

AUTOMATED TRACKING AND ANALYSIS OF AERIAL SURVEILLANCE DATA

A dissertation presented to
the faculty of the School of Engineering and Applied Science
University of Virginia

In partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering

by

Alla Aksel

December 2014

Automated Tracking and Analysis of Aerial Surveillance Data

A Dissertation

Presented to
the faculty of the School of Engineering and Applied Science
University of Virginia

in partial fulfillment
of the requirements for the degree

Doctor of Philosophy

by

Alla Aksel

December

2014

APPROVAL SHEET

The dissertation
is submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy


AUTHOR

The dissertation has been read and approved by the examining committee:

Scott T. Acton

Advisor

Zongli Lin

Peter Beling

Kevin Skadron

Stephen G. Wilson

Accepted for the School of Engineering and Applied Science:



Dean, School of Engineering and Applied Science

December
2014

"I almost wish I hadn't gone down that rabbit-hole – and yet – and yet – it's rather curious, you know, this sort of life!"

-- Alice



© Arthur Rackham [1]

ABSTRACT

As video sources from unmanned autonomous vehicles, surveillance cameras, and other platforms have become ubiquitous, robust methods for target detection and tracking are in increasing demand. The major challenge of such big data collection is that once the data are captured, a cumbersome, if not impossible, task remains for a human analyst to mine the collected data for valuable information. Consequently, automated tracking methods are required.

Towards this end, we present several tracking algorithms to tackle a variety of video sequences, along with a trackability measure to analyze video sequences for the purpose of tracking. We are specifically focused on persistent surveillance applications used to study target movements. Accordingly, the first major contribution of this work is that it provides two approaches for automated methods to track multiple targets in persistent surveillance video sequences. In the first approach, we develop an automated tracker for registered (stationary camera) video sequences; in the second approach, we present a new tracker for unregistered video sequences based on the morphological filter. The second major contribution is the introduction of a novel trackability measure that allows the user to quantify the difficulty of tracking in a variety of environments via an assortment of imaging sensors.

First, we demonstrate a tracking algorithm for registered video sequences: the *Snake Particle Filter* (SPF) tracker. The SPF tracker is applied to two data sets. The first data set is composed of 28 targets where the SPF tracker has on average an RMSE error of 7 pixels in the horizontal direction and 3.5 in the vertical. The second data set had 7 targets with the mean RMSE in the horizontal direction being 4.5 pixels and 4 pixels in the vertical direction.

Second, we develop a novel algorithm for unregistered video sequences. The algorithm is named the *Morphological Scale-Space Tracker* (MS²T). We compare the MS²T to a SIFT based

tracker, the *Automated SIFT Tracker* (ASIFT²), and also investigate the incorporation of SIFT into the MS²T. For all the methods here we utilized 34 targets and 2 measurements, namely the percentage of tracking and normalized root mean squared error (RMSE). The tracking results show that: a) ASIFT² has an average of 90% frames tracked and 0.45 (half of target width) normalized RMSE; b) MS²T has an average of 96% frames tracked and 0.27 (less than third of target width) normalized RMSE, and; c) incorporating SIFT into MS²T results in an improvement of 98% frames tracked and 0.28 normalized RMSE.

Lastly, we establish a novel quality measure for tracking, which we named the trackability measure. The trackability measure enables, for the first time, quantification of the difficulty of tracking for a given scenario, based on the target appearance, the target motion, and the video quality. Previous measures considered only video quality. Overall, tracker performance parallels the newly introduced trackability measure in terms of the Spearman correlation.

By developing both automated tracking algorithms and a trackability measure we provide a comprehensive approach to tracking. Furthermore, this dissertation introduces a broad set of tracking methodologies by tracking not only registered video sequences, but also unregistered sequences.

ACKNOWLEDGEMENTS

It is a cliché of acknowledgments to say that I could not have completed this dissertation without the support of others, but in this case it is very true. It is only by virtue of the encouragement of my advisor, colleagues, friends, and family that this work has been written

Foremost, I want to thank my advisor and collaborator, Dr. Scott Acton. Over many years, and numerous chapter revisions, Dr. Acton has never tired of me, and has been a continual source of encouragement and inspiration. Dr. Acton has indeed been instrumental in my successful completion of this thesis, in my unique graduate-studies path, and to my finding employment. I am truly and will always be very grateful.

A special thank you goes to my committee members Dr. Zongli Lin, Dr. Peter Beling, Dr. Barry Horowitz, Dr. Kevin Skadron, and Dr. Steve Wilson for their wisdom, consideration, and time. I also would like to thank Dr. Toby Berger for his helpful advice in developing the trackability measure used in this dissertation. Working with professors like these has been of great assistance, and indeed a unique pleasure.

I thank Virginia Image and Video Analysis members past and present for the creation of a productive and engaging work environment. As a long time Wahoo (undergraduate and graduate degrees), I would also like to thank everyone at the University of Virginia who has helped me along the way, which are too many to be named individually. The professors, post-docs, administrators, students, and staff at UVA are what make it such an extraordinary institution of learning.

I am deeply grateful to my parents and sister for their unfailing love and support. A sincere and heartfelt thanks go to Jack, Cayman, and my friends for their consistent loyalty and friendship. The Army Night Vision and Electronic Sensor Directorate and the Army Research

Office supported this research and provided the invaluable data used for parts of this work, for which I am indebted. This work was supported under Contract Number W911NF1010367.

Lastly, a very special thank you goes to my husband, Dr. Stephen Silverstein, who has been supportive and encouraging, while so graciously reviewing all my publications throughout my graduate career.

TABLE OF CONTENTS

ABSTRACT	v
ACKNOWLEDGEMENTS	vii
TABLE OF CONTENTS	ix
LIST OF FIGURES	xii
LIST OF TABLES	xiv
LIST OF SYMBOLS AND ABBREVIATIONS	xv
Chapter 1. Introduction.....	1
Chapter 2. Tracking Background	5
2.1 Measurement Based Tracking.....	7
2.1.1 Centroid Tracker	7
2.1.2 Template Matching	8
2.1.3 Active Contour Tracker	9
2.1.4 Geodesic Tracker	10
2.1.5 Kernel-Based Object Tracking.....	10
2.2 Dynamic Model Tracking	11
2.2.6 Kalman Filter.....	11
2.2.7 Extended Kalman Filter	13
2.2.8 Particle Filter.....	13
Chapter 3. Target tracking via a Snake Particle Filter.....	15
3.1 SPF Initialization	16

3.2 SPF Tracking.....	18
3.2.1 Active Contour.....	18
3.2.2 Vector Field Convolution.....	21
3.2.3 Particle Filter.....	22
3.3 Combined Snake PF (SPF).....	25
3.4 Data Sets.....	26
3.5 SPF Results.....	28
3.6 Conclusion	32
Chapter 4. Target tracking via Morphological Scale-Space Trackers	34
4.1 Video Sequences of Interest.....	36
4.2 Invariant Feature Transform	36
4.2.1 Scale Invariant Feature Transform (SIFT)	37
4.2.2 Automated SIFT Tracker (ASIFT ²).....	42
4.2.3 ASIFT ² Tracking Results.....	44
4.3 Connected Components.....	50
4.3.4 Morphological Scale-Space Tracker and the Connected Filter.....	50
4.3.5 Morphological Scale-Space Connected Components Features.....	56
4.3.6 Target Detection Using the Morphological Scale-Space	59
4.3.7 Target Tracking via Feature Matching in MS ² T.....	65
4.3.8 MS ² T Tracking Results	67
4.3.9 Investigating Descriptors in MS ² T.....	70

4.3.10 Statistical Significance Analysis	72
4.4 Incorporating SIFT into MS ² T	74
4.5 Conclusion	75
Chapter 5. Tracking Characterization	78
5.1 Background on Image Quality and Tracking Characterization	79
5.2 Trackability Measure	84
5.2.1 Trackability Theory	85
5.2.2 Quality of Signal-to-Template Match	86
5.3 Quality of Video	90
5.4 Results	92
5.4.3 Synthetic Tests to Demonstrate Efficacy of Trackability Measure	93
5.4.4 Trackability Results for Yuma, AZ Data	97
5.5 Conclusions	99
Chapter 6. Conclusion	103
Reference	107

LIST OF FIGURES

Figure 1 – Tracker flow/execution steps block diagram	5
Figure 2 – Example of surveillance images with targets marked with a yellow square.....	6
Figure 3 – Example of the Kalman filter failing to track a target.....	12
Figure 4 – Track initialization.....	17
Figure 5 – Snake example.....	21
Figure 6 – Vector field convolution (VFC).....	22
Figure 7 – SPF tracking results.....	27
Figure 8 – Example of tracked targets.....	28
Figure 9 – Tracking RMSE results, where $N_s = 100$	30
Figure 10 – Yuma, AZ SPF tracking RMSE	31
Figure 11 – Two octaves of scale-space and difference of Gaussians (DoG)	39
Figure 12 – Visualization of 3-D neighborhood [35].....	40
Figure 13 – Example of matched key points using SIFT with 10° rotation	41
Figure 14 – Example of SIFT key-points in target initialization	42
Figure 15 – Example of SIFT target matching	45
Figure 16 – SIFT scale-space example	48
Figure 17 –Example of a morphological scale-space.....	51
Figure 18 - One level of the MS^2T	52
Figure 19 – Example of a target extracted from nine consecutive frames.	53
Figure 20 – Example of a three gray-level image with associated Max-Tree.....	55
Figure 21 –Illustration to guide feature description	57
Figure 22 - Neighborhood histogram illustration.....	58
Figure 23 – Volume ROC Analysis.....	63

Figure 24 – Eccentricity ROC Analysis.....	64
Figure 25 – Example of shape comparison for original target with a tracked target.....	66
Figure 26 – Tracking results comparison between MS ² T and ASIFT ²	69
Figure 27 – Structural similarity (SSIM) index	81
Figure 28 – Signal to clutter example	84
Figure 29 – Illustration of mutual information by way of intersecting circles.	87
Figure 30 – Illustration of mutual information of signal and template given clutter	89
Figure 31 – An illustration of trivariate conditional mutual information.....	91
Figure 32 –Trackability Measure versus SSIM and SCR in varying clutter occlusion.....	93
Figure 33 –Trackability Measure versus SSIM and SCR in varying target occlusion	94
Figure 34 – Comparison of the Trackability Measure to SSIM and SCR in varying noise levels....	95
Figure 35 –Trackability versus SSIM and SCR in varying motion model error.....	96
Figure 36 – Regression Analysis for RMSE of MS ² T and ASIFT ² vs. Trackability.	99

LIST OF TABLES

Table 1 - ASIFT ² Tracking Results	47
Table 2 – MS ² T Detection Results.....	61
Table 3 - MS ² T Tracking Results.....	67
Table 4 -Spearman's correlation results for parameter analysis,	71
Table 5 - ANOVA Results to Compare ASIFT ² and MS ² T	72
Table 6 - Tracking results for MST ² with SIFT	74
Table 7 - Mutual Information calculation for five target examples	90
Table 8 - The overall tracker performance versus signal-template match metrics.	97
Table 9 - The overall tracker performance vs. video quality metrics Here, <i>TM</i> is calculated with w =600 to maximize RMSE to <i>TM</i> correlation.....	98

LIST OF SYMBOLS AND ABBREVIATIONS

α, β	Weighting Parameters
γ	Positive Parameter
$\delta(\cdot)$	Kronecker Delta Function
ε	Small Positive Constant
ζ	Parameterization Index
η	Target's Contrast Comparison
$\theta_g(\cdot)$	Gradient Orientation
θ_l	Object Orientation
κ	Structure Index
Λ	Number of Scale-Space Levels in MS ² T
λ	Scaling Factor
μ	Mean
v	Measurement Noise
o	Constant
Ξ	Gray Level Histogram
σ	Standard Deviation
ρ	Spearman's Rank Correlation Coefficient
τ	Image Template
v	Target's Structure Comparison
ψ	Target's Luminance Comparison
Ω	Neighborhood Around a Target
ω_i	Sample Weight
∇	Gradient Operator

\aleph, \beth	Ellipse Major and Minor Axis, Respectively
∂	Partial Derivative Operator
$a_{i,j}$	Weighting Parameter
ASIFT ²	Automated SIFT Tracker
B	Scale-Space Shape (3D Structure)
\mathbf{b}	Discrete Pixel Index from the Altered Target
C_1, C_2	SSIM Constants
c_t	Normalizing Factor
CDF	Cumulative Density Function
CM	Shape Comparison Measure
$D(\cdot)$, DoG	Difference of Gaussians
$d(\cdot)$	Dissimilarity Measure
DF	Difference in Scales
$E(\cdot)$	Energy Functional
E_{ext}	External Energy
E_{int}	Internal Energy
$Edge$	Edge Strength
$f_g(\cdot)$	Gradient Magnitude
F_n	Linear State Transition Model
f_n	Non-linear State Transition Model
FAR	False Alarm Rate
FOV	Field of View
FN	False Negative
FP	False Positive

$G(\cdot)$	2-D Gaussian kernel
GGVF	Generalized Gradient Vector Flow
g_x, g_y	Gradient Components
H_0	Null Hypothesis
H_A	Alternative Hypothesis
H_n	Linear Observation Model
h_n	Non-linear Observation Model
HM	Histogram Matching Measure
I or $I(x, y)$	Image
(i, j)	Spatial Indices
$\mathbf{k}(\cdot)$	Vector Field Convolution Kernel
k, l	Indices
KF	Kalman Filter
$L(\cdot)$	Image blurred by a Gaussian kernel
$m(\cdot)$	Vector Magnitude
MI	Mutual Information
MS^2T	Morphological Scale-Space Tracker
n	Time Index
$\mathbf{n}(\cdot)$	Unit Vector
$NCC(\cdot)$	Normalized Cross Correlation Coefficient
NCC	Normalized Cross Correlation
N_s	Number of Samples/Particles
$nI(\cdot)$	Connected Component Scale-Space
O	Object to Indicate Structure Parameters

O_v	Object Volume
O_{or}	Object Orientation
O_{ec}	Object Eccentricity
O_{nh}	Object Neighborhood Histogram
P	Total Number of Pixels in an Image
PD	Probability of Detection
PF	Particle Filter
q	Target State
R	Discrete Radial Distance
r	Radial Distance
ROI	Region of Interest
RMSE	Root Mean Squared Error
SCR	Signal to Clutter Ratio
s_i	Samples used in particle filter
SPF	Snake Particle Filter
SSIM	Structural Similarity Index
TN	True Negative
TP	True Positive
\mathbf{t}	Discrete Pixel Index from the Altered Target
(u, v)	Ranks of Datasets for Spearman Correlation
VFC	Vector Field Convolution
w	Process Noise
(x, y)	Spatial Indices
$\bar{\cdot}$	Mean Operator

$(X(\cdot), Y(\cdot))$	Contour Position Indices
z	Target measurement

CHAPTER 1.

INTRODUCTION

Visual surveillance of objects from various dynamic imagery data is integral for many applications and is commonly utilized in both the private and public sectors. In visual surveillance, the objects of interest range from people to vehicles, and uses include everything from personal security to police and military operations. The general framework of visual surveillance is comprised, in the first instance, of numerous image analysis tasks whose aim is to detect, track, and classify objects of interest from image sequences or video. In the second instance, visual surveillance seeks to understand and describe these objects' behavior [2, 3].

Within the framework of visual surveillance, military applications find target tracking to play a vital role, specifically in persistent surveillance. The successful tracking of targets in image sequences delivers a significant tool to characterize the actions of objects of interest in real-time. In persistent surveillance, tens of square miles may be monitored and each vehicle tracked. Manually tracking such targets becomes a cumbersome task for the human operator who is prone to inter- and intra-observer error. Automated tracking provides the benefit of studying the activity of targets over a significant period of time. Moreover, automated tracking provides reproducible results which eliminate human observer shortcomings.

This project seeks to develop an automated method to track multiple targets in persistent surveillance video sequences. Particular difficulties that limit target tracking success are presented by varying video qualities and complicated scenes. Such obstacles must be studied, in terms of frame rate and resolution for instance, in order to determine which imaging methods are a best fit for target tracking. Towards this end we identified the following specific objectives: to tackle target tracking in registered and unregistered video sequences, and to study video quality and tracking scenes for trackability. In more detail, these aims are as follows:

Specific Aim 1: To develop an automated tracking algorithm for a single moving target in surveillance video sequences by fusing a particle filter with the active contour

Target motion in surveillance sequences is multifaceted and often cannot be addressed with classical tracking methods such as the Kalman filter. For example, assuming constant velocity (constant speed and direction) is not practical given low temporal sampling and the erratic movement of targets. The particle filter (PF) has shown promising results in tracking targets with complex, dynamic motion models [4]. Essentially, the multitude of particles computed allows multiple hypotheses regarding target state.

We propose to incorporate the active contour or snake [5] into the PF as the weight measure. The active contour will be used to establish a likelihood model for each particle. Establishing the weights of the particles in the PF plays a vital role in the success of tracking. Since the targets in surveillance sequences often have strong edges, incorporating active contours into the weights of the PF utilizes the edge information to improve tracking results.

Specific Aim 2: To accomplish tracking with a morphological scale-space that allows detection and tracking of multiple targets in unregistered video sequences

Often, surveillance sequences acquired from aerial vehicles are not registered. Registration, moreover, adds not only further time and complexity to the tracking algorithm, but it can often be inaccurate. Registration inaccuracies cause the motion model to incorrectly predict where the target is moving. Consequently, a method which combines target information with the relative frame location of the target must be incorporated into the tracker. We propose to utilize area morphology to generate a scale-space for each frame, resulting in a set of connected

components. Then, we will employ region specific image features to describe targets and achieve tracking in unregistered video sequences. A variety of regional features for each connected component are combined to develop a unique identifier for each target which then in turn is used for matching the target in the consecutive frames. As registration information is absent, we propose to utilize our morphological scale-space to detect targets in each frame without the knowledge of the image background.

In addition, we propose to investigate an existing scale-space algorithm, Scale Invariant Feature Transform (SIFT), through the comparing and contrasting with our morphological scale-space method.

Specific Aim 3: To develop a measure of tracking difficulty (trackability) for a given target and sequence

The success of tracking results can be limited due to the quality of the video sequence. For instance, sequences with very low spatial resolution may not provide enough information for reliable results. We propose to develop a “trackability” measure that can be used to predict and evaluate the success of the tracker in complex imaging scenarios.

The trackability measure uses a theoretical information approach to evaluating the difficulty of matching a template to a target in the presence of clutter. The measure also takes into account resolution, noise, frame rate, registration error and motion model prediction error. This trackability measure is the first such attempt to quantify the difficulty of a given tracking experiment.

The remainder of this dissertation presents the theory and methods used to accomplish the specific aims stated above. Chapter 2 describes the pertinent background material, including the commonly used tracking methodologies, dividing them into two main tracking classes: measurement based and dynamic model based trackers. Chapter 3 discusses the approach taken to achieve aim 1. This chapter explains the use of the active contours and the particle filter as it is used towards our tracking solution. Chapter 4 presents our solution to aim 2 by employing area morphology to generate scale-space models for the targets and the video background of interest. The area morphology is then utilized to track targets in non-registered data sequences. Next, Chapter 5 addresses our solution to aim 3 that incorporates information about the video and the tracking circumstances to understand the success of a tracker. Finally, Chapter 6 offers concluding reflections and considers possible future research directions.

CHAPTER 2.

TRACKING BACKGROUND

Visual surveillance provides a vital tool for characterizing the actions of targets in real-time from video or other imagery. Increasingly, many organizations, predominantly in the public sector, utilize video cameras of various wavelengths and frequencies to collect data from around the world. Furthermore, with an increase in unmanned aerial vehicle integration into surveillance and mission execution, the ability to capture live video feed is unprecedented. For the purpose of this research, our application is specifically concerned with visual surveillance of motorized vehicles, and their locations and movements in terms of target detection and tracking.

The ubiquitous and inexpensive nature of video cameras allows for the collection of a vast amount of information. In video surveillance, the quantity of collected dynamic imagery grows continuously, making it difficult, if not impossible, for manual analysis by human operators. Consequently, an automated method is needed to detect and track targets [6]. Furthermore, an investigation of suitable video for tracking will guide the selection of hardware and the evaluation of particular data collectors for best tracking results.

Automated tracking provides the benefit of studying the activities of targets over a

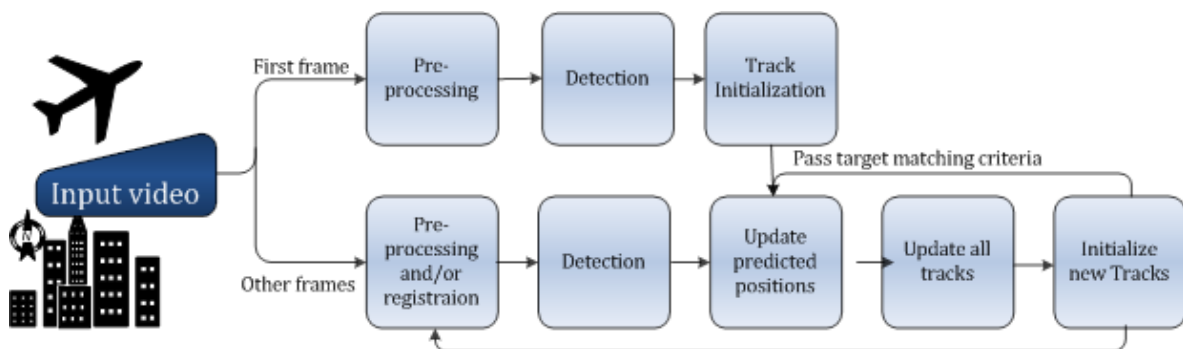


Figure 1 – Tracker flow/execution steps block diagram

This figure outlines the steps that a typical automated tracker might take towards accomplishing tracking. These steps are not comprehensive and some steps might be skipped depending on the algorithm of a given tracker.

significant period of time and alerting the operator when needed. There are several general steps performed by an automated tracker as illustrated in Figure 1. The steps outlined in Figure 1 are not comprehensive, but rather provide an overview of steps a given tracker might take towards accomplishing tracking. Vehicle tracking faces several challenges. In general, the imaging environment is often cluttered and the video acquisition process may suffer from insufficient temporal resolution. The clutter might include trees, bushes, buildings, other man-made stationary objects, etc. Furthermore, video sequences acquired from moving vehicles such as airplanes and helicopters with or without stabilizers require registration. Video registration is a time consuming method that can result in an inaccurate frame alignment. Consequently, methods which rely on video registration will fail to track the targets of interest in poorly registered video sequences.

Surveillance image sequences can vary significantly as shown in Figure 2. Figure 2 presents two examples of a single frame from (a) a higher resolution stationary camera and (b) a

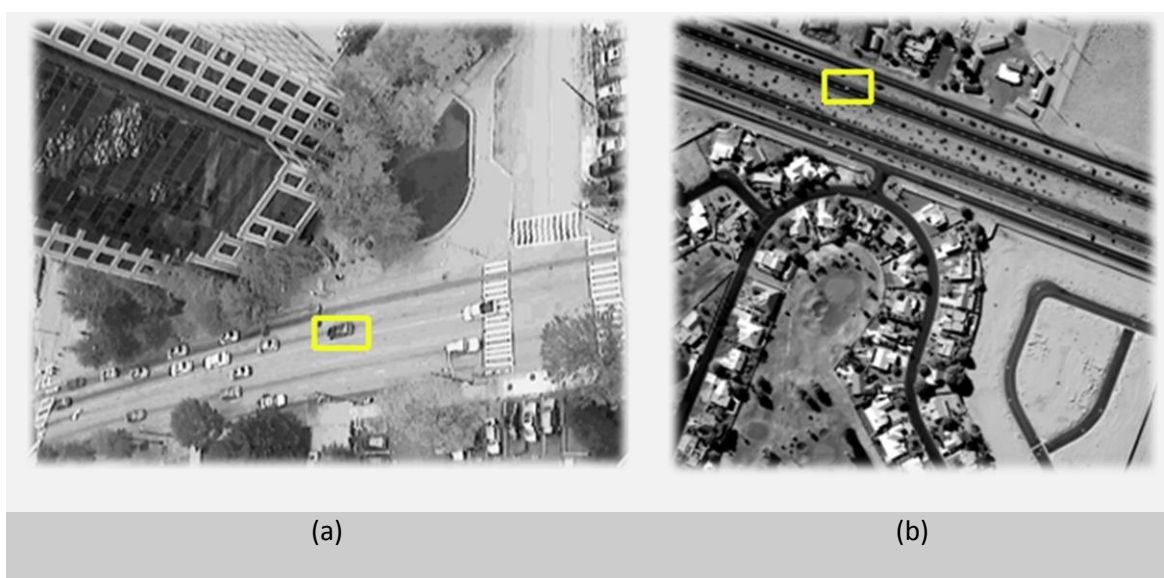


Figure 2 – Example of surveillance images with targets marked with a yellow square.

