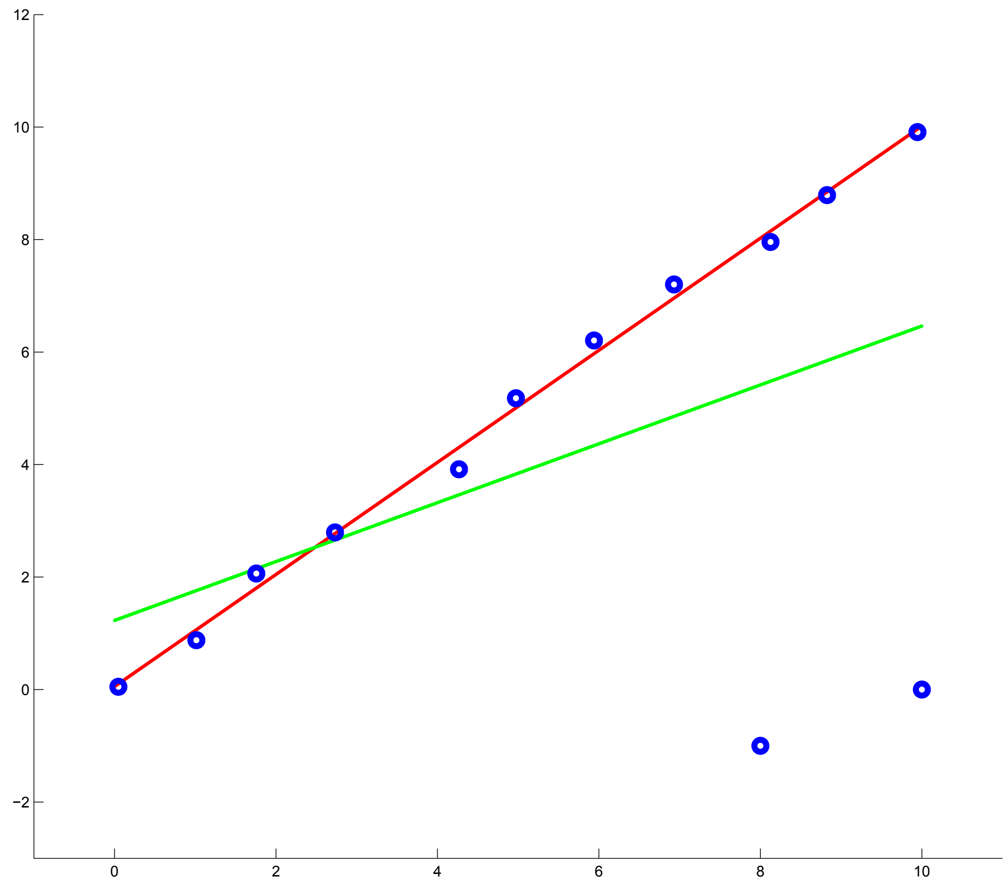


Figure 2.3: The data points (in blue) are modeled with RANSAC (red) and by a least-squares regression (green). RANSAC is able to identify and ignore the outlier points.

that could be used to estimate multiple signals. Reference [23] introduces R-RANSAC as a method of estimating multiple static signals that are updated via recursive-least squares. R-RANSAC is extended in [12] to estimate time-evolving signals. The conversion of R-RANSAC to an MTT algorithm is straightforward and is discussed in Chapter 4.

2.2.1 Filtering

R-RANSAC maintains a model set, a set of hypothesis tracks described by their state estimate $\mathbf{x}$, error covariance $\mathbf{P}$, and consensus set χ. At every time step, each hypothesis track is propagated forward using the predict step of the Kalman filter, given by

$$\mathbf{x}_{t|t-1} = A\,\mathbf{x}_{t-1} \tag{2.1}$$

$$\mathbf{P}_{t|t-1} = A\,\mathbf{P}_{t-1}A^{\top} + Q \tag{2.2}$$

where A is the state transition matrix of the assumed motion model and Q is the covariance of the process noise.

Once all tracks have been propagated forward, an inlier threshold τ_{R} is set around each track; the ℓ_1 norm is used as a threshold. An inlier region is created when the inlier threshold is applied ($I_{\mathrm{R}} = \{z : ||z - C\mathbf{x}_{t|t-1}||_1 < \tau_{\mathrm{R}}$). For a given track, all measurements from the current scan are classified as inliers or outliers to that track according to whether or not the measurements fall within the track's inlier region. Each inlier is used to update the track using the update equations of the Kalman filter, given by

$$\mathbf{x}_t = \mathbf{x}_{t|t-1} + K\left(z - C\,\mathbf{x}_{t|t-1}\right) \tag{2.3}$$

$$\mathbf{P}_t = (I - KC)\,\mathbf{P}_{t|t-1} \tag{2.4}$$

where z is the measurement, C is the measurement observation matrix that relates the measurement to the target states, I is an identity matrix of the same dimensions as A, and K is the Kalman gain and is given by

$$K = \mathbf{P}_{t|t-1}C^{\top}\left(R + C\mathbf{P}_{t|t-1}C^{\top}\right)^{-1} \tag{2.5}$$

where R is the covariance of the sensor noise. The term *inlier region* is phraseology borrowed from RANSAC. In the tracking community, this region is more commonly referred to as a gate or a *measurement validation region* (see, for example, [24]). R-RANSAC uses an inlier region of fixed volume and bases the size of the inlier region on the assumed measurement noise covariance.

2.2.2 Track Initialization

Measurements that are outliers to all existing tracks are used to initialize new tracks using a RANSAC-based method. R-RANSAC stores all the measurements from the past $\mathrm{N_w}$ time steps in a measurement history window. When a measurement is found to be an outlier to all existing tracks, a random subset of measurements (which includes the outlier measurement under consideration) is selected. Using this subset of measurements, a model is computed. The measurement history window is then searched for other measurements that support this model, i.e. that meet the inlier threshold. This process of creating hypothesis models is repeated iteratively and the model with the largest consensus set is stored. At the end of ℓ iterations, the model with the most support is then propagated forward in time to the current time step. As it is propagated forward, it is updated by the measurements that compose its consensus set. These propagation and update steps are performed with a Kalman filter. This new model is then appended to the model set. The RANSAC iterations are also terminated if the cardinality of a model's consensus set exceeds $\gamma = \frac{\tau_\rho}{\mathrm{Nw}}$. The equations used to generate the hypothesis models are given in [25]. Due to the RANSAC-based initialization method, tracks are often referred to as models in the R-RANSAC context. The two terms are used interchangeably here.

Figure 2.4 provides a snapshot of a single time step of R-RANSAC. At this time step, there are three tracks in the model set, labeled by blue X's. There are several incoming measurements, drawn as circles. The inlier region of each track is indicated by a red square. Measurements classified as inliers are drawn in cyan whereas outlier measurements are drawn in orange. Each orange-colored measurement will be used to generate a new, hypothesis track which will then be appended to the model set.

2.2.3 Track Management

The remaining steps of R-RANSAC fall under the umbrella of track management: identifying valid tracks, merging redundant tracks, and pruning low-support tracks.

At the beginning of every time step, the consensus set of each track in the model set is updated by removing older measurements that have left the measurement history window. Each track's inlier ratio ρ is also calculated by $\rho = \frac{|\chi|}{Nw}$ where $|\chi|$ is the cardinality of the consensus

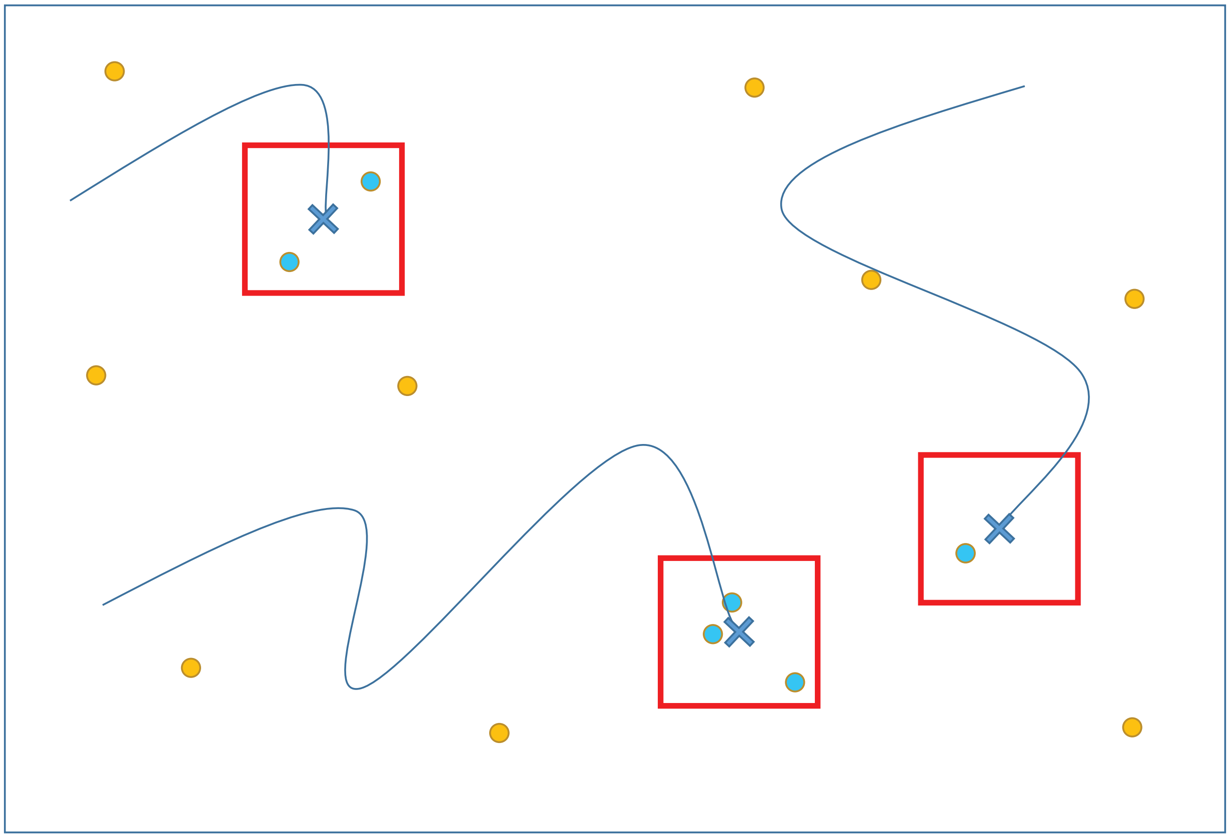


Figure 2.4: A single time step of R-RANSAC. Current state estimates are indicated by blue X's. Inlier measurements are indicated by cyan circles. Outlier measurements are indicated by orange circles. The inlier regions are displayed as red squares.

set. Good, or valid, tracks are identified with a good model threshold and a timeline threshold. The good model threshold τ_ρ is the minimum inlier ratio for a model to be considered a good model. Likewise, the timeline threshold τ_T is the minimum number of time steps a model must have existed to be considered a good model. Models that meet both thresholds are outputted as good models for that time step.

In order to limit the number of false positive tracks, redundant tracks must be identified and removed from the model set. The Mahalanobis distance is used as a merging criteria. If two tracks meet the merging criteria, the track with the higher inlier ratio is retained and the other one is discarded. At the end of each time step, the tracks in the model set are ordered by their inlier ratio. The model set has a fixed size M and is truncated at each time step, thereby removing the least probable tracks.

2.2.4 R-RANSAC: Alternative Viewpoints

R-RANSAC was initially developed as a standalone multiple object filter, a fully-integrated package that was mutually exclusive with other tracking approaches such as JPDA and MHT. However, in the process of further evolving R-RANSAC, it has been beneficial to view R-RANSAC in other ways.

R-RANSAC can be viewed purely as a track initialization algorithm, albeit one that imposes certain constraints on the filtering and data association blocks of the tracker. As a track initialization algorithm, R-RANSAC simply requires that each incoming measurement be classified as an inlier or an outlier. Given this requirement, any number of filters and data association techniques may be used with R-RANSAC. Consequently, R-RANSAC becomes a modular algorithm whose blocks can be substituted with application-specific replacements depending on the tracking scenario. This idea does not diminish the impact or utility of R-RANSAC; rather, it makes R-RANSAC highly adaptable, and applicable to an even greater number of situations. Figure 2.5 illustrates the modularity of R-RANSAC and includes several algorithms that can be used for data association and filtering.

R-RANSAC can also be viewed as a way of truncating the hypothesis tree generated by MHT. The un-truncated form of MHT would retain all track initialization, track death, and data association combinations from the beginning of the tracking scenario. However, it quickly becomes computationally intractable to maintain this tree. Consequently, the tree is continuously pruned so as to only propagate forward the most probable (and compatible) branch combinations. R-RANSAC, through its RANSAC-based track initialization method and fixed-size model set, only propagates forward what it deems to be the most probable hypothesis tracks. Hypothesis tracks that continue to receive support rise to the top of the model set whereas hypothesis tracks that fail to receive further support are eventually removed from the model set.

This section has provided a high-level overview of R-RANSAC. Implementation details including actual threshold values and motion models will be presented in later chapters as improvements and modifications to the original version of R-RANSAC are discussed. The chapters to follow will discuss enhancements made to virtually all aspects of the original R-RANSAC algorithm.

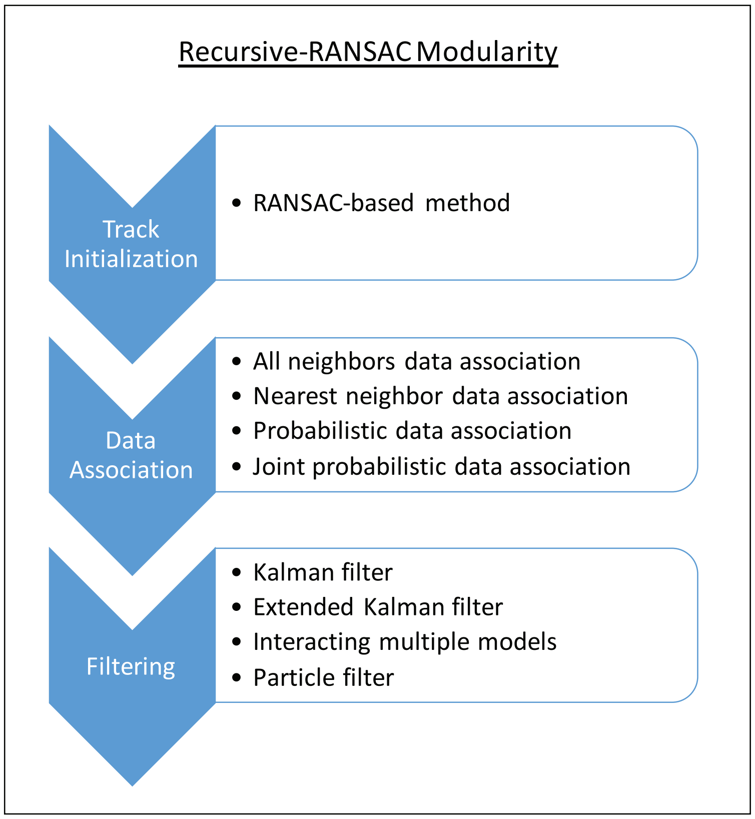

Recursive-RANSAC Modularity
Track Initialization
RANSAC-based method
Data Association
All neighbors data association
Nearest neighbor data association
Probabilistic data association
Joint probabilistic data association
Filtering
Kalman filter
Extended Kalman filter
Interacting multiple models
Particle filter

Figure 2.5: Recursive-RANAC as a modular algorithm.

CHAPTER 3. SIMULATION ENVIRONMENT

This chapter describes the general simulation environment used to test the variations of R-RANSAC developed in later chapters. Section 3.1 describes the test videos used in the simulations. Section 3.2 defines the metrics used to quantify tracking performance.

3.1 Test Videos

All of the videos described here are taken from static platforms high above the ground. These videos were chosen so as to approximate the type of video that a small UAV might capture. A UAV is a moving platform, or at the very least a jittery or shaky platform, and therefore video taken from a UAV-mounted camera would require motion compensation, a subject that is beyond the scope of this thesis. However, the techniques developed here are mostly applicable to motion compensated video. Furthermore, the techniques developed here are directly applicable to UAVs with automated landing zone detection capabilities that generally track objects after having landed.

Video data includes a plethora of tracking challenges and is an excellent way of stressing a tracking algorithm. Common tracking challenges present in video include:

1. *Occlusions*. Objects of interest often pass behind physical occlusions such as trees and light posts and are not detected by the camera. Additionally, shadows can fully occlude objects in low probability of detection scenarios. A robust MTT algorithm should have the ability to track targets through mild occlusions while maintaining consistent track labels.

2. *Missed Detections*. Missed detections, besides those that result from occlusions, occur frequently in the case of slowly moving and small targets. When viewed from an aerial platform, most targets appear to be small and slowly moving.

3. *False Positives*. False positives are especially prevalent in video data. In static, non motion-compensated video, false positives result from camera motion. Motion in the image frame

may results from motion of the camera platform or from the camera adjusting its focus. False positives in video can be particularly difficult to compensate for because they are usually not uniformly distributed throughout the field of view. For example, a slightly shifted camera produces false positive measurements along most distinct edges in the video.

4. *Closely Spaced or Interacting Targets*. Videos often contain many closely spaced or interacting targets. Video taken from an oblique angle might even contain targets that temporarily occlude other targets. This poses a significant challenge for the data association method.

The videos described below were all taken with a Canon PowerShot SX500 IS point-and-shoot camera at a frame rate of 29 frames per second (fps).

3.1.1 Pedestrian Video

The pedestrian video was taken from the fourth floor balcony of the Joseph F. Smith building on the BYU campus. It captures most of the open courtyard to the east of the building. The objects of interest in this video are pedestrians. Figure 3.1 shows one frame from this video. The resolution of this video is 1280 x 720. This video is 5 minutes and 48 seconds long (approximately 10,092 frames). A 600 frame sequence from this video was annotated. The beginning frame of the annotated sequence is frame 4347 and was chosen randomly. The annotated sequence contains 15 objects of interest. The first minute of the video includes several closely spaced targets and is used to test data association methods.

3.1.2 Vehicle Video

The vehicle video was taken from the top of Y Mountain. It is meant to simulate an aerial video. It captures a trapezoidal area of residential Provo with side lengths of approximately 194, 354, 220, and 255 meters. The objects of interest are vehicles. Figure 3.2 shows one frame from this video. The original resolution of this video was 1280 x 720, but it was post-processed to reduce the resolution to 854 x 480. This video is 8 minutes and 50 seconds long (approximately 15,370 frames). As in the pedestrian video, a 600 frame sequence was annotated beginning at frame 7344. The annotated sequence contains 18 objects of interest. The objects of interest are small and more difficult to detect. There are shadows throughout the video that fully occlude the targets.

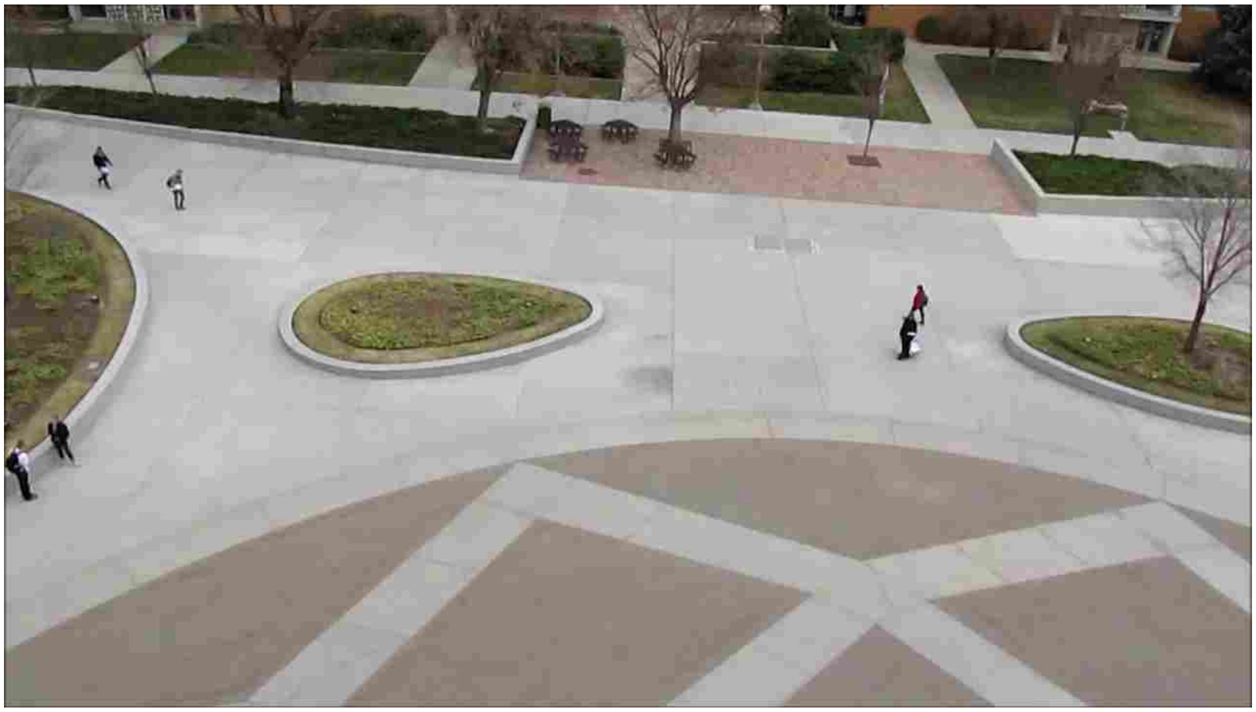


Figure 3.1: One frame from the pedestrian video.

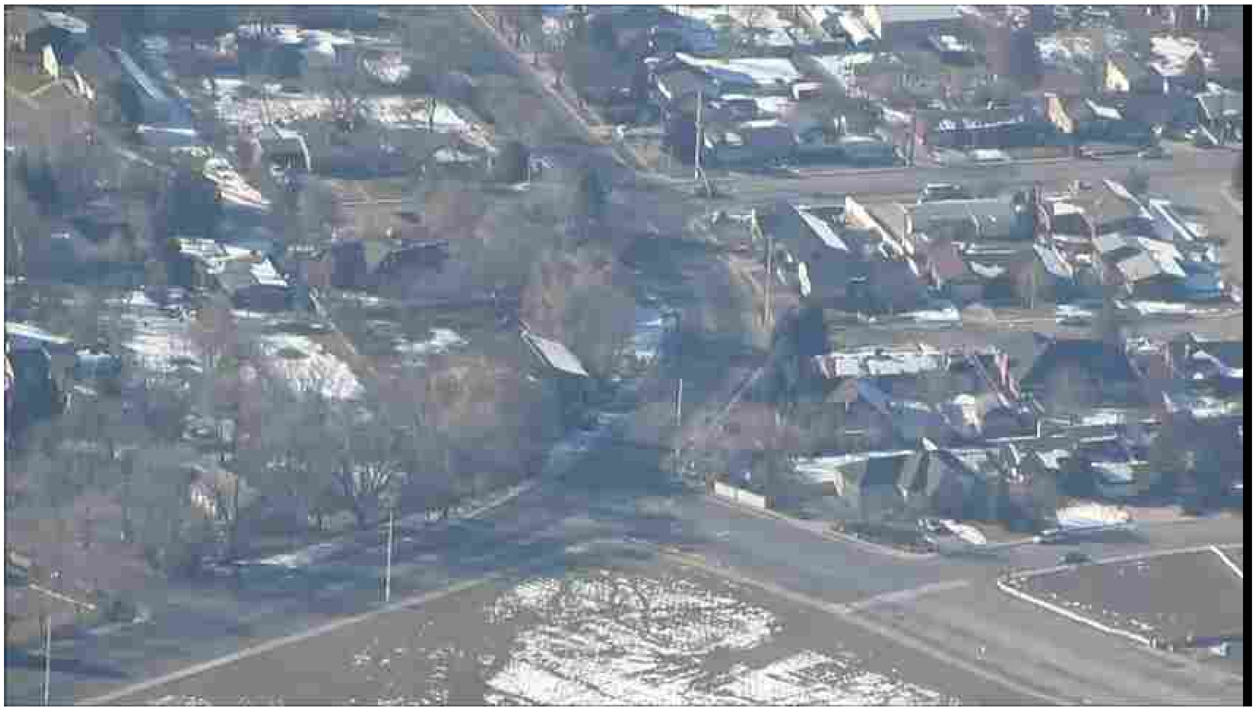


Figure 3.2: One frame from the vehicle video.

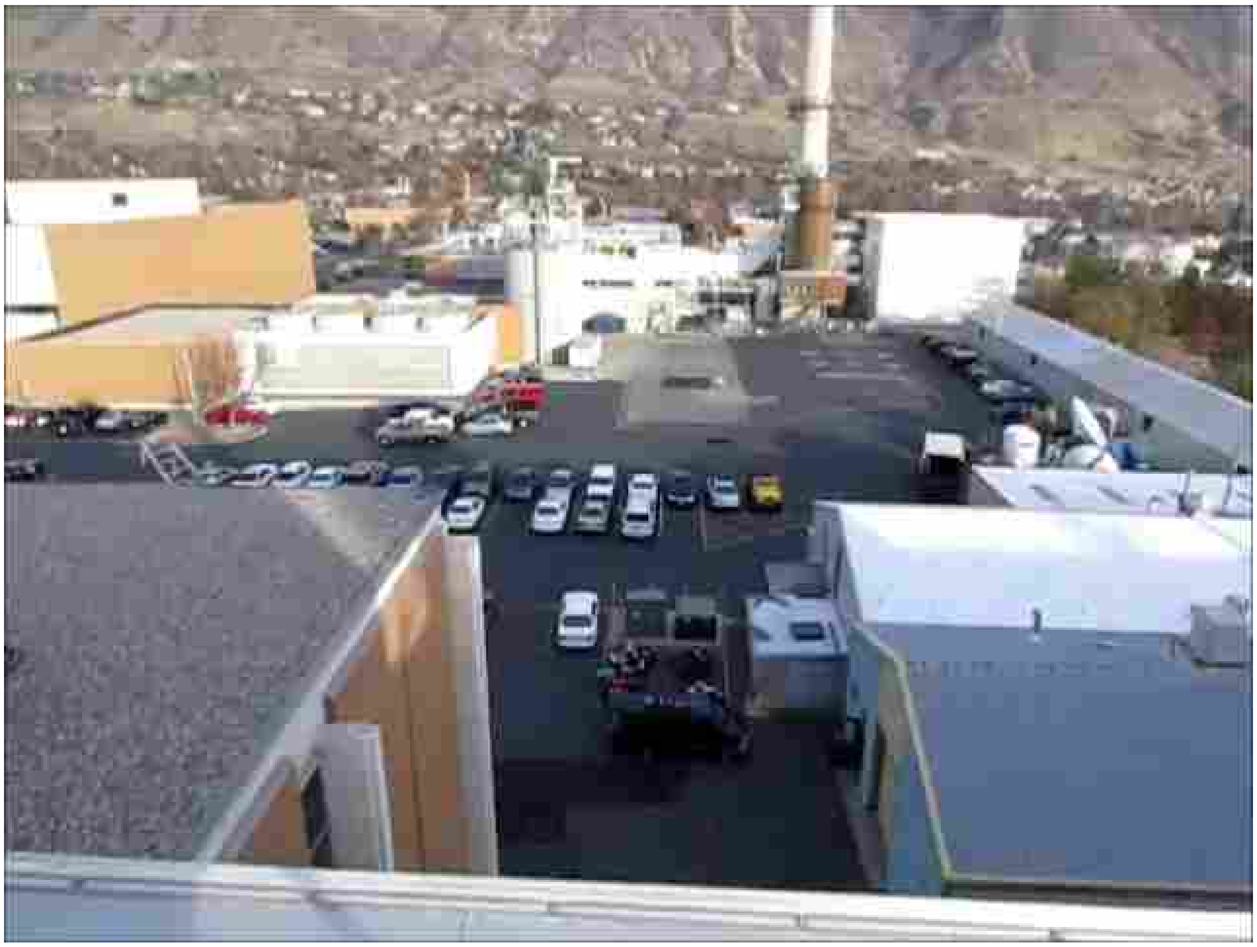


Figure 3.3: One frame from the first parking lot video.

3.1.3 Parking Lot Videos

Two videos were taken from the roof of the Clyde Building on the BYU campus overlooking the parking lot located between the Crabtree, Fletcher, and Clyde Buildings. Figures 3.3 and 3.4 show one frame from each video, respectively. Both videos were originally taken at a resolution of 640 x 480. The first parking lot video is 2 minutes and 35 seconds long and the second parking lot video is 18 minutes and 25 seconds long. Both videos contain moving objects of interest that become stationary (parking cars) and were meant to test R-RANSAC's ability to track stationary objects. Neither video was annotated.

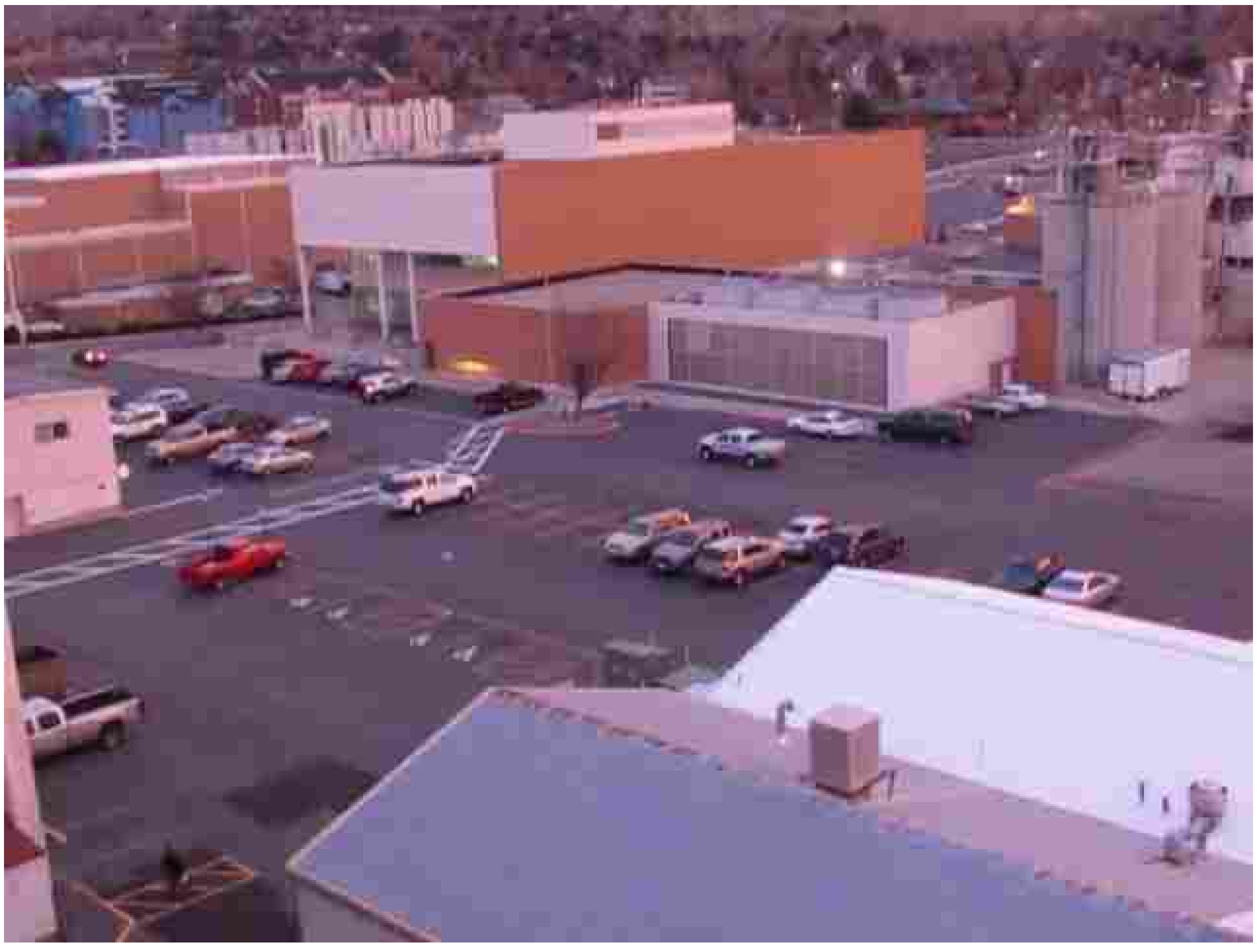


Figure 3.4: One frame from the second parking lot video.