Here image (a) illustrates a stationary camera capturing traffic at an intersection on Peachtree Street, Atlanta, GA, and (b) shows an image from a surveillance video acquired from a helicopter over Yuma, AZ, where the camera is not mounted on a stabilizer.

lower resolution camera mounted on a helicopter. This figure also illustrates the targets of interest in the example videos. In order to extract all the valuable information from these sequences a tracking algorithm must automatically monitor the activities of each target over time.

This chapter presents background material on existing tracking methods and related theory. Two categories of trackers are discussed: a) measurement based trackers which rely on image information such as intensity, target boundary, etc. for tracking, and; b) dynamic model based tracking which incorporates prior target state information such as location and velocity into the tracking model.

2.1 Measurement Based Tracking

Measurement based trackers use image information for detection and tracking. Here we discuss the literature on several widely used trackers. These trackers rely on the appearance, pixel intensities, and edges of the image to accomplish tracking.

2.1.1 Centroid Tracker

The centroid tracker computes the center of mass in terms of the pixel intensity profile of the target in a region of interest (ROI) in each frame. The ROI is initialized in the first frame and target location is estimated using the center of mass. The estimated target location from the current frame becomes the center of the ROI used to track the target in the following frame and target location estimation is repeated. Clusters of pixels can also be used to indicate separate targets and thus the centroid tracker can track several targets in an image sequence.

The centroid tracker has a low computational expense since only first and second moment pixel intensities are used to calculate the center of mass. However, the centroid tracker requires a large contrast between the target or targets and the background. Further, this tracker

cannot distinguish between targets that are close together, nor can it track complicated targets [7] [8].

2.1.2 Template Matching

The template matching or correlation tracker [9] is a tracking technique which utilizes previous knowledge of the target. Such a tracker can also detect the similarity or dissimilarity of one frame and another [8]. If the target's location in each frame is desired, the target becomes a template which is then used to search the image for a match. The best match of the template to the image is then offered as the estimated location of the target. If a target is moving at an angle or the viewpoint is changing, the template must be updated for successful matching.

Several methods can be used in the matching technique. Commonly, the matching is achieved via normalized cross correlation (NCC), but the sum of squared or absolute differences and the Hough transform have also been applied. When employing NCC in tracking, an ROI window is employed within the image to reduce the search area and speed up the execution of the tracker. The NCC coefficient is defined as in [10]:

$$NCC(i, j) = \frac{\sum_{x,y} [I(x, y) - \bar{I}_{i,j}] \cdot [\tau(x - i, y - j) - \bar{\tau}]}{\sqrt{\left\{ \sum_{x,y} [I(x, y) - \bar{I}_{i,j}]^2 \sum_{x,y} [\tau(x - i, y - j) - \bar{\tau}]^2 \right\}}} \quad (1)$$

At each point (i, j) under a window in the image, $I(x, y)$, the image is compared to the target template, τ . $\bar{\tau}$ is the mean of the template, and $\bar{I}_{i,j}$ is the mean of the image in the region under the template.

Template matching can also be incorporated into a dynamic model such as the Kalman or particle filters [4]. Since the template matching method is sensitive to the viewpoint and pose, it often fails when the camera is rotated while acquiring the video sequence. Furthermore, a

database of templates must be generated to account for the necessary viewpoints and/or poses and the algorithm must update the template accordingly.

2.1.3 Active Contour Tracker

Active contour trackers utilize the active contour or snake [5] to propagate the segmentation of the target from frame to frame. A snake is a parametric curve where each point on the curve is mapped to a location in the image (discussed further in Chapter 3). Tracking via a snake is achieved by determining the minimum cost of an energy functional. This is done, generally, by balancing the effects of image features or the external force (e.g. image edges) and the smoothness and rigidity of the curve or the internal force in each frame. In addition to locating the target in the image, this tracker allows establishing the shape of the target. Leymarie and Levine [11] have applied the snake tracker to a deformable cell to capture both the cell location and shape. These authors note that the external force in the snake plays a vital role in the ability of the snake to capture and track the targets. The active contour tracker is sensitive to initialization and noise in the image since noise can disturb the edges of the target and distract the snake from capturing the target of interest.

In order to improve the capture range of the active contours, the generalized gradient vector flow (GGVF) [12] has been incorporated into the external force. The GGVF enables tracking by improving the identification of the weak target boundaries in images with relative background homogeneity. Further, the use of the motion gradient vector flow (MGVF) [13] introduced the motion of the target into the external force which guided to the correct target boundaries of the moving target. Another active contour based method was employed in [14], which accounted for the size, shape, position, and sampling of the contour of the target. Lastly, the vector field

convolution (VFC) [15] has more recently been used as a more general form of GGVF and will be discussed in the following chapter as it is employed by our proposed method.

2.1.4 Geodesic Tracker

Geodesic active contours are a level set approach to segmentation. This segmentation method deforms contours based on the inherent geometric measures of the image. It unites the Kass snake, which is based on energy minimization, with the geometric active contour, which is based on the theory of curve evolution [16].

Later, the geodesic active contours were incorporated in [17] to be used as a tracker. The authors describe their tracker as a “model-free approach” for tracking that can deal with topological changes. Here the method utilizes the geodesic active region model to separate the target from the background and track it in the following frames. The authors note that the geodesic tracker applies a linear motion model, and errors caused by non-linear motion can propagate and cause discrepancies in the tracking results.

2.1.5 Kernel-Based Object Tracking

Kernel-based object tracking, proposed by Comaniciu et al. [18], is designed to address representation and localization of non-rigid objects or targets. The method selects an ellipsoidal area from the image to represent the object. The ellipsoidal region is then normalized to a unit circle which is smoothed by an isotropic and monotonically decreasing kernel. Next, the authors define a spatially-smoothing similarity function, thus reducing target search to a basin of attraction of the similarity function.

The kernel-based method can also be incorporated into a dynamic framework, such as the Kalman filter. The authors note that this method suffers because the basin of attraction may not

lead to the object of interest. Additionally, the kernel-based method does not handle occlusion. Consequently, other methods must be incorporated to address the occlusion problem.

2.2 Dynamic Model Tracking

Dynamic models play a vital role in target tracking. As mentioned in the previous section, the measurement trackers are often incorporated into a dynamic model for more successful tracking. Dynamic models allow incorporating the motion model into the tracker, thus improving the tracking accuracy. One of the more common dynamic models is the Bayesian based tracker which includes the Kalman filter, the extended Kalman filter, and the particle filter.

2.2.6 Kalman Filter

The Kalman filter (KF) [4] is an optimal filter for a linear system with a Gaussian model for the process noise, $w[n]$, and measurements noise, $v[n]$. This filter recursively estimates the state from a sequence of noisy measurements. Here the state transition model is:

$$q[n + 1] = F_n q[n] + w[n] \quad (2)$$

and the observation model is:

$$z[n] = H_n q[n] + v[n], \quad (3)$$

$q[n]$ denotes the state of the target and the measurement is represented by $z[n]$. F_n and H_n are matrices which define the linear mapping of the state and observation models, respectively. The goal of the tracker is to utilize the information we have, given all the measurements, $z[1:n]$ (up to time n), in order to estimate the state $q[n]$. In the KF this is achieved by building the model of the posterior density, $p(q[n]|z[1:n])$.

The KF has been utilized with active contours in the Kalman snake model [19] [20]. In Kalman snakes, the dynamic motion of the KF is incorporated into the energy of the snake, thus introducing a time varying term in the snake implementation. Other methods incorporated the

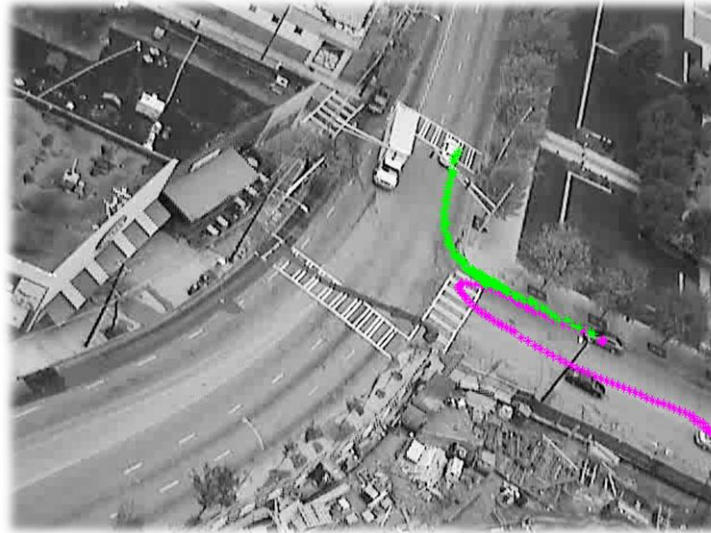


Figure 3 – Example of the Kalman filter failing to track a target.

The track colored in green represents the ground truth, while the track colored in magenta represents the Kalman filter (KF) results. Here the KF gets distracted by the second moving target (the truck which turns left and occludes the target) and switches tracks.

motion model into the external force, instead of dynamically changing the energy [21]. This allows the external force to guide the contour to the new boundary more accurately. Although snakes can provide a powerful tool for tracking, they tend to fail in a low temporal resolution. Typically, the initialization of the snake is based on the target's previous location. Therefore, if the target moves outside of the boundary of the initialization, it will not be captured. Additionally, if external force does not guide the curve to the target correctly, the tracker will typically fail.

With the KF, a single motion model (typically a linear model in the sense that the next state can be expressed as a matrix product with the previous state) is applied. Although the KF is an optimal filter for application with such a model, the surveillance tracking application is more complex [22]. In surveillance tracking, targets undergo discontinuities in acceleration and velocity, and maneuver. If the temporal resolution is low, this poses an added challenge. Figure 3 illustrates an example where KF fails due to the distraction presented by other targets around it.

When the target makes a turn, another target, the truck, crosses in front of it (to the camera), causing the tracker to switch and track the truck instead.

2.2.7 Extended Kalman Filter

The extended Kalman filter (EKF) was developed to address the linearity assumption of the KF, so the state and the observation model become:

$$q[n + 1] = f_n(q[n], w[n]), \text{ and} \quad (4)$$

$$z[n] = h_n(q[n], v[n]). \quad (5)$$

In the case of EKF, both the state transition and observation models do not need to be linear, but rather are differentiable functions. The EKF approximates or linearizes the state and observation models by using the Taylor series expansion. As in the KF, the EKF assumes that both the process and observation noises are white [4]; additionally for maximum a posteriori estimate, the noises are Gaussian. Although the EKF addresses the non-linearity of the KF, it still assumes that the noise is white. The EKF also fails if the state transition and observation models are not differentiable.

2.2.8 Particle Filter

Gordon et al. developed the idea of the particle filter (PF) or Bayesian bootstrap filter where the state density (posterior density) is approximated using measurements from the system [23]. The PF provides a method that removes the constraints inherent in conventional methods, such as that presented by the KF [8] on the system transition function and the measurement function (see more in-depth discussion in Chapter 3). In related applications, other researchers have incorporated the PF in video based target tracking. The CONDENSATION algorithm predicts the parameterized curve evolution using the PF [24] while [25] describes a method that combines the PF into geometric active contours. This method uses geometric active contours and affine

parameters of the contour as the particles in the PF and then computes the weights for each particle based on minimum energy configuration.

CHAPTER 3.

TARGET TRACKING VIA A SNAKE PARTICLE FILTER

Automated tracking is essential in understanding the behavior of targets. In persistent surveillance, target tracking provides an insight into the location of targets at each available point in time (frames) and allows the user to identify target origin and routes, which took place prior to a particular event. It also allows the user to study general patterns and detect pattern deviations. We commence by addressing the tracking problem in registered video sequences. It is beyond the purview of this thesis to address the registration problem or algorithms.

Two important steps must be taken in target tracking. First, the target must be detected in the first frame. This initializes the tracking process. Then the target is tracked from frame to frame. In this chapter we address aim 1. Here, we describe the theory and methods used for initialization techniques and our tracking approach. Initialization is developed using the background subtraction method, while tracking is achieved by utilizing the snake particle filter (SPF). The SPF builds upon the active contour tracking methods and employs active contour to obtain the weights for the particles as the particle filter (PF) is applied to track the targets. This chapter also discusses our SPF tracking results and provides a concluding discussion.

Towards successful tracking with the SPF method, we assume that we have registered video and that registration does not introduce new structures to the image (e.g. due to image warping). The temporal resolution must be high enough such that there is overlap in target location from one frame to the next, while the spatial resolution must be known to initialize and evolve the active contour from frame to frame. The target size must be large enough for an active contour to capture it.

3.1 SPF Initialization

Track initialization is an essential step in tracking applications. Without prior knowledge, the tracker cannot know where the targets are located, and no motion model can be applied in the initial frame. Consequently, for registered sequences, the initialization is performed by generating a background image via a pointwise median computation involving several frames. Here, registration plays a vital role because if registration is not performed correctly the median image would not be the correct background image and initialization will fail. Although a portion of the data used for this work was registered using manual registration, we will not address registration in this work. In this application, we assume that registered video data is available for tracking. Additionally, we assume that a sufficient number of frames is available to establish the median frame such that only stationary objects are present in the median frame.

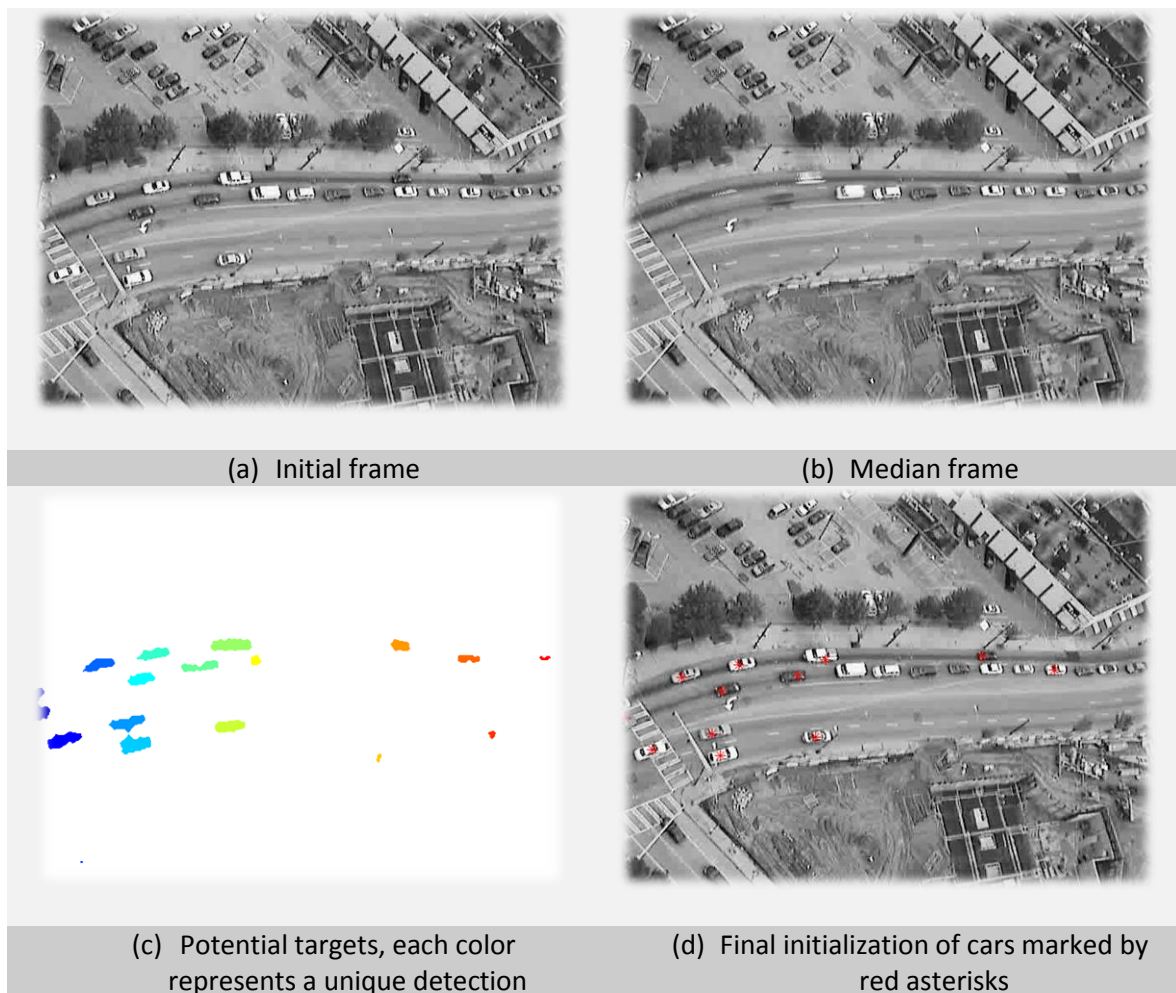


Figure 4 - Track initialization

Here the process of track initialization is illustrated by first (a) examining the frame, (b) taking the median of a set of frames to identify non-moving sections of the image, (c) subtracting the initial frame from the median image results in areas of possible targets noted in various colors where each color indicates a unique detection and lastly (d) the initialization of targets which have moved over the course of the image set which was included in the median calculation.

Once the background image is computed, shown in Figure 4 (b), it is subtracted from the initial frame to locate the moving targets. Now that the background is eliminated and only the moving targets are left, we can generate a logical image where the targets are marked as true and background is marked as false. Basic morphological features are used to remove the falsely detected pixels. The logical image now contains clusters, where each cluster represents a potential moving target, shown in Figure 4 (c). In Figure 4 (c), colors are used to distinguish between each target. Once the target areas are established, we employ the connected

component analysis method which determines the locations and the number of targets [26]. Additionally, small clusters (few pixels in size) are removed because they do not represent targets of interest. Finally, the (x, y) coordinates of each target are recorded. Figure 4 (d) illustrates the location of each target of interest, marked by a red asterisk.

3.2 **SPF Tracking**

We employ the results from the initialization step to set the first state of each target in our tracker. The snake is then applied to determine the shape of the target and to establish the motion model from the first three frames. Following the snake we utilize the particle filter to track the target using the shape result to determine the particle weights. This section describes the active contour and the particle filter – the foundation for our method – as well as the process of fusing these two methods to develop our tracker. Moreover, we discuss data sets used and tracking results.

3.2.1 **Active Contour**

The following discussion will focus on grayscale image segmentation, as this is the analysis relevant to the present study (it can, however, be extended to color images). In grayscale imaging, segmentation is usually based on either similarities or discontinuities in the image. Discontinuities are sharp changes in the image intensities or object edges in the image.

Edges in the image are commonly found using the gradient magnitude or the Laplacian of the image. The gradient of the image provides both magnitude and direction. ∂ is the partial derivative operator, ∇ is the gradient operator, and for an image I , it is defined as:

$$\nabla I(x, y) = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right], \quad (6)$$

while gradient magnitude can be found as follows:

$$|\nabla I(x, y)| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}. \quad (7)$$

The $|\nabla I|$ at a given pixel (x, y) provides the rate of change in the intensity. The Laplacian of the image I is defined as

$$\nabla^2 I(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}. \quad (8)$$

The Laplacian is the sum of second derivatives (in the x and y directions) of the two-dimensional image. From differential equations it is given that the sharpest rate of change of the gradient (first derivative) is the zero crossing of the Laplacian indicating an edge [27].

Snakes are deformable parametric curves that move through the spatial domain and are controlled by a force. This force is an iteratively updated energy functional, which directs the evolution of the snake towards a desired boundary. When the snake deforms to the desired feature it converges and ceases to evolve. The force controlling the movement of the snake is based on internal and external energies. Internal energy (E_{int}) is the elastic force of the contour (tension and rigidity), while the external energy (E_{ext}) is usually based on some image content. E_{ext} is typically user defined to be the edge map of the image. More sophisticated methods, such as generalized gradient vector flow (GGVF) [27], have been used to guide the snake to the appropriate edges. When $E_{ext} + E_{int}$ is minimized, the snake contour reaches a solution. As described in the previous section, active contours or snakes have been extensively used for tracking in various applications. Although snakes have shown promise in segmentation, they are not optimal to be the sole element used in tracking. If the target moves from one frame to another in an amount greater than half the target length, the snake will most likely lose the target because the edges will not be able to guide the snake to the boundaries of interest. It follows that we incorporate the snake into a dynamic model to improve tracking results.

Snakes are deformable parametric curves that move through the spatial domain and are controlled by a force. In order to describe the snake mathematically, we parameterize the contour using $(X(\zeta), Y(\zeta))$, where $\zeta \in [0,1]$, and describes pixel position in the image. The force controlling the snake is derived from an energy functional:

$$E(X, Y) = \frac{1}{2} \int_0^1 \alpha \left(\left| \frac{dX}{d\zeta} \right|^2 + \left| \frac{dY}{d\zeta} \right|^2 \right) + \beta \left(\left| \frac{d^2X}{d\zeta^2} \right|^2 + \left| \frac{d^2Y}{d\zeta^2} \right|^2 \right) d\zeta - \int_0^1 f[X(\zeta), Y(\zeta)] d\zeta. \quad (9)$$

The energy functional in (9) contains terms that combine both the internal and external energies. The internal energy (E_{int}) corresponds to the elastic force of the contour (tension and rigidity), described by the first integral in (9) and α and β control the stretching and bending energy of the curve. The external energy (E_{ext}) is usually based on some image content, represented by the second (bottom) integral in (9). E_{ext} is typically defined to be the additive inverse of edge strength. For an image $I(x, y)$, we can represent the edge strength as the squared image gradient magnitude,

$$f(x, y) = |\nabla I(x, y)| \text{ and} \quad (10)$$

$$Edge = f(x, y)^2. \quad (11)$$

When $E_{ext} + E_{int}$ is minimized locally, the snake reaches convergence. Figure 5 (a) illustrates an example of the snake initialization with a closed parametric curve. Figure 5 (b) presents the final segmentation result.

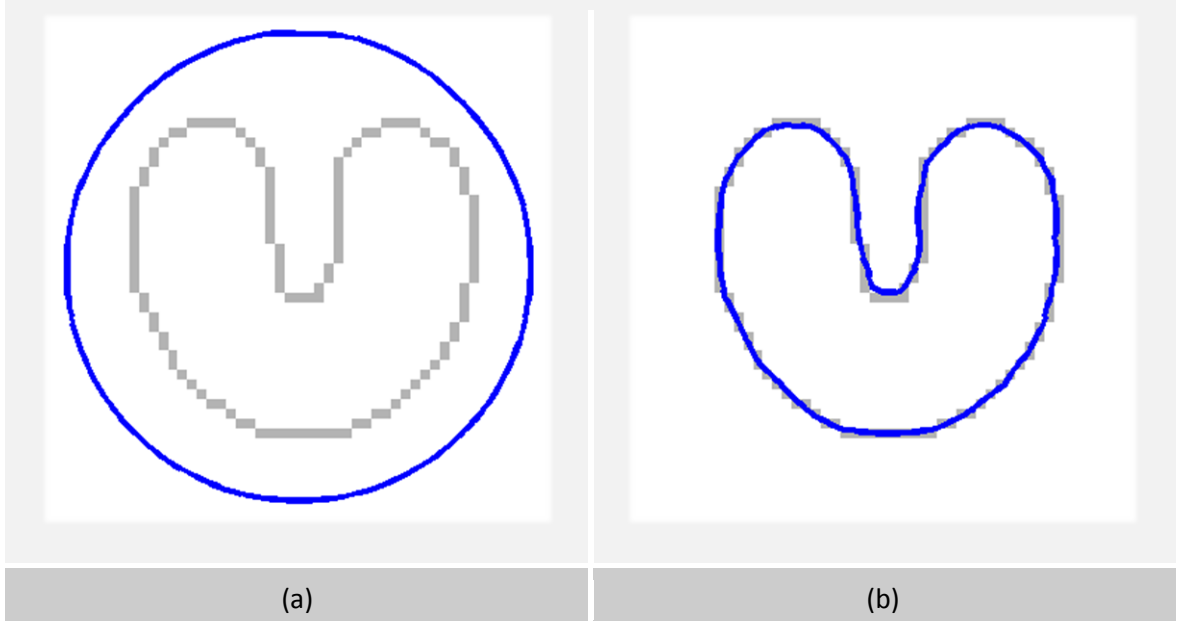


Figure 5 – Snake example

Here, (a) presents the snake initialization marked by the blue circle while the gray object is the object of interest and (b) shows the final segmentation of object of interest where blue overlaps the gray object of interest.

3.2.2 Vector Field Convolution

In order to improve the capture range of the snake we utilize the vector field convolution (VFC) [15] to guide the contour to the appropriate edges. The idea behind VFC is to extend the capture range of the snake by convolving a fixed vector field, $\mathbf{k}(x, y)$, with the edge map, $f(x, y)$. This convolution results in a field of vectors that locally point to the dominant edges. The VFC kernel \mathbf{k} is a tensor with vectors pointing to the center and is defined by

$$\mathbf{k}(x, y) = m(x, y)\mathbf{n}(x, y) \quad (12)$$

where $m(x, y)$ is the magnitude of the vector at (x, y) and $\mathbf{n}(x, y)$ is a unit vector pointing to the center of the kernel. The kernel $\mathbf{k}(x, y)$ is isotropic and has a magnitude, $m(x, y)$, that approaches zero at the periphery of the kernel. This magnitude is controlled via

$$m(x, y) = (r + \varepsilon)^{-\gamma}, \quad (13)$$

where γ is a positive parameter that controls the rate of decrease in magnitude ($\gamma = 2$ in this application), and r is the radial distance with respect to the center of the kernel. ε is a small

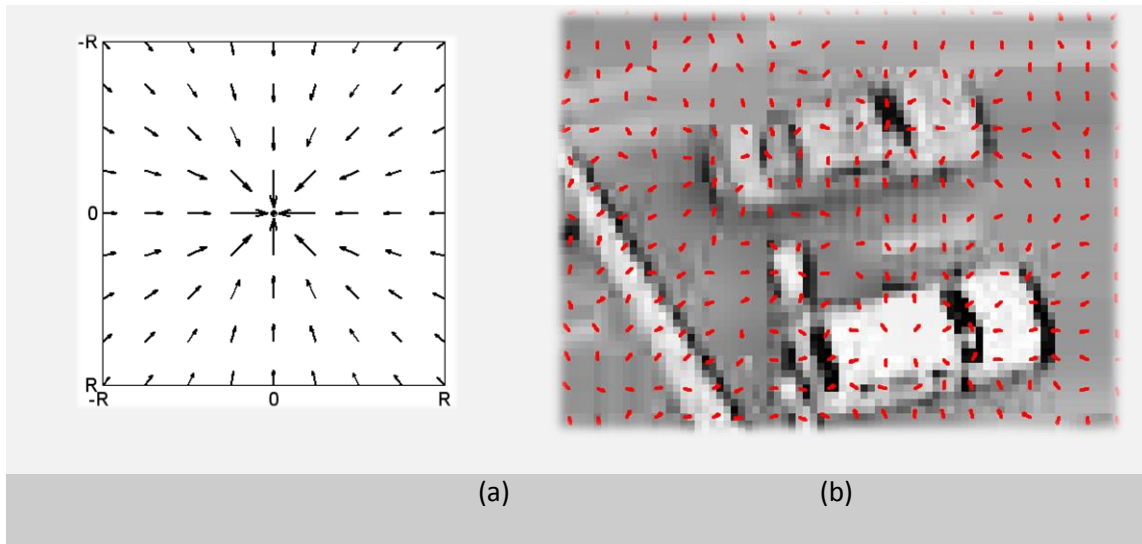


Figure 6 – Vector field convolution (VFC)

Here we present (a) an example of a discrete approximation of the VFC kernel, $\mathbf{k}(\mathbf{x}, \mathbf{y})$, with a radius of 4 and (b) an example of the VFC result (kernel from (a)) represented by red arrows; for clearer illustration purposes arrows are sampled at four pixels apart

positive constant that prevents division by zero at the origin. The continuous kernel \mathbf{k} is approximated using a discrete kernel \mathbf{k} with a predefined radius, R . A discrete kernel example is shown in Figure 6 (a). Figure 6 (b) demonstrates the capture range of VFC. The red arrows in the figure are pointing to the edges of the targets. We have chosen to use VFC over the commonly used generalized gradient vector flow [12] because of the improved and more controllable capture range, non-iterative and lower computational complexity, and the better performance of VFC in low signal to noise ratio scenarios [15].

3.2.3 Particle Filter

A tracker that solely relies on snake segmentation can suffer from target loss when the target moves more than half the target length from one frame to the next. For such a tracker to succeed, a high frame rate is required. By employing a dynamic model such as the particle filter (PF), the tracker will be more robust in lower temporal resolutions and will be able to handle occlusions. We also chose the PF over the KF because the PF is robust for a variety of state and observation models and it allows pursuing multiple hypotheses of the target state.

The PF is a sequential Monte Carlo numerical state estimation method which utilizes a likelihood model and a motion model to predict the subsequent state of the target. Unlike the Kalman filter family, which assumes a linear system with white noise, the PF is preferred in non-linear, non-white systems. The PF state is a nonlinear function of the previous state and the i.i.d. process noise, U , with known PDF is defined as:

$$Q_n = g(Q_{n-1}, U_{n-1}). \quad (14)$$

The noisy observations, Z_n , are related to the measurements via the measurement function:

$$Z_n = h(Q_n, v_n), \quad (15)$$

where v is the measurement noise with known PDF. Then, the PF is used to estimate the posterior density $p(Q_n|Z_{n-1})$ by a set number of samples or particles. The PF computes the state estimate based on random i.i.d. samples, s_i , drawn from the posterior (or importance) density of the previous state and the associated sample weights, ω_i . The number of samples or particles, N_s , can vary, but as N_s is increased, the PF approaches the optimal Bayesian estimate [4], [28].

Since the posterior density is unknown, the PF utilizes the particles and the weights to determine the posterior density of the current state. The discrete approximation of the posterior density at time point n is then defined as

$$p(Q_n|Z_{1:n}) \approx \sum_{i=1}^{N_s} \omega_i^n \delta(Q_n - s_i^n), \quad (16)$$

where δ is the Kronecker delta function and the weights are normalized such that they sum to one. Thus a region in the posterior density will result in high density if there are many particles in the region and/or the particles have high weights. The actual implementation of (16) takes the following form:

$$p(Q_n|Z_{1:n}) = c_t p(Z_n|Q_n) \sum_{i=1}^{N_s} \omega_i^{n-1} p(Q_n|Q_{n-1} = s_i^{n-1}), \quad (17)$$

where c_t is a normalizing factor. The measurement of the final state can be determined via one of the following calculations. First is the minimum mean squared error (MMSE) estimate:

$$Q_n = \sum_{i=1}^{N_s} \omega_i^n s_i^n, \quad (18)$$

and second is the maximum *a posteriori* estimate:

$$Q_n = s_j^n \text{ where } j = \arg \max_j \{\omega_1^n \dots \omega_{N_s}^n\}. \quad (19)$$

We choose the MMSE estimate from (18) as our preferred method.

An integral component to the success of the PF is the weight measurement. Several methods have been used to determine the weights of the particles, such as normalized cross-correlation [10] (NCC). NCC requires the knowledge of the target template, and if the target changes direction or the lighting changes it will impact the success of correctly generating high weights for particles on the target and low weights for those not on the target. We propose to use the contour from the snake segmentation to evaluate the optical flow for each particle for the calculation of each weight.

Lastly, PF suffers from a degeneracy problem. This problem occurs when after several iterations the weights of many particles become negligible. Consequently, we would like to remove particles with small weight and focus on those with large weight. In order to deal with this degeneracy phenomenon we employ the resampling scheme introduced in [29]. The method in [29] starts with a cumulative density function (CDF) of the weights. A starting point is drawn from

a uniform distribution $\mathbb{U}[0, N_s^{-1}]$. Then we move along the CDF and find a point that is lower than the starting point and assign it a higher weight.

3.3 Combined Snake PF (SPF)

The snake PF (SPF) algorithm commences by automatically initializing the moving targets, $Q[0]$, as described in the initialization section. Following the initialization we also generate contours for each target using the active contour in the first frame. Consequently, we use the location and the contour to initialize the snake for tracking in the subsequent frame.

Tracking is achieved by evolving the snake with VFC as an external force to track the target for the next two frames. Here we assume that the target size is large enough to allow a parametric curve to capture the target and that the approximate target size is known, which allows for snake initialization to capture the entire target. Below, Figure 7 (a) presents an example of the snake tracking results. Since the snake has a higher computational complexity and requires higher temporal resolution, we switch to the efficient PF for the remainder of a track. Switching to the PF permits us to track targets in lower temporal resolution if necessary. Further, the PF allows the tracking and investigation of a multitude of hypotheses regarding the target, whereas the snake accommodates only one target state at a given instant. The snake results from the first three frames, which establishes the motion model. The motion model is then used to continue the tracking with the PF and is updated in each frame.

While using the PF tracker, we adhere to the following procedures. Let the state be represented by the x-y position in the image:

$$Q_n = \begin{bmatrix} x[n] \\ y[n] \end{bmatrix}. \quad (20)$$

We generate particles that represent hypothetical positions of the target, $\hat{Q}[n]$. Here a Gaussian distribution is used to draw randomly-distributed samples in our first step of the tracker. Since the

motion of the target is established from the first three frames, it is incorporated into the PF model. We further utilize the snake result which produces the target contour to generate the weights for each particle. The weight is established by investigating the summed optical flow magnitude value for the area under the contour for a given particle. Let c_i^n be the contour that corresponds to particle s_i^n . Then, the weight is computed by way of a discrete-space approximation of

$$\omega_i^n = \oint_{c_i^n} |\nabla I| dc_i^n. \quad (21)$$

The final estimate of the state (target's location) is then determined by evaluating each particle based on the associated weight - finding the weighted average of the particle locations and their associated weights. Furthermore, the shape of the contour is also adjusted every 10 frames to account for any shape changes of the target.

3.4 Data Sets

We applied the SPF algorithm to track 28 targets that were acquired from a video camera mounted on a 30 story building located on Peachtree Street, Atlanta, GA. This is a stationary dataset with five and ten frames per second, taken at various locations along the street [30]. We also investigated the tracker on data that was captured in Yuma, AZ using an array of 2048X2048 pixel frames, with ground-sampling distance of 10 cm and visible cameras operating at 1 frame per second. The array was mounted on a rotary wing aircraft hovering 5000 feet directly above the field of regard. The sensors were aligned in a spiral pattern with 10% overlap in each FOV. The algorithm was only applied to the output selected FOV and seven targets were tracked with semi-manual registration.

Root mean squared error (RMSE) was used to evaluate the SPF algorithm. We measure the RMSE in sequences where the true target locations are not known. Consequently, the error measurement is based on the comparison with manually determined locations of the targets throughout the video sequence. We evaluated our SPF algorithm and compared it to the PF

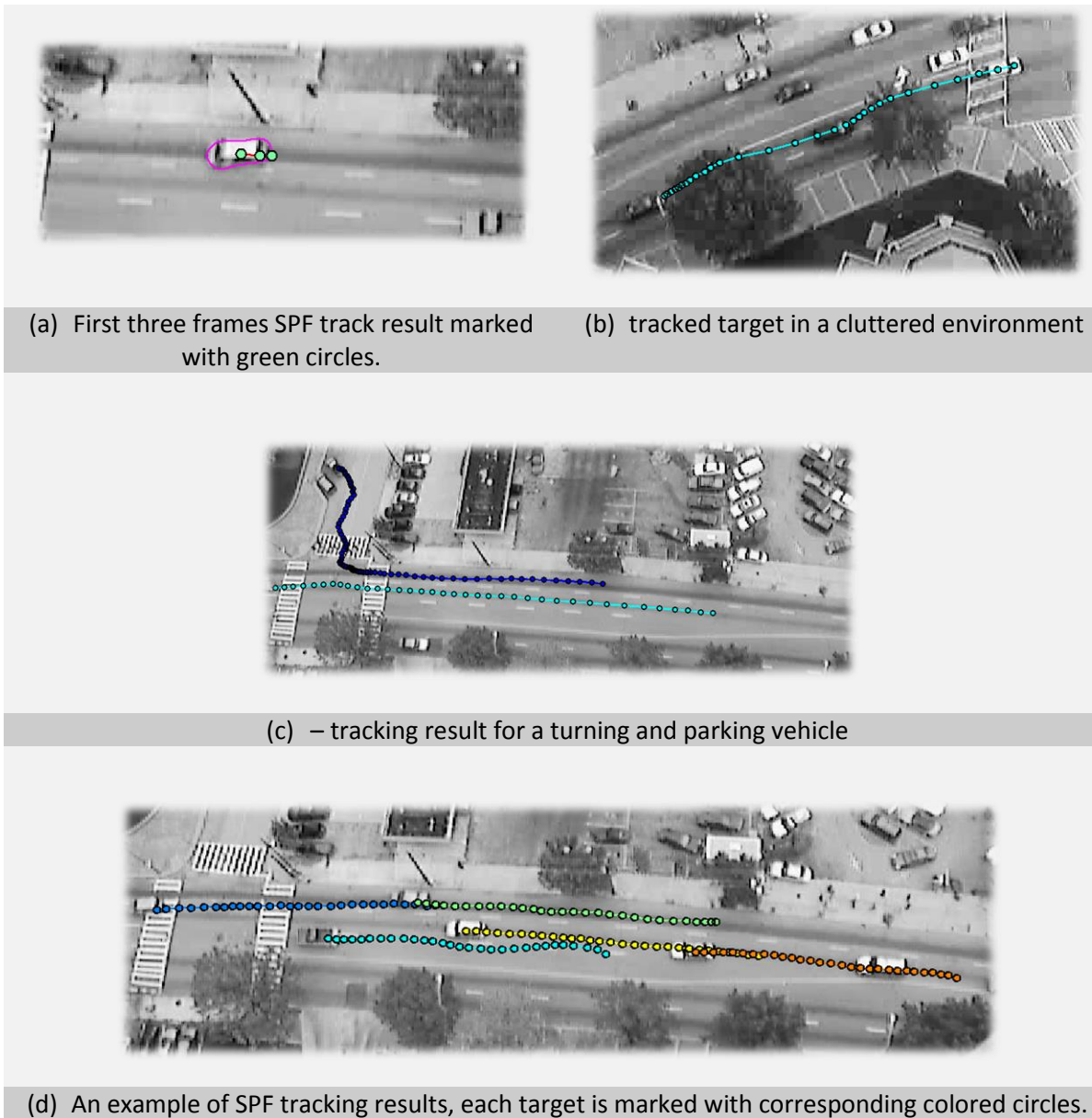


Figure 7 – SPF tracking results

The results first present (a) an example of the first three frames determined using the snake and followed by (b-d) examples of tracking results in 4 varying video sequences.

results in the given video sequences where the error is based on the manual tracking.

3.5 SPF Results

We first report the results for the SPF algorithm for the Atlanta dataset. Each target is approximately 30x10 pixels in size. Figure 7 (a) demonstrates the SPF tracking results for the first three frames. Figure 7 (b) provides an example of a single target tracked in a cluttered environment, while Figure 7 (c) illustrates the tracking result for a turning and parking vehicle. Furthermore, Figure 7 (d) demonstrates the tracking results for four targets in a 30 frame video sequence. Figure 8 (a) demonstrates another example of a tracked target in the Atlanta dataset,

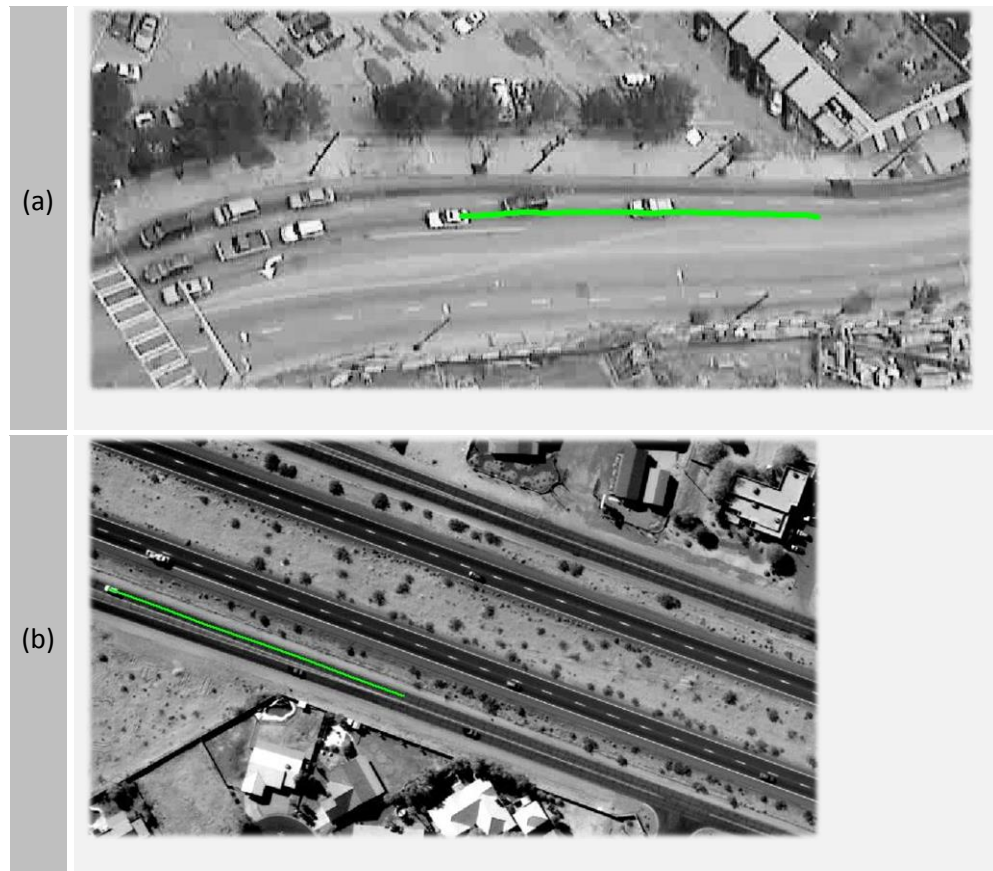


Figure 8 – Example of tracked targets

In this figure (a) presents another example from a result from the Atlanta dataset and (b) an example from a registered sequence from the Yuma, AZ data captured from a rotary wing aircraft. The paths the vehicles traveled are marked by the green lines.

while Figure 8 (b) illustrates tracking results in the Yuma dataset.

We compare the performance of the SPF tracker to a PF result with normalized cross correlation weighting. The SPF and the PF results are produced in Figure 9 comparing the trackers in the horizontal and vertical directions. In most cases the target movement is in the horizontal direction. We sought to investigate the difference in the trackers with the movement of the targets where there was not significant change in movement. The results demonstrate the superior performance of the SPF tracker compared to the PF. For the Atlanta data, the SPF tracker has on average an RMSE error of 7.0 pixels in the horizontal direction and 3.5 in the vertical, while the PF tracker has errors of 40.0 and 10.0 pixels, horizontally and vertically, respectively. The PF suffers from a dependence on image intensities and loses the target when there is clutter in the image. It performs poorly if the target nears another target or when the target passes close to a stationary object. It can be seen that the SPF method outperforms the PF.

The SPF algorithm was also used to track seven targets from the Yuma, AZ dataset, where the targets are approximately 20x10 pixels in size. In the Yuma, AZ (multi-camera) data set, the mean RMSE in the horizontal direction was 4.5 pixels and 4.0 pixels in the vertical direction. Figure 8 (b) provides an example of a target tracked in the Yuma, AZ dataset. Figure 10 illustrates the RMSE results for the seven targets from the Yuma, AZ dataset. RMSE was measured in both the horizontal and vertical directions. It can be noted that overall the RMSE was less than 10 pixels.

We used a Windows 7 64-bit home edition PC, with Intel® Pentium® P6100 dual core processor, with 3GB RAM. The execution time is on average 1.5 seconds for initialization, where calculating the frame median takes half of the total time of 0.7 seconds. Snake evolution for a 30x10 pixel target requires 0.1 seconds on average per target per frame and computation of 100 particles per target per frame entails approximately 1.5 seconds.

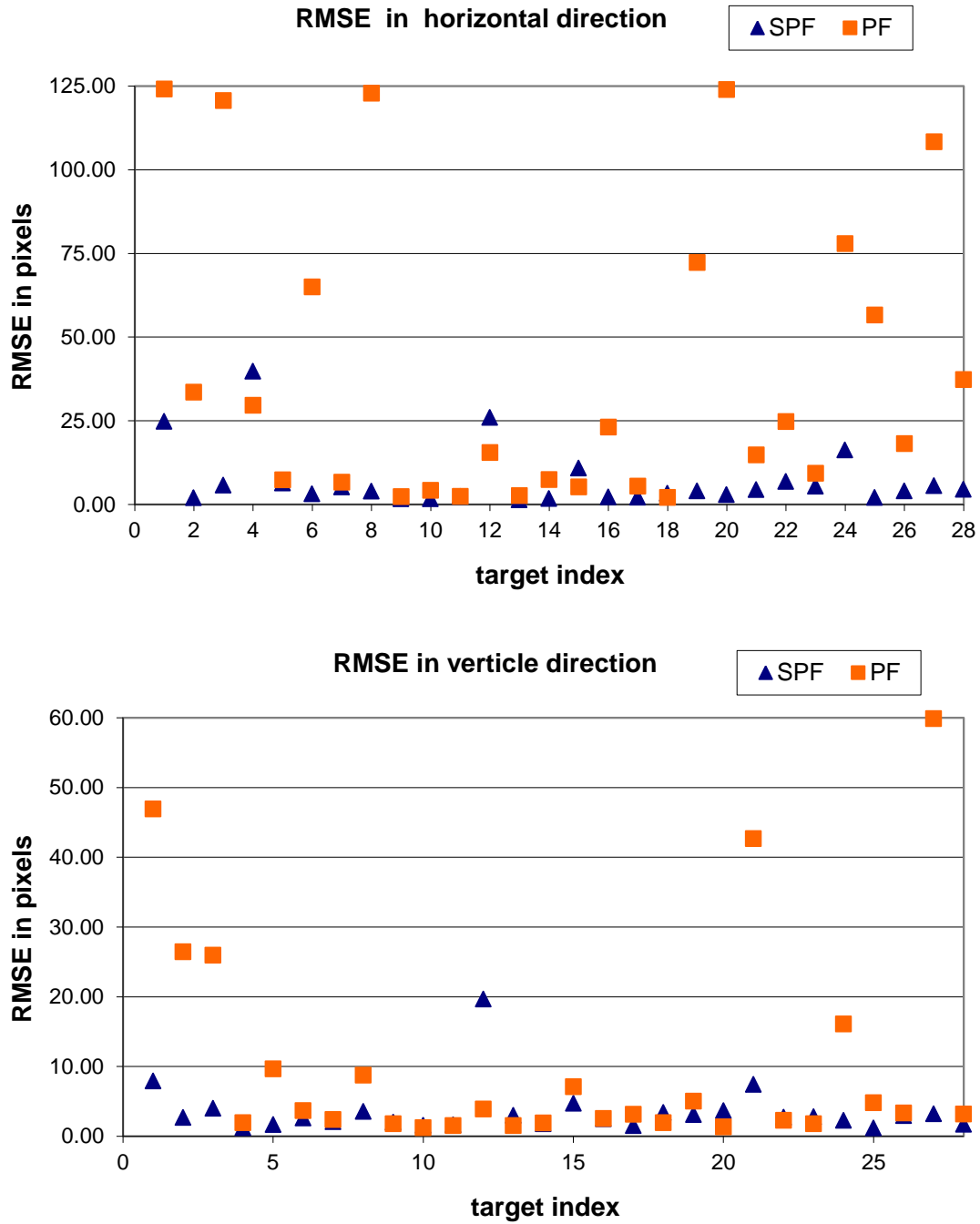


Figure 9 – Tracking RMSE results, where $N_s = 100$

These results illustrate the RMSE in the more common direction of movement (horizontal) and the vertical direction. It can be seen from the results that overall the SPF tracker outperforms the traditional particle filter in both directions. Furthermore, it can be observed that in the direction of the target's movement, the PF has a higher rate of error.

The success of the SPF algorithm requires the targets to overlap from frame to frame. If the target of interest moves more than one target size away in the next frame, the snake will not be able to capture the target. Additionally, the SPF algorithm utilized the expectation that image sequences of interest are either captured with a stationary camera or have been registered. While there is a significant amount of data from stationary cameras, persistent surveillance is often acquired from aircrafts, resulting in unregistered and jittery imagery. There exist many registration algorithms [31], but registration introduces another layer of complexity. In addition to increasing the overall processing time and complexity, registration tries to align all frames via a variety of transformations. These transformations introduce changes to the image that might augment the true target from frame to frame. Furthermore, poor registration will cause the SPF and other algorithms relying on registration to fail due to lack of a clear background image.