3.1.4 PETS 2006 Dataset

The PETS 2006 dataset was used to test R-RANSAC's ability to track abandoned objects. These videos were taken from a high-mounted static camera in a train station. This dataset can be found `http://www.cvg.reading.ac.uk/PETS2006/data.html`.

3.2 Metrics

The optimal subpattern assignment (OSPA) and OSPA-track (OSPA-T) and the CLEAR multiple object tracking (MOT) metrics are used to measure tracking performance. In computing these metrics, we let $\mathscr{M}_t^*$ be the set of good tracks, and let $i = 1, \ldots, |\mathscr{M}_t^*|$ be an index over the good tracks. We define $j = 1, \ldots, M_t$ as an index over all true targets at time t. We define $d(\hat{\mathbf{x}}^i, \mathbf{x}^j) = \left\| \hat{\mathbf{x}}_t^i - \mathbf{x}_t^j \right\|_2$ as the Euclidean distance between good tracks and true targets. Also, we define the cutoff distance as

$$d_c(\hat{\mathbf{x}}^i, \mathbf{x}^j) = \begin{cases} d(\hat{\mathbf{x}}^i, \mathbf{x}^j) & \text{if } d(\hat{\mathbf{x}}^i, \mathbf{x}^j) < c \\ \Delta & \text{otherwise,} \end{cases} \tag{3.1}$$

where $c > 0$ is a cutoff threshold describing the maximum allowable distance between estimated tracks and true targets and Δ is the distance value assigned when the distance exceeds the cutoff threshold. In the formulation of the MOT and OSPA metrics used here, $\Delta = c = 100$. The set of all cutoff distances is summarized by the $|\mathscr{M}_t^*| \times M_t$ matrix D^c, where each element $D^c_{ij} = d_c(\hat{\mathbf{x}}^i, \mathbf{x}^j)$. A modified Munkres algorithm [26] is used to solve the 2-D assignment problem that finds the best track-to-target assignment pairs, given by $\mathscr{A}_t = \mathsf{Munkres}(D^c)$, where $\mathscr{A}_t$ is the set of $a_t \leq M_t$ track-to-target pairs. For convenience $\mathscr{A}_t^j$ is defined as the track assigned to the j^{th} target.

3.2.1 OSPA and OSPA-T

The OSPA metric, first introduced in [27], is a single metric that measures the overall performance of multiple object filters. It was designed to overcome several of the weaknesses of the other concurrently available metrics. The authors of [27] note that competing metrics did not deal with cardinality differences between the ground truth and filter estimate in an intuitive manner, displayed geometric dependent behavior (i.e. differences in cardinality were penalized less heavily for targets that were closely spaced together than for more widely spaced targets), were undefined when the cardinality was zero, and were not actually metrics as defined by mathematical theory. The OSPA-T metric [28], an extension of the OSPA metric, measures the performance of MTT algorithms.

When $M_t \leq a_t$, the OSPA-T metric is formulated as

$$\begin{aligned} &\mathsf{OSPA\text{-}T} = \\ &\left[\frac{1}{a_t}\left(\sum_{i=1}^{M_t}\left(d_c(\hat{\mathbf{x}}^{\mathscr{A}_t^j}, \mathbf{x}^j) + \alpha_{\text{OSPA-T}}\bar{\delta}[\hat{\mathscr{L}}^{\mathscr{A}_t^j}, \mathscr{L}^j]\right)^p + c^p(a_t - M_t)\right)\right]^{\frac{1}{p}}, \end{aligned} \tag{3.2}$$

where $c = 100$ is the cutoff distance, $p = 2$ is the metric order parameter, and $\alpha_{\text{OSPA-T}} = 75$ determines the relative penalty assigned to incorrectly labeled tracks or mismatches; the OSPA-

T metric reduces to the OSPA metric when the parameter $\alpha_{\text{OSPA-T}} = 0$. If $M_t = a_t = 0$ then $\mathsf{OSPA} = \mathsf{OSPA\text{-}T} = 0$. Lower OSPA and OSPA-T scores correspond to better tracking performance. Similar OSPA and OSPA-T scores indicate strong track continuity. These score are computed for each frame in the annotated sequences and an average over all 600 frames is reported.

3.2.2 CLEAR MOT

The multiple object tracking precision (MOTP) and multiple object tracking accuracy (MOTA) metrics [29] describe the precision and accuracy of an MTT algorithm in such a way that the precision of the algorithm has no effect on its accuracy score, and vice versa. The MOTP metric is the sum of the root-mean square (RMS) errors of all of the successfully tracked objects in a consecutive video sequence. A successfully tracked object is defined as a ground truth target which has a track assigned to it by the modified Munkres algorithm. The metric is computed by

$$\mathsf{MOTP} = \sum_t \frac{\sum_{j=1}^{a_t} d(\hat{\mathbf{x}}^{\mathscr{A}_t^j}, \mathbf{x}^j)}{a_t}. \tag{3.3}$$

MOTA is a composite metric composed of the average ratios of missed detections, false positives, and mismatches (track label switches). $\mathsf{MD}_t = M_t - a_t$ is defined as the ratio of missed detections. $\mathsf{FP}_t = \left|\mathscr{M}^* / \mathscr{A}_t^j\right|$ is defined as the ratio of false positives or the number of all unassigned tracks, where $/$ is the set difference operator. Finally, $\mathsf{MM}_t = \sum_{j=1}^{a_t} \bar{\delta}[\hat{\mathscr{L}}^{\mathscr{A}_t^j}, \mathscr{L}^j]$ is defined as the ratio of label mismatches, where $\bar{\delta}$ is the complement of the Kronecker delta, i.e. $\bar{\delta}[\hat{\mathscr{L}}^i, \mathscr{L}^j] = 0$ if $\hat{\mathscr{L}}^i = \mathscr{L}^j$ and $\bar{\delta}[\hat{\mathscr{L}}^i, \mathscr{L}^j] = 1$ if $\hat{\mathscr{L}}^i \neq \mathscr{L}^j$. The MOTA metric is then given by

$$\mathsf{MOTA} = 1 - \sum_t \frac{\mathsf{MD}_t + \mathsf{FP}_t + \mathsf{MM}_t}{M_t}. \tag{3.4}$$

This formulation of the MOT metrics deviates slightly from that presented in [29]. When performing the track to ground truth mapping, [29] gives priority to previous labels; if the ground truth target was assigned a given track at the previous time step, the same assignment is made at the current time step provided that the distance between the ground truth target and the previously assigned track does not exceed the cutoff threshold. For computational ease, the formulation used here foregoes this step.

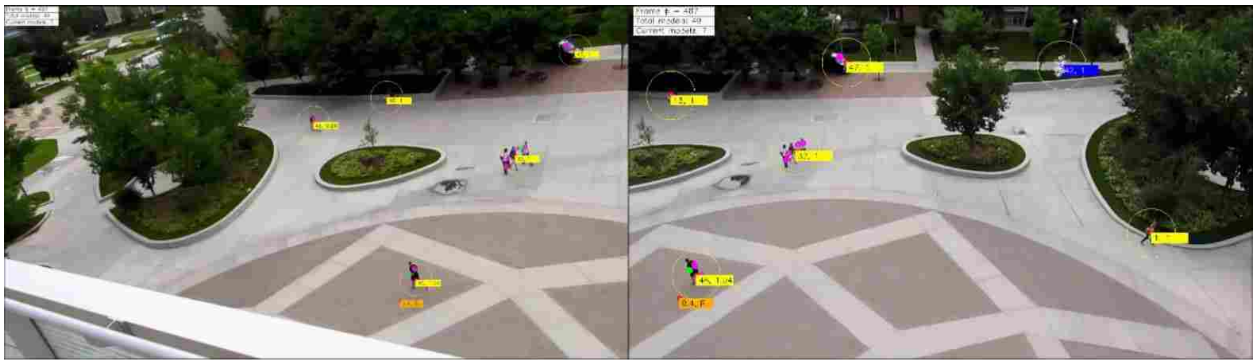


Figure 3.5: R-RANSAC can be used as a centralized tracker if measurements from multiple cameras are transformed into a common coordinate system. In this case, the measurements originating from the left panel are transformed into the coordinate system of the right panel. The homography is pre-calculated using manually selected feature pairs.

Lower MOTP and higher MOTA scores correspond to better tracking performance.

3.3 Other Notes on the Simulation Environment

In all of the experiments, tracking is performed in the image coordinate frame. Measurements are (x, y) pixel values and state estimates are in units of pixels. If the targets are constrained to a pseudo-planar surface, tracking in the world coordinate frame can be accomplished with a simple homography transformation. Similarly, measurements from multiple cameras can be transformed into a common coordinate frame, concatenated, and fed into a centralized R-RANSAC tracker in order to increase the effective field of view; this is illustrated in Figure 3.5.

Because R-RANSAC uses a random number generator to select measurements with which to initialize tracks, results may vary from simulation to simulation[1]. For many of the results, the R-RANSAC version being considered was run through several simulations. Information about the number of trials is included with the results. Details concerning video processing are also provided with the results. In general, the video is processed by a foreground detector that segments out moving objects and produces a foreground mask. This mask is then post-processed using a combination of dilation and erosion morphological operations. Finally, a blob detector identifies separate, contiguous blobs and outputs their centroids as measurements. The development of R-

[1]The results presented in Chapters 9 and 10 report the standard deviation of the OSPA-T metric for the various simulations. Additionally, complete simulations results can be found in Appendices A and B. As the number of RANSAC iterations ℓ increases, the results become more consistent, but computation time increases.

RANSAC occurred in both Matlab and C++. As results are presented, the language used will also be specified. All Matlab simulations use Matlab's Computer Vision System Toolbox for video processing. The C++ implementations use the OpenCV libraries for image handling and our own computer vision algorithms.

In many figures, the R-RANSAC tracks are superimposed over a single video frame. Different tracks are represented as different colors. Sometimes, a single track (indicated by the same color) is broken up into several fragments. This occurs when the inlier ratio drops below the inlier threshold (such as when an object passes behind an occlusion), and then rises above the inlier threshold when the target resumes being detected. This is preferable to having the object's path described by several, different tracks.

CHAPTER 4. TRACK MANAGEMENT IMPROVEMENTS

Several modifications were made to the track management system of R-RANSAC. These improvements transform R-RANSAC from a multiple object filter into a multiple target tracker and aid in maintaining track continuity. Improvements to track management also tend to have a positive impact on the ratio of false positives. Section 4.1 describes the track labeling system used. Section 4.2 describes the changes made to the track merging criteria. Section 4.3 describes changes made to the RANSAC-based track initialization method.

4.1 Track Labeling System

The original version of R-RANSAC did not have a formal track labeling system. It outputs a vector of valid model states at each time step, but the order of vector elements changed from time step to time step as models gained and lost support. Thus, R-RANSAC was a multiple object filter and not a multiple target tracker. The introduction of the good model number system transformed R-RANSAC into an MTT algorithm; this section describes this labeling system.

R-RANSAC is well-suited to multiple target tracking because it propagates and updates individual, discrete tracks at each time step; this makes implementing a track labeling system straightforward. At the beginning of a tracking scenario, a good model counter is initialized to 0. This good model counter ensures that all labels are unique. When a new model is initialized, the good model number track variable (*GMN*) is initialized to zero. At the time step in which that track first meets the good model criteria, its good model number is set to the good model counter and the good model counter is incremented by one. That model retains the same good model number during its entire existence unless it is merged with another track, which might cause a good model number swap.

4.1.1 Good Model Criteria

The good model criteria determines when a hypothesis track becomes a good or valid track. The criteria should be sufficiently restrictive to reject spurious tracks, but not so restrictive as to reject true tracks. In addition to the original inlier ratio and lifetime requirements, two other constraints were added: a maximum velocity constraint and a consecutive missed detections constraint. The maximum velocity constraint rejects tracks that do not behave according to assumptions made about target dynamics. The consecutive missed detections constraint prevents false tracks with overly inflated inlier ratios from being included in the good model set. Both of these constraints were applied early in the development of R-RANSAC and have since been rendered obsolete, to a degree, by the changes to the track initialization method described in Section 4.3 and by the changes concerning how the consensus set is updated described in Section 5.1.3. Still, these constraints, along with the original constraints, have served as excellent debugging tools. As demonstrated in Chapter 9, the optimal good model criteria is largely dependent on how sensor processing is performed.

4.2 Track Merging Criteria

Perfect track continuity occurs when an object of interest is tracked by a track, or set of tracks, that have the same model number. For this to happen, an effective track merging criteria is necessary. R-RANSAC originally used the Mahalanobis distance between tracks as the merging criteria. Although the Mahalanobis distance provided a rigorous way of using the state estimate and error covariance in determining model similarity, it proved to be unintuitive and severely limited customizing the merging criteria for different scenarios. The merging criteria presented in Algorithm 1 allows the user to set intuitive similarity thresholds on the percent difference in target speeds, τ_v, target headings, τ_θ, and target positions, τ_x and τ_y. The $\tau_{x_{\min}}$ and $\tau_{y_{\min}}$ parameters are necessary to merge duplicate models tracking a stationary target. When tracking a stationary or nearly stationary target, the speed estimates of the target will be very small, but the percent different in speed and the difference in heading might be quite large; these minimum position difference thresholds account for this. The track merging criteria is as follows (line 9 of Algorithm 1). Tracks are merged if their percent difference in speed and their absolute differences in heading, x position, and y position meet their respective thresholds ($\tau_v = 0.25$, $\tau_\theta = 15, \tau_x = 30$, and $\tau_y = 30$). Tracks may also be merged if their absolute differences in x position and y position meet the $\tau_{x_{\min}} = 3$ and $\tau_{y_{\min}} = 3$ thresholds.

In Algorithm 1, *GMN* stands for good model number, $\wedge$ is the logical *and* symbol, and $\vee$ is the logical *or* symbol. Much of the logic in Algorithm 1 ensures that the correct track label persists during merging. In deciding which track survives, the first and most important criteria is inlier ratio. The second criteria is lifetime; if both tracks meet the inlier threshold, preference is given to the older track. Giving preference to the older track occasionally results in a track label switch (track fragmentation), but it performs as intended in the majority of merging situations.

Although this track merging heuristic adds several new parameters, which is somewhat undesirable, it allows the user to customize the merging criteria for the situation at hand. Tracking objects where object shadows are present is one example in which the ability to customize the merging criteria is important. In this situation, the object and its shadow should be tracked by a single track, not by multiple tracks. To compensate for the shadow, the user can relax the τ_x and τ_y constraints and tighten the τ_v and τ_θ constraints, working under the assumption that the object and its shadow should have essentially the same velocity vector. Another example in which customization is important is in tracking targets confined to a network, such as a road system. In this case, the speed and heading constraints could be relaxed and the τ_x and τ_y constraints could be adjusted to reflect anticipated target size.

4.3 Changes to Track Initialization

Two changes were made to the RANSAC-based track initialization method. The first change is a way of using assumed target dynamics to produce more probable hypothesis tracks. The second change helps preserve the inlier ratio as a true measure of a track's support.

Given assumed target dynamics and a known measurement history window length Nw, a guided sampling threshold can be used in track initialization. The RANSAC algorithm is designed to find the best model of data containing gross errors. If some gross errors can be identified and removed from the data, it only increases the probability that RANSAC find the correct model. With the guided sampling threshold, a randomly chosen measurement must fall within a certain radius of the outlier measurement currently under consideration. This is a logical change because a slowly

Algorithm 1 Track Merging Heuristic

```
1: Place all active tracks in a new array $\rightarrow$ *activeModels*. An active track is one with a positive lifetime value.
2: Initialize another array $\rightarrow$ *mergedModels*.
3: while length(*activeModels*) $> 0$ do
4:     Select the model with the highest inlier ratio $\rightarrow$ *highestRhoModel*.
5:     for each model in *activeModels* except for *highestRhoModel* do
6:         Calculate the percent difference in speed between the current model and *highestRhoModel* $\rightarrow$ *percentSpeedDiff*.
7:         Calculate the absolute difference in heading between the current model and *highestRhoModel* $\rightarrow$ *absoluteHeadingDiff*.
8:         Calculate the absolute difference in x and y positions between the current model and *highestRhoModel* $\rightarrow$ *absoluteXdiff* and *absoluteYdiff*, respectively.
9:         if the track merging criteria is satisfied then
10:            if *highestRhoModel*.$\rho > \tau_\rho \wedge$ (current model).$\rho > \tau_\rho$ then
11:                if *highestRhoModel*.T $>$ (current model).T then
12:                    if *highestRhoModel*.*GMN* $== 0$ then
13:                        *highestRhoModel*.*GMN* $=$ (current model).*GMN*.
14:                    end if
15:                    Remove current model from *activeModels*.
16:                else
17:                    if (current model).*GMN* $== 0$ then
18:                        (current model).*GMN* $=$ *highestRhoModel*.*GMN*.
19:                    end if
20:                    Remove *highestRhoModel* from *activeModels*. Current model becomes *highestRhoModel*.
21:                end if
22:            else if *highestRhoModel*.$\rho > \tau_\rho \wedge$ (current model).$\rho < \tau_\rho$ then
23:                if *highestRhoModel*.*GMN* $== 0$ then
24:                    *highestRhoModel*.*GMN* $=$ (current model).*GMN*.
25:                end if
26:                Remove current model from *activeModels*.
27:            else
28:                if *highestRhoModel*.*GMN* $== 0$ then
29:                    *highestRhoModel*.*GMN* $=$ (current model).*GMN*.
30:                end if
31:                Remove current model from *activeModels*.
32:            end if
33:        end if
34:        Place *highestRhoModel* in *mergedModels*.
35:        Remove *highestRhoModel* from *activeModels*.
36:    end for
37: end while
```

moving target cannot have produced a measurement on the other side of the field of view within the most recent Nw time steps. This threshold reduces the number of hypothesis tracks generated because a measurement inside the sampling radius must be randomly selected to produce a new model. Because fewer tracks are being initialized, the ratio of false positives is improved at the slight expense of the ratio of missed detections. The threshold also inhibits the initialization of very high velocity tracks which reduces the number of false positive tracks without a negative effect on any other metric.

In the track initialization method of the original version of R-RANSAC, all inlier measurements are used to update the state estimate and consensus set of the new model as it is propagated forward to the current time step. In tracking scenarios with a high number of clutter measurements, with objects of interest that produce multiple measurements per time step, or with closely spaced targets, tracks are often initialized with inlier ratios greater than one. To normalize the inlier ratio and preserve it as a true measure of a track's support, the consensus set is only updated with a single measurement per time step as the model is propagated forward to the current time step. Which measurement is used for the update is inconsequential; the first measurement per time step is an easy-to-implement choice.

As will become more clear in Sections 5.2.2 and 5.2.3, an obvious potential change to the track initialization method would be to propagate the new, hypothesis track to the current time step using a more sophisticated data association method such as the probabilistic data association filter. This change was not explored because the current approach is efficient and its performance satisfactory. Another potential change would be to initialize tracks with the nearly constant jerk (CJ) or nearly constant acceleration (CA) motion models instead of the nearly constant velocity (CV) motion model. This change would allow tracks to be initialized for continuously maneuvering targets, but also has a significant disadvantage. When randomly selecting measurement subsets, all measurements need to be generated by the same target that generated the outlier measurement under consideration. The CV model only requires one target-generated measurement to be chosen whereas the CA and CJ models require two and three target-generated measurements to be chosen, respectively. In practice, this drastically reduces the likelihood of generating a track that correctly models an object of interest. Initializing tracks with the CV model does require that the target has constant velocity benign behavior and that it spends a majority of time in this mode. Fortunately, both of these assumptions are valid for the targets considered in this research. Consequently, the CV-based track initialization method is retained even when higher-order motion models are used for the filtering block. In these cases, the higher-order terms of the state estimate and error covariance are initialized to zero.

CHAPTER 5. DATA ASSOCIATION TECHNIQUES

Data association, the task of assigning measurements to tracks, is one of the essential blocks of a MTT algorithm. Although somewhat trivial in situations with easily-detectable and widely-spaced targets and low clutter, data association becomes one of the most difficult tasks associated with MTT in more complicated tracking scenarios. The modularity of R-RANSAC allows for the inclusion of practically any data association method, several of which are highlighted in Section 5.1. A comprehensive comparison of the different data association techniques is presented in Section 5.2.

5.1 Data Association Techniques

Four widely used data association techniques were incorporated into the R-RANSAC framework. They include the all neighbors (Section 5.1.1), nearest neighbor (Section 5.1.2), probabilistic data association (Section 5.1.3), and joint probabilistic data association (Section 5.1.4) methods. As mentioned in Section 2.2.4, the only requirement that R-RANSAC imposes on the data association method is that every incoming measurement be classified as an inlier or outlier. Instead of using an ℓ_1-norm based gate that results in a rectangular inlier region, an ℓ_2-norm based gate is used. The size of this gate is determined by the assumed measurement noise covariance. This results in an elliptical inlier region, or a circular inlier region if the measurement noise covariance is diagonal.

5.1.1 All Neighbors

The original version of R-RANSAC presented in Chapter 2 uses a gated all neighbors (AN) approach to perform data association. In this method, each of the validated measurements (measurements that fall within a track's inlier region) are used to update the track via the update step of the Kalman filter. The update step is performed for each validated measurement; the measurements are not combined into an average residual. The track's consensus set and inlier ratio are also updated by all validated measurements. When working with video data, where multiple detections of the same object are common (i.e. a person's torso and other appendages are detected separately), this leads to inlier ratios greater than one. This has the unintended consequence that larger targets often appear to have more support than smaller targets simply because their size leads to fragmented segmentation.

Although the AN approach is the simplest method, it is arguably the most consistent with the idea of R-RANSAC as a dynamic extension of RANSAC. Recall that when the iterations of RANSAC have been completed, the final model is computed by performing a least-squares regression using the entire consensus set. Similarly, with AN data association, R-RANSAC tracks are updated with each, fully-weighted measurement.

5.1.2 Nearest Neighbor

The nearest neighbor (NN) data association method integrated into the R-RANSAC framework is a gated, local version of the method as opposed to the popular global nearest neighbor (GNN) method. In this version of NN, the validated measurement with the smallest residual to the track's position is the sole measurement used in the update step of the Kalman filter; this is also the only measurement that updates the consensus set and inlier ratio. Using only the nearest neighbor measurement to update the consensus set prevents the inlier ratio from becoming overly inflated. Thus, the inlier ratio is preserved as a true measure of a track's support.

In GNN, the measurement-to-track assignment is performed using a Munkres- or Hungarian-type of algorithm that minimizes total measurement-to-track difference. The GNN method was not explored with respect to R-RANSAC.

5.1.3 Probabilistic Data Association Filter

The probabilistic data association (PDA) filter calculates the measurement association probability for each validated measurement based on a track's state estimate and error covariance. The validated measurements are then combined into a weighted residual which is used to update the

Kalman filter. The PDA filter also uses information about the probability of detection, gate probability, and the Poisson parameter describing the clutter distribution when updating the state estimate and covariance [24]. The PDA filter assumes each measurement originates from a single target and that a target can only produce one measurement. With video tracking, it should be noted that both of these assumptions occasionally break down. The association probabilities for the i^{th} validated measurement are given by

$$\beta_t^i = \begin{cases} \frac{L_t^i}{1-P_D P_G+\sum_{j=1}^{|\mathscr{M}_t^*|} L_t^j}, & \text{if } i = 1, ..., |\mathscr{M}_t^*|. \\ \frac{1-P_D P_G}{1-P_D P_G+\sum_{j=1}^{|\mathscr{M}_t^*|} L_t^j}, & \text{if } i = 0. \end{cases} \tag{5.1}$$

where P_D is the probability of detection, P_G is the gate probability and L_t^i is the likelihood ratio and is given by

$$L_t^i = \frac{N[z_t^i; \hat{z}_{t|t-1}, S_t]\, P_D}{\lambda} \tag{5.2}$$

where λ is the spatial density of the assumed Poisson clutter model, N represents a Gaussian probability density function (pdf), $\hat{z}_{t|t-1} = C\mathbf{x}_{t|t-1}$, and S_t is the covariance of the innovation and is given by

$$S_t = C\mathbf{P}_{t|t-1}C^\top + R. \tag{5.3}$$

The state estimate is given by

$$\mathbf{x}_t = \mathbf{x}_{t|t-1} + W_t\, \nu_t \tag{5.4}$$

where ν_t is the combined residual and is given by

$$\nu_t = \sum_{j=1}^{|\mathscr{M}_t^*|} \beta_t^j\, \nu_t^j, \tag{5.5}$$

where ν_t^j is an individual residual and is given by $\nu_t^j = z_t^j - \hat{z}_{t|t-1}$ and W_t is given by $W_t = \mathbf{P}_{t|t-1} C^\top S_t^{-1}$. The error covariance is given by

$$\mathbf{P}_t = \beta_t^0 \mathbf{P}_{t|t-1} + [1 - \beta_t^0] P_t^c + \tilde{P}_t \tag{5.6}$$

where $P_t^c = P_{t|t-1} - W_t S_t W_t^\top$, and $\tilde{P}_t$ is given by

$$\tilde{P}_t = W_t \, [\sum_{j=1}^{|\mathscr{M}_t^*|} \beta_t^j \, \nu_t^j \, (\nu_t^j)^\top - \nu_t \, \nu_t^\top] \, W_t^\top. \tag{5.7}$$

The equations presented above were taken from [30]. The PDA filter traditionally uses a covariance-based validation region, however, because R-RANSAC initializes a new hypothesis track with every outlier measurement, a static validation region is preferred. A covariance-based validation region causes the initialization of many redundant tracks as the covariance shrinks. In employing the PDA filter in the R-RANSAC framework, only the measurement with the highest measurement association probability is used to update the inlier ratio and consensus set.

In Section 5.1.1, it was noted that AN data association provides the closest analog to traditional RANSAC. The RANSAC analog of the PDA filter would use a weighted least-squares regression as opposed to an un-weighted regression. When the final, smoothed model is computed, members of the consensus set would be weighted according to their residual with the original model, i.e. the model estimated with the minimum subset of data.

The likelihood ratio of the PDA filter can be used to create a simpler weighted residual without taking into account detection and gate probabilities or clutter distribution. Furthermore, if position and velocity measurements are received, the likelihood ratio can indicate the nearest neighbor measurement in a probabilistic sense.

5.1.4 Joint Probabilistic Data Association Filter

The joint probabilistic data association (JPDA) filter calculates the measurement association probabilities *jointly* across all targets. The JPDA filter makes the same assumptions about the nature of the measurements as does the PDA filter. A validation matrix is constructed which helps to delineate all of the association hypotheses [6]. One known weakness of the PDA filter is its tendency to coalesce closely spaced tracks; the JPDA filter was designed in part to overcome this tendency. The JPDA filter was implemented using starter code found `http://www.control.isy.liu.se/student/graduate/TargetTracking/`. The consensus set and inlier ratio are only updated with the nearest neighbor measurement.

5.2 Comparison of Data Association Techniques

A comprehensive comparison of the four data association methods described in Section 5.1 was conducted using the vehicle video. This comparison used Matlab implementations of R-RANSAC. Two dynamic models were used in this comparison, the nearly constant jerk (CJ) model and the nearly constant velocity (CV) model; dynamic models are discussed in greater detail in Chapter 6. The R-RANSAC parameters used were $\mathrm{N_w} = 25$, $m = 30$, $\tau_\rho = 0.5$, $\tau_\mathrm{T} = 3$, $\tau_\mathrm{R} = 30$, $\ell = 40$, and $\tau_\mathrm{CMD} = 5$. The merging parameters used were $\tau_\theta = 25°$, $\tau_v = 0.2$, and $\tau_x = \tau_y = 35$. The PDA filter parameters used were $P_D = 0.80$, $P_G = 0.95$, and $\lambda = 10^{-2}$.

5.2.1 Numerical Results

Table 5.1 contains results for the MOTP and MOTA metrics which were taken from one experimental run. MOTP scores among the R-RANSAC implementations are all similar, with CJ-PDA R-RANSAC having the best score. The different implementations all have very similar ratios of missed detections. More discrepancy exists in the ratios of mismatches. NN data association is known to diverge in scenarios with noisy measurements and missed detections, a tendency illustrated by its higher ratio of mismatches. CV-AN R-RANSAC has the highest ratio of mismatches because it consistently experiences a track label switch during target maneuvers (tracking maneuverable targets is treated further in Chapter 6). There is also considerable range in the ratio of false positives, the best score being achieved by CJ-JPDA R-RANSAC.

Table 5.2 contains results for the OSPA metrics. These results are averaged over 10 experimental runs. The small difference between OSPA and OSPA-T scores indicates that R-RANSAC generally maintains track continuity very well. The OSPA scores show that the CJ-JPDA and CJ-PDA R-RANSAC implementations perform better than the other R-RANSAC implementations.

Table 5.1: MOT Results

	MOTP	MOTA	MD	FP	MM
RR CJ-JPDA	21.8574	0.8617	0.1063	0.0173	0.0147
RR CJ-PDA	17.7733	0.8594	0.1046	0.0211	0.0149
RR CJ-NN	25.3310	0.8437	0.1022	0.0367	0.0174
RR CJ-AN	23.1678	0.8509	0.1018	0.0308	0.0166
RR CV-AN	23.6848	0.8413	0.1185	0.0225	0.0177

Table 5.2: Average OSPA Scores

	OSPA	OSPA-T
RR CJ-JPDA	40.3093	41.7940
RR CJ-PDA	40.1244	41.3469
RR CJ-NN	41.0989	42.3409
RR CJ-AN	40.4496	42.0448
RR CV-AN	42.7384	44.2893

Table 5.3 reports the execution times per frame. These times are only for the trackers and do not include the accompanying video processing times. The execution times are all very similar, except for the JPDA implementation which takes about twice as long to run.

5.2.2 Track Examples

This section contains several examples of tracks produced by the various implementations of R-RANSAC. The tracks highlighted in this section are taken from the 600 frame annotated sequence of the vehicle video.

Figures 5.1 and 5.2 show the R-RANSAC tracks associated with the fifth target in the annotated video sequence for the CJ-NN and CJ-PDA implementations, respectively. The fifth target presents a difficult tracking challenge because, as it approaches the upper-middle intersection, it simultaneously crosses paths with two other targets and becomes occluded by a shadow. With NN data association, the track is updated by and converges to the measurements originating from the crossing targets. When the fifth target emerges from the occlusion and resumes being detected, its associated measurements are not used to update the original track because they are no longer the nearest neighbor measurements. In the CJ-PDA implementation, the post-occlusion measurements

Table 5.3: Execution Times - Seconds Per Frame

CJ-JPDA	CJ-PDA	CJ-NN	CJ-AN	CV-AN
0.0329	0.0188	0.0180	0.0169	0.0171

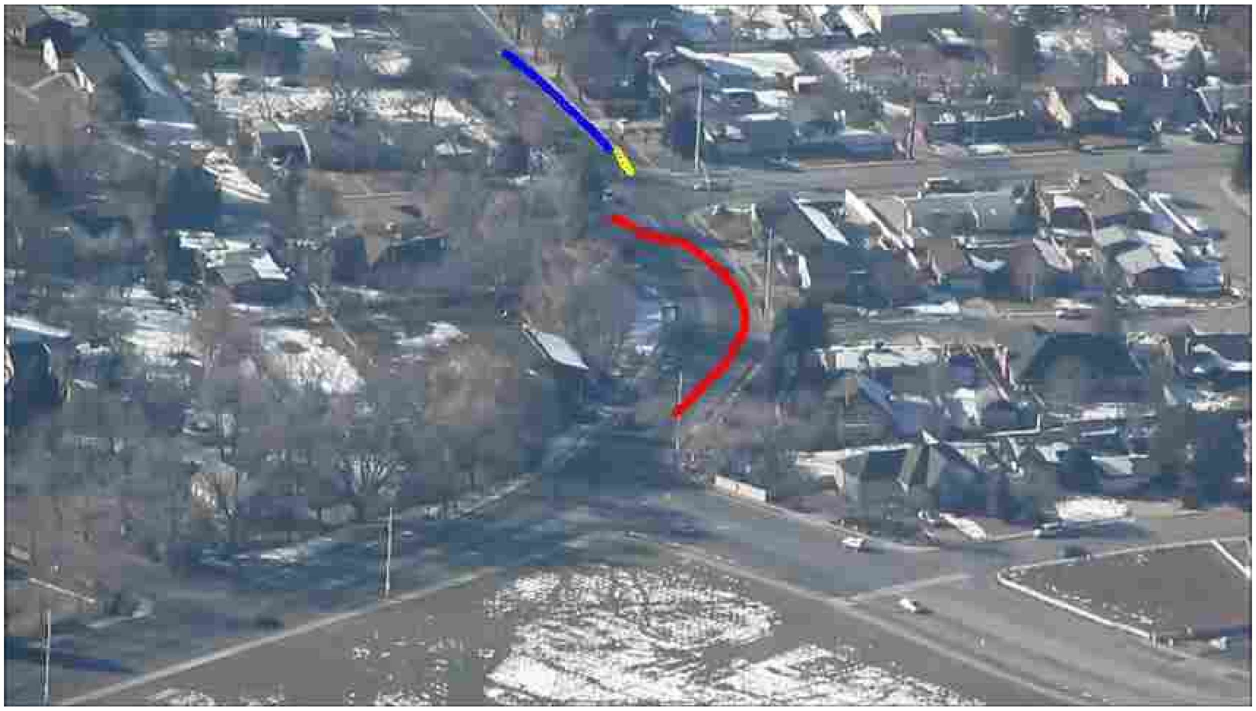


Figure 5.1: Three tracks produced by R-RANSAC with the nearly constant jerk model and nearest neighbor data association. The target of interest is briefly lost during the occluded intersection and then reacquired shortly thereafter.

are given sufficient weight to pull the original track back onto the fifth target's true trajectory. The CV-AN implementation also successfully tracks the fifth target in this situation because it uses all validated measurements to update the Kalman filter and because the dynamics of the CV model help prevent the track from converging to the incorrectly associated measurements.