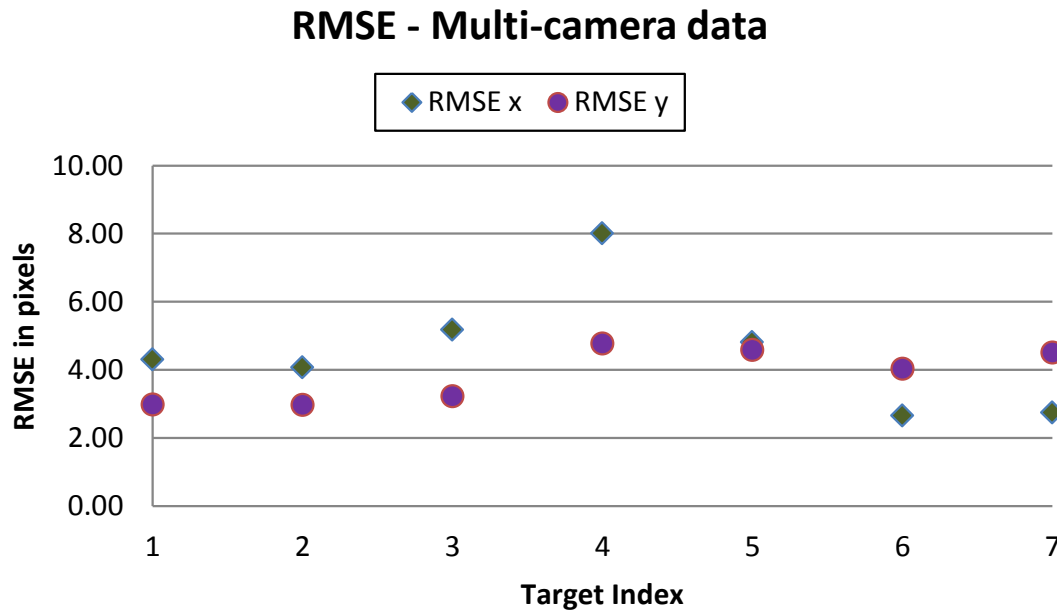


Figure 10 – Yuma, AZ SPF tracking RMSE

For these results we employed 100 samples (N_s). This figure illustrates results for seven targets with RMSE recorded in horizontal and vertical directions. The main motion of the targets was in the horizontal direction. The maximum error in the horizontal direction is less than the width or length of the target where the target size averages 15 pixels wide and 50 pixels in length.

3.6 Conclusion

Here we presented a novel tracker for video surveillance applications. This SPF tracker unites the snake and the PF. The snake is used to establish the motion model and to determine particle weights, while the PF is employed to handle non-linear, non-Gaussian systems. A low RMSE between the results of SPF and manual tracked targets was 7 pixels in horizontal and 3.5 pixels in vertical directions. As future work, SPF can be extended to include automatic monitoring for incoming targets, and the characterization of target-to-target interaction.

This chapter addressed tracking in registered video or video collected from stationary cameras. We next introduce a new tracking methodology for tracking in unregistered video sequences. Chapter 4 addresses detection and tracking in unregistered images.

Specific Aim 1: To develop an automated tracking algorithm for a single moving target in surveillance video sequences by fusing a particle filter with the active contour.

This chapter has addressed the development of an automated tracking algorithm for a single moving target in surveillance video. The algorithm, the snake particle filter (SPF), utilizes both the active contour and the particle filter to achieve tracking. SPF employs the active contours to establish the weight for each particle in the PF. We have demonstrated the tracking results using our SPF methodology in two datasets. We have shown here that our algorithm outperforms the PF where weights are defined by NCC. The SPF algorithm has been published in [32].

CHAPTER 4.

TARGET TRACKING VIA MORPHOLOGICAL SCALE-SPACE TRACKERS

Our first aim of tracking registered images was addressed in Chapter 3. Chapter 3 employed the background for target detection and tracking. Additionally, the stationary nature of the video camera in Chapter 3 allowed us to establish a motion model, which was incorporated into the particle filter. This chapter tackles a more complex problem – the tracking of targets in a non-registered video sequence, which is addressed in our second aim.

While registration has been a topic of great interest in the image analysis community, video registration is a time consuming process. As spatial and temporal resolutions of the video sequences increase, the computational overhead of registration grows. Moreover, registration still suffers from inaccurate frame alignments and other inaccuracies, such as drift. Drift can cause a tracking method which utilizes motion models to misrepresent target motion resulting in a flawed predicted target location. To address this concern, some methods attempt to provide an accuracy measure. With automated methods, however, it is challenging to distinguish between registration errors and actual changes that occur from one frame to the next [31]. Consequently, methods which rely on video registration will fail to track the targets of interest in poorly registered or unregistered video sequences.

Additionally, this work is not just used to address tracking in unregistered sequences, but in sequences with large displacement due to jitter. The movement caused by inter-frame jitter combined with that of the target, may be multiples of the target in size. For example, the Yuma, AZ data we introduced in Chapter 3, when unregistered, illustrates this image sequence behavior. In these imaging sequences, the target can move as far as 600 pixels in each direction (total

distance of more than 700 pixels) from one frame to the next. This location change includes camera movement as well as the movement of the target. From registering these video sequences we found that of the 700 pixels, approximately 100 pixels is the actual distance of travel and the rest is camera movement. Towards resolving this obstacle, we employ methods which try to mimic human detection and recognition of objects in a given scene.

The algorithm presented here assumes that we have unregistered data sequences with at least 150 pixels per target and that there is some contrast between the background and the target. The spatial resolution must be known to the algorithm as well as the size of the targets of interest. We further assume that if the target leaves the FOV it must re-enter within the next two consecutive frames, otherwise the track for that target will be terminated. No assumptions are made regarding the motion of the target or the background.

In David Marr's *Vision*, the author describes vision from the perspective of neuroscience, as a task of information processing [33]. Furthermore, it has been shown that one can observe that objects in the real-world have structures of varying scales [34]. Thus, one way to tackle information processing from an image is to decompose it to several scales – often referred to as 'scale-space.' In a nutshell, a scale-space is a collection of signals representing the same scene, which vary from fine to coarse. Significant attention in the object recognition and target tracking communities has been paid to the subject of scale-space

A scale-space is typically created by a scale generating filter. The most widely used filter for scale generation is the Gaussian filter. The Gaussian has been applied in the scale invariant feature transform (SIFT) algorithm [35]. In [36], Zhou et al. utilize SIFT and Mean Shift algorithms to track targets from stationary cameras where targets occupy a large number of pixels in the frame. This tracker uses a pre-defined area of interest to track and incorporate the color histogram into the descriptor for tracking. Furthermore, the tracker assumes the camera is

stationary because it uses the location from the previous frame to conduct the search for matching key-points in the current frame. The authors demonstrate a successful tracker, which encouraged us to investigate SIFT as a target tracker in unregistered video sequences with low spatial resolution per target. We seek to track specific targets in each frame and understand each target's unique movement as it changes through time. Towards this end we introduce the automated SIFT tracker (ASIFT²).

This chapter commences by introducing the dataset which is used for the purpose of this work. We then describe the high-level theory of SIFT and how it is incorporated into the ASIFT². We then introduce a new scale-space paradigm, built around a connected filter with several attractive properties. Following, we discuss our approach to detection and tracking using the connected component scale-space, provide tracking results, and discussion. Lastly, we investigate incorporating SIFT features into the connected component scale-space tracker and compare and contrast the tracking results with the individual methods.

4.1 Video Sequences of Interest

Tracking is conducted on datasets captured using an array of 16 2048X2048 pixel sensors, with ground-sampling distance of 10 cm and visible cameras operating at 1 frame per second. The array was mounted on a rotary wing aircraft hovering 5000 feet directly above the field of regard. The sensors were aligned in a spiral pattern with 10% overlap in each FOV. The results for algorithms presented here and in Chapter 5 were only applied to the output of a single camera at a time.

4.2 Invariant Feature Transform

Much work has been carried out on methods to extract image features that allow for matching similar objects in separate images. The Harris corner detector [37] has been applied to establish

features that can be compared in several images to find a match. The authors in [38] first found the corners in the image and then correlated a window around the edges to find likely matches. As research in this area progressed, researchers noticed that the Harris corner detector is sensitive to size and therefore two images of slightly different sizes might not match using this method. Belongie et al. [39] introduced shape descriptors that describe every point of the shape relative to all other points. Two descriptors are then compared to find the similarities between the two shapes. This method is based on the boundaries of the image rather than the intensities of the image, thus requiring an extra step of boundary extraction before the matching can be done. Consequently, poor boundary detection will lead to poor performance of this method.

Koenderink observed in [40] that a Gaussian kernel provides a tool to derive a one-parameter family of images from a single image. Koenderink introduced a scale-space using the Gaussian kernel as a foundation of extracting features that are not sensitive to noise, changes in illumination, scaling, rotation, or small changes in view point. This scale-space idea was the foundation of the method developed by Lowe [35] - the scale invariant feature transform (SIFT). SIFT provides a framework for establishing highly distinctive features. These features can be matched with high probability against a large dataset of features.

4.2.1 Scale Invariant Feature Transform (SIFT)

This section describes the high-level formulation of SIFT. (Readers can explore the complete theory and development steps of SIFT in [35].) The foundation of SIFT is scale-space, where scale-space is a set of grouped collections of Gaussian smoothed images and differences of Gaussians (DoGs). The DoGs are determined from differencing the image which was filtered by the two-dimensional Gaussian filter with varying standard deviations. Thus, the algorithm commences by creating a set of Gaussian filtered images, $L(x, y, \lambda_i \sigma)$:

$$L(x, y, \lambda_i \sigma) = G(x, y, \lambda_i \sigma) * I(x, y). \quad (22)$$

Here, an image, $I(x, y)$, is convolved with a Gaussian point spread function, $G(x, y, \lambda_i \sigma)$, for a given standard deviation, σ , and a varying scale factor, λ , where

$$\lambda_i = 2^{1/i}, \text{ where } i = \{1, 2, 3 \dots N_i\}, \text{ and} \quad (23)$$

N_i is the total number of scales in an octave. Also, the two-dimensional Gaussian is

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \quad (24)$$

Note that the standard deviation is the same for both x and y .

The next step is to generate the DoGs, $D(x, y, \lambda_i \sigma)$. Since the size of the Gaussian variance or kernel is directly related to spatial resolution, as we vary λ_i , $\lambda_i \sigma$ becomes the measure of spatial scale in the smoothed signal at scale $\lambda_i \sigma$ [41]. Note that this scale is relative, in the sense that increasing values of $\lambda_i \sigma$ will be increasingly coarser in scale. However, this quantification of scale does not provide a direct threshold on scale. For example, one cannot eliminate objects below a certain area or radius by specifying $\lambda_i \sigma$.

$$D(x, y, \lambda_i \sigma) = L(x, y, \lambda_i \sigma) - L(x, y, \lambda_{i-1} \sigma). \quad (25)$$

Here we start generating DoGs from the second smoothed image in the stack, $L(x, y, \lambda_2 \sigma)$. The DoGs are then collected into octaves, where each octave doubles the standard deviation. The first octave is at the original spatial resolution; the next one is downsampled by two and so on. We found that two octaves were sufficient in our application because with increased blurring and downsampling the targets blend in with the background, as we discuss later in this chapter. The scale-space of two octaves and four DoGs is illustrated in Figure 11. We can observe that as λ increases, the Gaussian kernel increases, resulting in increasing smoothing of the image. Furthermore, as we downsample the octaves, the smoothing also increases, since the smaller features now become smaller in scale relative to the Gaussian kernel.

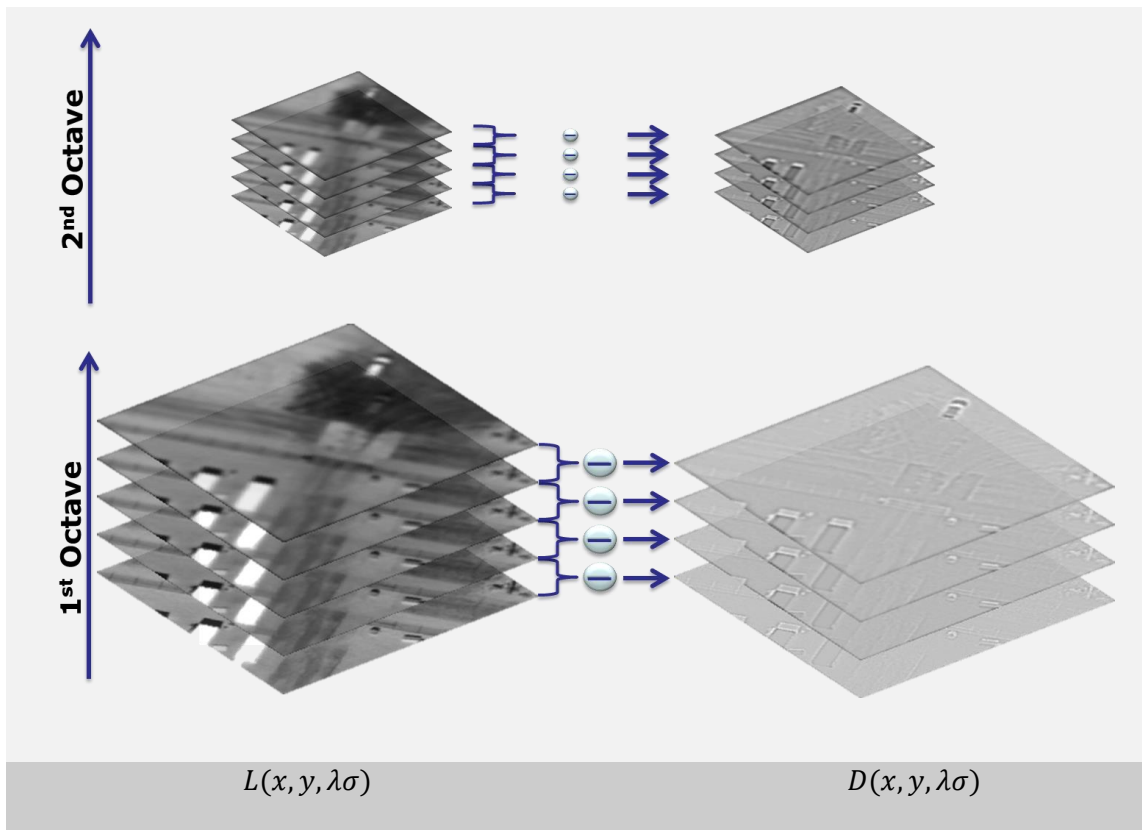


Figure 11 – Two octaves of scale-space and difference of Gaussians (DoG)

The stacks on the left illustrate the blurred images by a Gaussian, while the stacks of images on the right illustrate the DoGs as the adjacent sets of blurred images are subtracted. Additionally, the two smaller stacks on top are the downsampled versions of the images.

The next step is to find the features, also termed key-points, in this scale-space. Detection of key-points is accomplished by locating the minima and maxima in a 27-pixel region in the scale-space. That is, every pixel in each scale is compared to its 8 neighbors in the same scale and the 9 neighbors in the scale above and below, as demonstrated in Figure 12. This process generates a large number of points, and to make the features less sensitive to changes in the image, some key-points must be removed. Consequently, in order to eliminate unstable points, points with bad contrast and points that lay on edges in the image are eliminated.

At this point in the algorithm, the process of generating a descriptor for each key-point can commence. Towards this end we explore the gradient of the scale-space. For every scale we

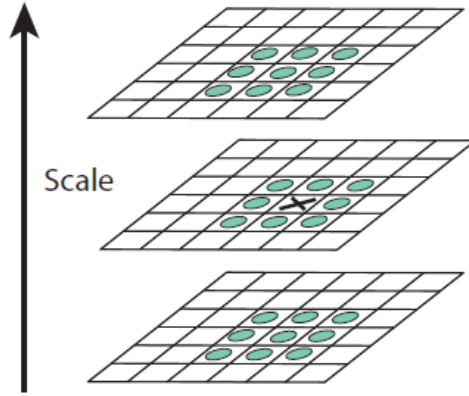


Figure 12 – Visualization of 3-D neighborhood [35]

In order to calculate the magnitude and orientation of the key-points, the 3 D neighborhood of each key-point is used. In this figure the pixel of interest (the key-point) is marked with an X and it shows the 8 neighboring pixels from the same scale, 9 from the scale above and 9 from the scale below.

determine the gradient magnitude, $f_g(x, y)$, and the orientation, $\theta_g(x, y)$, as follows. First we find the gradient for a given scale, $\lambda_i \sigma$:

$$[g_x, g_y] = \nabla L(x, y) \quad (26)$$

where g_x, g_y are the gradient components in x and y directions, respectively for the closest scale, L , to each key-point. Then

$$f_g(x, y) = \sqrt{g_x^2 + g_y^2} \text{ or in discrete form} \quad (27)$$

$$f_g(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2}$$

and

$$\theta_g(x, y) = \tan^{-1} \left(\frac{g_y}{g_x} \right) \text{ or in discrete form}$$

$$\theta_g(x, y) = \tan^{-1} \left(\frac{L(x, y + 1) - L(x, y - 1)}{L(x + 1, y) - L(x - 1, y)} \right). \quad (28)$$

The resulting magnitude, $f_g(x,y)$, and the orientation, $\theta_g(x,y)$, are used to construct a histogram in an 8x8 window around each key-point, with 10 degrees for each bin of the histogram, where each orientation sample added is weighted by its magnitude. The maximum bin is used to assign the orientation for the given key-point.

To match key-points in separate images, a region descriptor is needed. This descriptor is a collection of histograms associated with the key-point. A 16x16 window around each key-point is used for the region descriptor. The orientation associated with the key-point is used to rotate the gradient in order to align the window with the y-axis. The rotation is essential to preserve rotation invariance. Then the 16x16 pixel window is divided into 16 square sections. For each section an 8 bin orientation histogram is recorded; the gradient magnitude is normalized to unit length before constructing the histogram to remove illumination changes.

Finally, the matching is achieved by calculating the Euclidean distance between the key-point region descriptor histograms in our image of interest to all the key-points and their

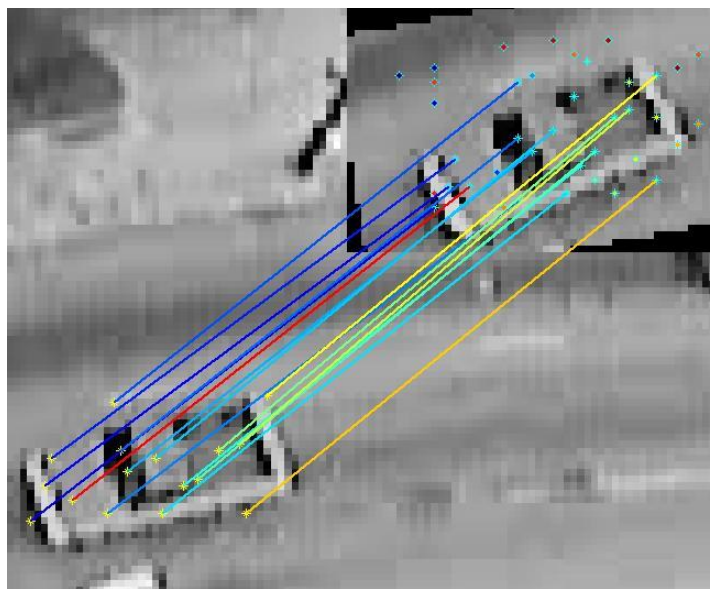


Figure 13 – Example of matched key points using SIFT with 10° rotation

An example of matching a rotated target (top right corner) and the current frame is illustrated. The target has all the key-points noted, while the current frame only shows matching key-points. Note that several points from the road (outside of the target) are matched in this case. This example uses the higher spatial resolution data from Atlanta, GA.

associated histograms in the image being matched. The minimum distance is the chosen matching feature. We have to make sure that there is one-to-one correspondence between all points. SIFT allows detecting targets in the image even if they have rotated, scaled, or changed illumination. Figure 13 shows an example of a SIFT match. In this figure we illustrate an example of taking the same target and rotating it by 10 degrees prior to matching. This example shows the key-points on the initialized target (top right corner) and their matches on the frame being tracked. Also, it is important to note that since SIFT does not distinguish between the target and the background, some key-points from the background (the road) are matched as well. This example uses the higher resolution data from Atlanta, GA.

4.2.2 Automated SIFT Tracker (ASIFT²)

In order to utilize SIFT in vehicle tracking we establish the Automated SIFT Tracker (ASIFT²) and we followed the subsequent steps. As SIFT does not offer a way to identify targets,

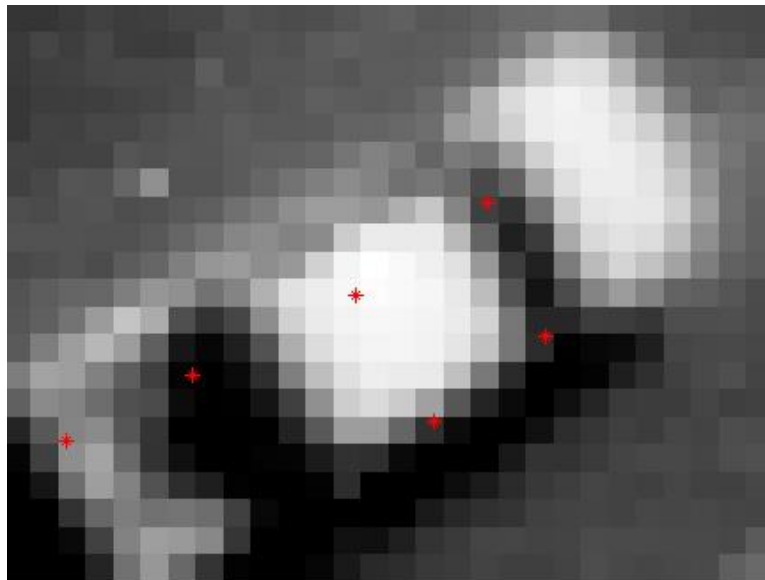


Figure 14 – Example of SIFT key-points in target initialization

We illustrate a target with 6 SIFT key-points marked in red. This example was created using a target mask to only capture key-points on the target and not the background. This example shows the lower resolution data from Yuma, AZ. Compared to the earlier figure, this example shows that the lower spatial resolution results in much fewer key-points on the target.

we used manual initialization or our Morphological Scale-Space Tracker detection as initialization (described later in this chapter). The manual initialization step for ASIFT² requires the target location as well as the target size in order to generate SIFT key-points to match to subsequent frames. We segment the target of interest from the image by employing the target's bounding box (the largest box that captures the entire target) and extract key-points from the entire box which may include key-points falling on the background as SIFT does not discern between target and background. Figure 14 illustrates an example of a target with SIFT key-points marked in red. In this figure, 6 SIFT key-points are detected. Figure 14 illustrates an example from the lower spatial resolution image compared to Figure 13. It can be observed that the lower spatial resolution target has fewer key-points, thus fewer matching opportunities for the SIFT tracker.

The next step in the tracking process is to calculate SIFT key-points for the consequent frames. Following the SIFT key-point detection in the following frame, a SIFT matching algorithm, as described earlier, is used to match all SIFT key-points in the image and the initialized target. Since we are using unregistered images and do not have a prediction of target location, the entire image must be analyzed. As a result this process is time consuming especially with high spatial resolution.

Lastly, one or more key-points are matched to determine where in the image the target is located. A match of initialized target and a cropped target (the rest of the image is removed for matching demonstration purposes only) from the consecutive frame is illustrated in Figure 15. Note here that there are 7 matches, but only 5 lines are shown. With SIFT, 2 key-points can overlap from different octaves.

Since SIFT does not capture the center of the target, the key-points can only provide an approximate center. If the key-points are matched covering the target area, the median of the points can give a relatively good center of mass estimate; if the points are skewed to one side of

the target as in Figure 15, the median is skewed to that side of the target. We have found that in some frames only 1 key-point is matched to the target. Furthermore, we chose to employ the median and not the mean to eliminate potential bad matches that matched elsewhere in the image.

4.2.3 ASIFT² Tracking Results

We have applied the SIFT tracker to 34 targets with varying numbers of frames for each target, including frames where the target is no longer in the field of view. Here, initialization is performed manually, although, given a library of SIFT features, the extension to automated initialization would be straightforward. The results are tabulated in Table 1. It is important to note that during initialization, great care went into capturing key-points on the target and not on the background next to the target. Additionally, since the key-points are scattered on the target, we utilized the median location for the horizontal and vertical positions and removed outliers assuming that most key-points were on the target.

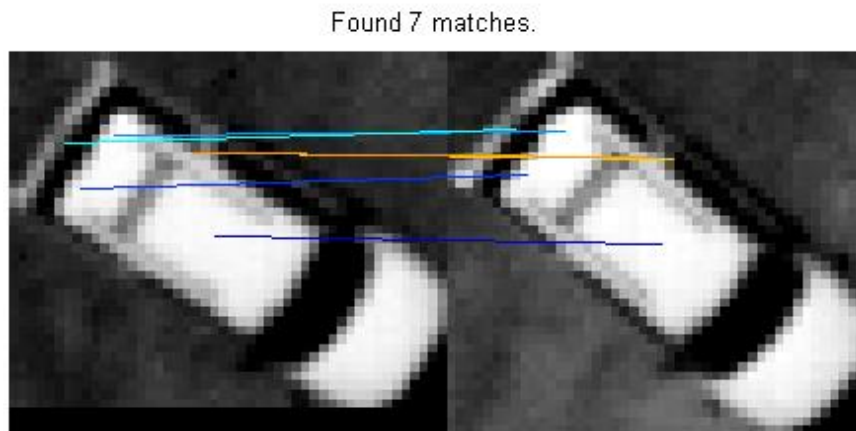


Figure 15 – Example of SIFT target matching

This figure illustrates a matching example of SIFT key-points. On the left is the originally initialized target; on the right is the matched target in the second frame. Here the target is zoomed in to illustrate the matching. Note that 7 matches were found, while only 6 lines are marked because one set of key-points overlap as they come from two distinct octaves.

The table demonstrates results for ASIFT² as described above. The table includes the targets, number of frames in each sequence of the tracked target, normalized RMSE results as compared to ground truth, the probability of detection – what percentage of the frames’ detection falls within the area of the target –, and lastly the normalized RMSE of only the targets that were tracked correctly. The normalized RMSE is the distance RMSE between the ground truth and the detected location, normalized by the target width. The first RMSE reported is for all frames including those that the tracking has missed; in the case of a missed target we use the previous location of correctly tracked target in the RMSE calculation.

There are two scenarios when the tracking was missed in the case of SIFT. In one, the target was tracked (matched) to a wrong target (or noise) and thus there is detection, but it is incorrect (false positive). In the other, the tracker was unable to find the target, but the target does exist in the frame (false negative). In this second case, the target location is recorded as zero for both x and y . Consequently, the last column, reporting only the RMSE for successfully tracked frames, is important because measuring an RMSE value for missed targets does not reflect the tracking error correctly. The error of missing a target is captured by the “% Detection of Frames”

measure. It is important to note that the tracked RMSE is on average about half target size (0.45) off the center of the target. This RMSE indicates that the detected location of the target center from tracker can be on the edge of the target and in some cases off the target, which is an undesirable result.

While SIFT has demonstrated positive results in matching large objects with a significant number of pixels, it often fails in matching smaller targets and thus in tracking our targets of interest because of the lack of spatial resolution. Since there are not enough pixels to describe the targets, only a few key-points can be used for matching, reducing the accuracy of SIFT. Often, only one key-point will be identified on the target. Furthermore, since the key-points are not necessarily in the center of the target, and unless provided in manual initialization, we have no *a priori* knowledge of the location boundaries of the target. Therefore, we cannot tell where the points are relative to the target. Accordingly, there is no clear way to combine the location of the matched key-points to accurately determine the center location of the target. This is especially challenging if two points are matched and are further than a target distance apart. One of the matched points could be on the target, but there is no way to tell which one is a false-positive match.

Moreover, SIFT results in a large number of key-points in a given image. The author noted that *"500x500 pixels will give rise to about 2000 stable features [35]."* As a result, since the motion of the target cannot be approximated and thus the search area cannot be reduced, we must search the entire image for matching points. Sophisticated parallel processing must be utilized to improve the execution time of this method.

Table 1 - ASIFT² Tracking Results

Target index	Number of frames	Normalized RMSE	% Frames tracked	Tracked normalized RMSE
1	13	3.63	83%	0.68
2	12	22.75	45%	0.37
3	11	3.14	90%	0.65
4	11	26.97	10%	0.49
5	8	0.85	100%	0.85
6	7	0.38	100%	0.38
7	10	0.36	100%	0.36
8	12	8.90	55%	0.31
9	12	1.52	91%	0.50
10	7	0.31	100%	0.31
11	6	0.14	100%	0.14
12	10	0.31	100%	0.31
13	18	0.80	100%	0.80
14	18	2.84	94%	0.40
15	9	0.38	100%	0.38
16	9	26.44	75%	0.28
17	9	0.26	100%	0.27
18	7	0.18	100%	0.18
19	6	0.21	100%	0.21
20	9	4.72	75%	0.57
21	12	6.75	91%	0.33
22	12	0.62	100%	0.62
23	8	0.70	100%	0.70
24	11	0.52	100%	0.52
25	10	3.54	90%	0.25
26	10	0.54	100%	0.54
27	10	0.36	100%	0.36
28	13	6.12	67%	0.61
29	8	0.76	100%	0.76
30	7	0.57	100%	0.57
31	9	0.64	100%	0.64
32	6	10.31	88%	0.39
33	9	0.41	100%	0.41
34	8	0.24	100%	0.24
Average		4.03	90%	0.45

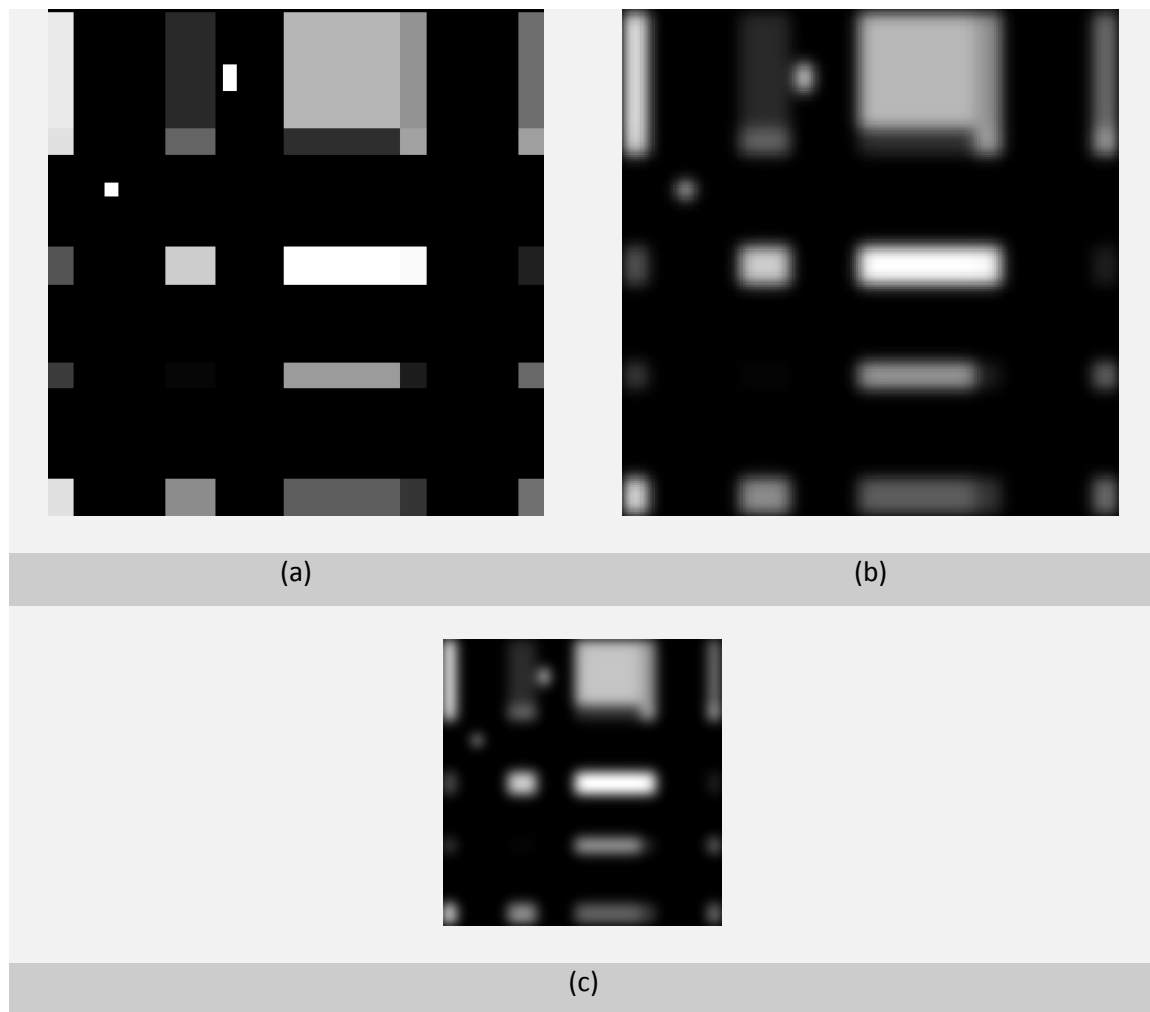


Figure 16 – SIFT scale-space example

The figure illustrates (a) the original image, (b) an example of scale-space level four in octave one, and (c) an example of scale-space level four in octave two. It is important to note the small objects in the image start to merge with the background, while the sharp edges of the larger objects become blurry. Thus it is hard to establish where the object ends and the background begins.

In addition, drawbacks of SIFT center around its constituent scale generating filter – the Gaussian. Using the Gaussian filter to construct the scale-space, we encounter feature drift (movement of edges) as the scale becomes coarser [42]. This distortion of edges across scale aggravates the correspondence problem (tracking features from fine to coarse), especially with poorer spatial resolution. The distortion also leads to ambiguity in the description of the target and to blurring and merging of the target with the background. Furthermore, as noted earlier, the Gaussian filter in SIFT uses a spatial scale parameter, $\lambda\sigma$, that specifies the standard deviation of

the kernel applied. This parameter does not directly specify the area or width and length of features that are eliminated by applying a larger scale version of the filter. So, another concern with the Gaussian scale-space is the inability to directly specify the scale of features that are eliminated in a coarse version of the signal. Figure 16 illustrates an example where image details can be blurred by the kernel and even merge with the surrounding objects. Note the small rectangle in the top left corner adjacent to another shape. This small rectangle connects to another shape in the image in the second octave at a courser scale.

As noted earlier, SIFT has been successfully employed as a tracker of entire frame tracking as well as in initial attempts at tracking large objects from stationary cameras [36]. ASIFT², on the other hand, is a novel approach for employing the SIFT algorithm for tracking small targets in unregistered video with low spatial resolution in large frames.

Whereas in ASIFT² we can utilize the SIFT algorithm to track targets from frame to frame, we cannot use it to detect targets in the initial frame. Since SIFT seeks to match sets of key-points without having to know to which objects the key-points belong, when employing SIFT for object tracking we must rely on the key-point description of the target in advance. Thus, the ASIFT² algorithm and other algorithms that may utilize SIFT for specific target tracking require a preset table of targets of interest, manual initialization, or exploitation of another initialization algorithm.

One way we could proceed would be to pre-process all the targets and record all objects and their associated key-points and descriptors. If such information is known, a database of features can be generated. Still, it is most likely that the user does not have all the possible targets of interest cataloged in a database. Such collection will require a considerable amount of time and resources. To address the shortcomings of tracking with SIFT, we present our novel algorithm, Morphological Scale-Space Tracker (MS²T). MS²T is developed by incorporating

connected components. It tackles the aforementioned shortcomings of SIFT, while allowing the detection of targets in unregistered video sequences.

4.3 Connected Components

Employing image analysis at the pixel level can be restrictive due to the large number of pixels in an average image. Using connected components introduces an abstraction with a lower number of regions of interest and, to the contrary, does not require the utilization of all the pixels. The MS²T employs a novel connected filter approach described below, which takes the same basic ideology of SIFT – both approaches use a scale-space to recognize invariant features that can be identified in various scales, viewpoints, and illumination. Both approaches find these features by differencing successive layers of the scale-space. Finally, both approaches allow correspondence (matching) of these features to achieve tracking.

4.3.4 Morphological Scale-Space Tracker and the Connected Filter

Given a binary image, a connected filter operates by keeping or removing connected components (connected blocks of ones or zeroes), where connectivity is defined by four-connected or eight-connected pixels. In a grayscale image, a connected filter operates on each gray level resulting in a stack of binary images, one for each grayscale level. Connected filters play a vital role in image analysis because they allow for adequate contour preservation.

In morphological operators, such as open and close, a structuring element is required to perform the operation. Using a structuring element introduces a drawback because the shape of the structuring element can introduce distortion to the image. The use of connected filters in area morphology removes the reliance on proper selection of the structuring element. Connected operators also do not introduce artifacts to the output such as new structure.

In order to extract targets of interest we employ the connected filter (area open) method for bright targets. Conversely, area close can be employed for dark targets. We employ the area open method on a grayscale image, I . The grayscale image is decomposed into a number of binary (thresholded) image representations called level sets. Each level set is thresholded based on a given area size; connected components for the given area are then retained or deleted based on the threshold. This is done by using the number of connected pixels to the current pixel; in two-dimensional images, and as noted above, there may be either of four or eight-connected pixels. Finally, the level sets are collected into one resulting image.

Our method, the **Morphological Scale-Space Tracker (MS²T)**, utilizes morphological scale-space to track. To generate the scale-space the area open method is applied to retain bright objects that have an area greater than a specified minimum area, I_{o1} , thus removing small undesirable objects such as noise. This area is determined to be less than the area of our target of interest, e.g. vehicles. The bright objects that are slightly larger than the target of interest, I_{o2} , are also retained. Consequently, we are able to retain the objects of interest by differencing:

$$DF = I_{o1} - I_{o2}. \quad (29)$$

By subtracting the smaller than target size objects we detect the objects of interest. The next step

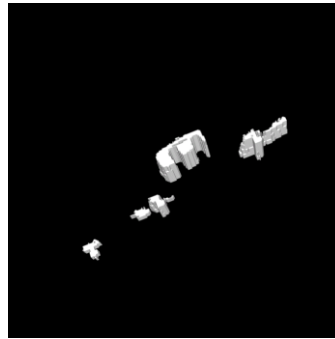


Figure 17 –Example of a morphological scale-space

A target of interest is cropped from an image to illustrate a four level morphological scale-space along with some debris next to the target.

is to generate the scale-space as shown in Figure 17. As with SIFT, this is achieved by selecting specific difference magnitude values and generating a scale-space $nI(x, y, \lambda)$.

$$nI(x, y, \lambda) = \begin{cases} 0, & DF < o\lambda \\ 1, & DF \geq o\lambda \end{cases}, \quad (30)$$

where o is a constant scalar. Each object in the image thus results in a three-dimensional structure. So, (30) invokes a lower bound (threshold) on the difference values. Figure 17 illustrates an example of four levels of the generated scale-space from an image subsection. Here the target is the largest structure, with several other structures (debris) around it. We will later discuss how we distinguish between the targets and the debris.

With the aim of target tracking we must be able to describe the target in each frame and associate it correctly through time. Towards this end, we utilize the difference to first generate a

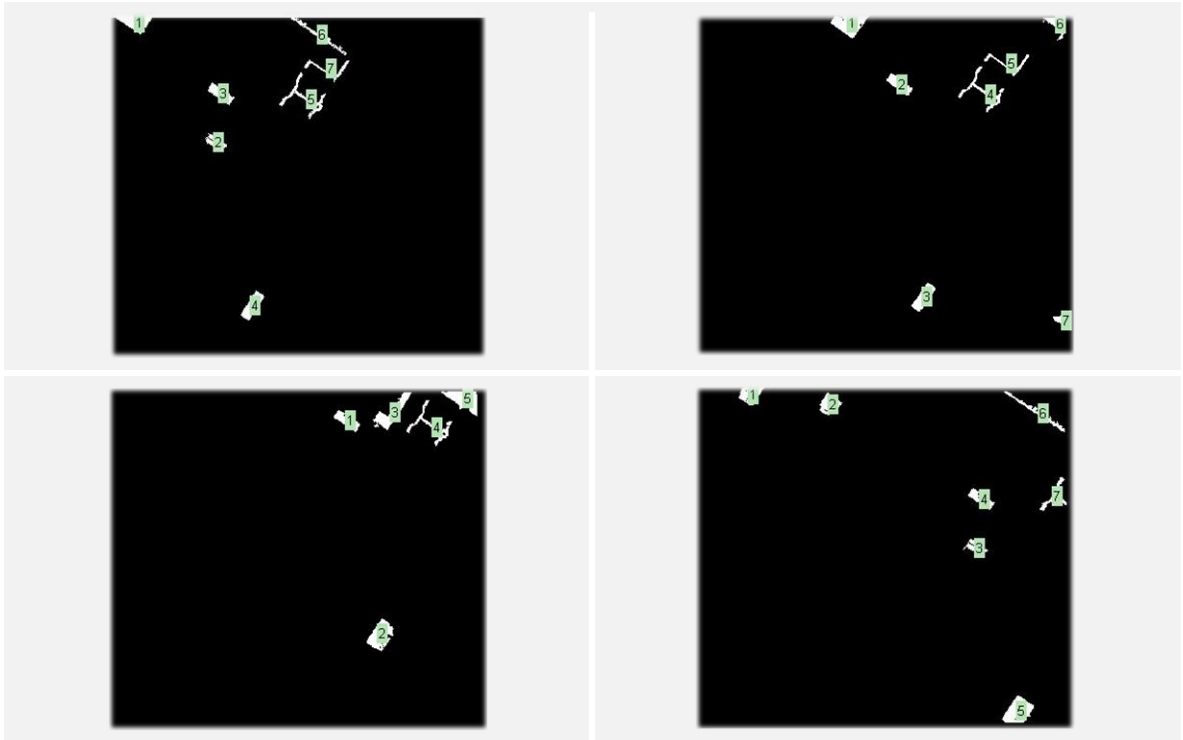


Figure 18 - One level of the MS²T

Four selected frames from a video sequence with all objects marked with a number are illustrated. The objects are extracted from one scale of the morphological scale-space. Note that the numbers marking each object do not track corresponding objects from frame to frame, but rather enumerate all the objects in each individual frame.

scale-space of each frame in a sequence. Employing connected-component filtering presents us with a foundation to apply tracking techniques exclusively to the structures of interest, and not the entire image. An example of such structures is presented in Figure 18. This figure illustrates four selected frames with components marked with numbers. Note that the numbers are merely used to illustrate each distinct object and not to match objects from one frame to the next. Figure 19 illustrates a target which was extracted and cropped from nine consecutive frames, illustrating the first scale (finest scale) in the scale-space.

It is important to note that traditional connected component implementation is slow and increases program execution time significantly. Towards overcoming these stumbling blocks, we have employed the region-based representation of the image called Max-Tree. The nodes of the Max-Tree describe the binary connected components which are generated by thresholding the

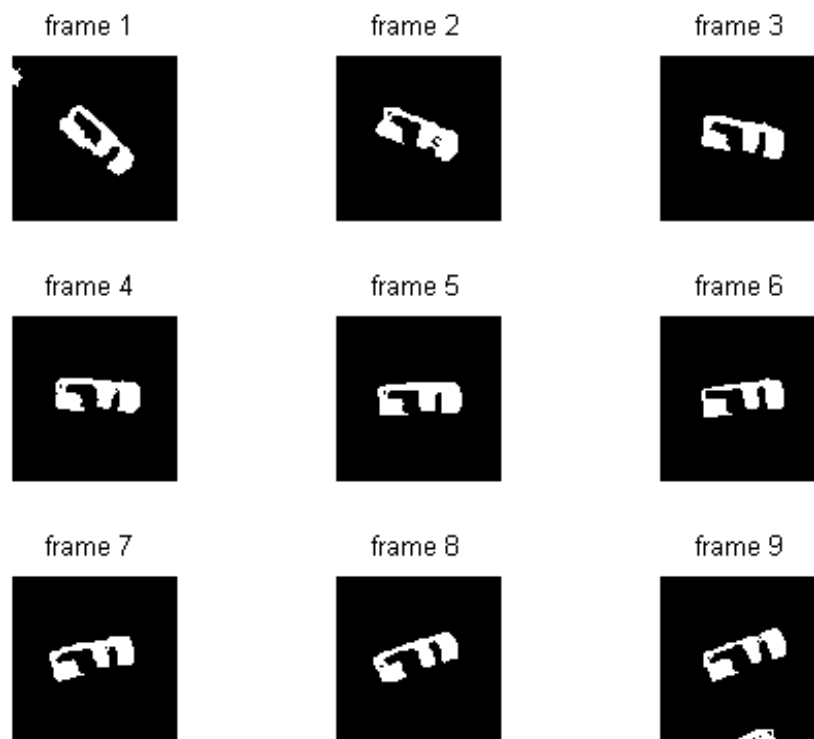


Figure 19 – Example of a target extracted from nine consecutive frames.

The target is illustrated in only one scale from the scale-space as it moves from frame to frame. It can be observed that while the target undergoes rotation change, the change in shape is minor.

original gray-scale image at all possible gray-level values. Since the tree is based on the gray-scale values, the leaves of the tree refer to the image maxima, while the root of the tree refers to the lowest gray scale level. Furthermore, the flat zones may be merged by following the links between the nodes. These relationships can be used to combine nodes based on a given criteria, such as flat-zone size [43].

The Max-Tree is only generated once per frame and then the processing is done on the Max-Tree in our MS²T algorithm. A simple example of a Max-Tree is demonstrated in Figure 20. Figure 20 shows an image with only three gray-scale values and the associated tree formation as the tree is built. Once we have established the structures in each frame using the Max-Tree algorithm, we can describe each target with unique attributes or features. The following section describes these features.

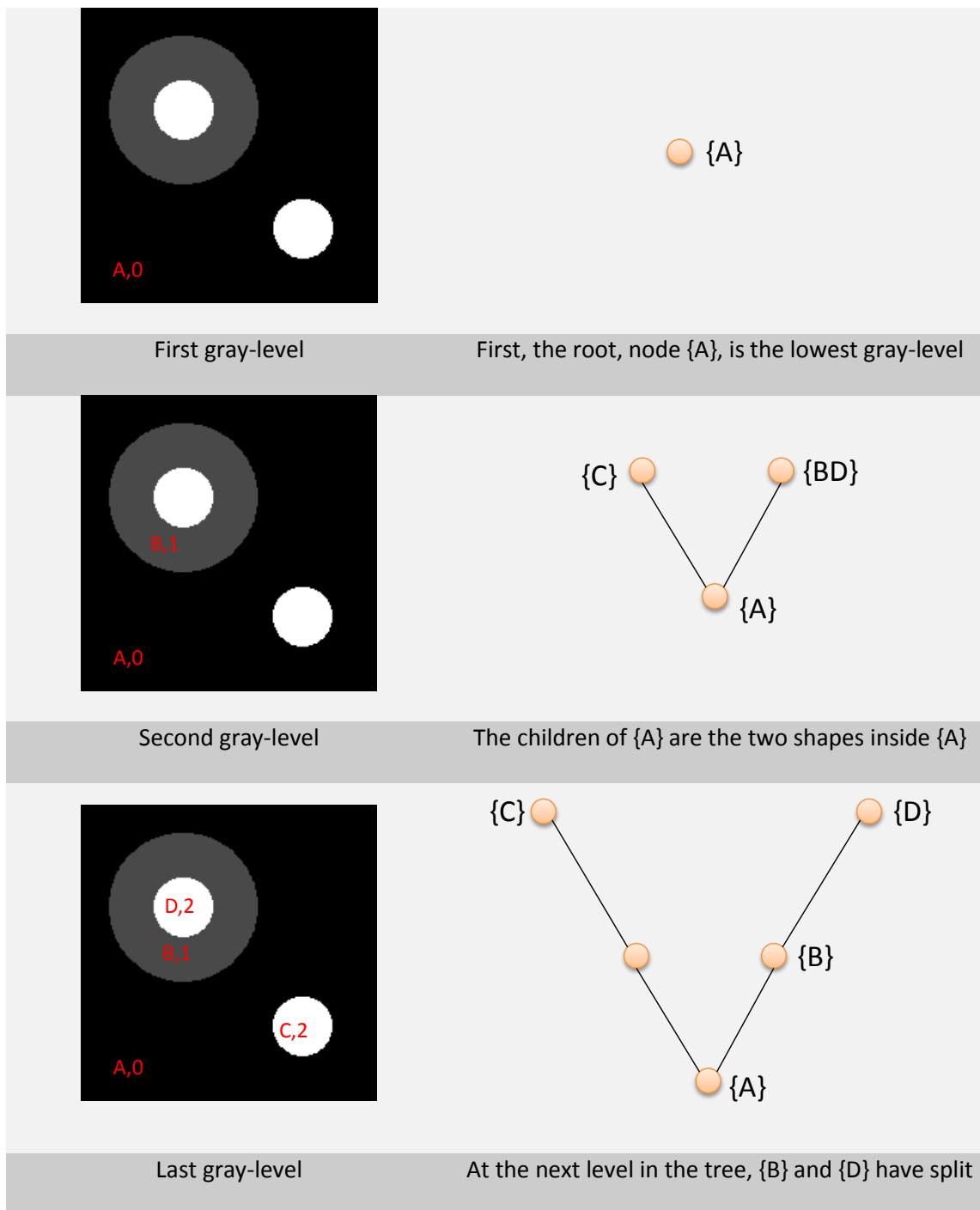


Figure 20 – Example of a three gray-level image with associated Max-Tree.

The development of the Max-Tree algorithm is illustrated. The algorithm starts with lowest gray-level value of 0 and region A, thus the root of the tree; it then processes the rest of the gray-levels growing the tree by adding each consecutive gray scale until all the gray levels have been added to the tree resulting in the tree illustrated in the lower right cell.