Figure 5.3 displays two advantages of the PDA filter approach over the AN approach. First, the PDA filter is able to maintain track continuity even in the presence of occluding shadows and other nearby targets. On the other hand, the AN track is unable to maintain track continuity and jumps from one target to another. Second, the measurement association probabilities calculated by the PDA filter are able to distinguish between correctly and incorrectly associated validated measurements, whereas AN data association has no mechanism to do this. In Figure 5.3, immediately

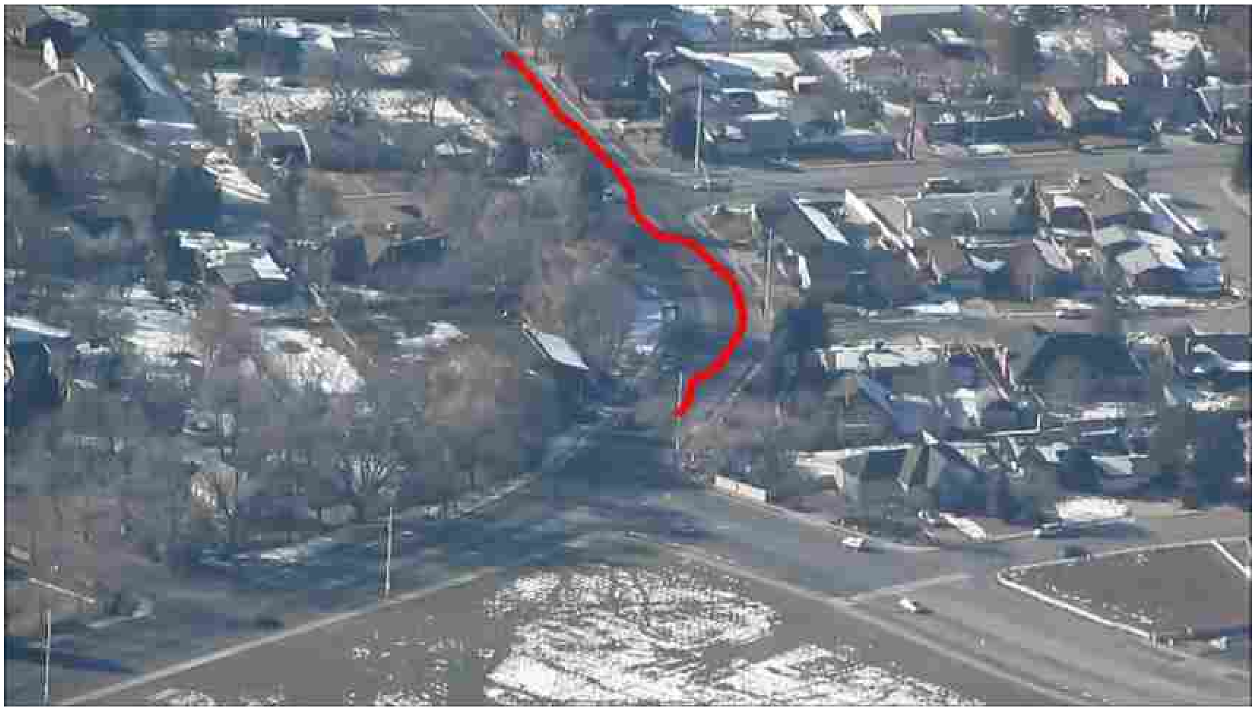


Figure 5.2: A single track produced by R-RANSAC with the nearly constant jerk model and probabilistic data association. The PDA filter provides robust tracking through the occluded intersection as evidenced by the single, continuous track.

after the vehicle makes the U-turn, it passes by two other vehicles. The measurements from the other vehicles fall within the inlier region of the U-turn vehicle. With AN data association, this causes the U-turn track to be "pulled" towards the other tracks, resulting in less tracking precision. This "pulling" effect may also lead to track fragmentation if the first track converges onto the second track; this event is very likely if the first target is not detected for one or more time steps during the track interaction.

5.2.3 Conclusions

R-RANSAC has been shown to be compatible with numerous data association techniques including NN, PDA, and JPDA. Although the JDPA filter was specifically designed to overcome some of the deficiencies of the PDA filter, CJ-PDA R-RANSAC achieved higher OSPA and OSPA-T scores than CJ-JPDA R-RANSAC. This shows that there exists a coupling between the RANSAC track initialization method of R-RANSAC and the data association method utilized. R-RANSAC is inherently subject to duplicate tracks because every outlier measurement generates a new track;

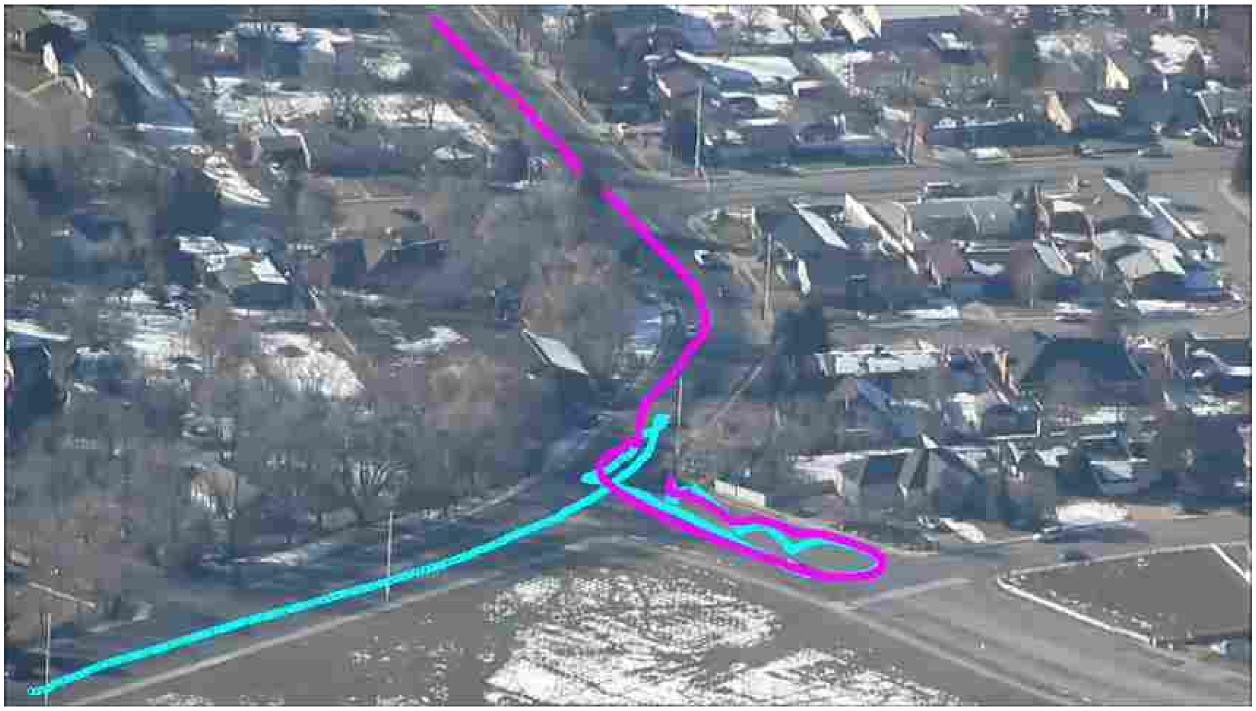


Figure 5.3: The CJ-PDA R-RANSAC is drawn in magenta and the CJ-AN R-RANSAC track is drawn in cyan.

outlier measurements located just outside of an existing track's validation region will likely generate redundant tracks. With the JPDA filter, both an existing and a newly-generated (possibly) redundant track will be updated by only partially weighted measurements. This lengthens the time needed for these tracks to meet the merging criteria. In essence, R-RANSAC relies on a certain degree of track coalescence to limit the number of false positive tracks. The CJ-PDA implementation of R-RANSAC is shown to be the best balance between performance and computational efficiency.

CHAPTER 6. TRACKING HIGHLY MANEUVERABLE TARGETS

This chapter describes improvements made to R-RANSAC to enhance its ability to track highly maneuverable targets. Section 6.1 discusses higher-order linear models that can be used with R-RANSAC. Section 6.2 reviews the interacting multiple models (IMM) algorithm and discusses its integration into the R-RANSAC framework. Finally, Section 6.3 provides concluding remarks about how to best track maneuvering targets with R-RANSAC.

6.1 Higher-Order Linear Models

The original version of R-RANSAC uses a nearly constant velocity motion model (hereafter referred to as the CV model) where the state transition matrix A in Eqns. (2.1) and (2.2) is given by

$$A_{\text{CV}} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.1}$$

where dt is the time step[1]. The process noise covariance Q accounts for variations in the velocity. This motion model is used both to initialize tracks and as the state transition matrix for the Kalman filter. Early experiments with R-RANSAC found that this model was well-suited to the task of initializing tracks, but was incapable of tracking maneuverable objects (see Figure 6.1). Instead of replacing the CV model with a non-linear model (and the Kalman filter with an extended Kalman filter) to increase R-RANSAC's ability to track maneuvering objects, two other options were explored: higher-order linear models and the IMM algorithm. These two options were favored over a non-linear model because they allow tracks to continue to be initialized by the CV model, whereas

[1]In video tracking, dt is the inverse of the video frame rate

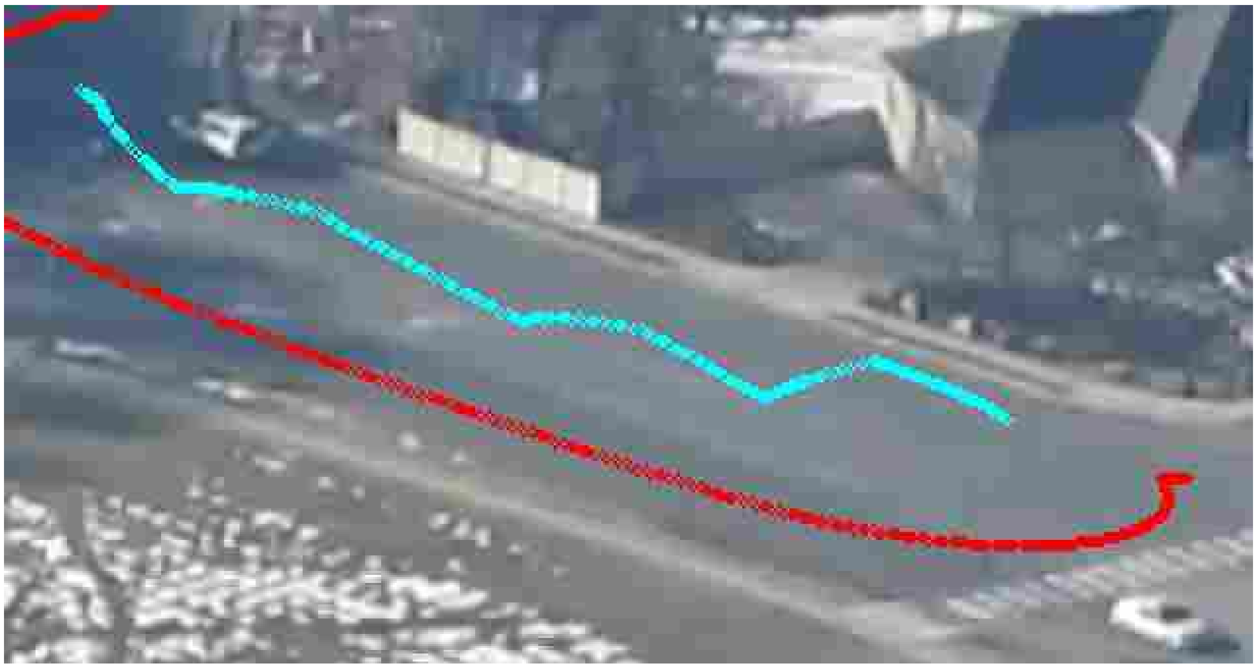


Figure 6.1: The R-RANSAC tracks associated with a car performing a U-turn. This version of R-RANSAC used the nearly constant velocity motion model.

a non-linear model combined with an EKF would have required a fundamental reworking of the RANSAC-based track initialization algorithm.

The nearly constant acceleration model and the nearly constant jerk model (referred to throughout as the CA and CJ models, respectively) were considered to replace the CV model. The state transition matrix A of the CA model is given by

$$A_{\text{CA}} = \begin{bmatrix} 1 & 0 & dt & 0 & \frac{dt^2}{2} & 0 \\ 0 & 1 & 0 & dt & 0 & \frac{dt^2}{2} \\ 0 & 0 & 1 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & dt \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.2}$$

and A of the CJ model is given by

$$A_{\text{CJ}} = \begin{bmatrix} 1 & 0 & dt & 0 & \frac{dt^2}{2} & 0 & \frac{dt^3}{3} & 0 \\ 0 & 1 & 0 & dt & 0 & \frac{dt^2}{2} & 0 & \frac{dt^3}{3} \\ 0 & 0 & 1 & 0 & dt & 0 & \frac{dt^2}{2} & 0 \\ 0 & 0 & 0 & 1 & 0 & dt & 0 & \frac{dt^2}{2} \\ 0 & 0 & 0 & 0 & 1 & 0 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.3}$$

These models, like the CV model, include process noise to account for variations in target dynamics. A comprehensive discussion of the CJ model, including its application to tracking highly maneuverable targets, can be found in [31]. Figures 6.2 and 6.3 show the increased ability to track maneuvering objects by replacing the CV model with the CJ model in the R-RANSAC framework.

6.2 Interacting Multiple Models

Multiple model (MM) algorithms are a useful tool for tracking targets that deviate from a single motion model. A simple example is that of tracking a vehicle constrained to an orthogonal road network. This vehicle exhibits acceleration, deceleration, constant speed, and turning behaviors, all of which would be difficult to characterize with a single model. MM algorithms use several motion models to characterize the various behavior modes. Figure 6.4 displays the general layout of an MM algorithm. The central feature of MM algorithms is a bank of m Kalman filters. At every time step, each filter is predicted forward and updated. The outputs of the individual filters are assigned a posterior probability which is used to combine the filter outputs into a fused state estimate and error covariance. Model switching is usually governed with a Markov transition matrix. The IMM algorithm is unique in that it stores the individual filter outputs from the previous time step and uses them to create “mixed estimates” which serve to re-initialize the Kalman filters

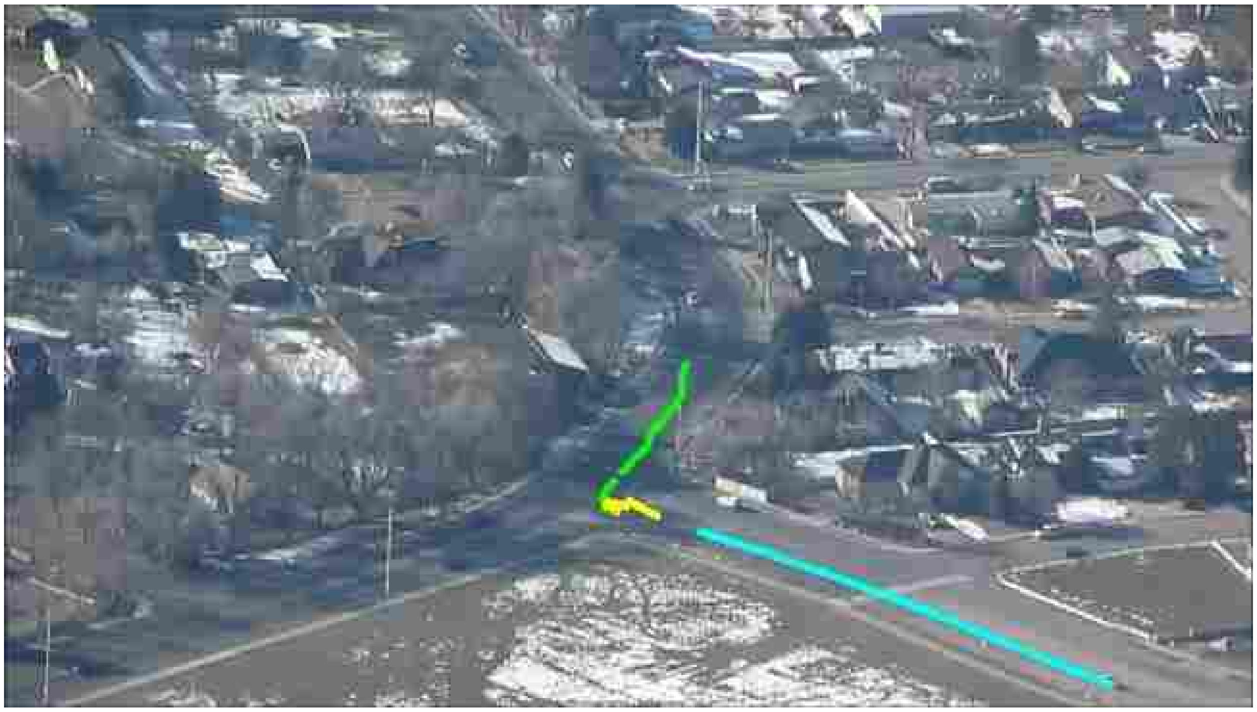


Figure 6.2: The R-RANSAC tracks associated with a car negotiating a left hand turn. This version of R-RANSAC uses the nearly constant velocity motion model and requires two new track initializations to complete the turn.

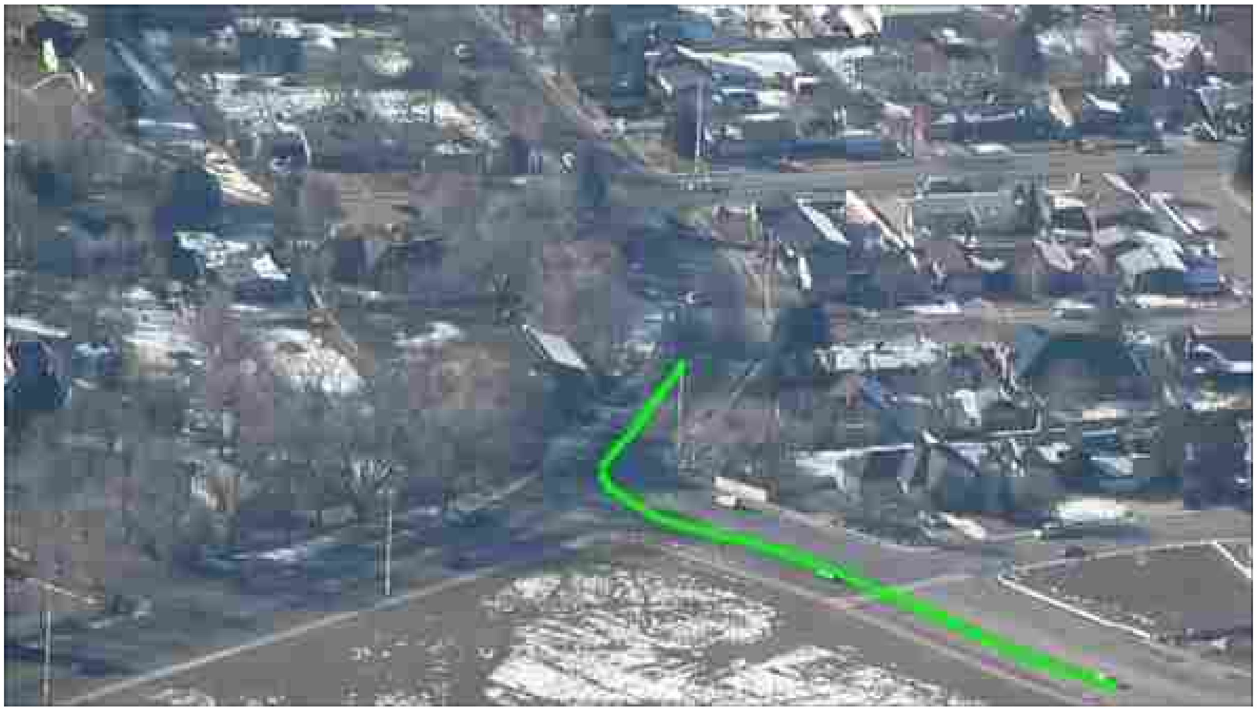


Figure 6.3: The R-RANSAC tracks associated with a car negotiating a moderate left hand turn. This version of R-RANSAC uses the nearly constant jerk motion model and completes the turn with a single track.

at the next time step. In the IMM, the filter outputs are fused according to

$$\mathbf{x}_t = \sum_{i=1}^{m} \mathbf{x}_t^i \, \mu_t^i \tag{6.4}$$

$$\mathbf{P}_t = \sum_{i=1}^{m} [\mathbf{P}_t^i + (\mathbf{x}_t - \mathbf{x}_t^i)\,(\mathbf{x}_t - \mathbf{x}_t^i)^\top]\,\mu_t^i \tag{6.5}$$

where $\mathbf{x}_t^i$ and $\mathbf{P}_t^i$ are the individual filter outputs and μ_t^i are posterior probabilities assigned to each model and are given by

$$\mu_t^i = \frac{\mu_{t-1}^i \, L_t^i}{\sum_{j=1}^m \mu_{t-1}^j \, L_t^j} \tag{6.6}$$

where L_t^i is the likelihood ratio given in Equation (5.2). The "mixed estimates" $\hat{\mathbf{x}}_{t-1}$ and $\hat{\mathbf{P}}_{t-1}$ are given by

$$\hat{\mathbf{x}}_{t-1} = \sum_{j=1}^{m} \mathbf{x}_{t-1}^j \, \mu_t^{j|i} \tag{6.7}$$

$$\hat{\mathbf{P}}_{t-1} = \sum_{j=1}^{m} [\mathbf{P}_{t-1}^j + (\hat{\mathbf{x}}_{t-1}^i - \mathbf{x}_{t-1}^j)\,(\hat{\mathbf{x}}_{t-1}^i - \mathbf{x}_{t-1}^j)^\top]\,\mu_{t-1}^{j|i} \tag{6.8}$$

where $\mu_{t-1}^{j|i} = \frac{\pi_{ji}\,\mu_{t-1}^j}{\mu_{t|t-1}^i}$, $\mu_{t|t-1}^i = \sum_{j=1}^m \pi_{ji}\,\mu_{t-1}^j$, and π_{ji} is the (j,i) element of the Markov transition matrix π.

Other well-known MM algorithms include the autonomous multiple models (AMM) and the first- and second-order generalized pseudo-Bayesian filters (GPB-1 and GPB-2). In the AMM, the filters are initialized with their own output from the previous time step. In GPB-1, the filters are initialized with the fused output from the previous time step. In GPB-2, the filters are initialized with each of the m individual outputs from the previous time step, resulting in m^2 filtering operations.

Three different IMM algorithms were implemented. The first implementation used three CV models, but with varying levels of process noise: medium, the same level of process noise used

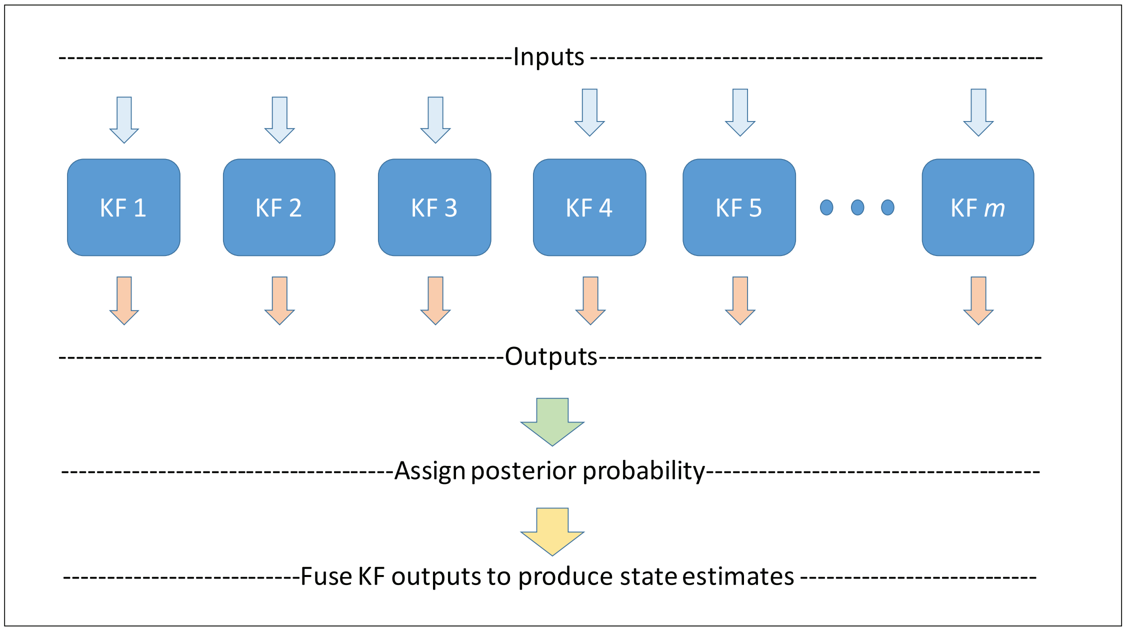

Inputs
KF 1
KF 2
KF 3
KF 4
KF 5
KF m
Outputs
Assign posterior probability
Fuse KF outputs to produce state estimates

Figure 6.4: The general layout of a multiple models algorithm.

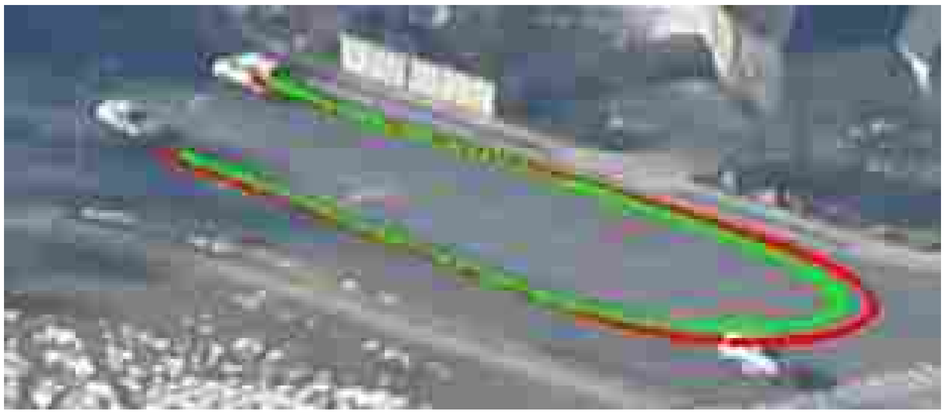
in the standard CV model; low, $1/100$ of the medium level process noise; and high, 100 times the medium level noise. The two other IMM implementations employed the CA and CJ models in the same way. Reference [32] conducts a comparison between IMM and several other popular MM algorithms and concludes that IMM is a good balance between tracking accuracy, computational complexity, and robustness to model mismatch. The IMM equations presented above are taken from [32]. Reference [32] uses a total of 9 models for its model set: a CV model along with four constant tangential acceleration models and four constant turn models. A contrasting approach is presented in reference [30] which suggests using only three models: one characterizing the target's benign behavior, one characterizing the maneuver behavior, and one characterizing the onset of a maneuver. Simulations run with both approaches showed that the approach in [30] performs better and, because it uses fewer models, was more computationally efficient. Therefore, the IMM implemented in this paper follows the three model approach, with the low, medium, and high process noise models corresponding to the benign, maneuvering, and onset of maneuvering behavior modes. Furthermore, each behavior mode is realized with a PDAF instead of a Kalman filter, resulting in an IMMPDAF.

Table 6.1: Magnitude in Changes in IMM Mode Weighting

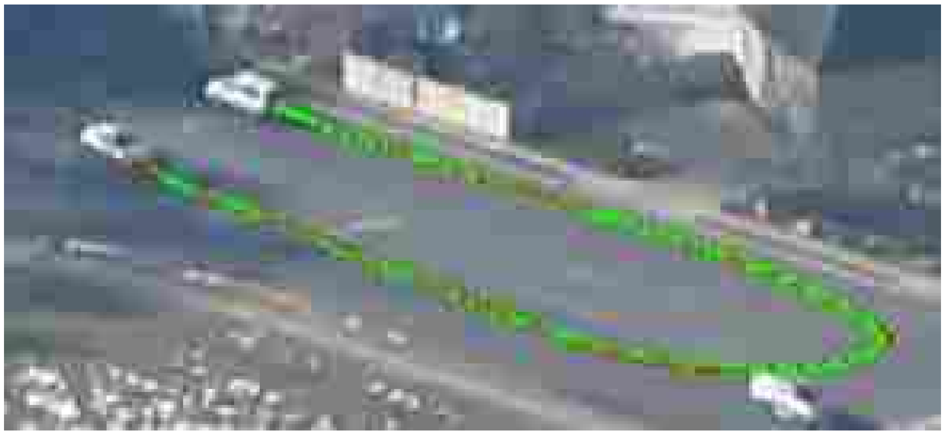
	CJ-IMM	CA-IMM	CV-IMM
Low process noise	5.3014e-07	6.6692e-07	4.2828e-05
Medium process noise	3.3619e-07	4.2933e-07	2.4902e-05
High process noise	1.9428e-07	2.3792e-07	1.7926e-05

Figures 6.5 and 6.6 illustrate the performance of the CV, CA, and CJ IMM filters. Figure 6.5 shows the path of a vehicle undergoing a U-turn. Figure 6.6 plots the weighting of the three different modes while the vehicle maneuvers for the CV-based IMM case. While the vehicle is in its benign motion mode, the weights settle to their steady-state values. During the maneuver, the low process noise mode is weighted less heavily and more weight is transferred to the medium and high process noise modes. The weights return to their steady-state values after the maneuver. It is also interesting to note the changes in IMM mode weighting among the three different IMM filters. Table 6.1 displays the magnitude of the weighting change for each mode in each IMM during the maneuver shown in Figure 6.5. As expected, the CV-based IMM must transfer more weight (about two orders of magnitude more) to its medium and high process noise modes in order to track the vehicle than the other IMMs do. Because the CA and CJ models are better approximations of the vehicle behavior during this maneuver, the CA- and CJ-based IMMs rely more on their underlying motion models to track the maneuvering vehicle than the medium and high process noise modes.

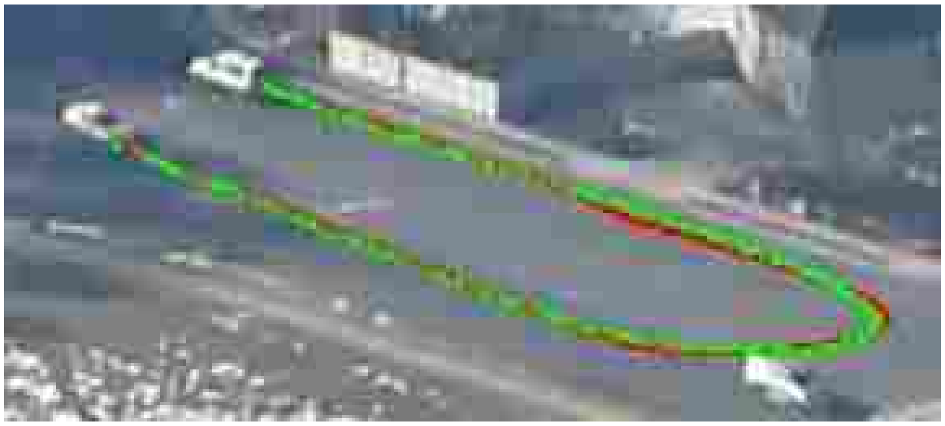
Figure 6.7 provides a simulated comparison between the CV, CA, and CJ models and their IMM counterparts. In this figure, the RMS error of the estimated path is plotted (see Figure 6.8 for the path). It is shown that a three model IMM using an n-order linear model exhibits a level of tracking performance between that of an $n+1$-order linear model and an $n+2$-order linear model. For example, the performance of the CV-based IMM falls in between that of the CA and CJ models. The CA- and CJ-based IMM algorithms showed acceptable tracking error throughout the duration of the path; both were able to track the true path during the benign and maneuvering sections. The CJ-based IMM algorithm was integrated into R-RANSAC and was extensively compared with the Gaussian mixtures probability hypothesis density filter (see Chapter 7.



(a) CV-based IMM



(b) CA-based IMM



(c) CJ-based IMM

Figure 6.5: Resulting tracks (red) and the associated measurements (green) for the 3 interacting multiple models implementations. The CV-based IMM struggles to track the vehicle during the U-turn (evidenced by the track overshoot) whereas the CA- and CJ-based IMMs prove more successful.

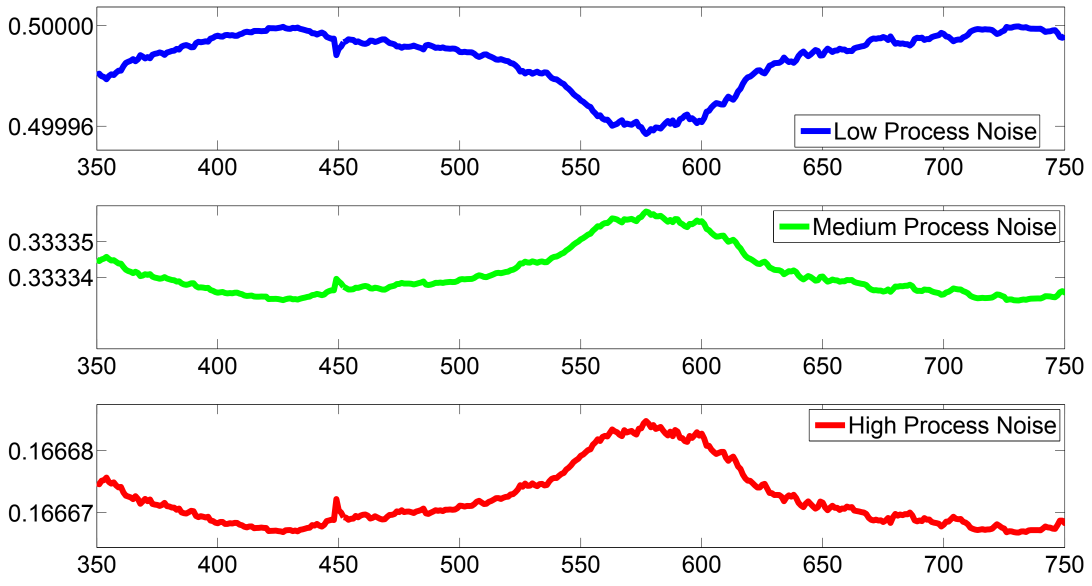


Figure 6.6: The weights given to the low, medium, and high process noise modes during the maneuver shown in Figure 6.5a for the CV-IMM implementation of R-RANSAC. The horizontal axis is the time axis. The change in weighting directly corresponds to the vehicle's maneuver.

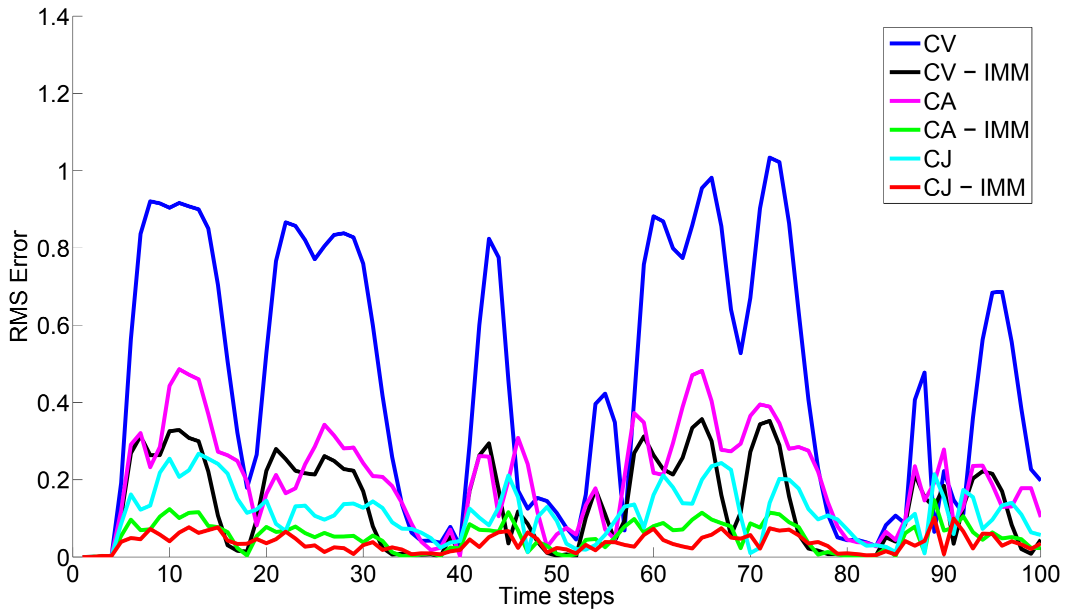


Figure 6.7: The RMS error over a randomly generated path consisting of 100 time steps for the different motion models and IMM implementations.

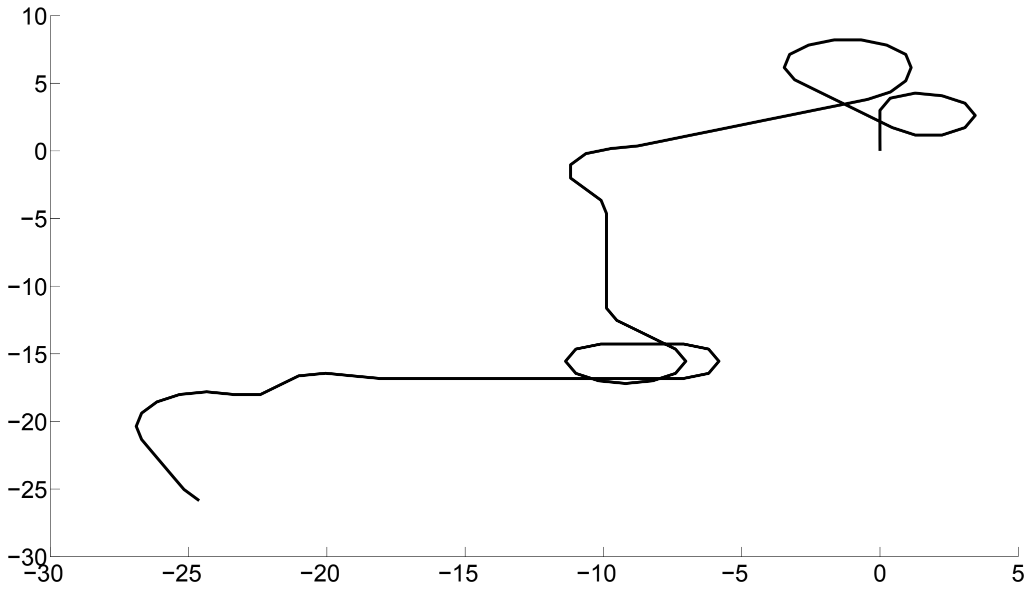


Figure 6.8: The randomly generated path used to produce the RMS error results of Figure 6.7. The path begins at $(0,0)$. The spikes in RMS errors, seen most clearly in the CV track, correspond to the more extreme maneuvers in the simulated path.

6.3 Conclusions

R-RANSAC has been shown to be compatible with different linear motion models and MM algorithms without requiring modifications to the track initialization or track management blocks. However, the IMM is not fully utilized in the R-RANSAC framework, as evidenced by the small variation in mode weights reported in Table 6.1 and plotted in Figure 6.6. For a greater degree of model switching to occur, the measurement noise covariance matrix R and process noise covariance matrix Q need to be more restrictive. However, this would negatively effect two other parts of the algorithm. First, R-RANSAC depends on smooth tracks in order to accurately predict target location at future time steps. A more restrictive R matrix would result in more jittery tracks and less accurate predictions. Second, the performance of the PDA filter degrades significantly as the values of the R matrix are reduced. With the current R matrix, the performance of the CJ-IMMPDAF R-RANSAC can be replicated by a CJ-PDAF R-RANSAC with a process noise matrix set to an approximate weighted average of the IMM process noise matrices. The IMM is also accompanied by a substantial increase in computational complexity; the increase in execution time is roughly proportional to the number of additional filters employed.

In light of these observations, development of the IMM R-RANSAC has been discontinued. A single, properly-tuned CJ model is capable of tracking most objects of interest viewed from a UAV platform. Additionally, it is a fast and elegant solution to the problem of tracking highly maneuverable objects. One potential use of the IMM in the R-RANSAC framework is in the identification of different behavior states or as an indication of when a target begins to deviate from its benign behavior. Figure 6.6 shows that the IMM mode weighting could be used to clearly indicate the onset and completion of the U-turn.

CHAPTER 7. COMPARISON WITH THE GAUSSIAN MIXTURE PROBABILITY HYPOTHESIS DENSITY FILTER

A comprehensive comparison was conducted between the CJ-IMMPDAF R-RANSAC implementation and the Gaussian mixture probability hypothesis density (GM-PHD) filter. The implementation of the GM-PHD filter is described in Section 7.1 and numerical results are presented in Section 7.2.

7.1 Implementation of the GM-PHD Filter

The implementation of the GM-PHD filter was performed by Dr. Peter C. Niedfeldt. A linear jump Markov model GM-PHD filter was implemented to compare the performance of R-RANSAC with a state-of-the-art alternative. The GM-PHD filter implementation closely follows the algorithm in [10, Table I]. The only difference is that the track labeling capability suggested in [33] is also incorporated, where each new Gaussian mode is assigned a label which is passed to its children modes from time step to time step. As with the R-RANSAC merging criteria, merged Gaussian modes keep the label of the mode with the longest lifetime to maximize track continuity. The linear jump Markov model GM-PHD filter was introduced in [34].

To make the comparison as valid as possible, the parameters used in this implementation of the GM-PHD filter parallel those of the CJ-IMMPDAF R-RANSAC implementation. The IMM parameters of motion models, noise covariance matrices, and Markov transition matrix used in the CJ-IMMPDAF R-RANSAC are carried over here. No assumptions are made about the birth distribution on the targets; instead, measurements from the previous time steps are used to seed the target birth models. This allows for rapid tracking of new targets at the expense of increased computational complexity, which is proportional to the number of measurements squared [12]. The probability of survival is 0.999, $P_D = 0.8$, and the clutter density is set to 5×10^{-6}. The Mahalanobis distance is used as the merging criteria; this threshold is set at 6. The pruning threshold is equal to 10×10^{-5}. The birth target weight is set to 0.1 and the birth target covariance is equal to $10^2 \begin{bmatrix} I_2 & \mathbf{0} \\ \mathbf{0} & I_2/4 \end{bmatrix}$. The maximum number of tracks stored between time steps (equivalent to the M parameter of R-RANSAC) is set to $\mathcal{J}_k = 50$.

7.2 Numerical Results

This comparison used the Matlab implementation of the CJ-IMMPDAF version of R-RANSAC. The R-RANSAC parameters used were $\mathrm{N_w} = 25$, $m = 30$, $\tau_\rho = 0.5$, $\tau_\mathrm{T} = 3$, $\tau_\mathrm{R} = 30$, $\ell = 40$, and $\tau_\mathrm{CMD} = 5$. The merging parameters used were $\tau_\theta = 25°$, $\tau_v = 0.2$, and $\tau_x = \tau_y = 35$. The PDA filter parameters used were $P_D = 0.80$, $P_G = 0.95$, and $\lambda = 10^{-2}$. The Markov transition matrix of the three IMM modes (low, medium, and high process noise) is given by

$$\pi = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.5 & 0.2 \\ 0.3 & 0.4 & 0.3 \end{bmatrix}. \tag{7.1}$$

7.2.1 MOTP & MOTA

Table 7.1 displays the MOTP and MOTA metrics for the pedestrian and vehicle videos. These results indicate that R-RANSAC and the GM-PHD filter are similarly *precise*, i.e. correctly tracked objects are tracked with comparable RMS errors. However, R-RANSAC produced significantly fewer missed detections than the GM-PHD filter. It should be noted here that because R-RANSAC continues to propagate tracks forward in time without receiving measurements and continues to classify those tracks as good tracks until their inlier ratios fall below the good model threshold, it has the ability to track objects through short occlusions. Because the two test sequences contain occlusions which cause missed detections, R-RANSAC should have a lower ratio of missed detections than the GM-PHD filter. The results verify this supposition, as indicated by column 4 of Table 7.1.

R-RANSAC and the GM-PHD filter produced a similar number of false positives in the pedestrian video; however, R-RANSAC produced noticeably fewer false positives in the vehicle video. The GM-PHD filter produced many duplicate tracks in the vehicle video leading to the

Table 7.1: MOT Results

Vehicle Video

	MOTP	MOTA	MD	FP	MM
GM-PHD	6.7180	0.4119	0.3149	0.1383	0.1349
R-RANSAC	6.3610	0.7306	0.1662	0.0932	0.0100

Pedestrian Video

	MOTP	MOTA	MD	FP	MM
GM-PHD	17.8665	0.3535	0.5122	0.0893	0.0449
R-RANSAC	19.2215	0.5550	0.3690	0.0726	0.0023

high number of false positives. The largest difference in the performance of the two algorithms can be seen in the ratio of mismatches (column 6, Table 7.1). This difference is most pronounced in the vehicle video: the closely spaced and frequently interacting targets lead to very poor track continuity for the GM-PHD filter. On the other hand, R-RANSAC proved to have very robust track continuity even in this challenging scenario. Because of the ratio of mismatches, R-RANSAC is much more *accurate* than the GM-PHD filter.

Figures 7.1 and 7.2 illustrate the stark contrast in the ratio of mismatches between the GM-PHD filter and R-RANSAC with an example from the pedestrian video. These figures plot the different tracks associated with a child walking along the border of a planter. R-RANSAC generates only one track whereas the GM-PHD filter generates 55 different tracks. Figures 7.3 and 7.4 present a similar example from the vehicle video.

7.2.2 OSPA & OSPA-T

Table 7.2 displays the average OSPA and OSPA-T scores over the 600 frame test sequences from the vehicle and pedestrian videos. R-RANSAC achieved a lower OSPA score than the GM-PHD filter in both cases. Because R-RANSAC has superior track continuity, its OSPA-T scores are only slightly higher than its OSPA scores. On the other hand, the poor track continuity of the GM-PHD filter translates into OSPA-T scores that are significantly higher than its OSPA scores.

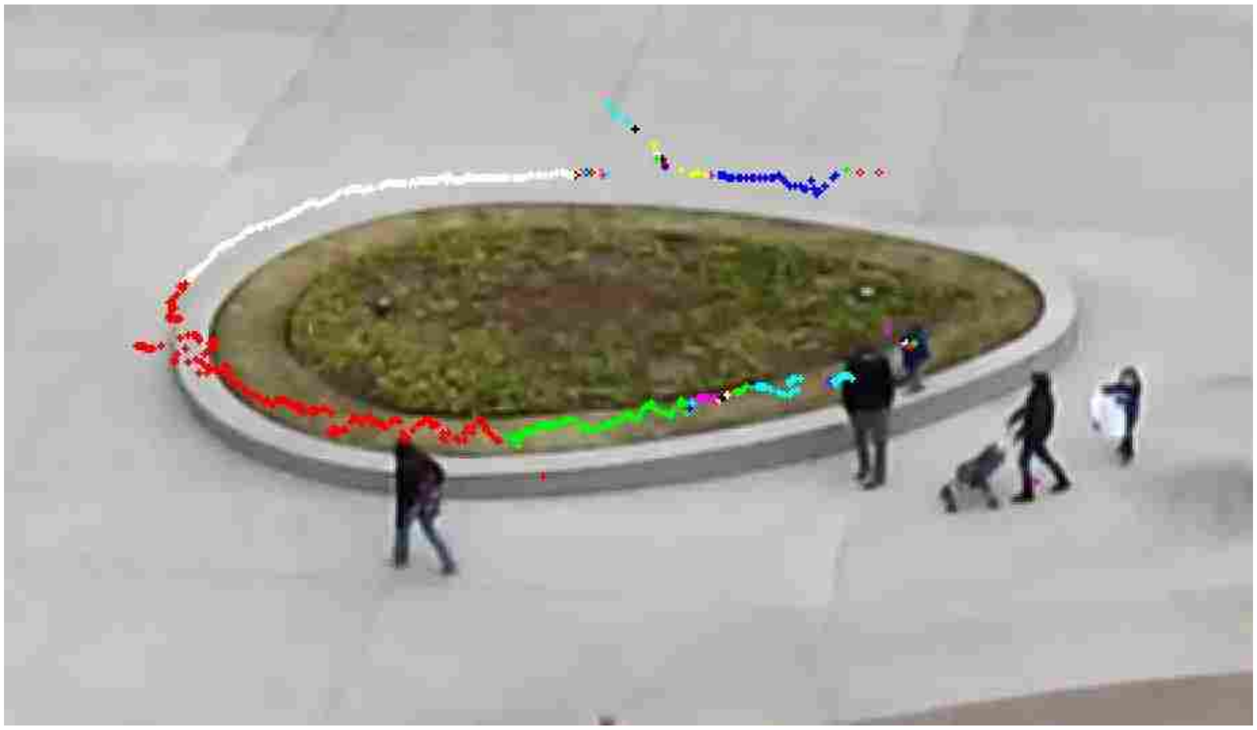


Figure 7.1: The GM-PHD tracks associated with one of the people from the annotated section of the pedestrian video.

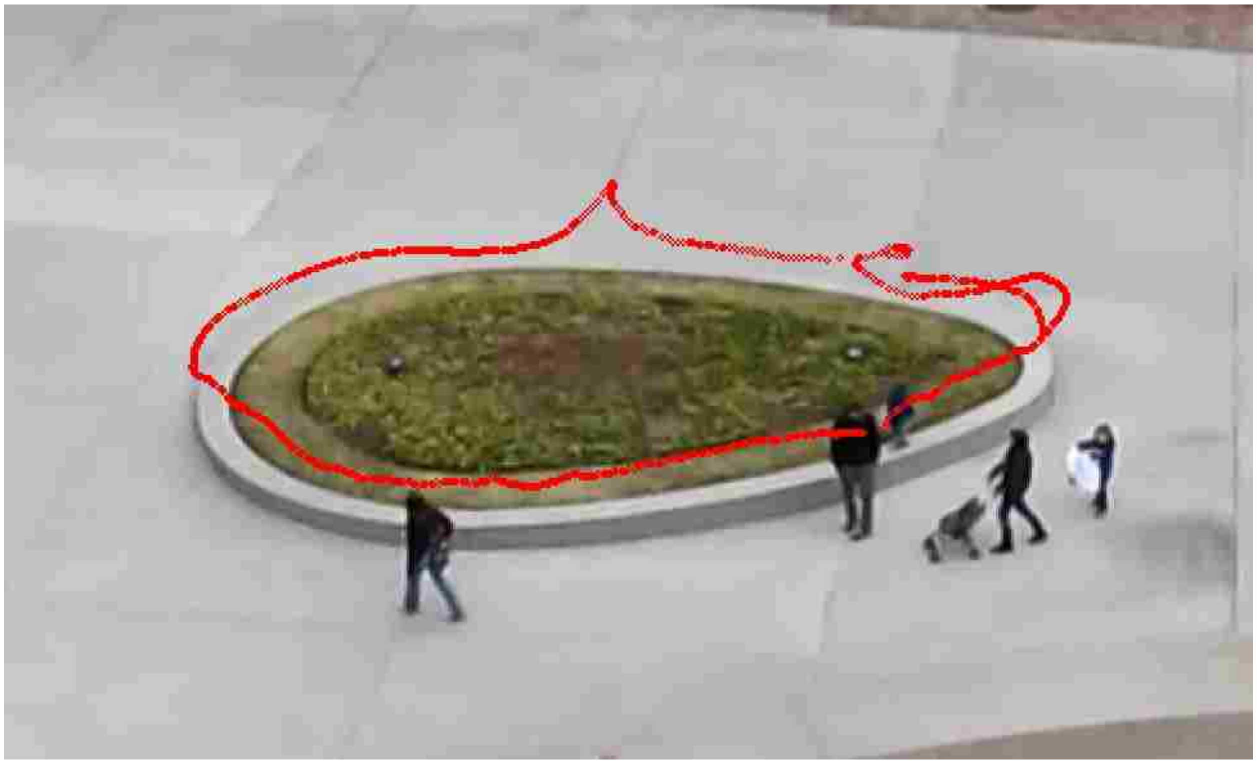


Figure 7.2: The single R-RANSAC track associated with one of the people from the annotated section of the pedestrians video.

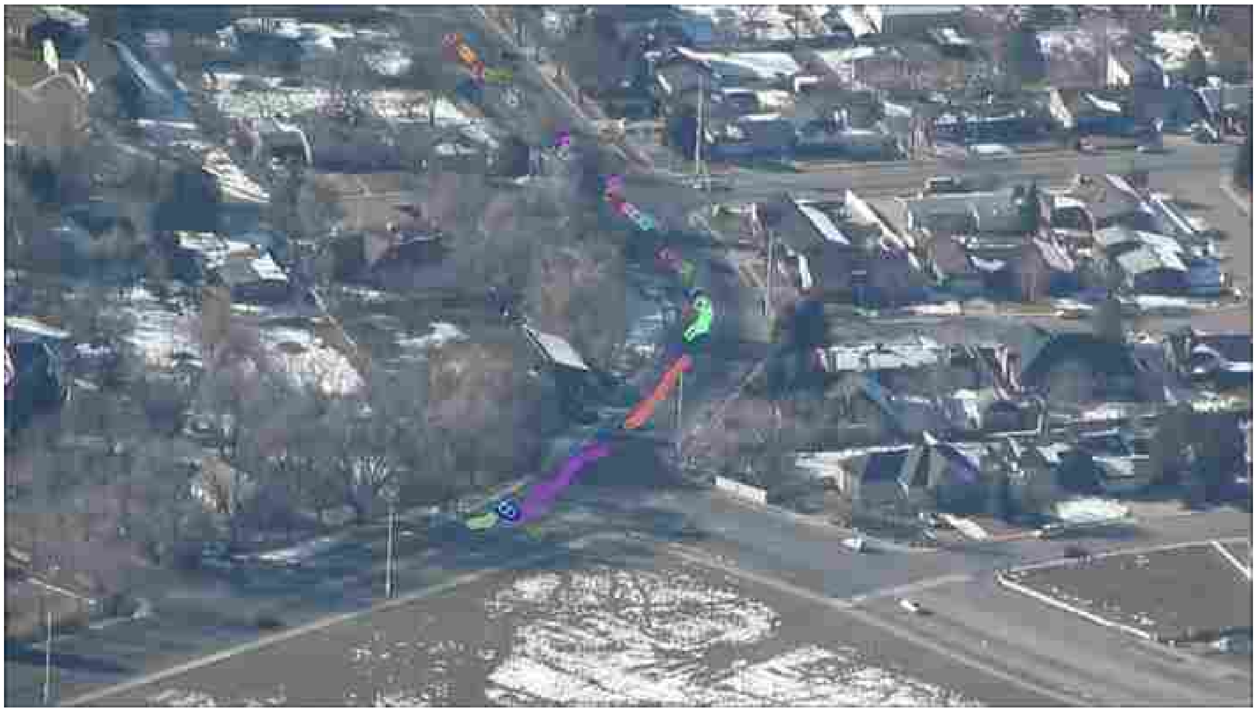


Figure 7.3: The GM-PHD tracks associated with one of the cars from the annotated section of the vehicle video.

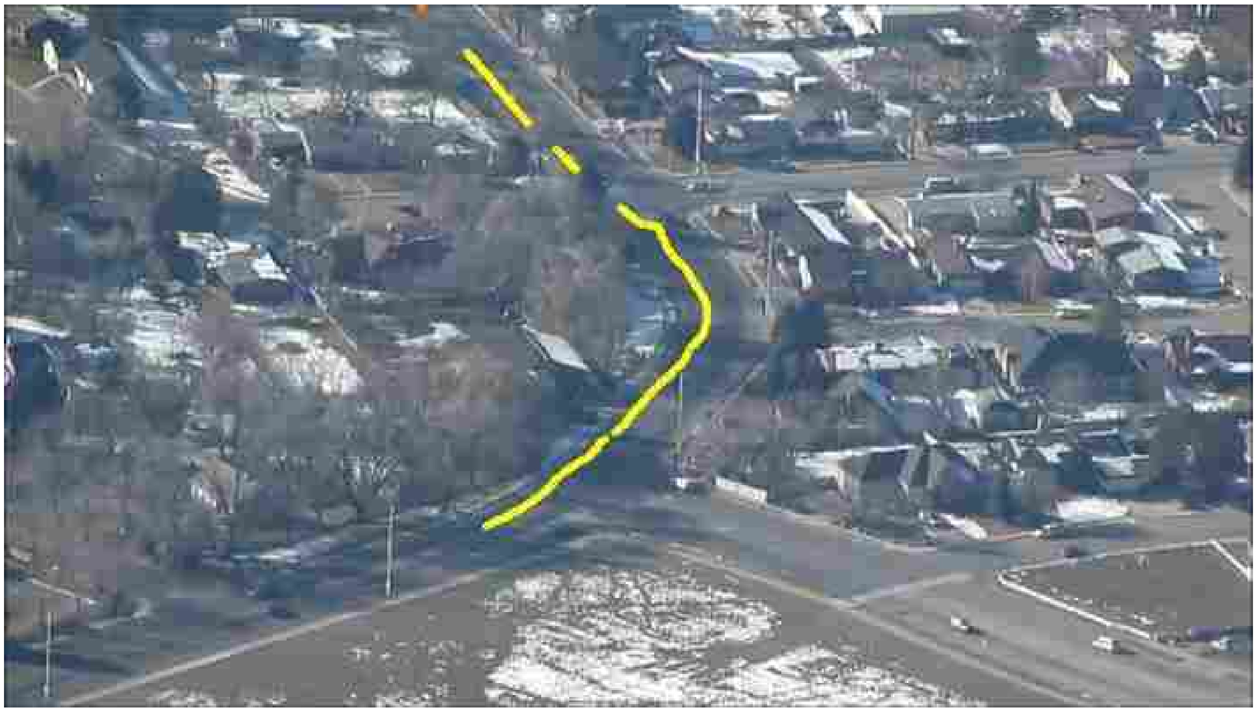


Figure 7.4: The two R-RANSAC tracks associated with one of the cars from the annotated section of the vehicle video. The first track is drawn in orange and appears on the top edge of the image.

Table 7.2: Average OSPA Scores

(a) Vehicle Video

	OSPA	OSPA-T
GM-PHD	55.0926	61.4158
R-RANSAC	40.2504	41.7389

(b) Pedestrian Video

	OSPA	OSPA-T
GM-PHD	70.3005	71.9871
R-RANSAC	59.5268	60.4030

Table 7.3: Execution Times - Time Per Frame

	Vehicle Video	Pedestrian Video
GM-PHD	0.0942	0.0738
R-RANSAC	0.0234	0.0230

7.2.3 Execution Times

The average execution time per frame for both algorithms (excluding video processing time) over the full videos is reported in Table 7.3. The execution times are generally comparable. Reference [12] reports that R-RANSAC becomes increasingly more efficient than the GM-PHD filter when the ratio of clutter to true target measurements is low; conversely, the GM-PHD filter is more efficient in high clutter environments. In high clutter environments, R-RANSAC forms many tracks modeling the clutter that are unused. When the clutter-to-target ratio is lower, R-RANSAC efficiently updates tracks without generating many new tracks. The results presented here support this finding. However, even though the PHD filter is more computationally efficient than R-RANSAC in high clutter scenarios, there is an obvious trade off with track continuity.

CHAPTER 8. BACKGROUND ON VIDEO PROCESSING

To understand the innovations to video processing presented in Chapter 9, standard video processing techniques are reviewed here. To produce measurements for the R-RANSAC tracker, the objects of interest are segmented from the image using a foreground detection / background subtraction algorithm. This step produces a foreground mask in which foreground pixels are white and background pixels are black. Foreground detection is described in Section 8.1. This mask might then be post-processed by a combination of morphological operations. The mask is then fed into a blob detector which detects contiguous groups of foreground pixels and outputs certain geometrical information about them. Blob detection is described in Section 8.2. The centroids of the blobs are used as (x, y) position measurements for the tracker.

8.1 Foreground Detection

Foreground detection involves classifying pixels as either foreground or background. Generally, foreground pixels are those that belong to moving objects, although this definition is extended in Chapter 9 to include pixels belonging to any object of interest, whether it be moving or stationary. A successful foreground detector is able to fully segment moving objects without leaving ghosts in the objects' previous positions, account for gradual changes in background lighting, and account for cyclical variations in the background.

Foreground detection algorithms range from the very simple to the vastly more complex. Basic techniques include frame-to-frame subtraction and subtraction from an empty background image. In frame-to-frame subtraction, the previous frame, or a frame from several time steps prior to the current time step, is subtracted from the current frame. A threshold is then applied to the difference image to identify foreground pixels. This method struggles to fully segment objects in video with high frame rate and leaves ghosts in video with low frame rate. Another simple method is subtraction from an empty background. In this method, an image of the empty background is obtained prior to tracking. This method can fully segment new additions to the scene, but cannot account for changes in scene lighting.

More effective methods of foreground detection construct a background model and compare incoming images with this model. The background is often modeled with an assumed pixel distribution or a sample set of pixels. The principal tenet of the latter approach is that the true pixel distribution inevitably deviates from the assumed distribution, and that it is more accurate to model the background with a sample set of actual pixel values. One interesting example in the sample set class of algorithms is ViBe [35]. ViBe uses temporal and spatial sub-sampling as well as a unique method of discarding samples to construct the background model. Incoming background-labeled pixels have a one in 16 chance of being included into the background model. When a pixel's sample set is updated, a neighboring pixel is selected and its sample set is also updated with that same pixel. Instead of using a first-in, first-out approach to discard sample pixels, the sample to be discarded is randomly chosen according to a uniform distribution. This allows older pixels, which might still contain useful background information, to remain in the sample set instead of being systematically discarded.

Although the sample set class of foreground detectors have certain desirable features, the most popular foreground detection algorithms use an assumed pixel distribution. In fact, the Gaussian mixture models (GMM) or Stauffer's method [36], has become the ubiquitous foreground detector. This method uses a mixture of Gaussian distributions to model each pixel. An incoming pixel that supports any of the existing Gaussian modes is used to update the mean, variance, and weight of those modes. An incoming pixel that does not match any of the existing modes is used to initialize a new mode that replaces the mode with the least support. The modes are ordered according to their sortkey and the highest ranked modes comprise the background. If a current pixel matches one of the background modes, then that pixel belongs to the background and is colored black in the foreground mask. Otherwise, the pixel is foreground and is colored white. GMM can be tuned to achieve excellent segmentation results, handles slow changes in environmental lighting well, and models cyclical background elements through its several Gaussian modes.