4.3.5 Morphological Scale-Space Connected Components Features

It is vital to describe each three-dimensional structure in the scale-space with unique features to allow for feature matching in MS²T. Each target has associated features: volume, number of scales where the object is none-zero, orientation, 3D- shape, eccentricity, gray-value histogram, and a neighborhood histogram. The volume is given by

$$O_v^\kappa = \sum_{i=1}^N \sum_{j=1}^M \sum_{l=1}^\Lambda B_{i,j,l}^k. \quad (31)$$

Here O denotes the parameter for each structure in the image, where the subscript (e.g. v for volume) specifies the parameter type and κ is the structure index. B indicates the 3D structure and N and M are the number of pixels in width and length, while Λ is the height or the number of scales in the MS²T's scale-space. We determine the orientation as

$$O_{or}^\kappa = \frac{1}{\Lambda} \sum_{l=1}^\Lambda \theta_l^k. \quad (32)$$

θ_l is the orientation of the structure from the horizontal axis to the center line of the target as shown in Figure 21(a) for each scale. In order to determine the eccentricity of each object, first an ellipse is fit to the filled object. A filled object has the same outline as the original object, but any holes or concavities are filled. Then, the major axis, \aleph , and minor axis, \beth , are estimated, as noted in the example in Figure 21(b). Consequently, the eccentricity of the object is defined as:

$$O_{ec}^\kappa = \frac{1}{L} \sum_{l=1}^L \sqrt{1 - \frac{(\beth_l^k)^2}{(\aleph_l^k)^2}}. \quad (33)$$

One can notice that if the structure closely resembles a circle the eccentricity will be approximately zero, while if it resembles a flat line, eccentricity will approach one. We can utilize this information to preserve or remove shapes of interest.

The next parameter is the intensity histogram, O_{ih}^k , of the target. In order to calculate the intensity histogram, we employ the top scale of the scale-space. Using the top scale allows us to isolate the pixels of the target from its surrounding. The histogram is then generated with 52 bins for 0-255 gray-scale images. We then normalize the histogram to allow for matching as described in the next section. The last feature we utilize in our method is the neighborhood histogram.

The neighborhood histogram, O_{nh}^k , is designed to compensate for lack of registration in the image. The histogram is based on the large background structures in the image. The neighborhood histogram is a unique way of describing the position of the target with respect to large structures in the image. Furthermore, it eliminates potential matches of similar targets that are located in different points in the image relative to the large stationary structures.

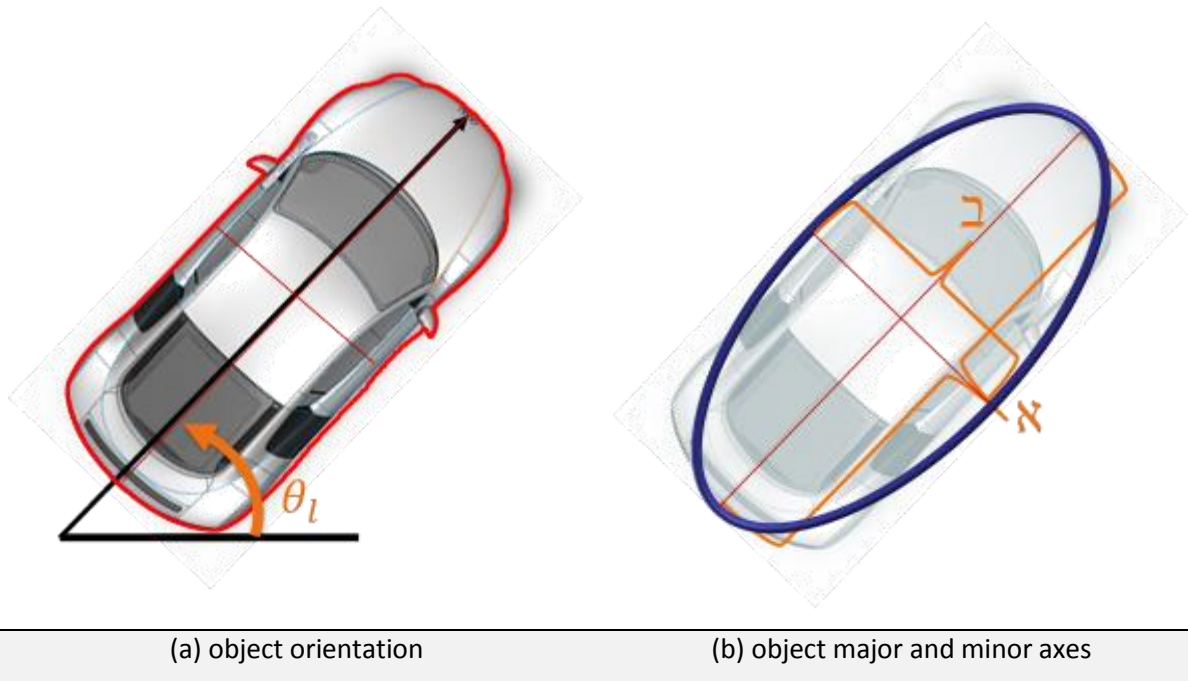


Figure 21 –Illustration to guide feature description

Where (a) illustrates how object orientation is determined, from the x-axis to the major axis of the object marked by θ_l , and (b) notes how we establish the major and minor axes marked by \aleph and \beth , respectively.

The same process of identifying the connected components of targets is used to identify large structures in the image. Those structures are assumed to correspond to stationary structures, as their size is ten or more times that of the largest targets (semi-trailer truck for example). This allows us to determine the neighborhood of the target in each frame. The

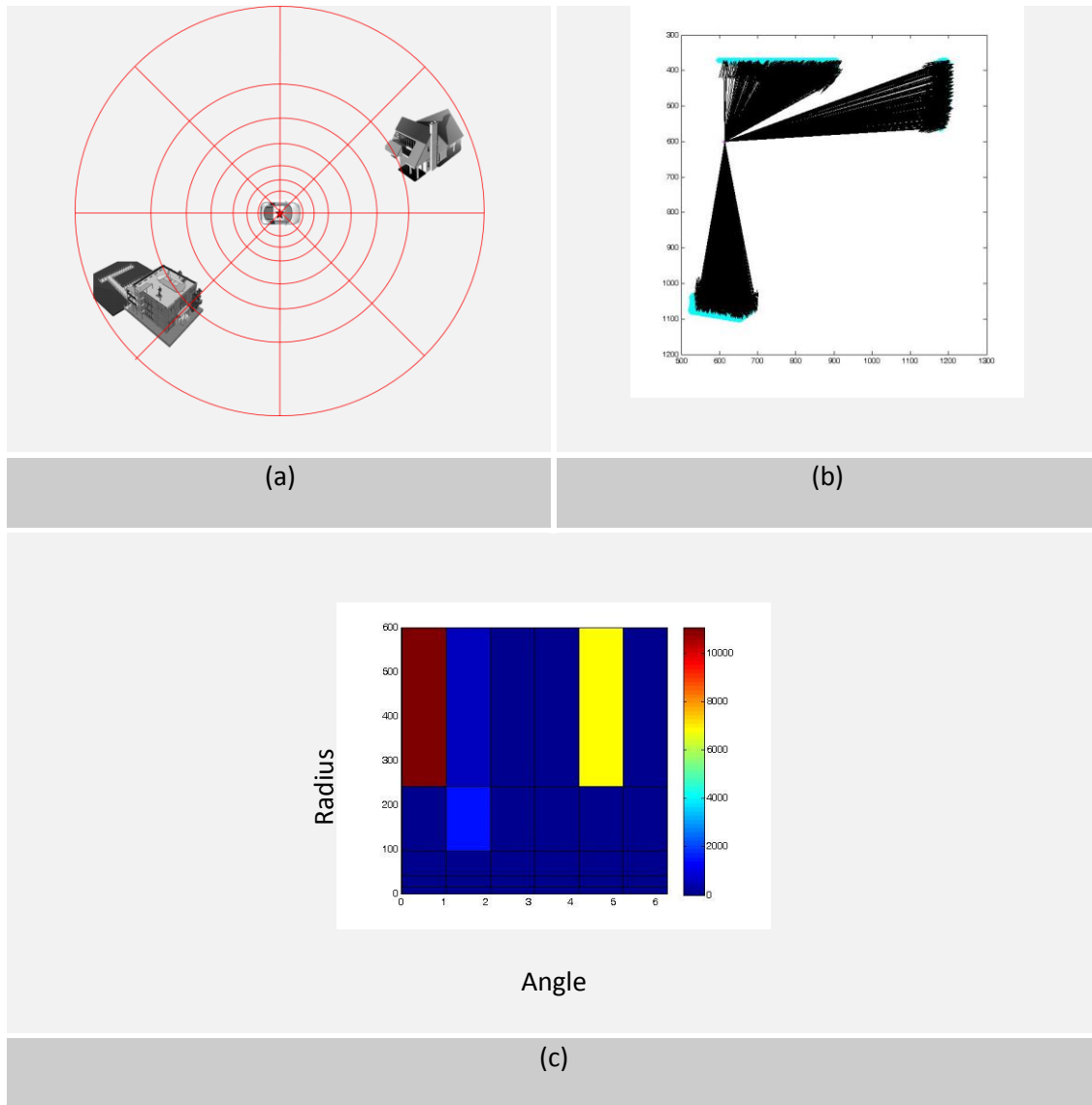


Figure 22 - Neighborhood histogram illustration

Here, (a) illustrates the target (car) of interest in the center and the bin distribution is marked by red lines with log-polar distance from the target, (b) illustrates three structures around a given target from a real image and radii to each pixel from the target in the center and (c) is the resulting 2D histogram, where the x-axis is the angle and y-axis is the radius from the center of the target to each pixel in the structures, and the cooler colors indicate lower bin values and hotter colors indicate higher bin values.

neighborhood histogram is then generated based on the method proposed by Belongie and Malik to match shapes [44]. While the authors used the histogram to describe and then match shapes, we use it to describe the neighborhood. Figure 22 (a) illustrates the way the histogram bins are defined – based on the log-polar distance from the target. The angles around the target are equally divided into six bins while the bins for the distance or radius from the target grow logarithmically. The values in each bin are assigned based on the number of pixels that each structure contributes to that bin. Figure 22 (b) shows an example of three large structures (buildings) and the radii to each pixel in the structure. These radii are then used to generate the two-dimensional histogram as demonstrated in Figure 22 (c), where dark blue (cold colors) demonstrates bins with few points and red (hot colors) demonstrates bins with a large number of points.

Now that we have defined unique features for each object, we can use these features to not only track the objects and/or targets, but also to detect targets in each frame. This process is described in the following section.

4.3.6 Target Detection Using the Morphological Scale-Space

Target detection is a vital step in target tracking. Automatic target detection allows for removing human interaction by identifying new targets entering the FOV. Common detection methods are based on stationary cameras or registered video sequences. An unchanging background permits the algorithm to determine what in the frame is approximately constant and what is changing. Thus by establishing the background, moving targets are detected, which is also known as temporal differencing [45, 46]. A related method was used in our methodology to achieve detection in Aim 1 of this dissertation. Due to lack of registration in our sequences, we do not have the background image. Thus we cannot detect targets by background subtraction, and

therefore must establish a new method for target detection. We utilize the connected component scale-space to do just that.

Since we know that our targets are vehicles, it allows us to identify shapes in the scale-space that are elliptical in nature. Furthermore, we can eliminate shapes that are approximately linear, which most often are lines from road markings. Thus, we utilize the eccentricity measurement, O_{ec}^k , and constrain it to high values to indicate more elliptical shapes. Also, knowing the approximate spatial resolution for each video sequence, we can retain the structures that are approximately the target size. This is achieved using our volume measurement, O_v^k .

Our detection method is applied to all frames in the multi-camera sequences from Yuma, AZ and our results are presented in Table 2. The table reports the normalized RMSE values for targets of interest as compared to ground truth. It is important to note that the RMSE results do not capture false detections. Hence, we note the number of missed detections separately, while not penalizing the RMSE for missed detections. This is to avoid arbitrarily setting values for missed detection locations in calculating the RMSE as noted in the discussion of Table 1. Overall, the normalized RMSE on average was 0.305 of target width with two frames total of missed targets among the 34 sequences.

Table 2 – MS²T Detection Results

Target index	Number of frames	Detection normalized RMSE	# of frames Missed	Target index	Number of frames	Detection normalized RMSE	# of frames missed
1	13	0.069	1	18	7	0.297	0
2	12	0.239	0	19	6	0.125	0
3	11	0.195	0	20	9	0.165	0
4	11	0.194	0	21	12	0.138	0
5	8	0.157	0	22	12	0.141	0
6	7	0.132	0	23	8	0.790	0
7	10	0.100	0	24	11	0.233	0
8	12	0.143	0	25	10	0.268	1
9	12	0.163	0	26	10	0.374	0
10	7	0.142	0	27	10	0.189	0
11	6	0.311	0	28	13	0.130	0
12	10	0.265	0	29	8	0.345	0
13	18	0.257	0	30	7	0.328	0
14	18	0.578	0	31	9	0.218	0
15	9	1.512	0	32	6	0.171	0
16	9	0.733	0	33	9	0.394	0
17	9	0.207	0	34	8	0.668	0

In order to address the missed detections, we provide an example of a sequence where probability of detection (PD) or true positive rate and false alarm rate (FAR) or false positive rate are recorded. PD is defined as the number of correctly detected targets in a frame over the total number of actual targets:

$$PD = \frac{TP}{TP + FN} \quad (34)$$

In (34), TP , true positive, is defined as the number of targets that were correctly detected by our method or a given one and FN , false negative, is defined as the number of targets that were not detected by our method or a given one. FAR is the number of wrongly detected targets in a frame.

FAR is defined as:

$$FAR = \frac{FP}{FP + TN}. \quad (35)$$

In (35), FP , false positive, is defined as the number of detections that were wrongly assigned; in other words, when our method detected a target that was not present. TN , true negative, is defined as the target candidate that was correctly assigned as a non-target. In our case, since there are many observations in the frame, if an observation is not a true target and was not assigned as a target, we consider it a TN . We compute PD and FAR with respect to ground-truth obtained via manual detection.

Figure 23 and Figure 24 demonstrate the receiver operating characteristics (ROC) for the FAR and PD for one, 13-frame sequence. Both figures provide ROC analysis where we investigate varying threshold values for both the minimum and maximum volumes normalized by the target size (target volume in scale space) and eccentricity as defined by our detection criteria. Both

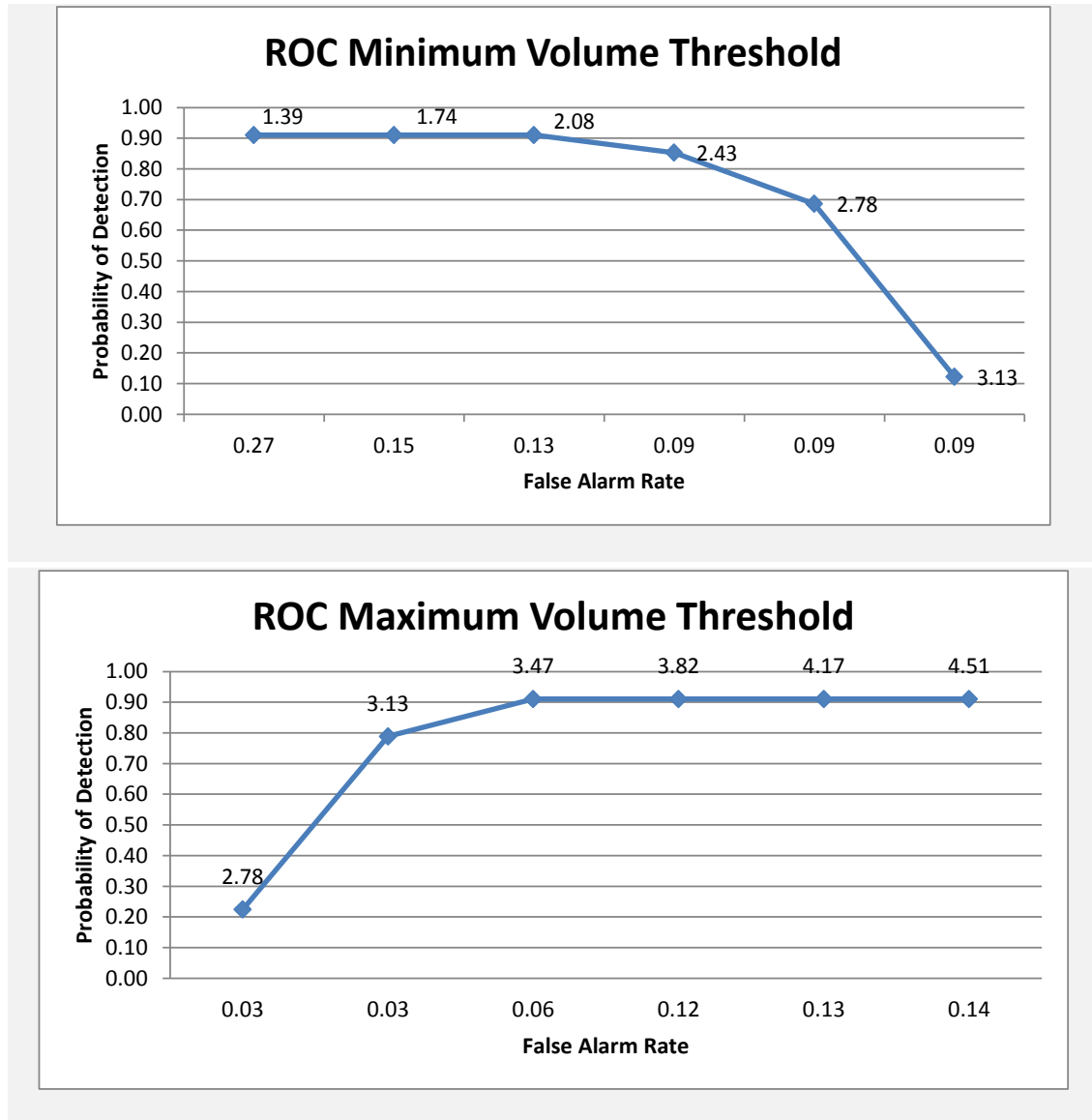


Figure 23 - Volume ROC Analysis

Here the threshold values for the minimum and maximum volume are varied. The labels on each point indicate the volume threshold value used, normalized by target size in scale-space. It can be seen that the minimum of approximately two times the target size and maximum of approximately three and a half times the target size defines a point where PD stays high while reducing FAR.

figures show results obtained by varying one threshold while keeping the other three thresholds constant.

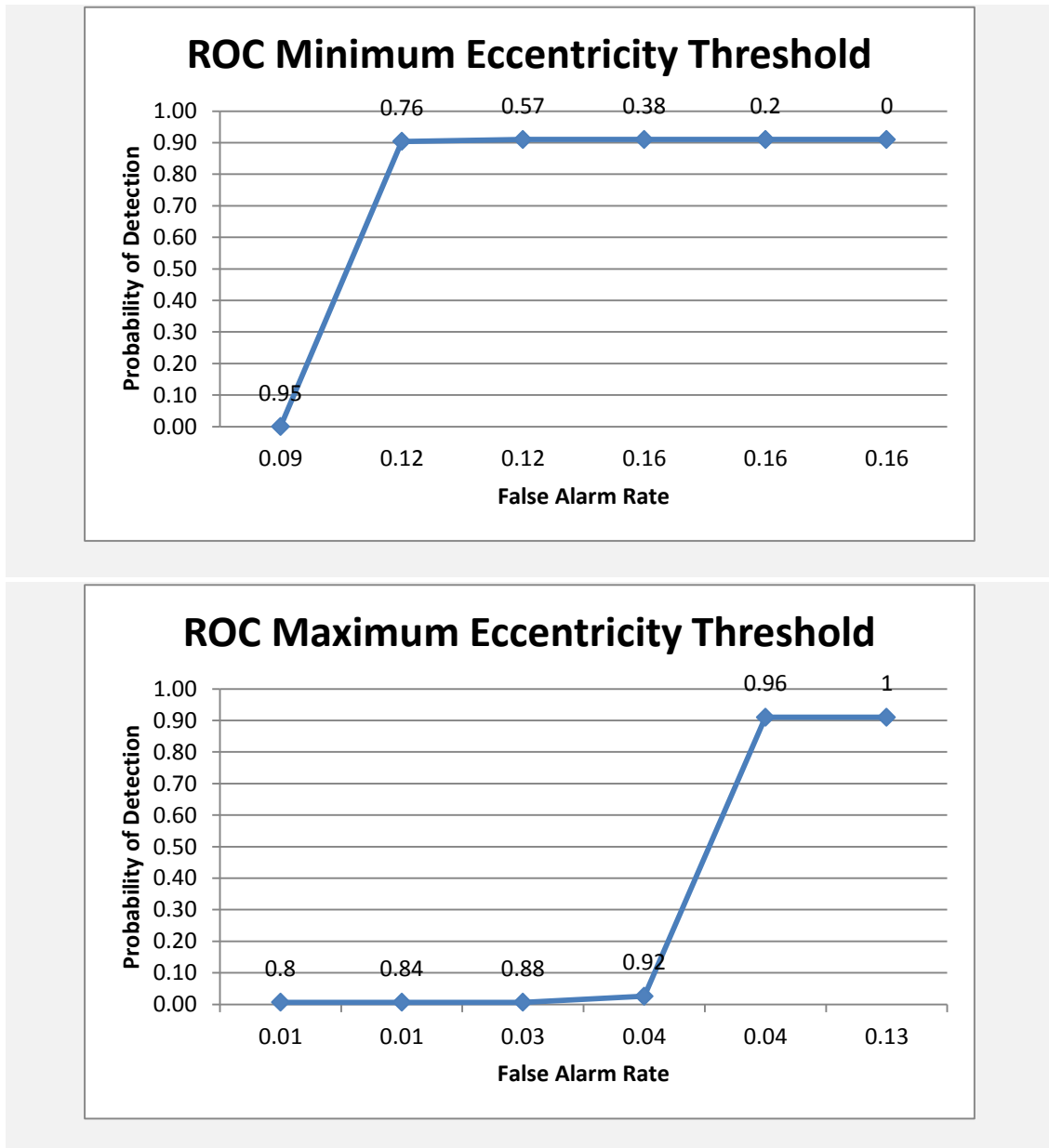


Figure 24 – Eccentricity ROC Analysis

Here the threshold values for the minimum and maximum eccentricity are varied. The labels on each point indicate the threshold value used. It can be observed that there is a clear cut off at the minimum between 0.7 and 0.75 eccentricity measures and maximum of 0.92 and 0.96 eccentricity measures.

From Figure 23 we observe that the minimum of approximately two times the target size and maximum of approximately three and a half times the target size defines a point where PD stays high while reducing FAR. In Figure 24 we observe that there is a clear cut off at the minimum between 0.7 and 0.75 eccentricity measures and maximum of 0.92 and 0.96 eccentricity measures.

Now that we have established the methodology for detection of our targets, we proceed by explaining our tracking approach.

4.3.7 Target Tracking via Feature Matching in MS²T

In order to use the above parameters for tracking in the MS²T, for each frame the structure of interest is matched to all structures in the subsequent frames based on the above criteria. We determine the minimum absolute difference for the volume, orientation, and number of levels where the structure is not zero. Furthermore, we compare the three-dimensional shape of the target of interest, B_t , with the structures in a given frame, B_j , with the following comparison measurement, CM :

$$CM = \frac{(B_t - B_t \cap B_j) \cup (B_j - B_t \cap B_j)}{B_t \cup B_j}. \quad (36)$$

All shapes are rotated based on the orientation, O_{or}^k , such that they are compared at the same orientation. Figure 25 provides a visualization of the comparison algorithm in one level of the scale-space. Here one can observe that the target of interest, B_t , is marked in green, and structures in a given frame, B_j , are marked in red. It is important to note that we specifically chose to illustrate the matched target in the given frame and not any other structure. Here, the overlapping pixels determined by performing a pixel-wise AND operation are illustrated in yellow. Furthermore, all structures/targets are centered in case of size variation.

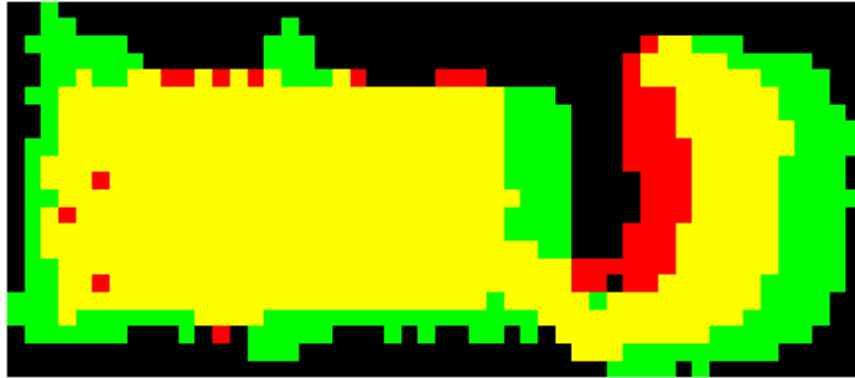


Figure 25 – Example of shape comparison for original target with a tracked target.