Many variations on the original GMM algorithm have been proposed. One variation of particular note was developed by Kaewtrakulpong and Bowden and was introduced in [37]. They add shadow detection capability to the GMM and modify the update equations to more quickly learn the background upon start up. Optimized implementations of their method are available in Matlab's Computer Vision System Toolbox and in the OpenCV libraries

GMM was re-implemented in C++ so that it could be modified to better detect slowly moving and stationary objects. Our implementation follows the OpenCV implementation and several of the key equations are given here. Each pixel is modeled with three Gaussian modes. The Gaussian modes are 3-dimensional, representing the red-green-blue (RGB) color channels. Each Gaussian mode is characterized by a mean array of length three, μ_{RGB}, a variance array of length three, σ_{RGB}, a weight, w, and a sortkey, s_k. In using individual variance values instead of a covariance matrix it is assumed that the color channels are independent from each other. This assumption also precludes a costly matrix inversion in the update step. Pixel p from the current frame matches the i^{th} Gaussian mode if

$$\Delta \cdot \Delta < \tau_{\text{similarity}} \left(\sigma^i_{\text{RGB}}[1] + \sigma^i_{\text{RGB}}[2] + \sigma^i_{\text{RGB}}[3]\right) \tag{8.1}$$

where Δ is the current pixel values subtracted from μ^i_{RGB} and $\tau_{\text{similarity}}$ is the similarity threshold and is set at 6.25. The weight update equation is

$$w^i = w^i + \alpha\left(1 - w^i\right) \tag{8.2}$$

the mean update equation is given by

$$\mu^i_{\text{RGB}} = \alpha\Delta \tag{8.3}$$

and element j of the variance is updated by

$$\sigma^i_{\text{RGB}}[j] = \max\left(\sigma^i_{\text{RGB}}[j] + \alpha\left(\Delta[j]\Delta[j] - \sigma^i_{\text{RGB}}[j]\right), \sigma_{\min}\right) \tag{8.4}$$

where the $\sigma_{\min}$ parameter is set at 225. The sortkey is re-calculated at each time step with

$$s^i_k = \frac{w^i}{\left|\sigma^i_{\text{RGB}}\right|}. \tag{8.5}$$

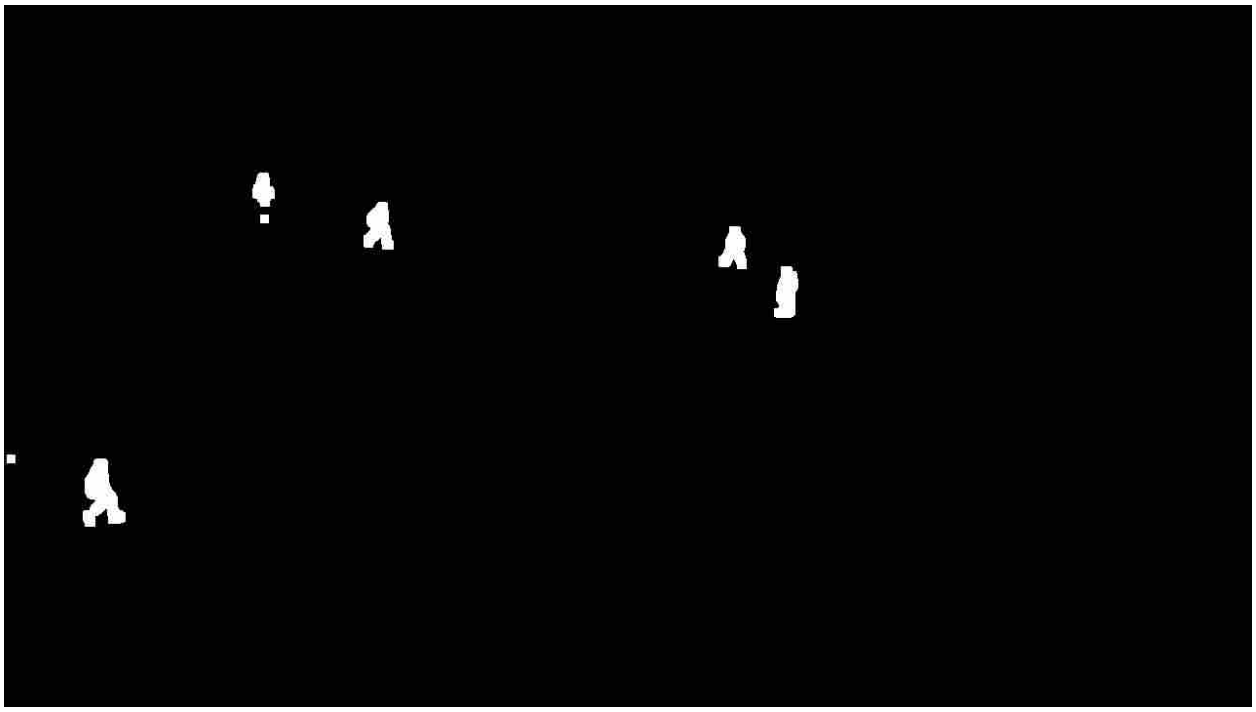


Figure 8.1: A foreground mask produced by the Gaussian mixture models foreground detection algorithm from the pedestrian video.

The following equation determines which Gaussian modes comprise the background model

$$B = \arg_b \min \left(\sum_{j=1}^{b} w^j > T \right) \tag{8.6}$$

where the parameter T is set at 0.7 and B represents the background model. In our implementation, we use three Gaussian modes and we only allow a maximum of two modes to represent the background. An example foreground mask produced using GMM is shown in Figure 8.1. This foreground mask has been post-processed using morphological operations. The basic morphological operations are dilation and erosion. The opening operation is performed by an erosion followed by a dilation and the closing operation is performed by a dilation followed by an erosion. An effective combination of morphological operations can reduce noise in the foreground mask and connect closely spaced blobs originating from the same object. One iteration of erosion with a 2 by 2 rectangular element followed by three iterations of dilation with the same element was performed on the foreground mask in Figure 8.1.

Algorithm 2 Blob Detector

```
1:  for each row of pixels do
2:      for each column of pixels do
3:          if current pixel is foreground then
4:              if neither pixel to the left or pixel above are foreground then
5:                  Initialize new blob class. Add current pixel.
6:                  In blob map, update current pixel's position with newly initialized blob class.
7:              else if pixel to the left is foreground $\wedge$ pixel above is background then
8:                  Add current pixel to blob class to which left pixel belongs.
9:                  Current pixel position in blob map assigned pointer to left pixel's blob class.
10:             else if pixel to the left is background $\wedge$ pixel above is foreground then
11:                 Add current pixel to blob class to which above pixel belongs.
12:                 Current pixel position in blob map assigned pointer to above pixel's blob class.
13:             else if both pixel to the left and pixel above are foreground then
14:                 if left and above pixels belong to the same blob class then
15:                     Add current pixel to shared blob class.
16:                     Current pixel position in blob map assigned pointer to shared blob class.
17:                 else if left and above pixels belong to different blob classes then
18:                     Add contents of left blob class to above blob class. Pixels belonging to left
                          pixel's blob class assigned pointer to above pixel's blob class in blob map.
19:                     Deactivate left blob class.
20:                     Add current pixel to shared blob class.
21:                     Current pixel position in blob map assigned pointer to shared blob class.
22:                 end if
23:             end if
24:         end if
25:     end for
26: end for
27: Apply minimum blob size threshold.
```

8.2 Blob Detection

A blob detector is a connected component labeling algorithm; it detects connected groups, or blobs, of foreground pixels. The blob detector we implemented in C++ is given in Algorithm 2. A key feature of this algorithm is the blob map. The blob map is a matrix of the same dimensions as the image that contains pointers to blob classes. The blob map is an efficient way of looking up which blob a given pixel belongs to, and of merging two newly connected blobs. After a new blob class is initialized, it is inserted into a linked list. When a blob class is deactivated, its *currBlob* variable is set to 0. In the last step of the blob detector, a minimum blob area threshold is applied that filters out purely noise associated blobs.

Blob detectors can generate a myriad of useful information about the blobs. For example, Matlab's blob detector can compute blob area, centroid, bounding box, orientation, eccentricity, extent, and several other geometrical measures (see Matlab's Blob Analysis documentation). Unfortunately, OpenCV's blob detector only outputs blob centroids and radii, thus necessitating the customized blob detector of Algorithm 2. The blob class of Algorithm 2 stores the entire set of pixels that compose the blob. After the foreground mask is processed, the centroid, area, and bounding box are calculated and stored in the blob class. The perimeter pixels of the blob are also stored in the blob class. Much of this information is used in Chapter 9 to improve video processing.

CHAPTER 9. FOREGROUND DETECTION VARIATIONS

Satisfactory tracking performance can be achieved with the CJ-PDAF R-RANSAC; a probabilistic data association filter is responsible for filtering and data association and the nearly constant jerk model is used as the motion model. The R-RANSAC algorithm could be further developed, but we are of the opinion that future improvements in performance would be only incremental. Consequently, our research focus shifted to refining the video processing block of the system. Section 9.1 describes ways of linking the tracker and video processing blocks through a feedback loop. Also included in Section 9.1 are methods to obtain velocity measurements (Section 9.1.5), a simple ghost suppression algorithm (Section 9.1.6), and concluding remarks about foreground detection 9.1.7. Section 9.2 discusses the application of R-RANSAC to a class of problems known as stationary object detection.

9.1 Tracker-Sensor Feedback

The foreground detection techniques developed and described in this section are based on the concept of a tracker-sensor feedback loop, an idea first presented in “Kalman Tracking with Target Feedback on Adaptive Background Learning” by Aristodemos Pnevmatikakis and Lazaros Polymenakos [38]. This concept is displayed in Figure 9.1. Normally, the tracker and sensor processing blocks of an MTT system are independent. In a tracker-sensor feedback loop, the tracking results are sent through a feedback loop and serve to inform how sensor processing is performed. In [38], the authors propose two modifications to Stauffer’s Method. First and foremost, they modify the background update rate α of each pixel according to its proximity to slowly moving targets. Pixels that are closer to such a target are updated more slowly (i.e. α is reduced) than pixels further away from such targets. This modification is designed to enhance GMM’s ability to detect slowly moving targets that would otherwise fade or be incorporated into the background. In [38], a target’s extent is approximated with its error covariance. The modified background update rate is

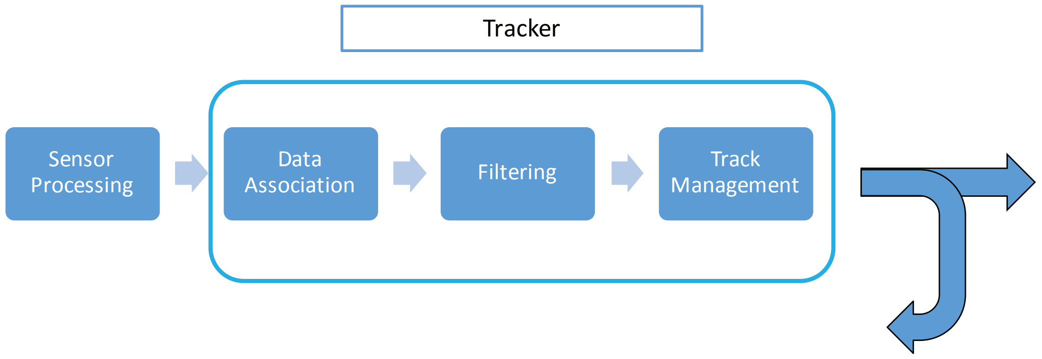


Figure 9.1: The tracker-sensor feedback loop.

applied to all pixels that fall within this estimated extent. Their second change involves varying the T parameter to account for situations with flickering backgrounds. Several techniques developed in this section extend their first modification.

The following sections describe the progression of foreground detection techniques explored during the course of this research. Tables 9.1 and 9.2 display the average MOT and OSPA scores for several simulation runs with the pedestrian video using the different foreground detectors. Full MOT and OSPA results for all simulation runs are included in Appendix A. Unless otherwise noted, the following parameters were used in the simulations. The R-RANSAC parameters used were $\mathrm{N_w} = 25$, $m = 30$, $\tau_\rho = 0.55$, $\tau_\mathrm{T} = 3$, $\tau_\mathrm{R} = 30$, $\ell = 25$, and $\tau_\mathrm{CMD} = 4$. The merging parameters used were $\tau_\theta = 15°$, $\tau_v = 0.25$, and $\tau_x = \tau_y = 40$. The PDA filter parameters used were $P_D = 0.80$, $P_G = 0.99$, and $\lambda = 5 \times 10^{-6}$.

9.1.1 Uniform Background Update Rate

The standard Stauffer's method was run with a uniform background update rate of 0.1 and 0.005 to establish baseline performance. These methods are abbreviated as "Baseline - high" and "Baseline - low" in the results tables. Five simulations runs were conducted with each method.

9.1.2 Non-Zero Adaptive Background Update Rate

In this variation, the standard background update rate is set at 0.1. All pixels belonging to the nearest neighbor blob of a valid R-RANSAC track during the previous time step have their background update rate reduced to 0.005. This method is abbreviated as "Adaptive" in the results table. Five simulation runs were conducted with this method.

This method is roughly equivalent to the first modification made by Pnevmatikakis and Polymenakos in [38]. Instead of estimating target extent with the error covariance, the target extent is directly measured from the foreground mask. This requires a custom blob detector (see Section 8.2). This technique is superior to that of [38] because only target-associated pixels have their update rates modified. Pnevmatikakis and Polymenakos concede that their method results in non-target associated pixels having their update rates modified and that this is undesirable. For this reason, they limit modifying the background update rate to slowly moving targets only. Using direct measurements of target extent allows the background update rate to be modified across all valid tracks without any negative side effects and thus eliminates a fast/slow target classification step.

9.1.3 Zero Background Update Rate

To be able to indefinitely detect stationary objects of interest, the background update rate must be zeroed at the associated pixel locations. This section's method is similar to that of Section 9.1.2 except that α is set to zero for all valid track-associated pixels from the previous time step. For pixel's whose update rates are set to zero, the current Gaussian modes of the GMM are retained; a new mode is not initialized for a non-matching current pixel.

Reference [35] provides an insightful discussion on the difference between *conservative* and *blind* background update schemes. In a conservative update scheme, only background labeled pixels are used to update the background. Conversely, all incoming pixels update the background model in a blind scheme. A conservative update scheme is inherently unstable because background pixels falsely labeled as foreground will be incorrectly labeled indefinitely. However, a conservative scheme is capable of indefinitely detecting stationary objects and a blind scheme is not. The method described in this section and in the following sections are examples of conservative update schemes.

This method was run with several combinations of different α and τ_T values. These include "Zero - $\alpha = 0.1, \tau_T = 3$", "Zero - $\alpha = 0.1, \tau_T = 20$", "Zero - $\alpha = 0.005, \tau_T = 3$", and "Zero - $\alpha = 0.005, \tau_T = 20$". Ten simulation runs were conducted with each method.

9.1.4 Zero Background Update Rate with Minimum Blob Area Threshold

This method is the same as that described in Section 9.1.3, but with the addition of an adaptive threshold on the minimum allowable blob area. This adaptive threshold is designed as a safeguard against background pixels being mistakenly labeled as foreground, i.e. it is a way of bringing stability to a conservative background update scheme. If a small group of background pixels is mistakenly labeled as foreground, the associated blob will not be considered a measurement and those pixels will continue to be updated until they are re-incorporated into the background. The distribution of blob sizes is modeled as a Kalman filter with $A = 1$ and $C = 1$, i.e.

$$\overline{\mathbf{B}}_{t|t-1} = \overline{\mathbf{B}}_{t|t-1} \tag{9.1}$$

$$\mathbf{P}_{t|t-1} = \mathbf{P}_{t-1} + Q \tag{9.2}$$

$$\overline{\mathbf{B}}_{t} = \overline{\mathbf{B}}_{t|t-1} + K\left(\mathbf{b}_t^i - \mathbf{P}_{t|t-1}\right) \tag{9.3}$$

$$\mathbf{P}_{t} = (I - K)\,\mathbf{P}_{t|t-1} \tag{9.4}$$

where $\overline{\mathbf{B}}$ is the average blob area, $\mathbf{b}_t^i$ is the i^{th} blob produced at the t^{th} time step, and K is the Kalman gain and is given by

$$K = \mathbf{P}_{t|t-1}\left(R + \mathbf{P}_{t|t-1}\right)^{-1}. \tag{9.5}$$

The parameters used are $R = 0.001$ and $Q = 10$. At each time step, the Kalman filter is updated with every blob $\mathbf{b}_t^i$ and the minimum blob area threshold is set at three variances $\mathbf{P}_k$ below the mean. Because the camera is stationary in these simulations, a fixed threshold would function just as well. However, if this MMT system were mounted on a UAV, an adaptive threshold would be required.

This method was run with several combinations of different α and τ_T values. These include "Zero$_{\text{min}}$ - $\alpha = 0.1, \tau_T = 3$", "Zero$_{\text{min}}$ - $\alpha = 0.1, \tau_T = 20$", "Zero$_{\text{min}}$ - $\alpha = 0.005, \tau_T = 3$", and "Zero$_{\text{min}}$ - $\alpha = 0.005, \tau_T = 20$". Ten simulation runs were conducted with each method.

9.1.5 Velocity Measurements

All of the previous methods use position-only measurements, i.e. the blob centroids. However, it is possible to obtain measurements with a velocity component, although it does add complexity to the algorithm. In obtaining velocity measurements, we follow the general procedure outlined in [39]. In [39], the authors use Harris corner detection to find image features in the previous frame and match those features in the current frame using the Lucas-Kanade optical flow algorithm. For each matched feature pair, the length and direction of the optical flow vectors are calculated. The optical flow vectors are then clustered using k-means block-based clustering. The authors select the most "scattered" cluster as the background cluster and label all other clusters as moving object clusters. Outliers are removed from the moving object clusters with RANSAC, and Delaunay Triangulation is used to produce a more fully segmented foreground mask. This technique is specifically meant for moving object detection with a non-stationary background, but it can be simplified and usefully applied to stationary background situations. In our application of this method, FAST features [40] are used instead of Harris corners because of their speed. Features in consecutive frames are matched using the Lucas-Kanade tracker implemented in OpenCV and the optical flow vectors are characterized by their speed and heading. The features in the current frame are then superimposed on the foreground mask and are pre-clustered according to the blob which they belong to. Features that do not fall on a blob are discarded (these features would be used to calculate the background transformation in the case of a moving camera). For each blob, the features are further clustered using sequential-RANSAC [14]. Our implementation of sequential-RANSAC is given in Algorithm 3. The similarity criteria (Algorithm 3, line 10) is very similar to the merging criteria used in Algorithm 1; several parameters are re-used and the similarity criteria accounts for moving and stationary targets. Measurements are similar if their percent difference in speed or their absolute difference in speed meet their respective thresholds

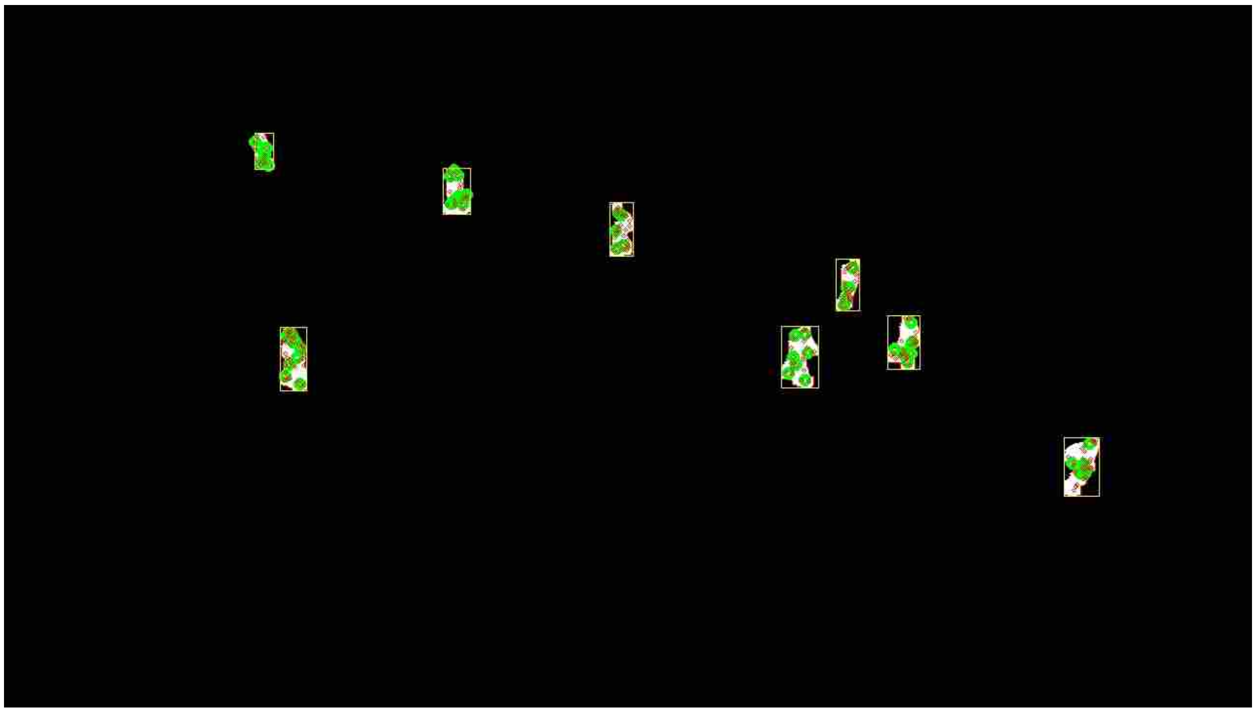


Figure 9.2: A foreground mask showing the foreground pixels (white), the blob bounding boxes (yellow rectangles), the matched features (small red circles), and clustered measurements (green circles).

($\tau_v = 0.25$ and $\tau_{|v|} = 10$) and their absolute differences in heading, x position, and y position meet their respective thresholds ($\tau_\theta = 15$, $\tau_{x_\text{cluster}} = 20$, and $\tau_{y_\text{cluster}} = 20$). Measurements are also similar if both of the measurement speeds meet the minimum velocity threshold (τ_{v_min}) and the absolute position differences meet their thresholds.

The average x and y velocity components and position for each cluster, as well as information indicating which blob the cluster belongs to, is sent to R-RANSAC as a measurement. Figures 9.2 and 9.3 display several steps of the process used to obtain velocity measurements.

Instead of using the full PDA filter with the velocity measurements, a simple weighted residual is computed; each measurement's association probability is calculated using the covariance of the innovation. Validation gating is still based solely on position, but the measurement association probabilities take velocity into account. Measurements that are outliers to all existing tracks are used to initialize hypothesis tracks with the standard CV track initialization scheme. The zero background update rate scheme of Section 9.1.3 is retained and a static minimum blob area threshold of 10 is used. This method was run with two combinations of α and τ_T values: "Velocity

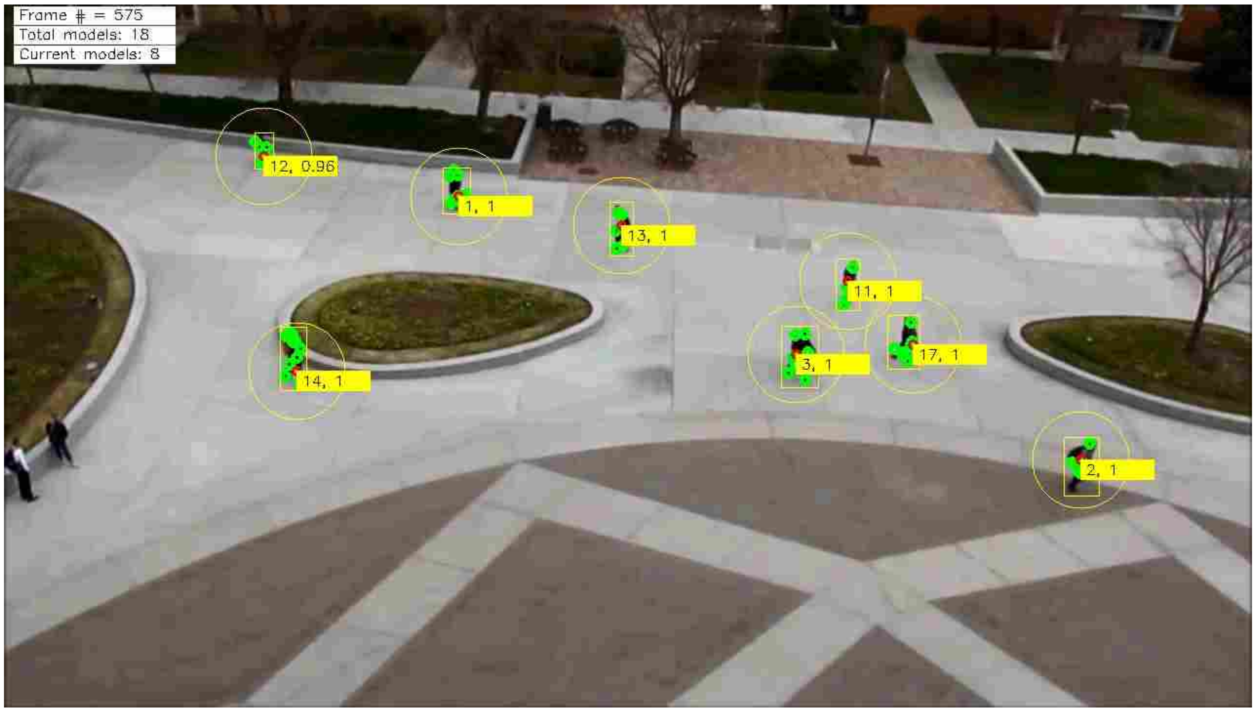


Figure 9.3: The video frame corresponding to Figure 9.2. Shown are the R-RANSAC tracks, their inlier regions, and the clustered measurements (in green).

- $\alpha = 0.005, \tau_{\mathrm{T}} = 3$” and “Velocity - $\alpha = 0.005, \tau_{\mathrm{T}} = 20$”. Ten simulation runs were conducted with each method. In these simulations, the guided sampling threshold for the RANSAC-based track initialization scheme was set to 100. Using this threshold improved the average OSPA-T scores by 2.1601 and 7.3506 for the two methods, respectively.

The method described above allows us to solve the merged measurement problem. A merged measurement occurs when two targets pass sufficiently close to each other that their individual blobs merge into a single blob during the morphological operations. Figure 9.4 shows a merged measurement from the pedestrian video. The resulting measurement does not accurately describe the position of either closely spaced target and using this measurement in the track update can have serious consequences. Figure 9.5 displays a set of tracks from Trial 2 of the “Zero$_{\mathrm{min}}$ - $\alpha = 0.005, \tau_{\mathrm{T}} = 3$” simulations (the best performing trial from this set). In this portion of the video, multiple targets pass near each other and produce merged measurements. The effects of using these merged measurements in the track update is obvious in Figure 9.5; several R-RANSAC tracks are momentarily pulled off of the true path of their target and several R-RANSAC tracks switch targets entirely. When the method described in this section is used, a merged measurement

Algorithm 3 Sequential-RANSAC

```
1: for each blob do
2:     while less than 80% of the features have been accounted for do
3:         for $\ell_{\text{sequential}}$ iterations do
4:             Randomly pick a feature.
5:             for all other unaccounted for features in that blob do
6:                 Calculate the percent difference in speed between the randomly selected and
                   current features, percentSpeedDiff.
7:                 Calculate the absolute difference in speed between the randomly selected and
                   current features, absoluteSpeedDiff.
8:                 Calculate the absolute difference in heading between the randomly selected and
                   current features, absoluteHeadingDiff.
9:                 Calculate the absolute difference in x and y positions between the randomly
                   selected and current features, absoluteXdiff and absoluteYdiff, respectively.
10:                if the measurement similarity criteria is satisfied then
11:                    Add current feature to the consensus set of the new cluster.
12:                end if
13:            end for
14:            if the new consenus set is larger than the previously largest consensus set then
15:                The new consensus set becomes the largest consensus set.
16:            end if
17:        end for
18:        Remove features belonging to the new cluster from the remaining feature set.
19:    end while
20:    Calculate average speed, heading, and position information for the refined clusters.
21: end for
```

is decomposed into several measurements based upon the distribution of optical flow vectors. This allows targets traveling in different direction to pass closely by each other without negatively affecting the tracking. Figure 9.6 demonstrates this by showing the same set of tracks taken from Trial 1 of the “Velocity - $\alpha = 0.005, \tau_T = 20$” simulations.

9.1.6 Ghost Suppression

Ghosts can be defined as foreground artifacts located in previous target locations. For example, if each new frame were subtracted from the first frame of a video sequence, foreground pixels would appear in the current locations of moving objects as well as the location of all moving objects present in the first frame. Ghost suppression is the task of removing these artifacts. Additionally, ghost suppression aids in removing falsely labeled background pixels from

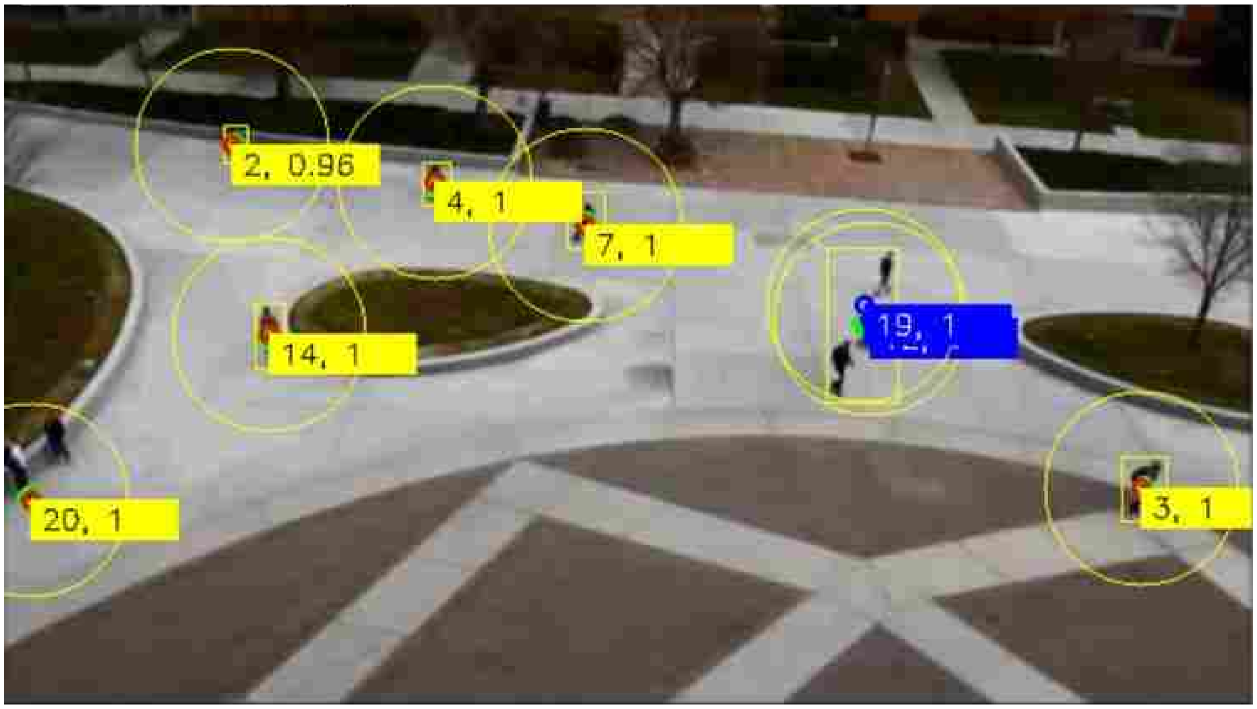


Figure 9.4: Three closely-spaced targets produce a single merged measured measurement (in blue).

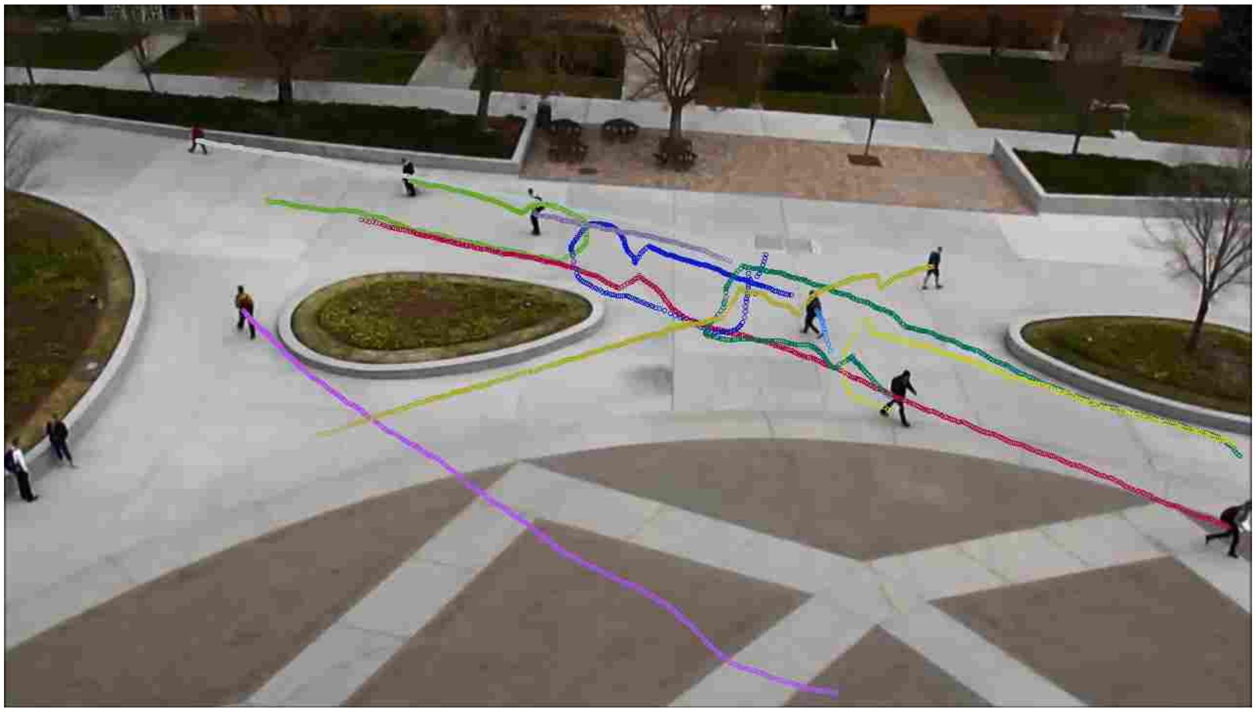


Figure 9.5: Updating tracks with merged measurements causes R-RANSAC tracks to deviate from the true path of their targets (all but the purple track) and oftentimes causes tracks to switch targets (green, blue, and yellow tracks).

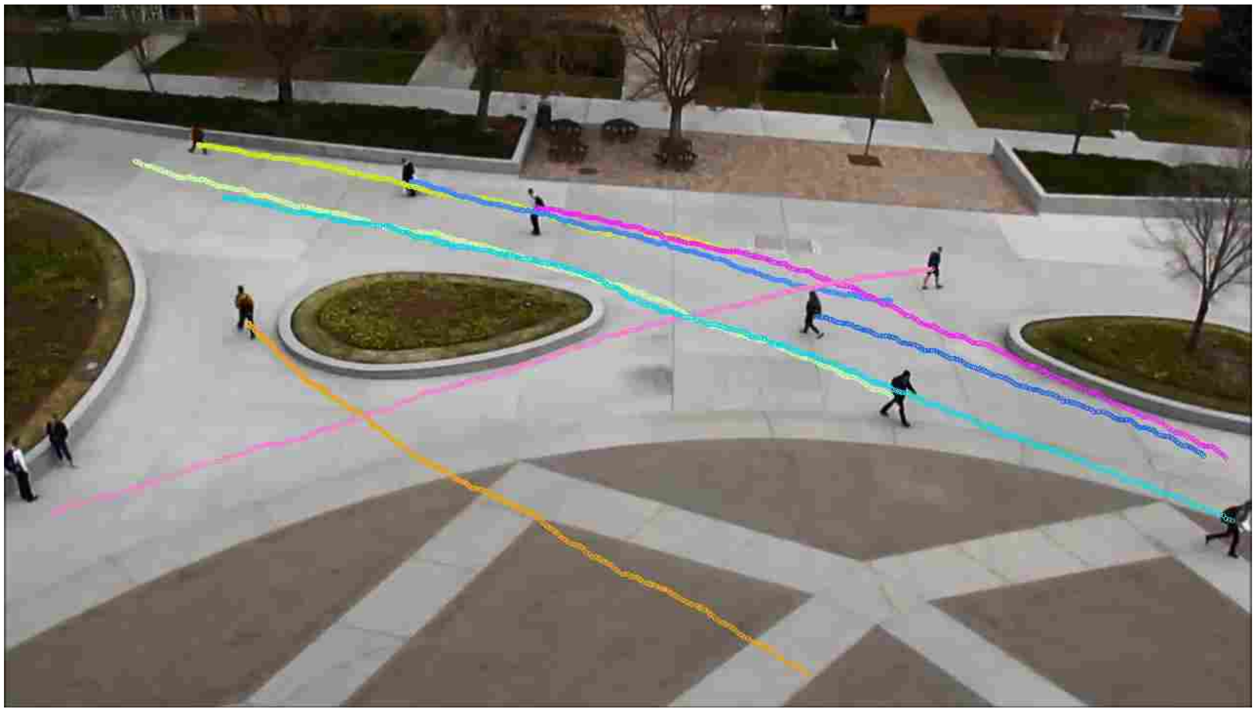


Figure 9.6: Tracks updated with velocity measurements. The tracks of closely spaced targets are not affected by each other.

the foreground mask. In our ghost suppression algorithm, a foreground pixel can be updated by a Manhattan neighbor background pixel if the pixels meet a similarity threshold. This similarity threshold prevents true stationary objects of interest from fading into the background. The ghost suppression algorithm was run with the methods described in Sections 9.1.4 and 9.1.5. These simulations are labeled $\text{Zero}_{\text{min}_{\text{ghost}}}$ - $\alpha = 0.005, \tau_{\text{T}} = 3$; $\text{Zero}_{\text{min}_{\text{ghost}}}$ - $\alpha = 0.005, \tau_{\text{T}} = 20$; $\text{Velocity}_{\text{ghost}}$ - $\alpha = 0.005, \tau_{\text{T}} = 3$; and $\text{Velocity}_{\text{ghost}}$ - $\alpha = 0.005, \tau_{\text{T}} = 20$. Each set of simulations included ten trials.

9.1.7 Conclusions on Foreground Detection

Instead of analyzing the results of every simulation, we will present more general statements about how to optimize tracking performance through improvements to video processing.

- Given a stable camera platform and a mostly stationary background, a low α value is preferable to a high α value. The pedestrian video meets both of these requirements and consequently, the “Baseline - low” method performed much better than the “Baseline - high”

Table 9.1: MOT Results - Foreground Detection Techniques

Foreground Detection Method	MOTP	MOTA	MD	FP	MM
Baseline - high	8.0027	0.6927	0.2441	0.0112	0.0520
Baseline - low	11.5430	0.8251	0.0770	0.0635	0.0344
Adaptive	10.3353	0.7907	0.1597	0.0173	0.0324
Zero - $\alpha = 0.1, \tau_T = 3$	10.9410	0.8590	0.0463	0.0806	0.0141
Zero - $\alpha = 0.1, \tau_T = 20$	10.7260	0.8599	0.0981	0.0292	0.0128
Zero - $\alpha = 0.005, \tau_T = 3$	11.1266	0.8290	0.0070	0.1540	0.0100
Zero - $\alpha = 0.005, \tau_T = 20$	11.2642	0.8630	0.0155	0.1120	0.0096
Zero$_{\text{min}}$ - $\alpha = 0.1, \tau_T = 3$	11.0029	0.8741	0.0736	0.0389	0.0134
Zero$_{\text{min}}$ - $\alpha = 0.1, \tau_T = 20$	11.0820	0.8759	0.0736	0.0389	0.0116
Zero$_{\text{min}}$ - $\alpha = 0.005, \tau_T = 3$	11.0992	0.8851	0.0223	0.0844	0.0083
Zero$_{\text{min}}$ - $\alpha = 0.005, \tau_T = 20$	11.1878	0.8934	0.0363	0.0630	0.0072
Velocity - $\alpha = 0.005, \tau_T = 3$	9.0759	0.8828	0.0656	0.0329	0.0188
Velocity - $\alpha = 0.005, \tau_T = 20$	8.6681	0.8721	0.1203	0.0011	0.0064
Zero$_{\text{min}_{\text{ghost}}}$ - $\alpha = 0.005, \tau_T = 3$	11.0997	0.8689	0.0465	0.0750	0.0096
Zero$_{\text{min}_{\text{ghost}}}$ - $\alpha = 0.005, \tau_T = 20$	11.1283	0.8201	0.0440	0.1215	0.0144
Velocity$_{\text{ghost}}$ - $\alpha = 0.005, \tau_T = 3$	8.7794	0.8805	0.0770	0.0242	0.0184
Velocity$_{\text{ghost}}$ - $\alpha = 0.005, \tau_T = 20$	8.4825	0.8575	0.1341	0.0012	0.0071

Table 9.2: Average OSPA Scores - Foreground Detection Techniques

Foreground Detection Method	OSPA	OSPA-T	Standard Deviation (OSPA-T)
Baseline - high	57.4211	58.5812	0.5042
Baseline - low	46.9291	48.7273	1.4966
Adaptive	49.5333	50.9470	0.5131
Zero - $\alpha = 0.1, \tau_T = 3$	39.0721	39.6491	3.0584
Zero - $\alpha = 0.1, \tau_T = 20$	42.3569	43.2007	3.9522
Zero - $\alpha = 0.005, \tau_T = 3$	37.2976	37.7397	1.8199
Zero - $\alpha = 0.005, \tau_T = 20$	35.4954	36.0667	3.5353
Zero$_{\text{min}}$ - $\alpha = 0.1, \tau_T = 3$	40.1993	41.3096	4.7428
Zero$_{\text{min}}$ - $\alpha = 0.1, \tau_T = 20$	40.1993	41.3096	4.7428
Zero$_{\text{min}}$ - $\alpha = 0.005, \tau_T = 3$	31.9620	32.5188	3.9421
Zero$_{\text{min}}$ - $\alpha = 0.005, \tau_T = 20$	32.8622	33.4488	4.5913
Velocity - $\alpha = 0.005, \tau_T = 3$	35.3888	36.8415	3.5248
Velocity - $\alpha = 0.005, \tau_T = 20$	36.4746	36.8038	1.1841
Zero$_{\text{min}_{\text{ghost}}}$ - $\alpha = 0.005, \tau_T = 3$	36.2479	36.9295	4.1253
Zero$_{\text{min}_{\text{ghost}}}$ - $\alpha = 0.005, \tau_T = 20$	41.3798	42.3361	4.3892
Velocity$_{\text{ghost}}$ - $\alpha = 0.005, \tau_T = 3$	36.6000	37.3656	4.1585
Velocity$_{\text{ghost}}$ - $\alpha = 0.005, \tau_T = 20$	38.2099	38.8664	3.7430

method. Furthermore, a low, uniform α value performs better than the "Adaptive" method that alternates between high and low α values depending on target position.

- Zeroing out α for pixels associated with targets brings significant improvements in performance over a low, uniform α value. Zeroing out α without taking appropriate measures to stabilize the algorithm generally results in a high ratio of false positives and a very low ratio of missed detections. Adding a minimum blob area threshold helps to restore the balance between the ratio of false positives and missed detections. With respect to the metrics used here, the zero background update rate methods with a minimum blob area threshold achieve the best performance.

- The methods involving velocity measurements also produced excellent tracking results. They consistently give greater tracking precision (less tracking error), as was seen in Figure 9.6. The "Velocity" methods, when tuned correctly, can deliver phenomenal track continuity as well. The one weakness of the "Velocity" methods is their tendency to merge close spaced stationary targets; this is the source of their high ratios of missed detections. The "Velocity" methods require that the targets be large enough and distinctive enough to obtain several features on the target.

- The ghost suppression algorithm resulted in poorer performance with every algorithm it was applied to. Its effect on the "Zero$_{\text{min}}$" methods was more serious than its effect on the "Velocity" methods. We conclude that the ghost suppression algorithm should not be used continuously in a tracking scenario. Rather, it should be used in the beginning of the scenario to remove foreground artifacts caused by moving objects present in the first frame and occasionally, but infrequently, throughout the tracking scenario to maintain an unpolluted foreground mask.

Given the results presented in the previous sections, we further tuned the R-RANSAC parameters and ran two final sets of simulations. The first method, "Zero$_{\text{opt}}$", is the "Zero$_{\text{min}}$" method with $\alpha = 0.005$ and $\tau_{\text{T}} = 10$. A guided sampling threshold of 100 was implemented for this method as well. The second method, "Velocity$_{\text{opt}}$", is the "Velocity" methed with $\alpha = 0.005$ and $\tau_{\text{T}} = 10$ and the Kalman filter-based adaptive minimum blob area threshold of the "Zero$_{\text{min}}$" methods. For both methods, the number of RANSAC iterations was increased to 35 and the following merging parameters were used: $\tau_\theta = 45°$, $\tau_v = 0.25$, and $\tau_x = \tau_y = 15$. For the "Velocity$_{\text{opt}}$" method, $\tau_{x_{\text{cluster}}} = \tau_{y_{\text{cluster}}} = 30$. The full simulation results for these methods are presented in Tables 9.3 and 9.4 respectively. Through a subtle adjustment of a few key parameters and the inclusion of the guided sampling and adaptive blob thresholds, we were able to obtain phenomenal tracking performance. The "Velocity$_{\text{opt}}$" method achieved lower ratios of false positives and mismatches along with greater tracking precision than the "Zero$_{\text{opt}}$" method achieved at the expense of a slightly higher ratio of false positives. The full results video of the "Velocity$_{\text{opt}}$" method can be viewed at `https://www.youtube.com/watch?v=VTcFrkFSkko`.

Table 9.3: Full Simulation Results - Zero$_{\text{opt}}$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	10.6970	0.9243	0.0679	0.0027	0.0051	26.7741	27.0538
Trial 2	11.0014	0.9233	0.0541	0.0191	0.0035	27.5537	28.0510
Trial 3	10.7252	0.9263	0.0673	0.0001	0.0063	28.3572	28.6917
Trial 4	10.8108	0.9389	0.0518	0.0059	0.0034	26.8634	27.3925
Trial 5	10.9063	0.9185	0.0655	0.0079	0.0081	28.4710	28.8572
Trial 6	10.9014	0.9224	0.0736	0.0001	0.0039	29.1099	29.4908
Trial 7	10.9562	0.9295	0.0648	0.0003	0.0054	27.1458	27.4584
Trial 8	10.9221	0.9262	0.0691	0.0012	0.0035	26.9714	27.2982
Trial 9	10.6869	0.9270	0.0655	0.0020	0.0055	27.5627	27.8690
Trial 10	10.8834	0.9174	0.0755	0.0039	0.0032	28.8240	29.1175
Average	10.8491	0.9254	0.0655	0.0043	0.0048	27.7633	28.1280

Overall, we are most impressed with and see the most potential in the "Velocity" methods; these methods are also the most applicable to video taken from a moving platform. A true parameter analysis of the R-RANSAC parameters and further tuning of the sequential-RANSAC parameters would likely yield further improvements in performance.

9.1.8 Execution Times

We report the average execution times for the "Position$_{\text{opt}}$" and "Velocity$_{\text{opt}}$" methods on the 1280 x 720 pedestrian video. The simulations were performed in Visual Studio 2010 on an Intel i7-2600 CPU @3.40 GHz with 8.00GB of RAM. For the "Position$_{\text{opt}}$" method, the entire

Table 9.4: Full Simulation Results - Velocity$_{\text{opt}}$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	10.0862	0.9169	0.0787	0.0000	0.0044	28.2333	28.4372
Trial 2	10.1924	0.9201	0.0763	0.0000	0.0036	27.7801	28.0205
Trial 3	10.2260	0.9294	0.0675	0.0000	0.0031	27.0152	27.2210
Trial 4	10.0941	0.9170	0.0807	0.0000	0.0023	27.8676	28.0497
Trial 5	10.2893	0.9259	0.0710	0.0000	0.0031	27.3188	27.5216
Trial 6	10.2287	0.9227	0.0751	0.0000	0.0023	27.0621	27.2455
Trial 7	10.1593	0.9259	0.0713	0.0000	0.0028	27.0661	27.2484
Trial 8	10.2930	0.9245	0.0724	0.0000	0.0031	27.2270	27.4104
Trial 9	10.1977	0.9245	0.0726	0.0000	0.0028	27.1272	27.3352
Trial 10	9.9890	0.9251	0.0721	0.0000	0.0028	26.7647	26.9494
Average	10.1756	0.9232	0.0738	0.0000	0.0030	27.3462	27.5439

code takes 0.15885 seconds per frame to run. For the "Velocity$_{\text{opt}}$" method, the entire code takes 0.17678 seconds per frame to run. The R-RANSAC portion of the code is identical for both methods and takes 0.0010 seconds per frame to run. The bulk of the execution time is due to the un-optimized foreground detector code. This portion of the code, when fully optimized, should only be incrementally slower than OpenCV's implementation of the GMM foreground detector. With OpenCV's foreground detector, the system runs at about 20 Hz.

9.2 Stationary Object Detection

R-RANSAC in conjunction with a foreground detector employing tracker-sensor feedback can be used to address a class of problems known as stationary object detection (SOD). SOD is an important topic in tracking because objects of interest often remain stationary for extended periods of time (e.g. a parked car). SOD in its truest sense, however, is not directly related to tracking. Rather, it is the problem of locating newly placed objects in a scene. SOD is especially relevant to the surveillance of public areas such as airports and train stations where abandoned luggage is a serious security concern.

Reference [41] provides a comprehensive survey on proposed stationary foreground object detection algorithms. Some notable examples include the use of dual backgrounds [42], SOD via tracking [43], and foreground mask sampling [44]. In [42], two background models are maintained, a short-term and a long-term model. Both models are based on the GMM architecture, but the long-term model is updated at a lower frequency than the short-term model. Temporarily stationary objects are quickly assimilated into the short-term model, but persist as foreground for a longer duration in the long-term model. Eventually, stationary objects fade into the background of both models. In [43], the single background model is based on a mixture of average and running average models. A connected component algorithm groups foreground pixels into blobs which are then tracked over time. A blob which has remained in the same place for 50 seconds is considered a stationary object. The authors do not specify how frequently the background model is updated, but it appears that stationary objects eventually fade into the background. Reference [44] uses foreground mask sampling to detect stationary objects. Six frames from the most recent 30 seconds of video are sampled and corresponding foreground masks are created by subtracting the six images from the background model. Stationary objects are objects that appear in all six of the foreground masks. The authors do not mention how their background model is created or updated. We observe that the examples cited here are only capable of temporarily detecting stationary objects. R-RANSAC in conjunction with the tracker-sensor feedback loop is capable of indefinitely detecting stationary objects.

9.2.1 Detecting Parked Cars

R-RANSAC was applied to two parking lot videos; the "Zero$_{\text{opt}}$" method was applied to the first video and the "Velocity$_{\text{opt}}$" methd was applied to the second video. In the first video, the parking cars are well-separated thus allowing R-RANSAC to both track the individual cars and detect areas occupied by stopped objects. These results can be seen in Figure 9.7. Several points of interest should be noted in this figure. Track 3, a moving and then parked car, persists during the entire length of the video (almost 3000 frames); it can been seen in the top right and bottom panels. Track 2, seen in the top right and bottom left panels, is a motorcyclist donning his protective gear and then riding away. Track 14, a moving and then parked car, is seen driving through the parking lot in the bottom left panel and then fully parked in the bottom right panel.

The results for the second parking lot video are shown in Figure 9.8. In this video, R-RANSAC is able to track individual cars for much of the video, but as the concentration of parked cars increases, tracks begin to coalesce. Still, by using R-RANSAC as the underlying mechanism, we are able to detect areas occupied by parked cars. Panels 1-3 of Figure 9.8 show tracks

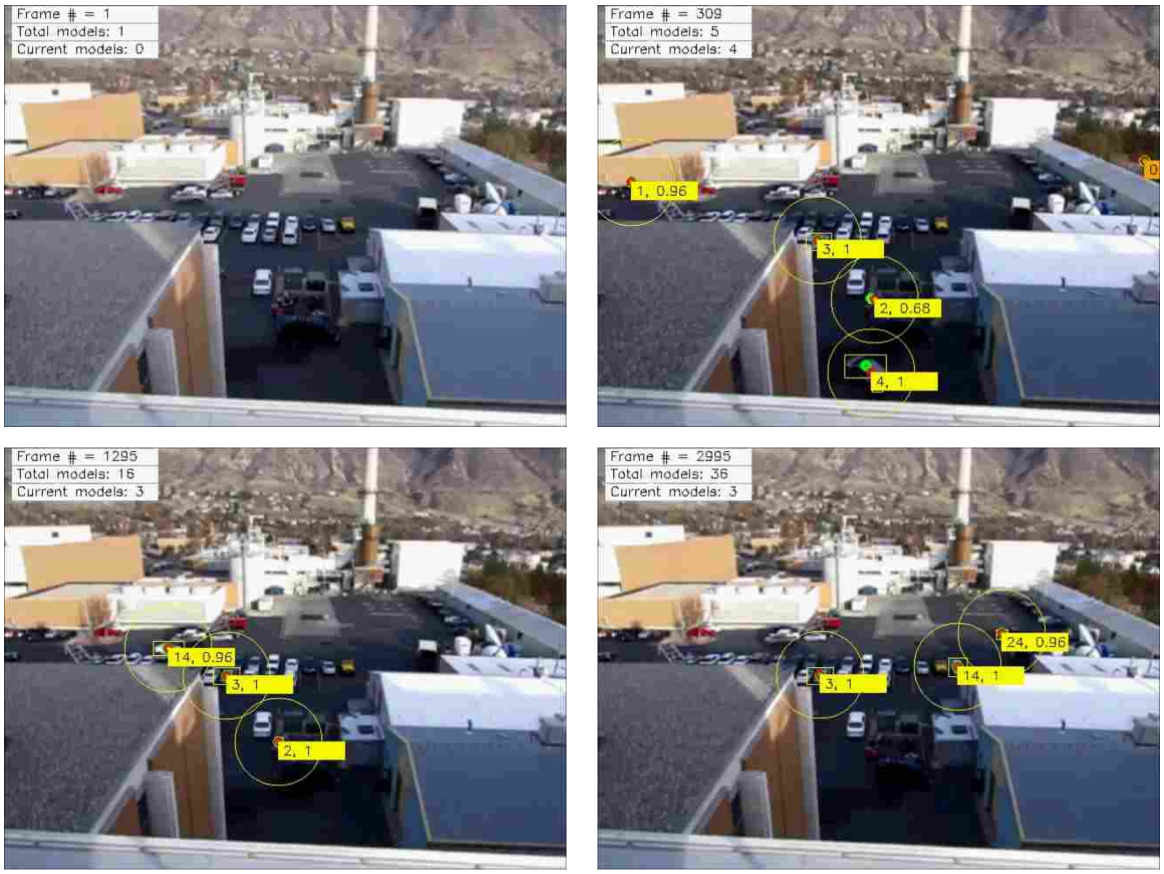


Figure 9.7: Parked car detection in the first parking lot video.

corresponding to individual cars. The tracks in the fourth panel are more convoluted, but the foreground mask clearly shows areas occupied by parked cars. Also of note is the stark change in environmental lighting during the video, transitioning from a sunrise orange to a shadowy blue.

9.2.2 Abandoned Luggage Detection

The "Velocity$_{\text{opt}}$" method was applied to the "3" image sets of the PETS 2006 dataset. In all seven of the image sets, R-RANSAC is able to detect the abandoned piece of luggage throughout the entire duration of the image sets. We present example images from the four most difficult image sets: S7-T6, S4-T5, S6-T3, and S2-T3. Corresponding results videos can be viewed at `https://www.youtube.com/playlist?list=PLxxDVxiyDNvt7tMJIEOSpa5btnppKlHip`. As described on the PETS 2006 website, the S7-T6 image set "contains a single person with a suitcase who loiters before leaving the item of luggage unattended. During this event five other people move in close proximity to the item of luggage.” Figure 9.9 shows that the abandoned piece of luggage is detected prior to the five people passing in its vicinity and continues to be detected after this event as well.

The S4-T5 image set “contains a person placing a suitcase on the ground. Following this a second person arrives and talks with the first person. The first person leaves the scene without their luggage. Distracted by a newspaper, the second person does not notice that the first persons luggage is left unattended.” Figure 9.10 shows that the abandoned piece of luggage as well as the two people are detected during the entire image set.

The S6-T3 image set “contains two people who enter the scene together. One person places a rucksack on the ground, before both people leave together (without the rucksack).” Figure 9.11 shows the people entering and leaving the scene together; the rucksack continues to be detected after their departure.

The S2-T3 image set “contains two people who enter the scene from opposite directions. One person places a suitcase on the ground, before both people leave together (without the suitcase).” Figure 9.12 shows the people entering and leaving the scene as well as the continued detection of the suitcase. The detections seen above the suitcase are from the movement of a chain of garbage bins behind the fence.

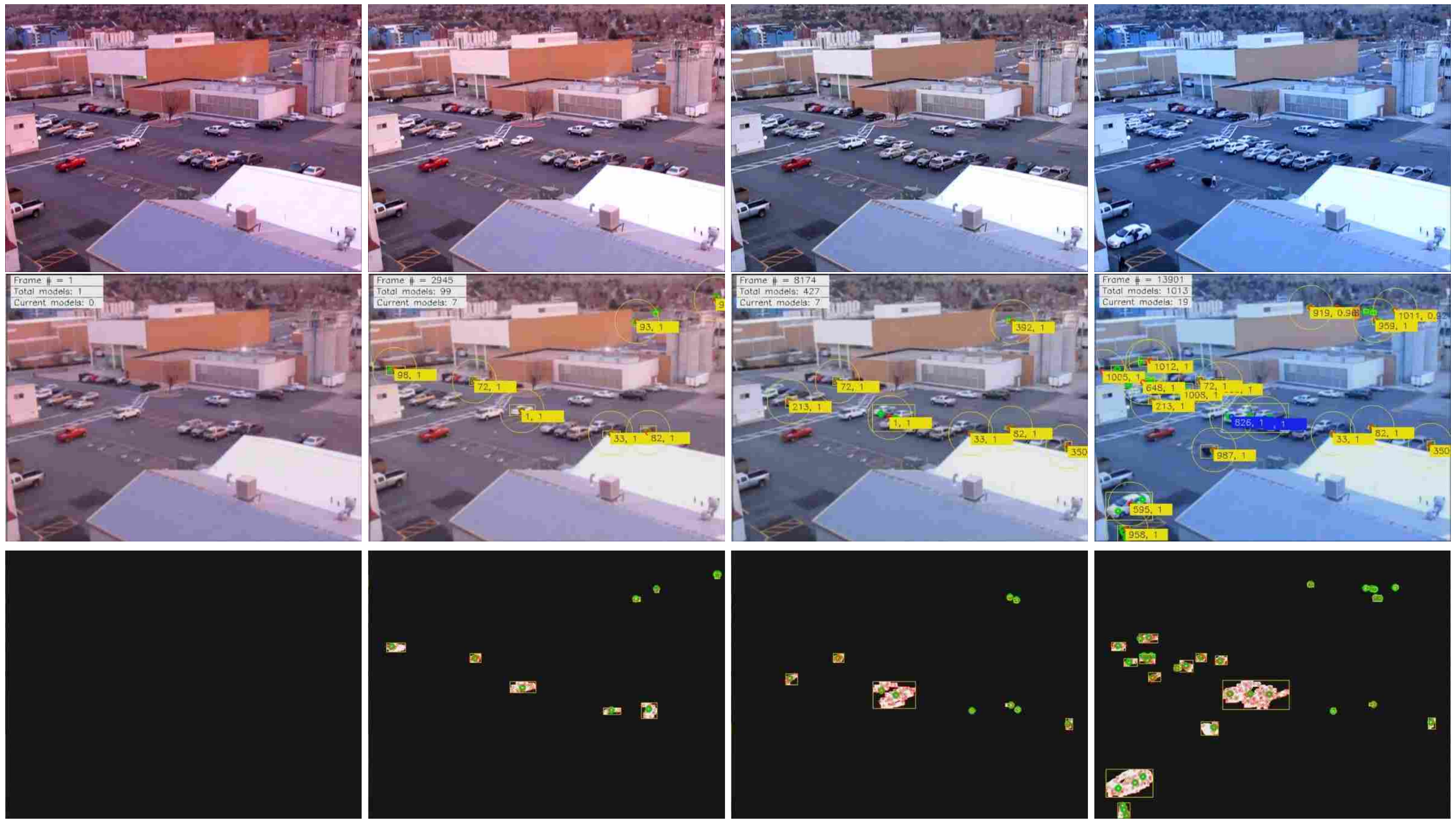


Figure 9.8: Parked car detection in the second parking lot video. Row 1 shows the original video, row 2 shows the R-RANSAC tracks, and row 3 shows the foreground mask. Column 1 shows images at frame 1, column 2 shows images at 2945, column 3 shows images at 8174, and column 4 shows images at frame 13901.

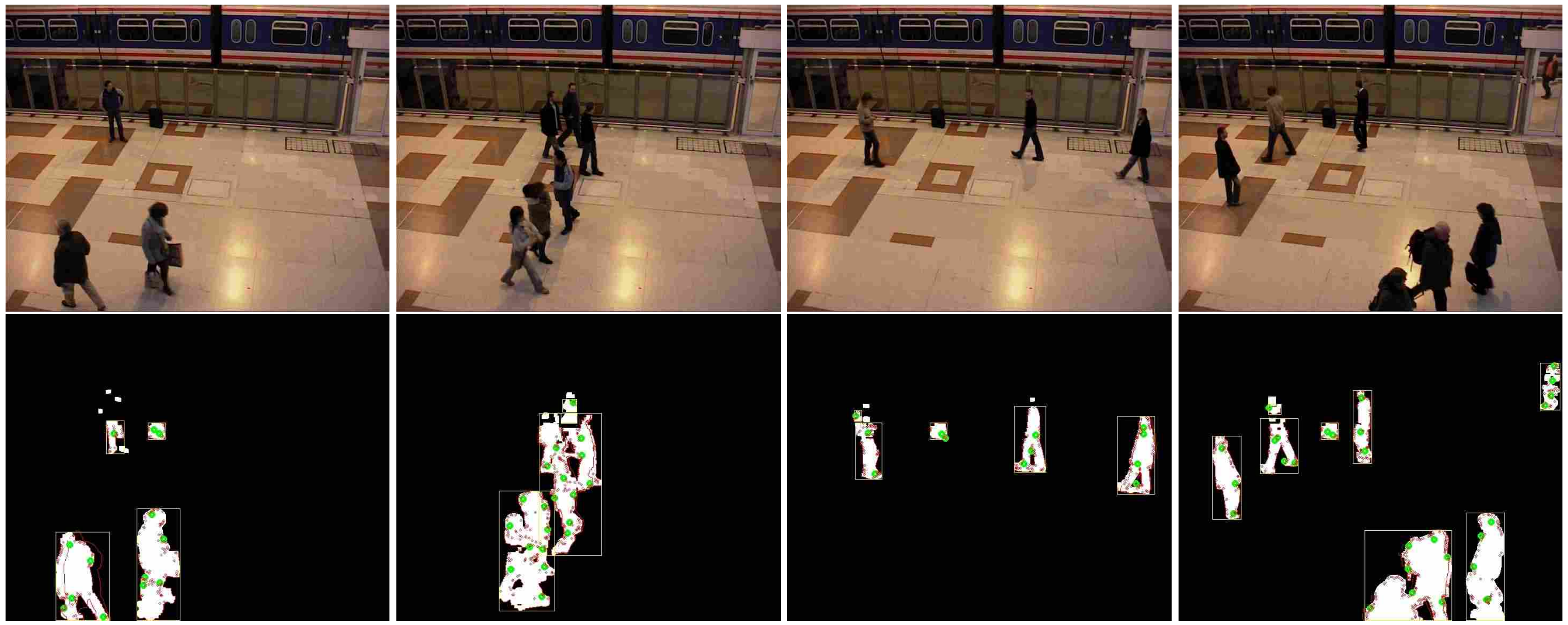


Figure 9.9: Abandoned luggage detection in the S7-T6 image set. Row 1 shows the original video and row 2 shows the foreground mask. Column 1 shows images at frame 1269, column 2 shows images at 1717, column 3 shows images at 2469, and column 4 shows images at frame 3273.