This figure illustrates the shape matching process. The initialized target and the current target of interest are aligned and the intersection is observed. In this figure the initialized target is marked in green, the current frame target is marked in red, and yellow is the overlap. Only one scale is illustrated here, but this is done for all scales/levels.

Other parameters used for matching are the target gray-level histogram and neighborhood histogram for each target, discussed in detail in section 4.3.5. We match the histograms by determining the minimum sum of the Euclidian distances between each bin of the target, Ξ_t^n , and the frame structures, Ξ_j^n . Histogram match, HM , is defined as:

$$HM = \sqrt{\sum_{n=1}^{NB} (\Xi_t^n - \Xi_j^n)^2}, \quad (37)$$

where NB indicates the number of bins.

In order to combine all these measurements into one, we order each measurement individually and assign it a score with the highest match having the highest score. If the highest match falls below a set threshold, no score is assigned. Each descriptor contributes to the score, and the overall decision is made based on the sum of the scores from all descriptors. We assign a match to the structure of interest based on the highest score. If the overall score falls below a set threshold, no match is made and we assume that the target left the FOV. A counter is used to allow the target to move out of FOV for two frames before the track is terminated. To validate our

method, we have investigated data where the target leaves the FOV for one frame and then returns.

4.3.8 MS²T Tracking Results

We report the results for the MS²T algorithm described in Table 3 and compare it to Table 1. The normalized RMSE is reported in pixel distance from ground truth and normalized to target width. It can be observed that our method tracks 96% of the frames versus 90% of the frames with the ASIFT². Furthermore, for the successfully tracked frames, MS²T has a tracked normalized RMSE of 0.27 pixels, compared to 0.45 with ASIFT². It is important to note that there are two types of normalized RMSE values reported. Once again here, as in the case of the ASIFT², we capture the normalized RMSE in the last column to remove the penalty of missed tracking since if a target is not detected the value cannot be assigned to its location, and normalized RMSE calculation becomes arbitrary (for more discussion see sections 4.1.2 and 4.2.3).

Table 3 - MS²T Tracking Results

Target index	Number of frames	Normalized RMSE	% Frames tracked	Normalized tracked RMSE
1	13	2.53	92%	0.06
2	12	0.20	100%	0.20
3	11	0.29	90%	0.15
4	11	4.61	90%	0.61
5	8	0.19	100%	0.19
6	7	0.25	100%	0.25
7	10	0.08	100%	0.08
8	12	0.09	100%	0.09
9	12	0.15	100%	0.15
10	7	0.14	100%	0.14
11	6	0.32	100%	0.32
12	10	0.36	100%	0.36
13	18	3.20	94%	0.10
14	18	2.57	94%	0.63
15	9	1.02	100%	1.02
16	9	0.54	100%	0.54

17	9	1.89	88%	0.16
18	7	0.25	100%	0.25
19	6	3.64	80%	0.10
20	9	0.14	100%	0.14
21	12	0.10	100%	0.10
22	12	0.14	100%	0.14
23	8	0.67	100%	0.67
24	11	14.76	70%	0.23
25	10	6.64	80%	0.18
26	10	0.26	100%	0.26
27	10	0.15	100%	0.15
28	13	0.11	100%	0.11
29	8	1.82	86%	0.24
30	7	0.35	100%	0.35
31	9	0.30	100%	0.30
32	6	0.19	100%	0.19
33	9	0.33	100%	0.33
34	8	0.55	100%	0.55
Average		1.44	96%	0.27

To calculate these results, we utilized a Windows 7 64-bit home edition PC, with Intel® Pentium® P6100 dual core processor, with 3GB RAM. MS²T was applied to frames of 2048x2048 pixels in size. The algorithm takes, on average, 1.2 seconds to generate the scale-space per frame. The rest of the algorithm—calculating object features and matching to targets of interest—takes, on average, 10 seconds per frame. Note that a parallel implementation of the feature calculation and matching will have a significant decrease in execution time.

MS²T performs well in video where there is contrast between the target and the background. The algorithm can be applied to any target motion, whether the target moves throughout the FOV or is stationary. As we employ a neighborhood histogram as one of the features, the algorithm performs better if there is high frame rate relative to the movement of the target. In low frame rate, the neighborhood histogram can be removed from the overall score. Next we present a graphical comparison of the results for the ASIFT² and the MS²T.

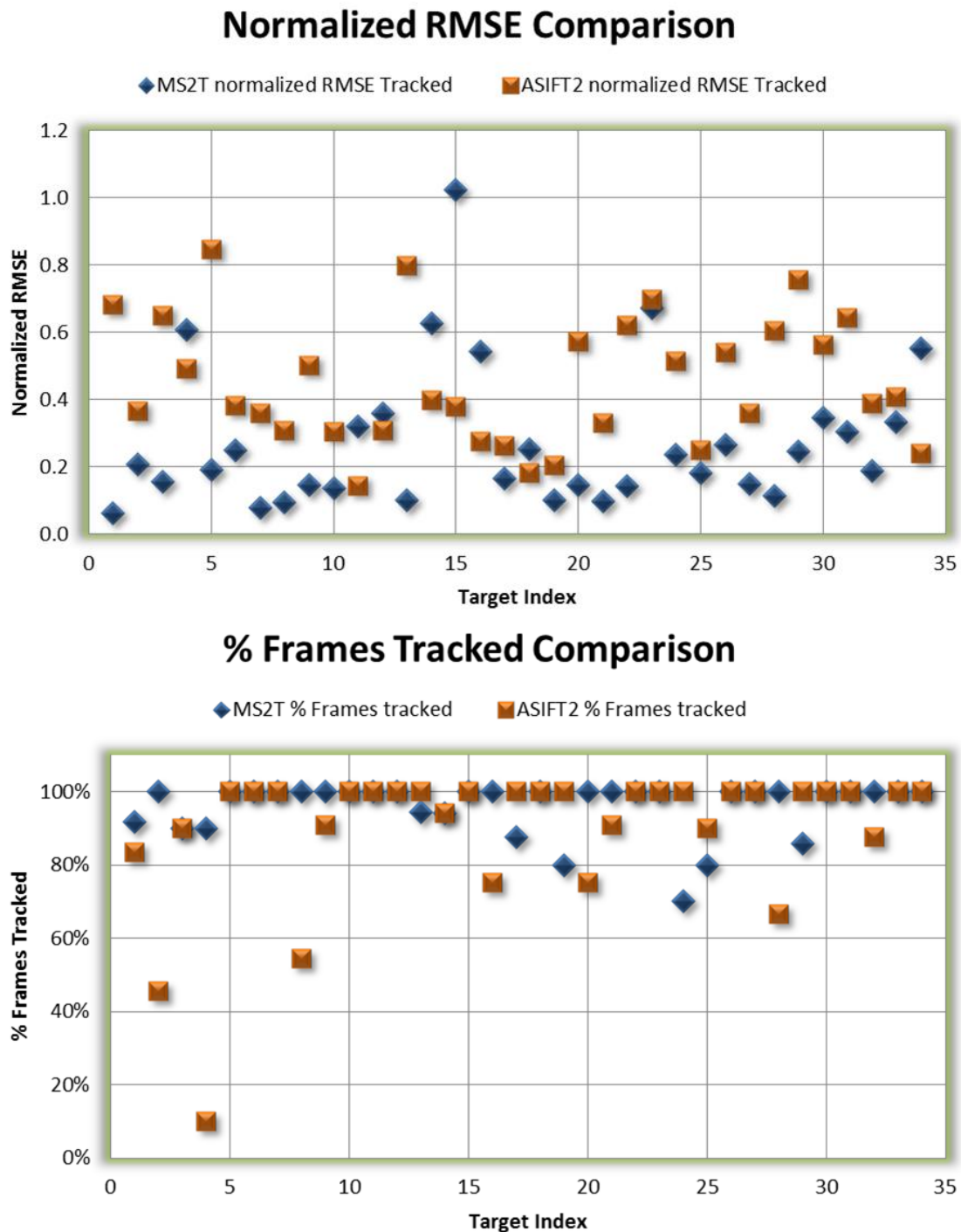


Figure 26 – Tracking results comparison between MS²T and ASIFT²

Here we show the tracking results from the Yuma, AZ data for both the normalized RMSE and percentage of tracked frames. The results show that overall the MS²T algorithm performs better than ASIFT² in terms of successful tracking through consecutive frames and locating the center of the target in tracked frames.

Our results demonstrate that the MS²T outperformed the ASIFT² both in the percentage of tracked frames and the tracking accuracy. We find that since the placement of the key-points in the SIFT methodology is not associated with any defined geometrical relationship to the target, it is difficult to detect the center of the target with ASIFT². This behavior can be observed in the results, where in the tracked frames ASIFT² misses the center of the target by approximately half a target on average. On the other hand, when the contrast is not significant from the background, ASIFT² demonstrates a higher match rate.

4.3.9 Investigating Descriptors in MS²T

As MS²T incorporates several parameters to achieve tracking, we aim to investigate the significance of the contribution of each descriptor in the success of tracking. We hypothesize that if we can correlate one descriptor with high scores to successfully tracked frames we can utilize it in determining which frames are successfully tracked based on that one descriptor.

We employed Spearman's rank correlation coefficient to determine the parameter(s). The parameters are related to the normalized RMSE value, where lower normalized RMSE indicates best match. In our case some of the relationships were monotonically increasing (correlation of one) and others were monotonically decreasing (correlation of negative one). For example, the absolute difference between volume and orientation is monotonically increasing because lower difference is scored higher. On the other hand, shape matching will have a higher comparison measure with matching shapes and a lower one with two targets that are unmatched, thus having a monotonically decreasing relationship.

To simplify our analysis, we seek to keep all correlations positive. Thus, any calculations that result in a negative correlation are inverted (multiplied by -1) and then the correlations are calculated. Only successfully tracked frames are used towards this analysis. We remove any

incorrectly tracked frames as they can skew the correlation to incorrectly favor one parameter or another.

We calculate the Spearman's rank correlation coefficient, ρ , as follows:

$$\rho = \frac{\sum_i (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum_i (u_i - \bar{u})^2 \sum_i (v_i - \bar{v})^2}}, \quad (38)$$

where (u, v) are the ranks for the normalized RMSE and parameter values we are correlating, and i is the index of all the values for each parameter. The calculated correlations for each parameter are given in Table 4.

Table 4 -Spearman's correlation results for parameter analysis, correlating each parameter used in the MS²T tracker with the resulting normalized RMSE.

Parameters	Correlations
Volume:	0.354
Orientation:	0.151
Eccentricity:	0.406
Context histogram:	0.126
Shape Match:	0.429
Gray Level Histogram:	0.366
Volume/Perimeter ² :	0.242

It can be observed from the tabulated correlations in Table 4 that while eccentricity and shape match show the highest correlation, none of the parameters demonstrate a strong correlation with the normalized RMSE. Since our aim was to determine if any of these parameters can determine the success of the tracker, we conclude that due to the low correlation, no one parameter or descriptor can clearly indicate that the tracker correctly tracked a target. As a result, we cannot use any one of the parameters in isolation to determine the success of the tracker. All parameters combined provide tracking success.

4.3.10 Statistical Significance Analysis

We further analyzed the results of our MS²T method and ASIFT² for statistical significance by utilizing the one-way analysis of variance (ANOVA) by comparing both the percentage of tracked frames and the normalized RMSE. The null hypothesis, H_0 , is

$$H_0: \mu_c = \mu_s, \quad (39)$$

where μ_c is the mean of MS²T, while μ_s is the mean of ASIFT² for both the percentage of tracked frames and the normalized RMSE. Then the alternative hypothesis, H_A , states that the means are not equal

$$H_A: \mu_c \neq \mu_s. \quad (40)$$

Here we assume a significance level of 0.05. Results are tabulated in Table 5. The table presents the degrees of freedom, df , sum of square deviations from the mean, SS , mean squares, MS , F -value from the F -distribution, F , F -ration which determines the critical value, F_{crit} , and P -value.

Table 5 - ANOVA Results to Compare ASIFT² and MS²T
One-way ANOVA for Percentage of Tracked Frames

SUMMARY

Groups	Count	Sum	Mean, μ	Variance, σ^2
% Detection ASIFT ²	34	30.54	90%	3.88%
% Detection MS ² T	34	32.63	96%	0.56%

ANOVA

Source of Variation	SS	df	MS	F	P -value	F_{crit}
Between Groups	0.065	1	0.065	2.922	0.092	3.986
Within Groups	1.465	66	0.022			
Total	1.815	67				

One-way ANOVA for Normalized RMSE

SUMMARY

Groups	Count	Sum	Mean, μ	Variance, σ^2
--------	-------	-----	-------------	----------------------

RMSE Detected ASIFT ²	34	288.14	8.47	17.79
RMSE Detected MS ² T	34	169.52	4.99	15.70

ANOVA

Source of Variation	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	206.91	1	206.91	12.357	0.0008	3.986
Within Groups	1105.1	66	16.74			
Total	1418.19	67				

As the *F*-value compared to *F*-critical and *P*-value demonstrate, there is statistical significance for our normalized RMSE results. Conversely, the percentage of tracked frames does not show statistical significance for the chosen significance level, but we utilize this measure as an alternative metric for our tracker. In the case of percentage of tracked frames, the *F*-value is 2.92 while the *F*-value for RMSE is 12.36. When compared to the *F*-critical of 3.986, it is clear that the RMSE measurements reject the null hypothesis, while the percentage of tracked frames measurements accepts the null hypothesis that the means are the same. The same conclusion can be achieved from the *P*-value when compared to our assumed significance level of 0.05. It should be noted that if the significance level were chosen to be 0.1, both measurements would reject the null hypothesis.

We can thus conclude that the two methods, ASIFT² and MS²T, perform equally well when it comes to tracking the target, but MS²T outperforms ASIFT² in locating the correct center of the target with statistical significance. The poor performance of ASIFT² in terms of RMSE is due to the fact that SIFT is not concerned with the objects themselves, but rather with specific regions in the image.

From our results we also noted that there were cases where one tracker (ASIFT² or MS²T) did better than the other; in other words, the trackers did not fail in the same sequences. To further investigate this behavior, we combined the two trackers to determine if further

improvement can be made to our tracking results. The next section discusses our method for doing so and the results.

4.4 Incorporating SIFT into MS²T

We tackle the SIFT shortcomings noted earlier – the inability to use SIFT for detection, ambiguity in target center, and key-points association with the target – by incorporating SIFT key-points as one of the descriptors in our MS²T algorithm. As a result, we are able to benefit from SIFT’s performance in low contrast targets while successfully employing the algorithm for detection, target association, and target center understanding.

We use the MS²T mask of the connected component from the first scale of area open and detect the key-points inside only the MS²T mask, thus removing any possible background key-points. We incorporate the number of matched SIFT points to our tracker as another parameter. Lastly, as with MS²T, each descriptor contributes to an overall score including the SIFT match. When incorporating SIFT, we utilize the number of key-points that match to the target as well as the matching quality provided by SIFT. The key-points and their matching quality allows us to determine the score for the SIFT descriptor as it contributes to the overall tracker score. Consequently, we found that our percentage of tracked frames has improved by two percent, to 98%, with a normalized RMSE value of 0.28. The slight increase in the normalized RMSE is due to more frames being incorporated into the RMSE calculation.

Table 6 - Tracking results for MS²T with SIFT

Target index	% Frames tracked	RMSE normalized MS ² T+SIFT
1	92%	0.06
2	100%	0.20
3	100%	0.19
4	90%	0.61
5	100%	0.19

6	100%	0.25
7	100%	0.08
8	100%	0.09
9	100%	0.15
10	100%	0.14
11	100%	0.32
12	100%	0.36
13	100%	0.17
14	94%	0.63
15	100%	1.02
16	100%	0.54
17	100%	0.17
18	100%	0.25
19	80%	0.10
20	100%	0.14
21	100%	0.10
22	100%	0.14
23	100%	0.67
24	100%	0.22
25	80%	0.18
26	100%	0.26
27	100%	0.15
28	100%	0.11
29	86%	0.24
30	100%	0.35
31	100%	0.30
32	100%	0.19
33	100%	0.33
34	100%	0.55
Average	98%	0.28

4.5 Conclusion

This chapter illustrated three novel ideas towards successful tracking: the automated SIFT tracker, the morphological scale-space tracker (MS²T), and the incorporation of SIFT into the morphological scale-space tracker. We demonstrated the improvements in tracking as we

developed each tracker. Furthermore, this chapter discussed how our morphological scale-space is applied towards successfully detecting the target in every frame.

Future investigation for this work can include incorporating relative key-point location to the target center in order to enhance the ASIFT². This enhancement requires more pre-processing of initialized targets, understanding where the target center is in the initialization of the tracker, and then establishing the location of SIFT key-points relative to the center. Knowing the location of the SIFT key-points will allow the SIFT tracker to track the center of targets more accurately.

Another idea that can be investigated is to use SIFT as a global tracker from one frame to the next. The result would be to acquire registration and establish the global motion of the frame. Knowing the motion of the frame will allow extracting the targets that are not moving with the general motion.

Lastly, GPU implementations of both SIFT and the connected filter should be investigated. The recent applications of GPU algorithm implementation have demonstrated a significant improvement in execution time. Since many aspects of both algorithms could run in parallel, both algorithms will have shorter execution time if implemented using GPUs.

Specific Aim 2: To accomplish tracking with a morphological scale-space that allows tracking multiple targets, in unregistered video sequences

This chapter has addressed the development of a novel algorithm which utilizes the morphological scale-space. The concept of morphological scale-space is introduced as well as its utilization in tracking. We illustrate a set of target parameters which are combined to achieve successful tracking. Furthermore, we describe how the morphological scale-space is used towards successful detection. This work has been submitted for publication [47].

CHAPTER 5.

TRACKING CHARACTERIZATION

Up to this point we have addressed the issue of tracking targets with several automated tracking algorithms. In Chapter 3 we did so for registered video sequences and in Chapter 4 for non-registered sequences. While our various tracking methodologies demonstrate successful tracking in a variety of video sequences, our next aim is to understand how image or video quality impacts tracking. Towards this end, the work presented in this chapter seeks to develop a measure that quantifies the difficulty of target tracking. As will be reviewed, no such quantitative measure exists that judges the difficulty of a given tracking experiment. Such a model needs to consider video quality, sampling rates in space and time, clutter, and the motion of the target.

Here, we assume that the target template is known. We further assume that any transformation of the template (e.g., rotation, scaling) is also known for the ground truth sequences. Finally, we assume that there is a motion model for the target and that the SNR is either known or can be computed. Under these assumptions, we attempt to define a trackability measure – a measure of the difficulty of tracking a target in a given video.

We combine several image characteristics to determine the ability of the tracker to accomplish tracking in a given sequence, defined here as trackability and quantified as bits per second. Our trackability model incorporates target matching in a given video frame as well as video quality. The combination of these parameters will return a metric that evaluates tracking success.

We commence this chapter by describing the methodology related to understanding how well a tracking algorithm will perform in a given video sequence. Next we describe our trackability method. As mentioned, currently there is no measure that quantifies trackability. Several

parameters do exist to describe the quality of the image or video and the complexity of the background in a given image or scene.

5.1 Background on Image Quality and Tracking Characterization

Image quality assessment methods seek to determine the degradation of the image after the image has undergone some change such as signal loss due to transmission or compression. Quality assessment is useful in testing video processing systems and algorithms and can be incorporated in the image analysis system to improve the algorithm's performance. Traditional image quality assessment methods compare the original, good quality image to the resulting transmitted image. If the original image is available, these methods are referred to as full-reference and are the practices most commonly used to assess the image quality. Methods that have no-reference (NR) images have been presented in literature, but less often. The NR methods must rely on information within the given image to determine image quality [48] [49].

The structural similarity (SSIM) index [49] is an image or video quality assessment method. This method is a full reference technique that was introduced to investigate the deterioration in the quality of the image or video due to JPEG compression or transmission. Assuming we have an uncompressed, high quality image, we can compare the reference target from the high quality image to a target altered in our new image. The compressed or transmitted target, which here we refer to as 'altered target', can be impacted by such things as noise and blurring. The theory of SSIM is founded on the assumption that such degradation in quality can be modeled as a combination of changes in contrast, luminance, and structure, in which structure is quantified by local changes in correlation (with the undistorted image). In [49], contrast, η , luminance, ψ , and structure, v comparison measures are defined as:

$$\eta(\mathbf{t}, \mathbf{b}) = \frac{(2\sigma_t\sigma_b)}{(\sigma_t^2 + \sigma_b^2)}, \quad (41)$$

$$\psi(\mathbf{t}, \mathbf{b}) = \frac{2\mu_t\mu_b}{\mu_t^2 + \mu_b^2} \text{ and} \quad (42)$$

$$v(\mathbf{t}, \mathbf{b}) = \frac{\sigma_{tb}}{\sigma_t\sigma_b}. \quad (43)$$

Here, \mathbf{t} and \mathbf{b} are discrete pixel indices from the target image and from the altered target image, respectively, σ_t , σ_b are standard deviations and μ_t , μ_b are means in each region. Setting all images to be the same number of pixels, P , the authors define σ_{tb} as:

$$\sigma_{tb} = \frac{1}{P-1} \sum_{i=1}^P (t_i - \mu_t)(b_i - \mu_b), \quad (44)$$

Where t_i is a pixel from the original image such that $\mathbf{t} = \{t_i | i = 1, 2, \dots, P\}$ and b_i is a pixel from the altered image such that $\mathbf{b} = \{b_i | i = 1, 2, \dots, P\}$. Putting all these measures together, the authors introduce SSIM:

$$SSIM(\mathbf{t}, \mathbf{b}) = \frac{(2\mu_t\mu_b + C_1)(2\sigma_{tb} + C_2)}{(\mu_t^2 + \mu_b^2 + C_1)(\sigma_t^2 + \sigma_b^2 + C_2)}, \quad (45)$$

where C_1 and C_2 are constants. Refer to [49] for further discussion on the constants. It is important to note that the SSIM index is no greater than one and if two images are identical, they will result in an SSIM index of one for each pixel. In the algorithm implementation a sliding 8x8 window is used to calculate the SSIM map, which is a quality measure of how well the two images match in the given window. Consequently, the resulting map is ten pixels smaller in height and width since only the center of the window is used to calculate the index for each pixel.

Figure 27 demonstrates an example of the target and possible ways that this target may become degraded in video – with additive noise or blurring. The second row illustrates the SSIM map with the corresponding mean SSIM index. The blurred image resulted in a higher mean

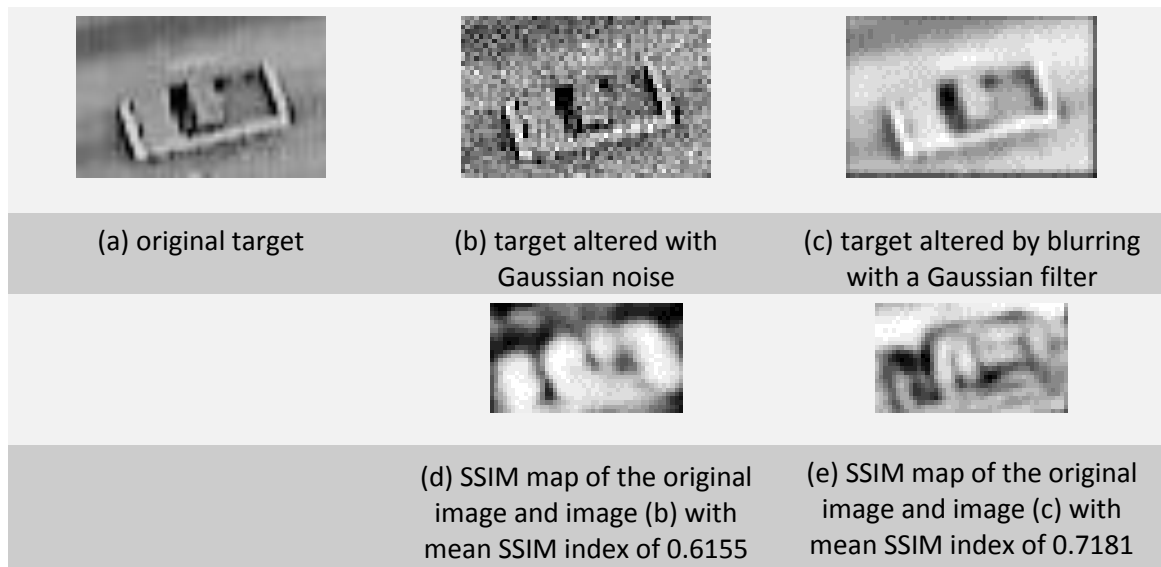


Figure 27 – Structural similarity (SSIM) index

The figure illustrates the images used to illustrate SSIM. Image (a) shows the original image. Here we assume this is the reference image, or the unaltered high quality image. Image (b) is altered by Gaussian noise and (c) is blurred by a Gaussian filter. Image (d) presents the resulting SSIM map for comparing the original image to the noisy image with the mean SSIM index of 0.6155 compared to (e) which illustrates the SSIM map for the blurred image with a mean SSIM index of 0.7181. It should be noted that the blurred image has a higher similarity than the noisy image.

quality measure of 0.72 versus that of the image with noise of 0.62. With the SSIM index, only image quality is reported and it requires a full reference image to be known, but such an image is unavailable in our tracking application. Furthermore, there exist other factors, such as temporal sampling rate and target motion, that are not considered by SSIM or other related structural quality measures.