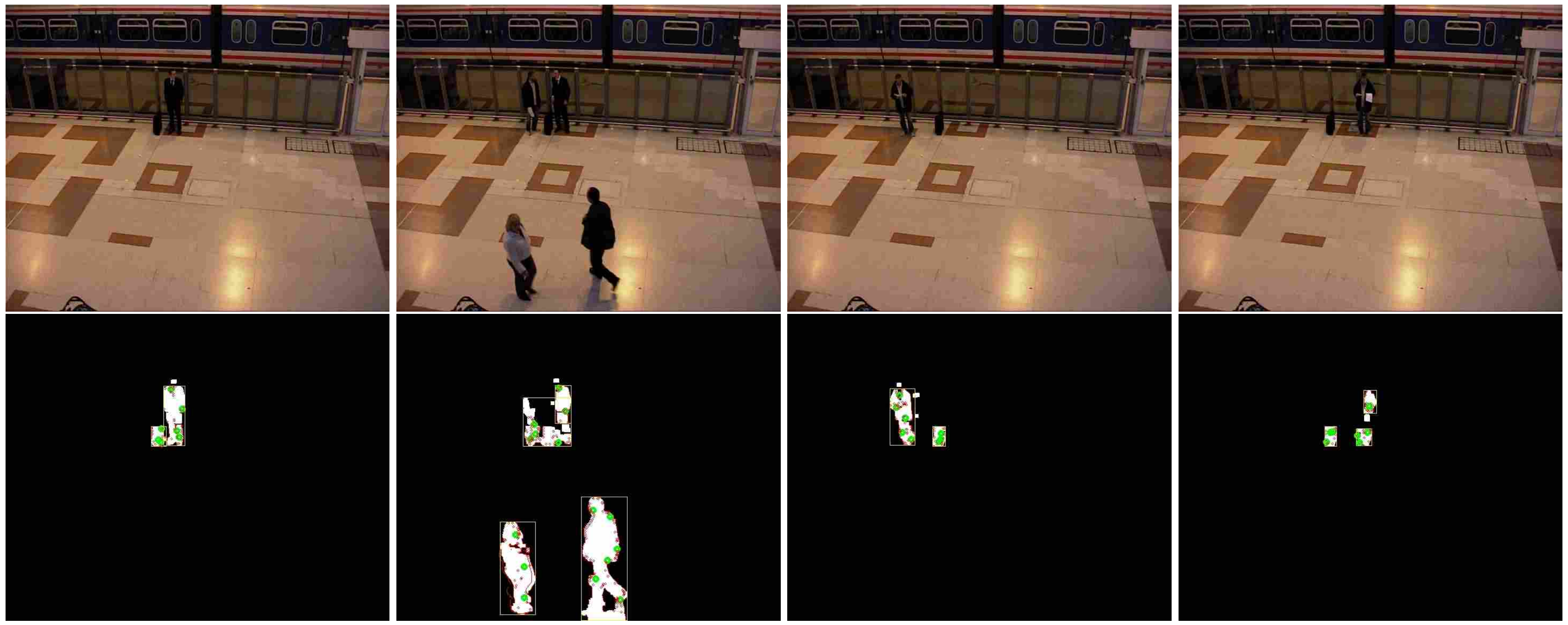


Figure 9.10: Abandoned luggage detection in the S4-T5 image set. Row 1 shows the original video and row 2 shows the foreground mask. Column 1 shows images at frame 909, column 2 shows images at 150, column 3 shows images at 2037, and column 4 shows images at frame 2773.

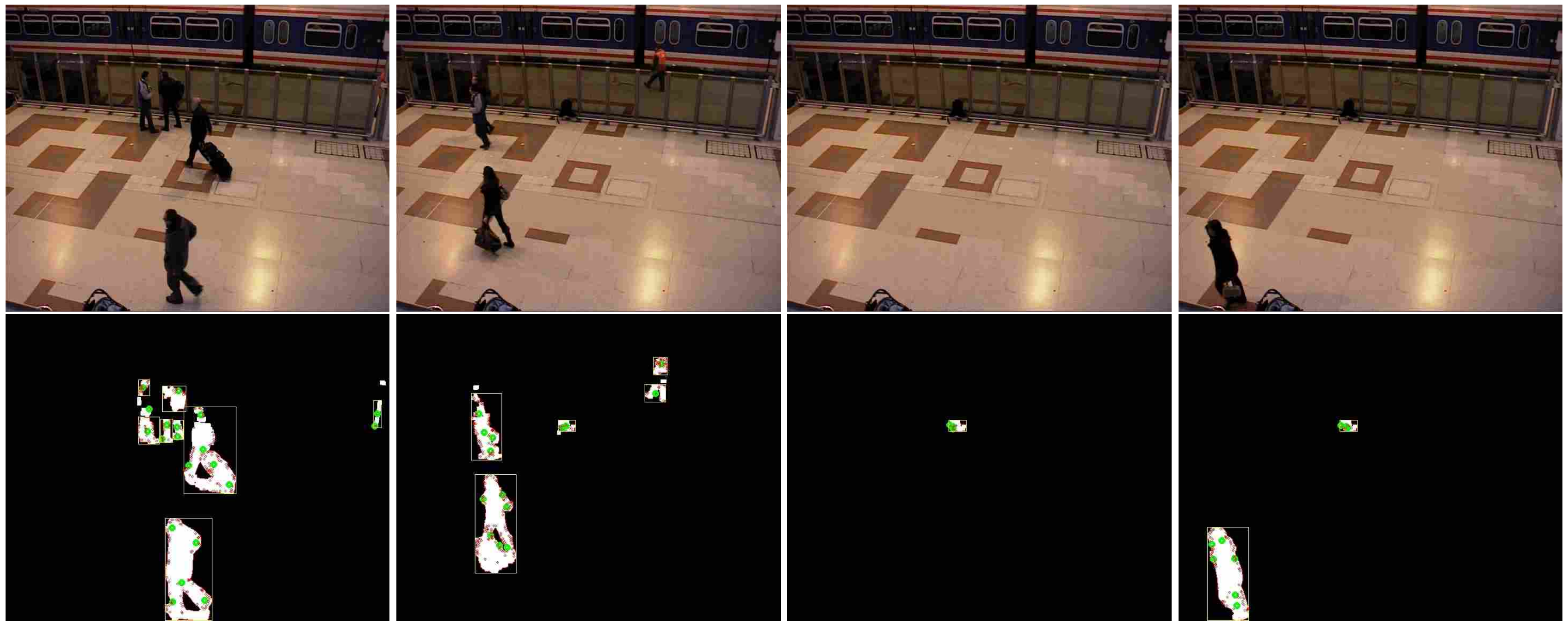


Figure 9.11: Abandoned luggage detection in the S6-T3 image set. Row 1 shows the original video and row 2 shows the foreground mask. Column 1 shows images at frame 1241, column 2 shows images at 1669, column 3 shows images at 2121, and column 4 shows images at frame 2609.

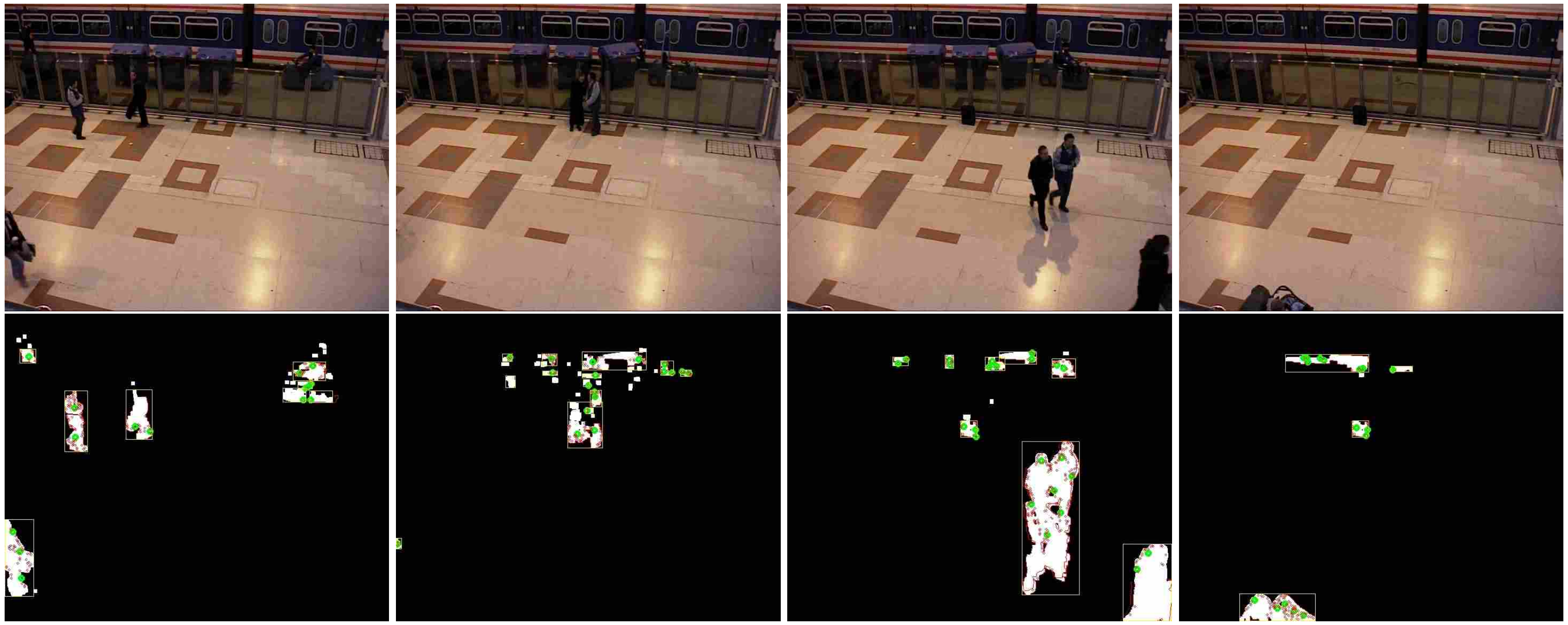


Figure 9.12: Abandoned luggage detection in the S2-T3 image set. Row 1 shows the original video and row 2 shows the foreground mask. Column 1 shows images at frame 881, column 2 shows images at 1273, column 3 shows images at 1577, and column 4 shows images at frame 2273.

CHAPTER 10. MACHINE LEARNING TO IMPROVE TRACKING

This chapter describes how tracking performance can be improved using machine learning. Section 10.1 briefly describes the machine learner used here, the Sequence Model. Section 10.2 explains how the Sequence Model was incorporated into the R-RANSAC framework. Section 10.3 describes the simulation that was developed to test the Sequence Model/R-RANSAC (SM/R-RANSAC) algorithm. Section 10.4 presents simulation results and Section 10.5 discusses conclusions about the SM/R-RANSAC algorithm and future research directions.

10.1 Sequence Model

The Sequence Model is an extension of the Sequence Memoizer, a machine learner first proposed by Wood, Gasthaus, et. al. in [45]. The Sequence Memoizer is a hierarchical Bayesian model designed to capture long-range dependencies in discrete data. The Sequence Memoizer is very appropriately named: it learns *sequences* of data. A classic application of such a learner would be in language prediction.

The SM was applied to a multiple agent, MTT problem in an urban environment in [1]. The SM, like the Sequence Memoizer, learns sequences of discrete data. Tracking usually takes place in a continuous field of view, so the first step in applying the SM to tracking is to discretize the field of view. In tracking, the SM learns sequences of target locations instead of continuous target paths. The grid size chosen to discretize the field of view is very important; an overly fine discretization makes it more difficult for the SM to learn sequences whereas an overly coarse discretization reduces the amount of usable information in the SM's prediction.

The SM creates a belief model of the field of view. For each grid location in the belief model, the probability that a target occupies that location in the future is estimated. The SM constructs this belief model in a Monte Carlo-like way by propagating forward several probable trajectories into the future. Propagating more possible trajectories forward creates a belief model

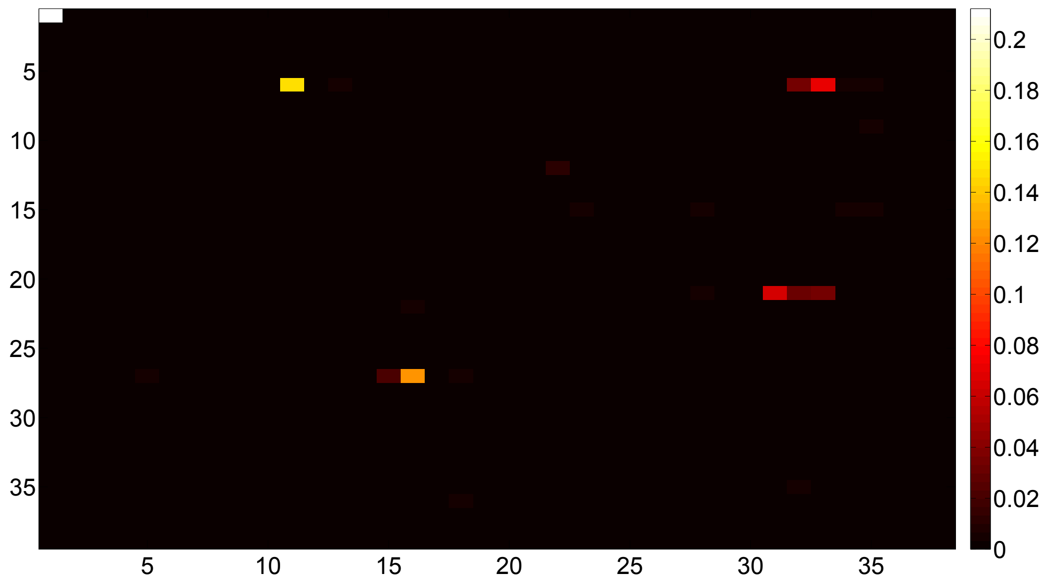


Figure 10.1: An example belief model constructed by the Sequence Model. The probability of a target occupying a given grid location at the next time step is indicated by the color scale.

that better estimates the true distribution, but also results in longer execution times. The number of possible trajectories was set at 100 in our simulations. The value of the belief model at a given grid location is the proportion of probable trajectories that passed through that location. An example belief model from the simulations presented in Section 10.3 can be seen in Figure 10.1.

Conceptually, the SM can be a powerful tool in tracking. It has the natural ability to learn road networks, including details like the locations of stoplights, one-way streets, and common U-turn locations. The SM, because it requires a discretization of the environment, can also easily incorporate prior knowledge one might have of the tracking environment. Grid locations corresponding to no travel areas (buildings instead of streets, for example), can have their probability mass zeroed out or greatly reduced. Incorporating prior knowledge about the environment into a Kalman filter estimate is much more difficult.

The SM does, however, present some difficulties when applying it to tracking. First, the SM, like all machine learning algorithms, requires a training period before it begins to make accurate predictions. Care must also be taken to only train the SM with tracks that have a high probability of representing true targets to prevent spurious tracks from corrupting the training data.

The most significant shortcoming of the SM in its current state is that it produces a single belief model describing the distribution of future locations for *all* targets. In [1], the SM is used to perform path planning for a cooperative team of UAVs with the goal of maximizing the number of targets seen. The shared belief model does not conflict with this objective. The authors also assume perfect data association; unfortunately, data association is one of the most difficult problems in real tracking scenarios. Because of this shared belief model, R-RANSAC cannot leverage the SM to associate measurements of closely spaced targets; the SM can only be used to distinguish between clutter measurements and target-associated measurements of widely spaced targets.

10.1.1 Other Learning Approaches

Machine learning is often applied to video-based target tracking in order to learn a visual appearance model of the targets. Reference [46] provides a representative example of this class of algorithm; when targets are not interacting, they are tracked with individual trackers and an appearance model is constructed. This appearance model includes positive examples (templates of the correct target) and negative examples (templates of other targets). When targets interact, the appearance models are used to distinguish between the targets. This use of machine learning is fundamentally different to the use proposed here. We seek to learn target trajectories as opposed to target appearances.

In our literature review, we only came across one other example of using machine learning to learn target trajectories. In [47], a motion map is used to learn non-linear motion patterns in the scene. This motion model helps to connect small sequences of associated measurements known as tracklets into larger sequences known as tracks. An affinity score is calculated between tracklets and learned motion patterns using the head and tail positions and velocities of the different segments. The motion pattern that receives the highest affinity score is used to connect the tracklets. This is an online learning algorithm; the motion model is constructed during tracking with high confidence tracklets. Reference [48] extends this method by using a Conditional Random Field to model the track affinities and dependencies, and by calculating the affinity scores globally. This method differs from our method in that it is a post-processing algorithm.

10.2 Incorporating the Sequence Model with R-RANSAC

When incorporating the SM into the R-RANSAC framework, we primarily use it as a tool to improve data association. We expect using the SM to be most advantageous in situations with infrequent measurement updates and a high proportion of clutter measurements. R-RANSAC relies entirely on its assumed dynamic model to perform data association. In the case of infrequent measurement updates, a target can deviate significantly from its assumed model, thereby making it much less probable that R-RANSAC correctly associates measurements. The SM does not share this inherent weakness of R-RANSAC; its belief model is constructed entirely from past observations of the target. Consequently, provided the SM has been sufficiently trained and the target does not deviate from its previous paths, the SM should be able to correctly associate measurements independent of the time between measurement updates. The simple example of a road with a right-hand turn clearly illustrates this point. Just prior to the turn, R-RANSAC propagates the state estimate forward and off the road. The SM, on the other hand, has never observed a target continuing straight on this section of road. Instead, it has observed targets making a right hand turn at that location and thus it assigns probability mass to the right.

R-RANSAC measurement association probabilities are calculated using the covariance of the innovation, as was done in Section 9.1.5. The SM association probabilities are calculated by interpolating between grid locations in the belief model. Both sets of association probabilities are normalized. A weighting between the R-RANSAC and SM association probabilities is computed. Several approaches have been suggested on how to best weight these probabilities including using mean-based, entropy-based, or random weightings. Ideally, the weighting would be based on the certainty of each prediction. We attempt to approximate this ideal weighting by allowing the SM weight to grow linearly from zero to an upper bound $P(\mathrm{SM})_{\max}$. The SM weight reaches $P(\mathrm{SM})_{\max}$ when the track's age reaches ℓ_{SM} time steps; after ℓ_{SM} time steps we are confident the SM has been sufficiently trained. In the simulations presented here, $P(\mathrm{SM})_{\max} = 0.9$ and $\ell_{\mathrm{SM}} = 300$.

When used with R-RANSAC, the SM is initialized with an arbitrarily large number of targets. All valid tracks outputted by R-RANSAC are used to update the SM. The R-RANSAC track with good model number i updates the belief model for the i^{th} target in the SM. A high τ_{T} value of 15 is used to prevent spurious tracks from corrupting the SM. The SM is updated with the highest probability measurement associated with a track instead of the track's state estimate.

Valid tracks that were not updated at the current time step are not used to update the SM. The inlier region of R-RANSAC is expanded to account for possibly extreme intra-time step target maneuvers ($\tau_R = 70$). A grid size of 15 is used to discretize the environment.

Because the SM/R-RANSAC algorithm is designed to excel in tracking situations with infrequent measurement updates, the CV model is used in place of the CJ model. The CJ model performs poorly in these situations because its sensitive higher-order terms result in inaccurate predictions of future target locations far into the future. The CV model, although it does not model turning behavior, is less sensitive to errors in the state estimate and oftentimes more accurately predicts future target locations. The CV model also behaves in a more stable fashion when a target has maneuvered between time steps. In the case of a right hand turn, the CJ model requires several post-turn measurements to converge to the true path. Conversely, the CV model snaps to the true path after receiving a single post-turn measurement.

10.3 Learning Simulation Environment

As described in Section 10.1, the SM is still under active development and can only be reasonably expected to improve tracking performance in certain situations. To fairly evaluate the performance of the SM/R-RANSAC combination, the simulated MTT scenario was designed to showcase the expected strengths of the new algorithm.

A MTT scenario was simulated in Matlab in order to test the SM/R-RANSAC algorithm. The simulation environment consists of a 550 x 550 field of view with four targets. Each target travels in a rectangular path, with each path being located almost wholly in one of the quadrants of the field of view. The target measurements are corrupted by normally-distributed noise with a standard deviation of 0.5. The simulation begins with a 600 time step learning period; this learning period contains no clutter measurements. After the learning period, alternating 400-time step periods with clutter and 200-time step periods without clutter occur until time step 2800 (the end of the simulation). The time periods with clutter are referred to as the "jamming" periods throughout this discussion. The periods without clutter measurements are designed to allow R-RANSAC to reacquire the targets and establish context for the learner. Figure 10.2 displays the number of clutter measurements over the entire simulation. The number of clutter measurements during each "jamming" period builds to a peak and then falls back to zero. Each successive "jamming" period

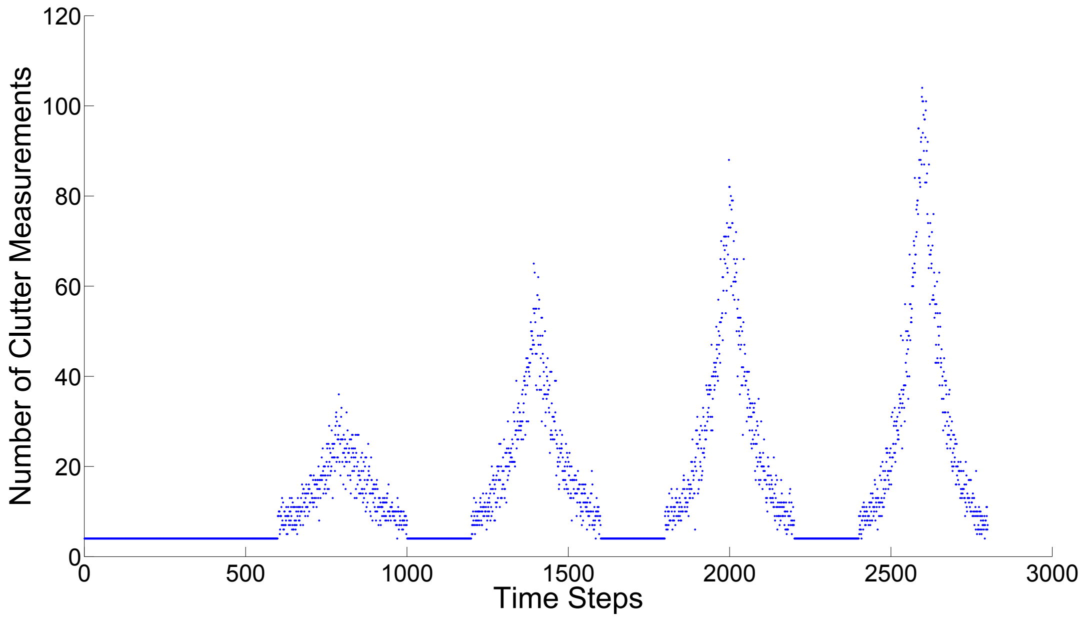


Figure 10.2: The number of clutter measurements at each time step in the learning simulations.

rises to a higher maximum number of clutter measurements resulting in more difficult tracking as the simulation progresses. Figure 10.3 shows a snapshot of the simulation environment during a period of clutter measurements.

To make R-RANSAC compatible with this particular simulation, two modifications are necessary. First, tracks' consensus sets are updated with *all* of the inlier measurements instead of only with the nearest neighbor measurement. Second, an adaptive τ_ρ is used. A baseline value of 0.8 is assumed for τ_ρ; this baseline value is added to the average number of expected inlier measurements based on the average size of the measurement scans in the measurement history window, the area of the field of view, and the inlier region area. This adaptive τ_ρ appears to work well during all stages of the simulation. These two modifications are necessary to distinguish valid tracks from spurious tracks during the "jamming" stages of the simulation.

10.4 Learning Results

Three sets of simulations were run in order to compare R-RANSAC with the SM/R-RANSAC algorithm. The simulations are differentiated by the length of their time steps: 9, 13, and 17 seconds, respectively. In Tables 10.1 and 10.2, the simulations are labeled as "Learning - *dt*" and

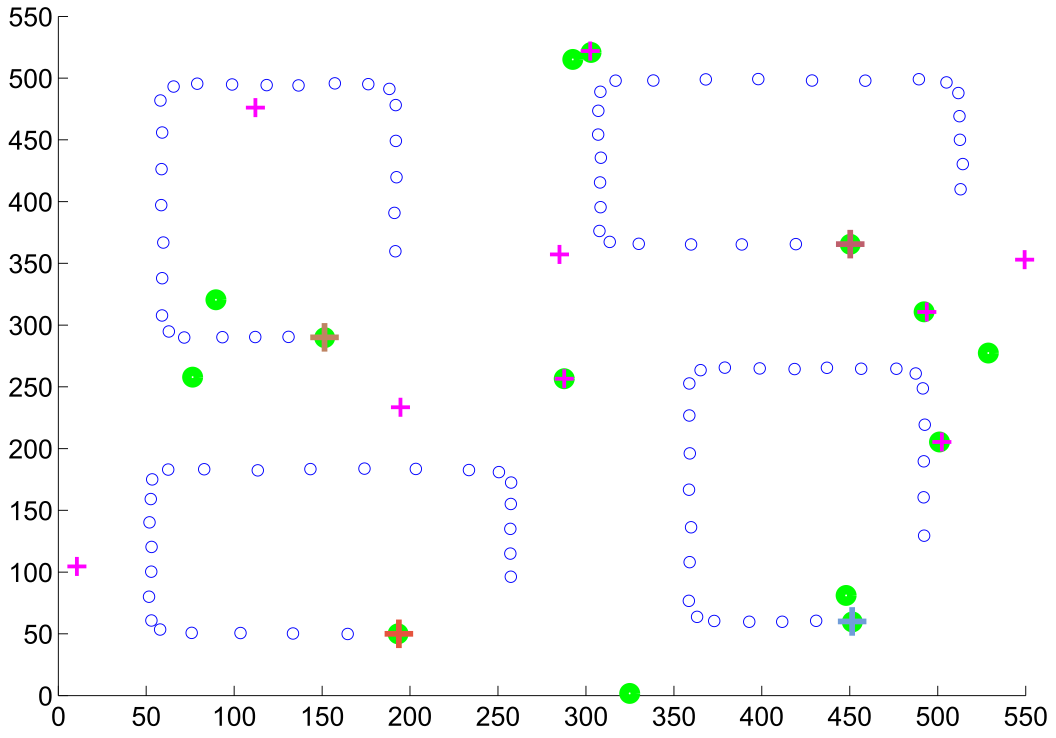


Figure 10.3: A snapshot of the simulation environment during a period of clutter measurements. The measurements of the current time step are shown as green circles. The most recent 25 measurements generated by the true targets are shown as smaller, blue circles. The good R-RANSAC tracks are displayed as randomly colored, larger crosses. The hypothesis tracks are displayed as smaller, magenta crosses.

"No Learning - *dt*" where *dt* indicates the time step length. Five trials were run with each simulation; full simulation results can be found in Appendix B. Results were only extracted from the post-learning period interval of the simulation (i.e. the last 2200 time steps). The R-RANSAC parameters were kept constant across all of the trials, except for the measurement noise covariance which was increased by one order of magnitude for the 17 second time step trials.

For all time step lengths, the SM/R-RANSAC algorithm outperformed R-RANSAC. As expected, the disparity in performance increases as the time step grows larger. This is because the Kalman filter estimate of future target location used by R-RANSAC becomes increasingly less trustworthy with longer time steps, whereas the SM estimate is less affected by longer time steps. One interesting observation is that SM/R-RANSAC's performance remained relatively constant

Table 10.1: MOT Results - R-RANSAC with the Sequence Model

Simulation	MOTP	MOTA	MD	FP	MM
Learning - 9	0.4960	0.9793	0.0000	0.0203	0.0004
No Learning - 9	0.4452	0.9428	0.0207	0.0281	0.0083
Learning - 13	1.1883	0.9818	0.0000	0.0174	0.0007
No Learning - 13	1.7101	0.7896	0.1500	0.0088	0.0515
Learning - 17	11.2009	0.8690	0.0085	0.0911	0.0314
No Learning - 17	15.3591	0.5766	0.2718	0.0233	0.1284

Table 10.2: Average OSPA Scores - R-RANSAC with the Sequence Model

Simulation	OSPA	OSPA-T	Standard Deviation (OSPA-T)
Learning - 9	4.1499	4.1786	0.4883
No Learning - 9	12.8371	13.0207	1.0472
Learning - 13	4.3613	4.3839	0.5872
No Learning - 13	35.8561	36.4852	1.2125
Learning - 17	20.3627	20.5861	0.7797
No Learning - 17	60.5101	61.4912	0.3926

between the 9 and 13 second simulations. This result is especially relevant to UAVs which often have strict limits on computational power: the same level of tracking performance can be achieved even when receiving measurements and updating the model set much less frequently.

Figure 10.4 plots the average OSPA-T scores at each time step for the five trials of the "Learning - 17" and "No Learning - 17" simulations. During the periods without clutter measurements, R-RANSAC tracks very well and SM-R-RANSAC tracks perfectly. During the periods with clutter measurements, SM/R-RANSAC performs noticeably better than R-RANSAC. Figure 10.4 also shows that R-RANSAC's performance degrades as soon as the clutter measurements begin and its performance only improves after the clutter measurements have ended. SM/R-RANSAC, on the other hand, maintains excellent performance through the early stages of the "jamming" periods and strongly recovers before the "jamming" periods are over.

There is a significant time penalty associated with using the SM. R-RANSAC averaged 2.6111×10^{-6} seconds per time step whereas the SM/R-RANSAC algorithm averaged 2.3889×10^{-5} seconds per time step.

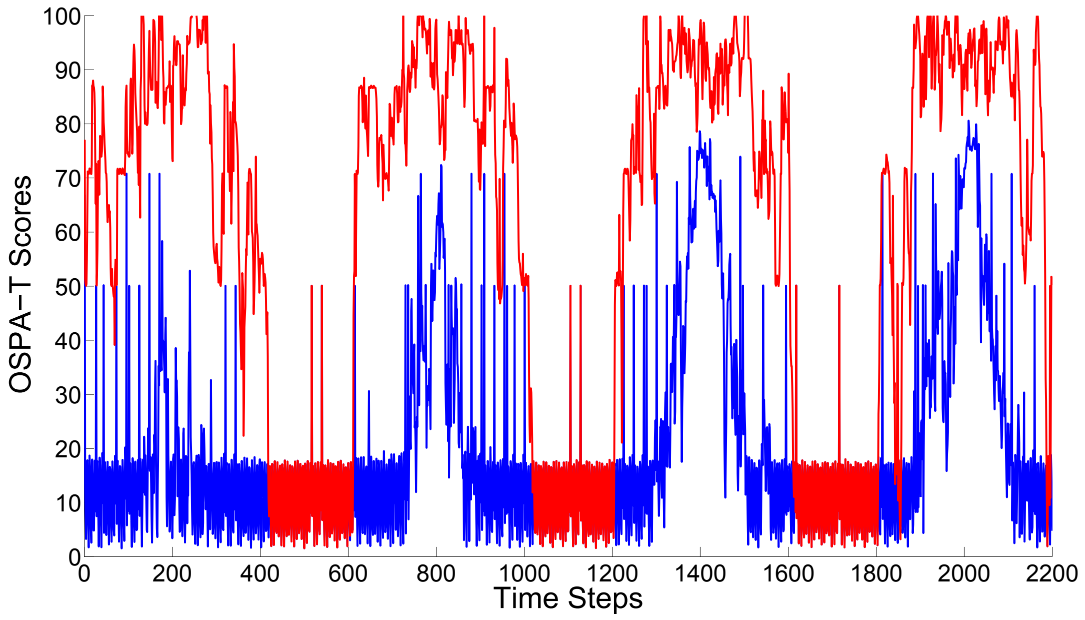


Figure 10.4: The OSPA-T scores for a single trial of "Learning - 17 (blue)" and "No Learning - 17" (red).

10.5 Conclusions

The ideas and results presented in this chapter are more proof-of-concept than fully developed. The simulation results showed that the SM significantly improved data association in situations with infrequent measurement updates and high amounts of clutter. However, this simulation was very simplistic: the targets were constrained to easily-learned and widely spaced paths. Future research should include simulations with interacting targets that deviate occasionally from their nominal paths. Eventually, experiments should be run on video where the objects of interest do not follow constrained paths. Future work will also need to look at determining the optimal discretization of the environment and the optimal way of combining the SM and R-RANSAC belief models. All of this future work, though, hinges on the continued development of the SM. The SM needs to be able to maintain separate belief models for individual targets for it to be successfully applied to more realistic tracking scenarios.

CHAPTER 11. CONCLUSION

This thesis has three primary contributions: completion of the development of R-RANSAC, the development of several novel stationary foreground detection techniques, and the application of machine learning to the MTT problem.

We focused our development of R-RANSAC in three areas: data association, filtering, and track management. Our work with data association consisted of:

- Incorporating the all neighbors, nearest neighbor, probabilistic data association, and joint probabilistic data association filters into the R-RANSAC framework.
- Conducting a comparison between the data association methods which included specific track examples demonstrating the strengths and weaknesses of the different methods.

We conclude that, of the data association methods investigated, the probabilistic data association filter is the most well-suited to R-RANSAC. Track examples show that is is more robust in situations with closely spaced targets and occlusions than less sophisticated data association methods. In our data association study, we also observed that R-RANSAC relies on track coalescence to a certain degree to reduce the number of false positive tracks. For this reason, the JPDA filter performs worse than the PDA filter in the R-RANSAC framework.

Our work with filtering consisted of:

- Replacing the nearly constant velocity model with higher-order linear models.
- Incorporating the interacting multiple models algorithm into R-RANSAC.

We conclude that the nearly constant jerk model is capable of efficiently tracking most objects of interest. We also observed that the IMM filter is not fully utilized in the R-RANSAC framework. R-RANSAC requires a large sensor noise covariance to produce smooth tracks and accurately predict future target locations. We demonstrated that a large sensor noise covariance significantly limits the amount of IMM mode switching, even during extreme maneuvers.

Our work with track management consisted of:

- Converting R-RANSAC from a multiple object filter into a multiple target tracker through the good model number track labeling system.
- Introducing an intuitive and customizable track merging heuristic.
- Using a guided sampling threshold in the RANSAC-based track initialization algorithm to improve the quality of new hypothesis tracks.
- Changing the way the consensus set is updated to preserve the inlier ratio as a true measure of a track's support.

We also explored the concept of a tracker-sensor feedback loop: R-RANSAC tracking results inform how video processing is performed. We developed several variations of the Gaussian mixture models foreground detector with the goal of being able to detect stationary objects and resolve merged measurements produced by interacting targets. By zeroing out the background update rate of target-associated pixels we are able to indefinitely detect stationary objects. Furthermore, by placing a threshold on the minimum blob area we are able to prevent large groups of false positive foreground pixels from persisting.

In order to resolve merged measurements, we borrowed a technique from moving camera foreground detection. We perform frame-to-frame feature matching, superimpose the foreground mask, and cluster the optical flow vectors according to which blob they belong to. The optical flow vector clusters are further refined using sequential-RANSAC, thus decomposing a single merged measurement into several distinct, target-associated measurements. Additionally, by using sequential-RANSAC inside the R-RANSAC framework, we irrefutably show that the two algorithms are fundamentally different.

The final contribution of this thesis is an explanation and demonstration of how machine learning can improve target tracking. The Sequence Model learns target trajectories and is leveraged to improve data association in tracking scenarios with infrequent measurement updates and a high proportion of clutter measurements.

REFERENCES

[1] Cook, K., Bryan, E., Yu, H., Bai, H., Seppi, K., and Beard, R., 2013. "Intelligent Cooperative Control for Urban Tracking." *Journal of Intelligent & Robotic Systems,* **74**(1-2), Sept., pp. 251–267. 1, 84, 86

[2] Ahmad, A., Nascimento, T., Conceição, A. G. S., Moreira, A. P., and Lima, P., 2013. "Perception-Driven Multi-Robot Formation Control." In *2013 IEEE International Conference on Robotics and Automation*, IEEE, pp. 1851–1856. 1

[3] Dames, P., Thakur, D., Schwager, M., and Kumar, V., 2014. "Playing Fetch with Your Robot: The Ability of Robots to Locate and Interact with Objects." *IEEE Robotics & Automation Magazine,* **21**(2), June, pp. 46–52. 1

[4] Adams, M., Vo, B.-N., Mahler, R., and Mullane, J., 2014. "SLAM Gets a PHD: New Concepts in Map Estimation." *IEEE Robotics & Automation Magazine,* **21**(2), June, pp. 26–37. 1

[5] Lee, C. S., Nagappa, S., Palomeras, N., Clark, D. E., and Salvi, J., 2014. "SLAM with SC-PHD Filters: An Underwater Vehicle Application." *IEEE Robotics & Automation Magazine,* **21**(2), June, pp. 38–45. 1

[6] Fortmann, T., Bar-Shalom, Y., and Scheffe, M., 1980. "Multi-Target Tracking Using Joint Probabilistic Data Association." In *1980 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, IEEE, pp. 807–812. 2, 33

[7] Reid, D., 1979. "An Algorithm for Tracking Multiple Targets." *IEEE Transactions on Automatic Control,* **24**(6), Dec., pp. 843–854. 2

[8] Crouse, D. F., Willett, P., and Bar-Shalom, Y., 2011. "Developing a Real-Time Track Display That Operators Do Not Hate." *IEEE Transactions on Signal Processing,* **59**(7), July, pp. 3441–3447. 2

[9] Ba-Ngu Vo, Sumeetpal Singh, and Doucet, A., 2003. Sequential Monte Carlo Implementation of the PHD Filter for Multi-Target Tracking. 3

[10] Vo, B.-N., and Ma, W.-K., 2006. "The Gaussian Mixture Probability Hypothesis Density Filter." *IEEE Transactions on Signal Processing,* **54**(11), Nov., pp. 4091–4104. 3, 50

[11] Reuter, S., Ba-Tuong Vo, Ba-Ngu Vo, and Dietmayer, K., 2014. "The Labeled Multi-Bernoulli Filter." *IEEE Transactions on Signal Processing,* **62**(12), June, pp. 3246–3260. 3

[12] Niedfeldt, P. C., 2014. "A Novel Multiple Target Tracking Algorithm in Clutter: Recursive-RANSAC." PhD thesis, Brigham Young University. 3, 9, 50, 55

[13] Fischler, M. A., and Bolles, R. C., 1981. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography." *Communications of the ACM,* **24**(6), June, pp. 381–395. 3, 6

[14] Torr, P. H. S. "Geometric Motion Segmentation and Model Selection.". 3, 66

[15] Wang, C.-C., 2009. "Multiple-Model RANSAC for Ego-Motion Estimation in Highly Dynamic Environments." In *2009 IEEE International Conference on Robotics and Automation*, IEEE, pp. 3531–3538. 3

[16] Tanaka, K., and Kondo, E., 2006. "Incremental RANSAC for Online Relocation in Large Dynamic Environments." In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, pp. 68–75. 4

[17] Vedaldi, A., Favaro, P., and Soatto, S., 2005. "KALMANSAC: Robust Filtering by Consensus." In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Vol. 1, IEEE, pp. 633–640 Vol. 1. 4

[18] Comaniciu, D., and Meer, P., 2002. "Mean Shift: A Robust Approach Toward Feature Space Analysis." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **24**(5), May, pp. 603–619. 5

[19] Bouguet, J.-y. "Pyramidal Implementation of the Lucas Kanade Feature Tracker.". 5

[20] Zheng, B., Xu, X., Dai, Y., and Lu, Y., 2012. "Object Tracking Algorithm Based on Combination of Dynamic Template Matching and Kalman Filter." In *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics*, Vol. 2, IEEE, pp. 136–139. 5

[21] Adhikari, G., and Das, B., 2011. "A Fast Template Matching Algorithm for Aerial Object Tracking." In *2011 International Conference on Image Information Processing*, IEEE, pp. 1–6. 5

[22] Huang, T., Zeng, Z., Li, C., and Leung, C. S., eds., 2012. *Neural Information Processing.*, Vol. 7667 of *Lecture Notes in Computer Science* Springer Berlin Heidelberg, Berlin, Heidelberg. 5

[23] Niedfeldt, P., and Beard, R., 2013. "Recursive RANSAC: Multiple Signal Estimation with Outliers." In *Nonlinear Control Systems*, Vol. 9, pp. 430–435. 9

[24] Bar-Shalom, Y., Daum, F., and Huang, J., 2009. "The Probabilistic Data Association Filter." *IEEE Control Systems Magazine,* **29**(6), Dec., pp. 82–100. 10, 32

[25] Niedfeldt, P. C., and Beard, R. W. "Multiple Target Tracking using Recursive RANSAC.". 11

[26] Munkres, J., 1957. "Algorithms for the Assignment and Transportation Problems." *Journal of the Society for Industrial and Applied Mathematics,* **5**(1), pp. 32–38. 20

[27] Schuhmacher, D., Vo, B.-T., and Vo, B.-N., 2008. "A Consistent Metric for Performance Evaluation of Multi-Object Filters." *IEEE Transactions on Signal Processing,* **56**(8), Aug., pp. 3447–3457. 20

[28] Ristic, B., and Clark, D., 2011. "A Metric for Performance Evaluation of Multi-Target Tracking Algorithms." *IEEE Transactions on Signal Processing,* **59**(7), July, pp. 3452–3457. 20

[29] Bernardin, K., and Stiefelhagen, R., 2008. "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics." *EURASIP Journal on Image and Video Processing,* **2008**, Feb., pp. 1–10. 21

[30] Bar-Shalom, Y., Willett, P., and Tian, X., 2011. *Tracking and Data Fusion: A Handbook of Algorithms*. YBS Publishing. 33, 44

[31] Mehrotra, K., 1997. "Jerk Model for Tracking Highly Maneuvering Targets." *IEEE Transactions on Aerospace and Electronic Systems,* **33**(4), pp. 1094 – 1105. 41

[32] Pitre, R. R., Jilkov, V. P., and Li, X. R. "A Comparative Study of Multiple-Model Algorithms for Maneuvering Target Tracking." *Proc. SPIE 5809, Signal Processing, Sensor Fusion, and Target Recognition XIV, 549*. 44

[33] Clark, D., Panta, K., and Vo, B.-n., 2006. "The GM-PHD Filter Multiple Target Tracker." In *2006 9th International Conference on Information Fusion*, IEEE, pp. 1–8. 50

[34] Pasha, S. A., Vo, B.-N., Tuan, H. D., and Ma, W.-K., 2009. "A Gaussian Mixture PHD Filter for Jump Markov System Models." *IEEE Transactions on Aerospace and Electronic Systems,* **45**(3), July, pp. 919–936. 50

[35] Barnich, O., and Van Droogenbroeck, M., 2011. "ViBe: A Universal Background Subtraction Algorithm for Video Sequences." *IEEE Transactions on Image Processing,* **20**(6), June, pp. 1709–24. 57, 64

[36] Stauffer, C., and Grimson, W., 1999. "Adaptive Background Mixture Models for Real-Time Tracking." In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, Vol. 2, IEEE Comput. Soc, pp. 246–252. 57

[37] P. Kaewtrakulpong, R. B. "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection." *2nd European Workshop on Advanced Video Based Surveillance Systems, AVBS01, VIDEO BASED SURVEILLANCE SYSTEMS: Computer Vision and Distributed Processing*. 57

[38] Pnevmatikakis, A., and Polymenakos, L., 2006. *Machine Learning for Multimodal Interaction.*, Vol. 4299 of *Lecture Notes in Computer Science* Springer Berlin Heidelberg, Berlin, Heidelberg. 62, 64

[39] Kim, J., Wang, X., Wang, H., Zhu, C., and Kim, D., 2012. "Fast Moving Object Detection with Non-Stationary Background." *Multimedia Tools and Applications,* **67**(1), Apr., pp. 311–335. 66

[40] Leonardis, A., Bischof, H., and Pinz, A., eds., 2006. *Computer Vision ECCV 2006.*, Vol. 3951 of *Lecture Notes in Computer Science* Springer Berlin Heidelberg, Berlin, Heidelberg. 66

[41] Bayona, A., SanMiguel, J. C., and Martinez, J. M., 2009. “Comparative Evaluation of Stationary Foreground Object Detection Algorithms Based on Background Subtraction Techniques.” In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, IEEE, pp. 25–30. 75

[42] Porikli, F., 2007. “Detection of Temporarily Static Regions by Processing Video at Different Frame Rates.” In *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, IEEE, pp. 236–241. 75

[43] Miguel, J. C. S., and Martínez, J. M., 2008. “Robust Unattended and Stolen Object Detection by Fusing Simple Algorithms.” In *2008 IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance*, IEEE, pp. 18–25. 75, 76

[44] Liao, H.-H., Chang, J.-Y., and Chen, L.-G., 2008. “A Localized Approach to Abandoned Luggage Detection with Foreground-Mask Sampling.” In *2008 IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance*, IEEE, pp. 132–139. 75, 76

[45] Wood, F., Gasthaus, J., Archambeau, C., James, L., and Teh, Y. W., 2011. “The Sequence Memoizer.” *Communications of the ACM,* **54**(2), Feb., p. 91. 84

[46] Song, X., Cui, J., Zha, H., and Zhao, H., 2008. *Vision-Based Multiple Interacting Targets Tracking via On-Line Supervised Learning - Computer Vision ECCV 2008.*, Vol. 5304 of *Lecture Notes in Computer Science* Springer Berlin Heidelberg, Berlin, Heidelberg, Oct. 86

[47] Yang, B., Huang, C., and Nevatia, R., 2011. “Learning Affinities and Dependencies for Multi-Target Tracking Using a CRF Model.” In *CVPR 2011*, IEEE, pp. 1233–1240. 86

[48] Nevatia, R., 2012. “Multi-Target Tracking by Online Learning of Non-Linear Motion Patterns and Robust Appearance Models.” In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, pp. 1918–1925. 86

APPENDIX A. FULL SIMULATION RESULTS FROM CHAPTER 9

Table A.1: Full Simulation Results - Baseline- high

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	7.5822	0.6990	0.2641	0.0023	0.0347	57.4381	58.5284
Trial 2	7.5999	0.6696	0.2717	0.0145	0.0441	57.7698	58.6401
Trial 3	8.7440	0.6961	0.2190	0.0152	0.0697	57.6521	59.0511
Trial 4	8.0920	0.6957	0.2405	0.0147	0.0491	57.7091	58.9248
Trial 5	7.9957	0.7033	0.2250	0.0096	0.0621	56.5362	57.7615

Table A.2: Full Simulation Results - Baseline - low

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	11.7485	0.8175	0.0783	0.0860	0.0183	48.9670	50.1087
Trial 2	11.4553	0.8524	0.0533	0.0498	0.0445	45.0487	46.2762
Trial 3	11.8237	0.8315	0.0756	0.0718	0.0211	46.6476	48.8707
Trial 4	11.2651	0.8197	0.0815	0.0550	0.0437	47.1359	49.7353
Trial 5	11.4224	0.8043	0.0964	0.0550	0.0443	46.8461	48.6458

Table A.3: Full Simulation Results - Adaptive

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	10.2743	0.7963	0.1512	0.0100	0.0425	48.4815	50.7288
Trial 2	10.7178	0.8039	0.1359	0.0397	0.0206	50.2255	51.6448
Trial 3	10.2809	0.7836	0.1410	0.0269	0.0486	48.7575	50.3411
Trial 4	10.1307	0.7801	0.1967	0.0059	0.0174	50.6869	51.2779
Trial 5	10.2729	0.7895	0.1737	0.0040	0.0328	49.5152	50.7423

Table A.4: Full Simulation Results - Zero - $\alpha = 0.1, \tau_T = 3$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	10.7194	0.8884	0.0648	0.0408	0.0061	36.8021	37.1853
Trial 2	10.8873	0.8738	0.0531	0.0538	0.0192	36.1374	36.3658
Trial 3	11.1601	0.8844	0.0530	0.0560	0.0066	38.4471	39.5978
Trial 4	10.9012	0.7959	0.0316	0.1494	0.0230	42.0576	42.5725
Trial 5	10.7703	0.8758	0.0794	0.0393	0.0055	37.1769	37.8734
Trial 6	10.8518	0.8278	0.0270	0.1258	0.0194	42.4530	42.8581
Trial 7	11.3942	0.8206	0.0206	0.1525	0.0063	42.7442	43.3957
Trial 8	10.8412	0.8917	0.0560	0.0311	0.0213	36.4326	36.7550
Trial 9	10.9195	0.9185	0.0437	0.0315	0.0063	35.5460	36.5712
Trial 10	10.9651	0.8130	0.0338	0.1256	0.0276	42.9238	43.3160

Table A.5: Full Simulation Results - Zero - $\alpha = 0.1, \tau_T = 20$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	10.8341	0.9054	0.0838	0.0058	0.0050	35.3456	36.1758
Trial 2	10.8347	0.8917	0.0543	0.0350	0.0190	37.6649	38.4433
Trial 3	10.3914	0.8242	0.1676	0.0016	0.0066	44.7717	45.4048
Trial 4	10.5373	0.8526	0.1107	0.0215	0.0152	45.6632	46.7002
Trial 5	11.3600	0.8436	0.0654	0.0834	0.0077	43.0433	44.0596
Trial 6	10.4612	0.8306	0.1470	0.0102	0.0121	44.9793	45.9544
Trial 7	10.8627	0.9097	0.0642	0.0192	0.0069	38.5395	38.8406
Trial 8	10.5977	0.8310	0.1036	0.0369	0.0285	47.3138	47.8596
Trial 9	10.6208	0.8666	0.1077	0.0172	0.0085	43.5148	44.5180
Trial 10	10.7600	0.8438	0.0765	0.0612	0.0184	42.7328	44.0505

Table A.6: Full Simulation Results - Zero - $\alpha = 0.005, \tau_T = 3$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	11.4075	0.8002	0.0000	0.1979	0.0019	41.0846	41.4172
Trial 2	11.2557	0.8413	0.0132	0.1298	0.0157	35.0781	35.6587
Trial 3	11.1032	0.8345	0.0003	0.1629	0.0023	36.4955	36.9032
Trial 4	11.3082	0.8374	0.0035	0.1442	0.0149	36.9775	37.5921
Trial 5	11.1636	0.8126	0.0036	0.1812	0.0026	39.6542	39.9542
Trial 6	11.0314	0.8110	0.0028	0.1723	0.0139	37.8119	38.3079
Trial 7	11.0176	0.8514	0.0126	0.1332	0.0028	36.4305	36.8051
Trial 8	10.9914	0.8168	0.0118	0.1529	0.0184	38.0112	38.3493
Trial 9	11.0180	0.8341	0.0063	0.1349	0.0246	35.6971	36.3612
Trial 10	10.9692	0.8506	0.0160	0.1303	0.0031	35.7349	36.0481

Table A.7: Full Simulation Results - Zero - $\alpha = 0.005, \tau_T = 20$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	11.0739	0.8511	0.0188	0.1275	0.0026	36.3337	36.9539
Trial 2	11.2492	0.8200	0.0184	0.1474	0.0141	38.2564	38.9401
Trial 3	11.1336	0.8468	0.0047	0.1460	0.0026	38.8191	39.1225
Trial 4	11.0901	0.8936	0.0098	0.0819	0.0147	31.1897	31.7011
Trial 5	11.1958	0.8867	0.0017	0.1087	0.0028	33.4682	33.8183
Trial 6	11.1482	0.8998	0.0194	0.0686	0.0122	32.8995	33.8968
Trial 7	11.5115	0.9029	0.0414	0.0513	0.0044	35.0763	35.6078
Trial 8	11.0683	0.8820	0.0038	0.1002	0.0140	32.0743	32.4885
Trial 9	11.4672	0.8610	0.0157	0.1196	0.0036	34.4249	34.9311
Trial 10	11.7045	0.7856	0.0207	0.1687	0.0250	42.4117	43.2065

Table A.8: Full Simulation Results - Zero$_{\text{min}}$ - $\alpha = 0.1, \tau_T = 3$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	11.5172	0.8816	0.0798	0.0309	0.0077	40.6464	41.8900
Trial 2	10.6984	0.8937	0.0913	0.0040	0.0109	36.4449	36.7442
Trial 3	10.7624	0.9040	0.0757	0.0151	0.0052	35.3957	36.7978
Trial 4	10.7615	0.8672	0.0772	0.0417	0.0139	38.4657	39.9197
Trial 5	10.9192	0.8041	0.1223	0.0597	0.0139	51.6094	52.1750
Trial 6	11.2688	0.8540	0.0488	0.0678	0.0293	42.8794	43.8836
Trial 7	10.9328	0.9056	0.0474	0.0416	0.0055	36.4340	36.8229
Trial 8	10.8717	0.9167	0.0599	0.0108	0.0126	36.9677	39.5807
Trial 9	11.0082	0.8772	0.0911	0.0254	0.0063	40.2323	40.6879
Trial 10	11.2889	0.8372	0.0428	0.0916	0.0284	42.9172	44.5943

Table A.9: Full Simulation Results - Zero$_{\text{min}}$ - $\alpha = 0.1, \tau_T = 20$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	11.5172	0.8816	0.0798	0.0309	0.0077	40.6464	41.8900
Trial 2	10.6984	0.8937	0.0913	0.0040	0.0109	36.4449	36.7442
Trial 3	10.7624	0.9040	0.0757	0.0151	0.0052	35.3957	36.7978
Trial 4	10.7615	0.8672	0.0772	0.0417	0.0139	38.4657	39.9197
Trial 5	10.9022	0.7875	0.1223	0.0597	0.0305	51.6094	52.1750
Trial 6	11.5835	0.8769	0.0488	0.0678	0.0065	42.8794	43.8836
Trial 7	10.9122	0.8966	0.0474	0.0416	0.0145	36.4340	36.8229
Trial 8	10.8717	0.9167	0.0599	0.0108	0.0126	36.9677	39.5807
Trial 9	11.0082	0.8772	0.0911	0.0254	0.0063	40.2323	40.6879
Trial 10	11.8030	0.8574	0.0428	0.0916	0.0082	42.9172	44.5943

Table A.10: Full Simulation Results - Zero$_{\text{min}}$ - $\alpha = 0.005, \tau_T = 3$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	10.9250	0.8284	0.0323	0.1254	0.0140	37.1294	37.8760
Trial 2	11.0898	0.9284	0.0066	0.0628	0.0022	27.0845	27.5767
Trial 3	11.0034	0.8555	0.0151	0.1160	0.0135	35.4752	36.2123
Trial 4	11.1282	0.8896	0.0262	0.0815	0.0027	31.5082	32.1901
Trial 5	10.9062	0.8999	0.0301	0.0608	0.0091	29.9748	30.2187
Trial 6	11.0314	0.8816	0.0429	0.0726	0.0028	33.7401	34.0207
Trial 7	11.1937	0.8590	0.0215	0.1043	0.0152	33.4871	34.3611
Trial 8	11.1496	0.9376	0.0192	0.0397	0.0035	26.3833	26.8060
Trial 9	11.5192	0.8516	0.0148	0.1160	0.0176	36.4004	36.7720
Trial 10	11.0457	0.9193	0.0141	0.0646	0.0020	28.4367	29.1544

Table A.11: Full Simulation Results - Zero$_{\text{min}}$ - $\alpha = 0.005, \tau_T = 20$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	11.2794	0.8241	0.0685	0.1037	0.0038	42.0920	42.7454
Trial 2	11.2309	0.8523	0.0714	0.0581	0.0182	39.0569	39.3263
Trial 3	11.2547	0.9088	0.0222	0.0665	0.0026	29.2292	29.7014
Trial 4	10.9515	0.9045	0.0354	0.0495	0.0106	31.5353	32.2946
Trial 5	11.0251	0.9142	0.0118	0.0713	0.0027	32.5043	32.7694
Trial 6	11.2093	0.9050	0.0375	0.0492	0.0082	30.8149	31.7922
Trial 7	11.2095	0.8729	0.0296	0.0938	0.0038	35.3686	36.1199
Trial 8	11.1493	0.9298	0.0417	0.0214	0.0071	27.3853	28.1566
Trial 9	11.2533	0.9026	0.0245	0.0698	0.0031	31.1935	31.4603
Trial 10	11.3151	0.9200	0.0204	0.0472	0.0124	29.4415	30.1220

Table A.12: Full Simulation Results - Velocity - $\alpha = 0.005, \tau_T = 3$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	8.7964	0.8789	0.0674	0.0288	0.0249	35.7624	39.8710
Trial 2	9.0627	0.9022	0.0398	0.0465	0.0114	31.8726	32.5877
Trial 3	8.9928	0.8676	0.0725	0.0382	0.0217	36.4804	37.2353
Trial 4	9.5302	0.9104	0.0313	0.0490	0.0093	30.8351	31.3276
Trial 5	9.3188	0.8805	0.0600	0.0358	0.0237	35.9392	37.9527
Trial 6	9.0237	0.8484	0.1036	0.0284	0.0196	41.4958	42.3791
Trial 7	8.7584	0.8760	0.0776	0.0229	0.0235	35.2213	35.7987
Trial 8	9.3145	0.8820	0.0753	0.0262	0.0164	37.1895	40.7401
Trial 9	8.9922	0.9031	0.0525	0.0289	0.0155	33.7638	34.6261
Trial 10	8.9690	0.8784	0.0759	0.0239	0.0218	35.3279	35.8971

Table A.13: Full Simulation Results - Velocity - $\alpha = 0.005, \tau_T = 20$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	9.0142	0.8979	0.0960	0.0001	0.0059	34.1064	34.5179
Trial 2	8.7665	0.8764	0.1162	0.0005	0.0069	36.0859	36.3887
Trial 3	8.7632	0.8597	0.1361	0.0000	0.0042	38.1389	38.3296
Trial 4	9.0270	0.8703	0.1204	0.0016	0.0077	36.4818	36.8726
Trial 5	8.2211	0.8750	0.1189	0.0003	0.0058	35.7138	36.0790
Trial 6	8.8402	0.8719	0.1161	0.0044	0.0075	36.8910	37.2414
Trial 7	8.6675	0.8625	0.1297	0.0003	0.0075	37.4888	37.7733
Trial 8	8.4561	0.8664	0.1273	0.0001	0.0062	36.6757	37.0652
Trial 9	8.6743	0.8792	0.1103	0.0016	0.0089	35.2488	35.6300
Trial 10	8.2509	0.8620	0.1324	0.0017	0.0039	37.9146	38.1402

Table A.14: Full Simulation Results - Zero$_{\text{min}_{\text{ghost}}}$ - $\alpha = 0.005, \tau_T = 3$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	11.1100	0.8472	0.0305	0.1056	0.0167	35.8987	37.0320
Trial 2	10.9301	0.8422	0.1108	0.0441	0.0028	39.7450	39.9684
Trial 3	11.2189	0.8647	0.0651	0.0586	0.0116	36.7196	37.4342
Trial 4	11.2272	0.8932	0.0174	0.0865	0.0030	33.5375	34.2893
Trial 5	11.1872	0.9240	0.0301	0.0331	0.0128	26.9779	28.0600
Trial 6	10.9044	0.8141	0.1157	0.0654	0.0048	43.8552	44.2550
Trial 7	10.8594	0.8793	0.0098	0.0936	0.0172	36.6675	37.1632
Trial 8	11.3404	0.8879	0.0553	0.0521	0.0047	37.4255	37.8949
Trial 9	10.9876	0.8364	0.0180	0.1275	0.0180	36.8599	37.8545
Trial 10	11.2320	0.8995	0.0125	0.0838	0.0042	34.7922	35.3431

Table A.15: Full Simulation Results - Zero$_{\text{min}_{\text{ghost}}}$ - $\alpha = 0.005, \tau_T = 20$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	10.8791	0.8600	0.0623	0.0546	0.0231	42.2754	42.6909
Trial 2	10.9804	0.8319	0.0562	0.1069	0.0050	41.0038	41.4043
Trial 3	10.7066	0.7846	0.1180	0.0737	0.0237	45.0680	45.4858
Trial 4	11.8990	0.7759	0.0484	0.1691	0.0066	44.6145	47.7048
Trial 5	11.1095	0.7600	0.0091	0.2077	0.0231	42.9632	43.3607
Trial 6	11.2778	0.9157	0.0359	0.0432	0.0052	31.1265	32.0680
Trial 7	10.8116	0.8089	0.0311	0.1375	0.0226	43.8788	44.9683
Trial 8	11.2057	0.8388	0.0180	0.1390	0.0042	38.1748	39.1816
Trial 9	11.3983	0.8175	0.0077	0.1500	0.0249	39.8669	41.1724
Trial 10	11.0153	0.8078	0.0530	0.1338	0.0054	44.8262	45.3243

Table A.16: Full Simulation Results - $\text{Velocity}_{\text{ghost}}$ - $\alpha = 0.005, \tau_T = 3$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	8.2183	0.8284	0.1344	0.0116	0.0257	43.7518	44.2249
Trial 2	8.0944	0.9194	0.0592	0.0124	0.0090	29.6698	30.0724
Trial 3	8.6719	0.8483	0.0935	0.0312	0.0270	39.7048	40.0592
Trial 4	9.0550	0.9058	0.0483	0.0350	0.0109	32.2706	33.4757
Trial 5	9.0467	0.9009	0.0542	0.0296	0.0153	33.9578	34.6820
Trial 6	9.4188	0.8917	0.0695	0.0214	0.0174	35.7672	37.8909
Trial 7	8.5143	0.8491	0.1051	0.0203	0.0256	41.8884	42.2496
Trial 8	8.6617	0.8898	0.0693	0.0257	0.0152	35.5528	36.2906
Trial 9	8.6590	0.8750	0.0786	0.0233	0.0231	37.2588	37.8664
Trial 10	9.4544	0.8963	0.0576	0.0317	0.0144	36.1780	36.8443

Table A.17: Full Simulation Results - $\text{Velocity}_{\text{ghost}}$ - $\alpha = 0.005, \tau_T = 20$

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	8.5582	0.8828	0.1073	0.0011	0.0087	34.7554	34.9799
Trial 2	8.9095	0.8773	0.1156	0.0019	0.0052	36.7952	37.3108
Trial 3	8.6067	0.8733	0.1189	0.0001	0.0077	36.1418	36.5784
Trial 4	7.8213	0.8051	0.1886	0.0000	0.0063	45.2819	45.6383
Trial 5	8.6588	0.8710	0.1172	0.0020	0.0098	36.7155	37.1597
Trial 6	8.5598	0.8616	0.1312	0.0013	0.0059	37.6023	38.0048
Trial 7	8.5792	0.8706	0.1181	0.0034	0.0079	36.6521	39.8571
Trial 8	8.6190	0.8707	0.1244	0.0000	0.0048	36.4084	36.7789
Trial 9	7.9821	0.7973	0.1936	0.0000	0.0091	45.1434	45.5250
Trial 10	8.5306	0.8656	0.1266	0.0026	0.0052	36.6034	36.8314

APPENDIX B. FULL SIMULATION RESULTS FROM CHAPTER 10

Table B.1: Full Simulation Results - Learning - 9

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	0.4925	0.9736	0.0000	0.0264	0.0000	4.6875	4.7111
Trial 2	0.4960	0.9810	0.0000	0.0183	0.0007	3.8230	3.8567
Trial 3	0.4995	0.9784	0.0000	0.0216	0.0000	4.5513	4.5850
Trial 4	0.4963	0.9794	0.0000	0.0194	0.0011	4.1712	4.1950
Trial 5	0.4959	0.9840	0.0000	0.0160	0.0000	3.5164	3.5452

Table B.2: Full Simulation Results - No Learning - 9

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	0.4460	0.9449	0.0261	0.0209	0.0081	12.9076	13.1079
Trial 2	0.4365	0.9538	0.0147	0.0252	0.0064	11.0228	11.1990
Trial 3	0.4449	0.9406	0.0206	0.0309	0.0080	13.2314	13.4156
Trial 4	0.4520	0.9394	0.0211	0.0290	0.0105	13.5637	13.7473
Trial 5	0.4463	0.9356	0.0211	0.0345	0.0088	13.4601	13.6339

Table B.3: Full Simulation Results - Learning - 13

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	1.1867	0.9857	0.0000	0.0139	0.0005	3.9609	3.9904
Trial 2	1.1875	0.9747	0.0000	0.0238	0.0016	5.3056	5.3351
Trial 3	1.1857	0.9833	0.0000	0.0156	0.0011	3.7952	3.8247
Trial 4	1.1950	0.9819	0.0000	0.0176	0.0005	4.4193	4.4434
Trial 5	1.1867	0.9836	0.0000	0.0164	0.0000	4.3256	4.3256

Table B.4: Full Simulation Results - No Learning - 13

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	1.6484	0.7768	0.1660	0.0061	0.0510	36.4406	37.0450
Trial 2	1.6780	0.7955	0.1520	0.0082	0.0443	36.3604	36.9657
Trial 3	1.7240	0.7784	0.1603	0.0103	0.0509	37.0060	37.6476
Trial 4	1.7008	0.8005	0.1322	0.0111	0.0563	33.8424	34.5008
Trial 5	1.7992	0.7969	0.1397	0.0083	0.0551	35.6311	36.2673

Table B.5: Full Simulation Results - Learning - 17

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	11.1847	0.8784	0.0082	0.0832	0.0302	19.5406	19.7282
Trial 2	11.3019	0.8653	0.0108	0.0903	0.0335	21.2663	21.5220
Trial 3	11.1630	0.8727	0.0081	0.0875	0.0317	19.6854	19.9011
Trial 4	11.1782	0.8733	0.0076	0.0886	0.0305	20.3652	20.6013
Trial 5	11.1768	0.8552	0.0076	0.1059	0.0313	20.9560	21.1778

Table B.6: Full Simulation Results - No Learning - 17

	MOTP	MOTA	MD	FP	MM	OSPA	OSPA-T
Trial 1	15.4279	0.5852	0.2642	0.0250	0.1256	60.3042	61.2909
Trial 2	15.3267	0.5711	0.2772	0.0225	0.1292	60.8655	61.8626
Trial 3	15.1853	0.5788	0.2770	0.0202	0.1240	60.1878	61.1615
Trial 4	15.4464	0.5692	0.2716	0.0282	0.1310	61.0113	61.9687
Trial 5	15.4093	0.5786	0.2689	0.0203	0.1322	60.1818	61.1723