Alternative assessment methods do address the NR image problem. Wang et al. proposed a method that analyzes blocking artifacts and blurring that result from JPEG compression [50]. Specifically of interest are the blurring artifacts that were established by determining the zero-crossing rate in the image. Another NR method introduced by Li in [51] analyzed edge strength in the image using step edges to find out how blurry the objects are in the images. A step edge defines image edges that result due to large discontinuity in pixel intensity.

Whereas image assessment investigates the quality of the image, other obstacles that the tracker must face to successfully track the targets in the image are assessed, to a certain degree, by image complexity or clutter measures. Complex or cluttered scenes might reduce the probability of detecting the targets, thus degrading tracking results. Image complexity has been addressed in literature specifically to analyze automatic target recognizers [52].

The signal-to-clutter ratio (SCR) that we will now present has been used in our laboratory to analyze the success of leukocyte trackers [14], and has been employed by researchers with the U.S. Army [53] in military applications. SCR investigates the surrounding area of the target and determines the complexity of tracking based on other objects present. A low SCR indicates a dense population of objects around the target of interest. SCR is defined as follows. Let ROI be the region of interest and T the target template. A neighborhood around the target is defined as:

$$\Omega = \{(i, j): \sqrt{(i - x_0)^2 + (j - y_0)^2} \leq 2\rho\}, \quad (46)$$

where (i, j) define the pixel coordinates, (x_0, y_0) is the estimated target location, and ρ is the target radius. Then SCR is defined as:

$$SCR = \frac{d(T, ROI_{0,0})}{\sum_{i,j \in \Omega} a_{i,j} d(T, ROI_{i,j}) / \sum_{i,j \in \Omega} a_{i,j}}, \quad (47)$$

where $ROI_{i,j}$ is an ROI centered at (i, j) , $d(T, ROI_{i,j})$ is a dissimilarity measure between the template T and $ROI_{i,j}$. A dissimilarity measure can be provided by methods such as normalized cross-correlation (NCC) or sum of square differences. In (47), $a_{i,j}$ is a weighting that depends on the distance of the clutter from the target center, e.g. Gaussian weighting. Figure 28 illustrates an example of SCR where: figure (a) is the target of interest; (b) is the clutter – including objects that are similar to our target; (c) illustrates an example of the target radius that is used in SCR

calculation; (d) presents the doubling of the target radius, which in turn is used in (e) to define the weighting function – a two-dimensional Gaussian function.

The SCR has the advantage over the signal-to-noise ratio (SNR) of including the effects of clutter; however, the SCR by itself, like the other methods presented above, fails to tell the whole story, which would include, in our view, the quality of the video and the predictability of the target motion. Furthermore, the SCR is concerned with clutter close to the target of interest, but in unregistered video the entire image's clutter is significant because the target's track is unknown and the whole image has to be considered as the domain in which the tracker can be distracted from identifying the target of interest.

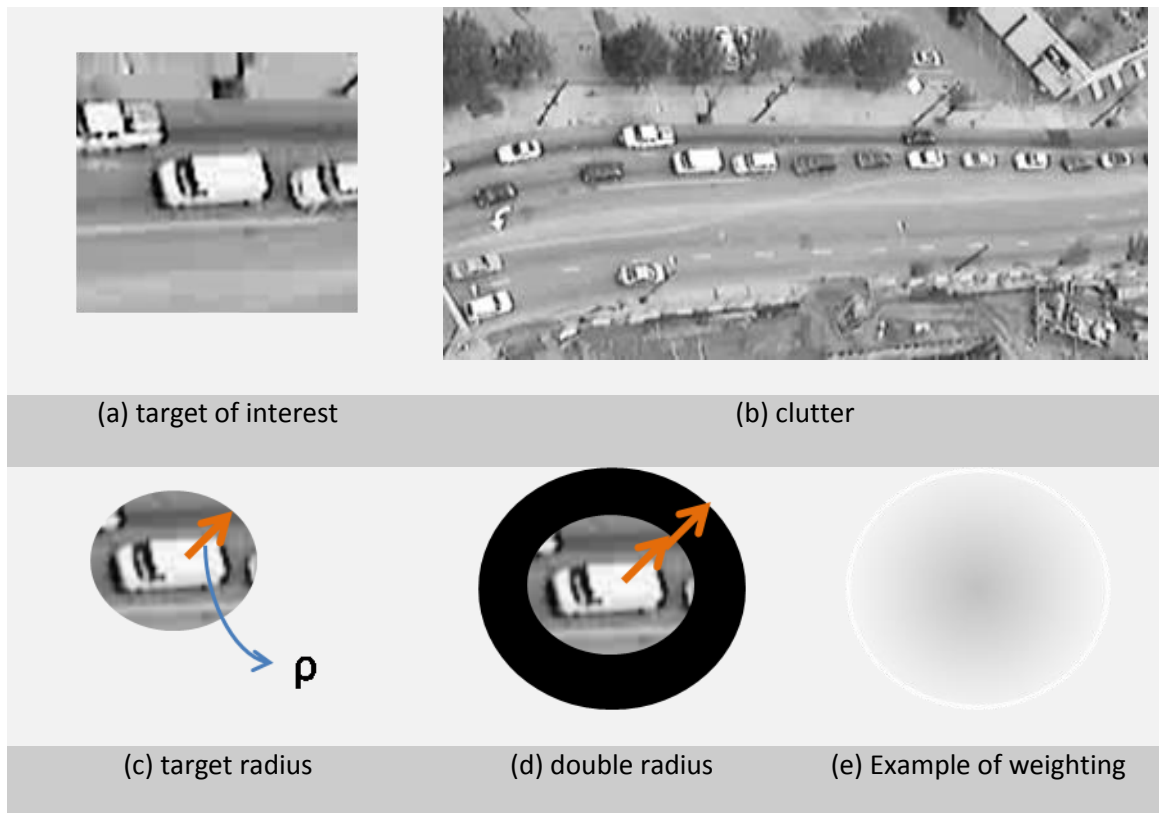


Figure 28 – Signal to clutter example

An example of how the signal to clutter ratio (SCR) is being calculated. We start with (a) the target of interest and the image we are investigating with clutter, shown in (b). Note that in (b) there are several objects that appear similar to the target in (a). As part of the SCR calculation we define the target radius, which is illustrated in (c), and twice the radius, illustrated in (d). Image (d) is the area that will be considered in calculating the SCR. Lastly, (e) is an example of the weighting function, $a_{i,j}$, that weights closer clutter higher than clutter further away.

5.2 Trackability Measure

Up to this point we presented very rudimentary components from literature which could be used to measure trackability. It is important to note that, until now, no one has investigated the notion of trackability; the measures presented are merely documented ways to investigate image quality. We, instead, take an information theoretic approach to calculating trackability. The application of mutual information to image processing is not a novel one – consider the widespread use of mutual information in registration [54]. Based on this work and other such

work that utilizes information theory in image processing, we develop a measure for tracking difficulty.

5.2.1 Trackability Theory

The trackability measure is computed in units of bits per second. In this model, there are two components: signal-to-template match and the overall video quality. Assuming we can obtain a video signal for a given target and that we have a template for that target signal in the given viewpoint and orientation, the first term, Q_{ST} , or the signal-to-template quality is computed for a given video. If this quality is found for a given frame in terms of information theoretic bits (where a bit is the amount of information associated with a binary variable in which both states are equally likely), then Q_{ST} is computed by multiplying the average of frame-wise quality by the frame rate (in frames per second) r_F . We consider two formulations for Q_{ST} and compare them to tracker performance.

The second component of our trackability measure reflects the overall video quality, Q_V , in bits per second. In considering all the contributing factors to video quality as bandwidth, we also take into account the spatial resolution, the temporal resolution, the quality of quantization, and the effect of noise. Here, we approach the effect of noise with a traditional measure, the SNR.

The second contributing factor to the video quality relates to the fidelity of motion of the target. Again, we assume that the video can be registered or unregistered. We also assume that a motion model for the target exists, even if such a model is trivial (estimating the target to be stationary, for example). So, this factor quantifies the uncertainty in the target position after registration (or the absence of registration) and prediction.

5.2.2 Quality of Signal-to-Template Match

In our trackability model, three important subimages exist. Each subimage is identical in size to an $M \times N$ grayscale image. The three subimages are the signal S , the template T , and the clutter C . The signal S is the representation of the target that actually appears in the video sequence. This signal may be partially occluded, lighter, darker, rotated, scaled, distorted, etc. The template T is the knowledge of the target appearance by the tracker. We assume that this template has been prepared according to the viewpoint and appearance of the actual signal S , so that the scale and sampling are identical. The clutter C represents the best matching subimage in the track gate that is not overlapping with the signal. We call this best matching subimage the dominant clutter. This subimage is chosen by finding a subimage that is non-overlapping with the actual target, which maximizes the mutual information with the template. There may be more than one “close match” within the gate, but we consider only one such match. The motivation for considering just one close match is that the possibility of incorrect identification of a target does not increase with multiple possible matches. The clutter subimage C is the exact same size and sampling as the signal S in the same gate.

Consider a single image in a video sequence in which the target appears as signal S . After the appropriate adjustment for viewpoint (scale, rotation, etc.), we attempt to match the signal with template T . The match between a signal S and a template T can be measured using mutual information. This is a measure of the information, $MI(S, T)$, shared between S and T , or likewise, the mutual dependence between S and T [55]:

$$\begin{aligned}
 MI(S, T) &= H(S) + H(T) - H(S, T) \\
 &= - \int p(S) \log p(S) dS - \int p(T) \log p(T) dT + \iint p(S, T) \log p(S, T) dS dT \\
 &= \iint p(S, T) \log \frac{p(S, T)}{p(S)p(T)} dS dT,
 \end{aligned} \tag{48}$$

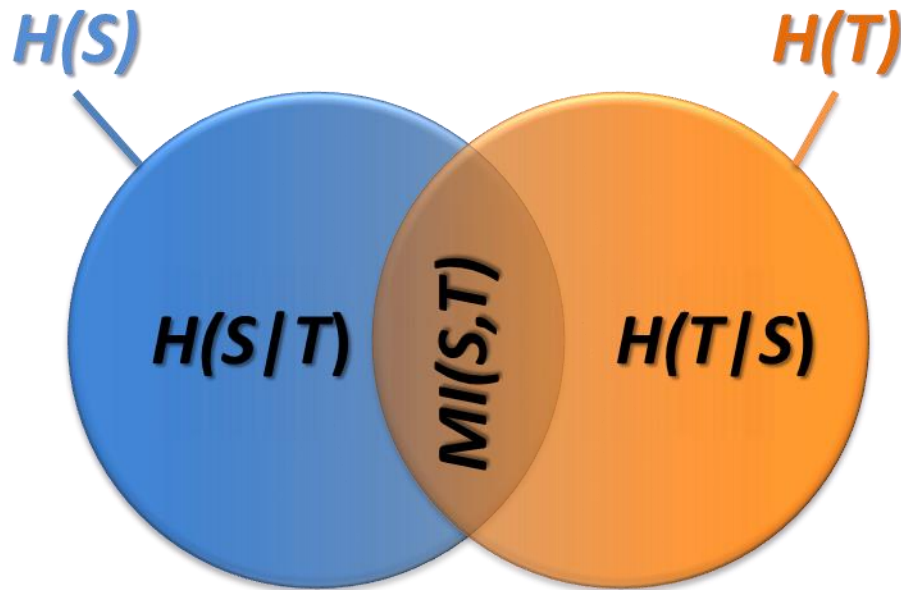


Figure 29 – Illustration of mutual information by way of intersecting circles.

Here, H is the entropy function, MI is mutual information, S is the signal, and T is the template.

where $p(S)$ and $p(T)$ are the probability densities for S and T , respectively. Both probability densities are intensity histograms within the template and signal. Empirical intensities are counted and normalized to probabilities. The joint probability, $p(S,T)$, is a two-dimensional histogram of the signal and the template where the frequency of gray-level value is matched from signal to template and counted.

The mutual information is the Kullback-Leibler distance (or relative entropy) between the product distribution and the joint distribution. The mutual information between S and T can be illustrated by way of a Venn diagram, as shown in Figure 29. Figure 29 presents two circles which represent the entropy of S (i.e., $H(S)$) and the entropy of T ($H(T)$). Their mutual information, $MI(S,T)$, lies in the intersecting portion, shown in the center of Figure 29. Also, outside the center, we have the conditional entropy of $H(S|T)$ on the left and of $H(T|S)$ on the right.

Mutual information is adequate to describe the similarity between the target signal S and the template T . However, multivariate analysis must be pursued if the interaction of the clutter C

is to be considered. McGill's 1954 work [56] and Fano's 1961 study [57] observe that the conditional entropy can be applied to extend the two-variable mutual information definition. We seek the mutual information, $MI(\{S, T\}|C)$, which gives the mutual information between S and T given C . This conditional mutual information can be written as [57] [56]:

$$MI(\{S, T\}|C) = [H(C, T) - H(C)] - [H(S, C, T) - H(S, C)]. \quad (49)$$

$MI(S, C, T)$ is what McGill [56] calls the "mutual interaction." The computation of $MI(\{S, T\}|C)$ subtracts this mutual interaction from the mutual information of S and T . The result, in terms of our trackability analysis, is a measure of the match between signal S and template T under the effect of clutter C .

One attractive feature of the formulation of (49) is that the joint entropy terms are straightforward to compute:

$$H(X_1, X_2, \dots, X_N) = - \sum_{x_1} \dots \sum_{x_N} P(x_1, x_2, \dots, x_N) \log_2 [P(x_1, x_2, \dots, x_N)]. \quad (50)$$

Now, we will investigate $MI(\{S, T\}|C)$ in assessing the signal-to-template match in a given video frame. Figure 30 illustrates how we generate $MI(\{S, T\}|C)$, where the mutual information between signal, S , and template, T , is conditioned on clutter, C . The three circles represent $H(S)$, $H(T)$, and $H(C)$, the entropies of the signal, template, and clutter, respectively. The area in common to all three entropies is $MI(S, T, C)$. The computation of $MI(\{S, T\}|C)$ essentially subtracts this mutual interaction from the mutual information of S and T . This measure is in information theoretic bits. For overall signal-to-template matching in bits per second, we multiply the average value of this mutual information term by the frame rate:

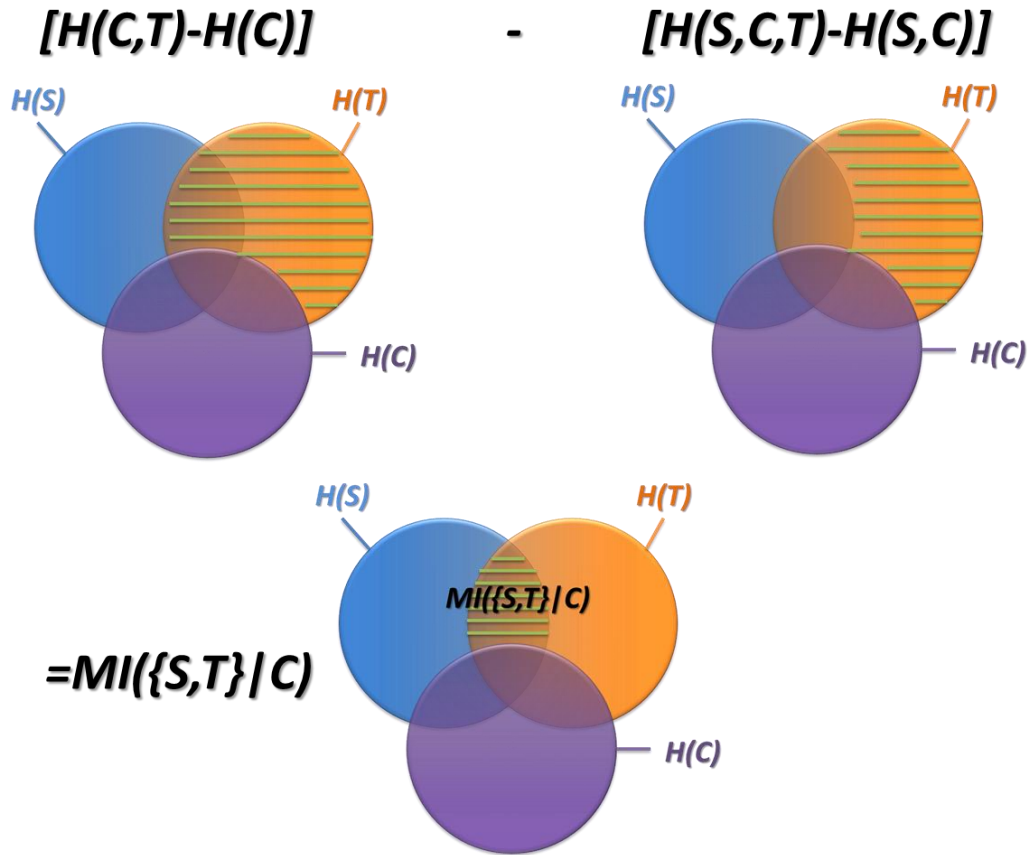


Figure 30 – Illustration of mutual information of signal and template given clutter $MI(\{S,T\}|C)$.
















In this figure the three circles represent $H(S)$, $H(T)$, and $H(C)$, the entropies of the signal, template, and clutter, respectively. The top row shows the result for the two equations above shaded with green lines. Then we illustrate the resulting mutual information of signal and template given clutter on the bottom row. This is our measure of the match between signal, S , and template, T , under the effect of clutter, C .

$$Q_{ST} = \overline{MI(\{S,T\}|C)} r_F \quad (51)$$

Next, we illustrate $MI(\{S,T\}|C)$ in order to demonstrate the mutual information for several types of examples. Table 7 provides examples of mutual information analysis from five of the tracking sequences. All datasets are from the Yuma, AZ data presented in earlier chapters. The template, T , is shown along with the signal, S (taken in the middle frame of the temporal sequence). From this same middle frame, we show the clutter, C — that is, the highest mutual information match that does not overlap the signal but is located inside the track gate. Mutual

information between the signal and template is shown along with the conditional mutual information that considers the template. Also given is the difference in mutual information between signal-to-template and clutter-to-template.

Table 7 - Mutual Information calculation for five target examples

Template T	Signal S	Clutter C	$MI(S,T)$ (bits)	$MI(\{S,T\} C)$ (bits)	$MI(S,T)-MI(C,T)$ (bits)
			3.1	2.8	0.5
			2.3	2.7	0.3
			3.6	3.7	0.5
			3.3	3.6	2.1
			3.1	3.9	1.2

Additionally, Figure 31 illustrates the specific example of the fourth target in the table above. The diagram shows that the template and signal are close in appearance but not identical. Consequently, $MI(\{S, T\}|C)$ results in 3.6 bits.

5.3 Quality of Video

We investigate four factors to determine video quality: spatial resolution of the target signal, the frame rate (temporal resolution), the quality of quantization (bit depth), and the effect of noise. For spatial resolution, we count the average number of pixels, N , (in a bounding box) of the target

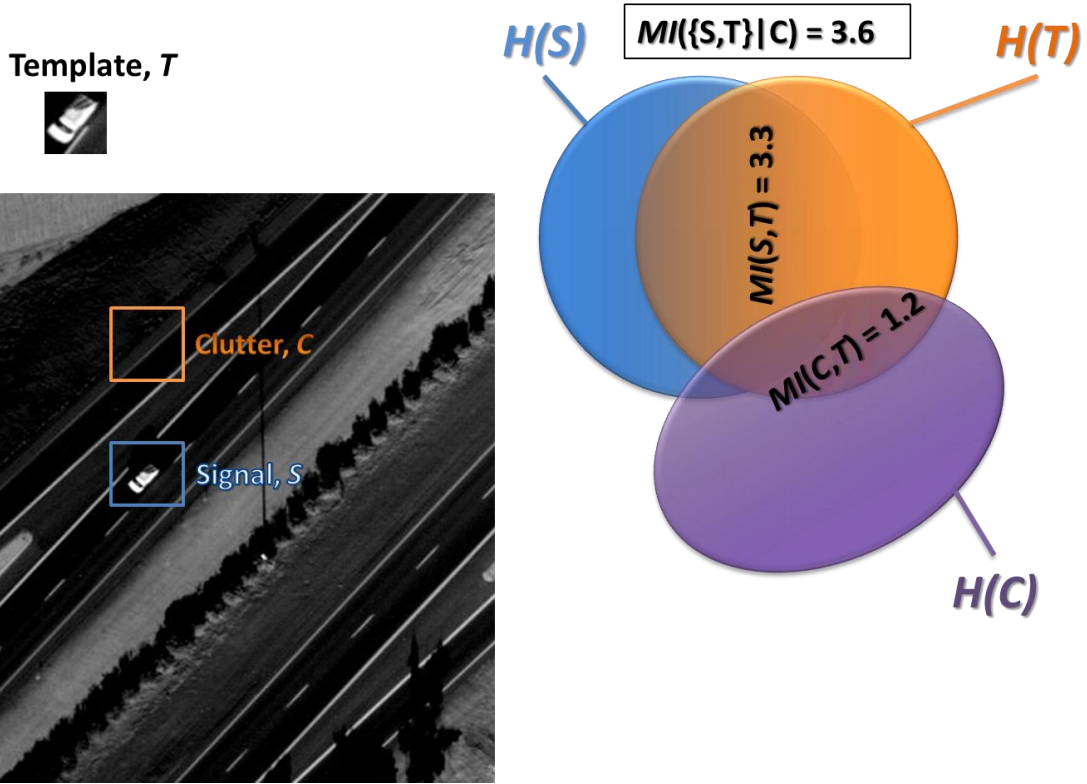


Figure 31 – An illustration of trivariate conditional mutual information.

In this figure, the fourth example from the results tabulated in Table 7 is illustrated. Here the track gate (bottom left), template (top left), and entropy diagram are shown. It can be seen that the target and signal are similar but not identical, thus having high mutual information. On the other hand, the clutter and the target are dissimilar, resulting in low mutual information.

signal. The frame rate is r_F in units of frames per second, and the bit depth is given as B_P in bits per pixel. Finally, in this work the effect of noise is characterized by the unitless ratio SNR. We consider the overall throughput of the noise-free video as a bandwidth, BW :

$$BW = Nr_F B_P. \quad (52)$$

Then, using the form developed by Shannon Hartley [58], we compute a data rate of $BW \log_2(1 + SNR)$.

Given registration (or the lack of registration) and the motion model, the error in the predicted position (the center of the track gate) for frame k is given by e_k . Here, we assume that $\{e_k\}$ is a normal random process with zero mean and standard deviation σ_e :

$$e_k \sim \mathcal{N}(0, \sigma_e^2). \quad (53)$$

Lastly, for a frame rate of r_F and a bit depth of B_p , we compute the uncertainty in position (due to motion of the target and/or video frame) in units of bits per second as $B_p r_F \sigma_e$, where this contribution to the video quality, Q_V , is negative:

$$Q_V = BW \log_2(1 + SNR) - B_p r_F \sigma_e. \quad (54)$$

As with Q_{ST} , Q_V has units of bits per second.

Although we have no rigorous theory with which to combine the two quantities, Q_{ST} and Q_V , we analyzed the possibility of maximizing the magnitude of anticorrelation of such a combination with the tracker performance as described in the results section below. The combination yielding an overall trackability measure was implemented as:

$$TM = wQ_{ST} + Q_V. \quad (55)$$

Here, w is a weighting constant.

5.4 Results

Towards analyzing the trackability measure, we first provide synthetic tests to determine robustness of our trackability measure. As no other comparable measure exists, we employ the synthetic data and compare it to SSIM and SCR, where each provides one of the elements of our trackability measure, but not a comprehensive measure as demonstrated by our work. We then demonstrate the results of the trackability measure using the Yuma, AZ data as described in Section 3.4. Essentially, our methods assume that registration is not possible, due to low overlap between frames and erratic movement of the sensor. Although we demonstrate efficacy on the Yuma dataset, the methods described apply to any rigid body tracking in unregistered video.

5.4.3 Synthetic Tests to Demonstrate Efficacy of Trackability Measure

To compare our trackability measure to another, similar measure is difficult, given that no such equivalent measure exists (one that considers video, motion model, quality of the template, presence of clutter, etc.). Therefore, we have constructed four synthetic experiments that use the aerial image of a car to test the basic differences of the trackability measure with the measures of SSIM and SCR.

Each experiment starts with the actual image of a vehicle taken from an airborne sensor.

Then, we test four basic scenarios.

3.4.5.1 The effect of gradual increase in clutter

Here, we assume an identical object in the neighborhood of the target we want to track. This

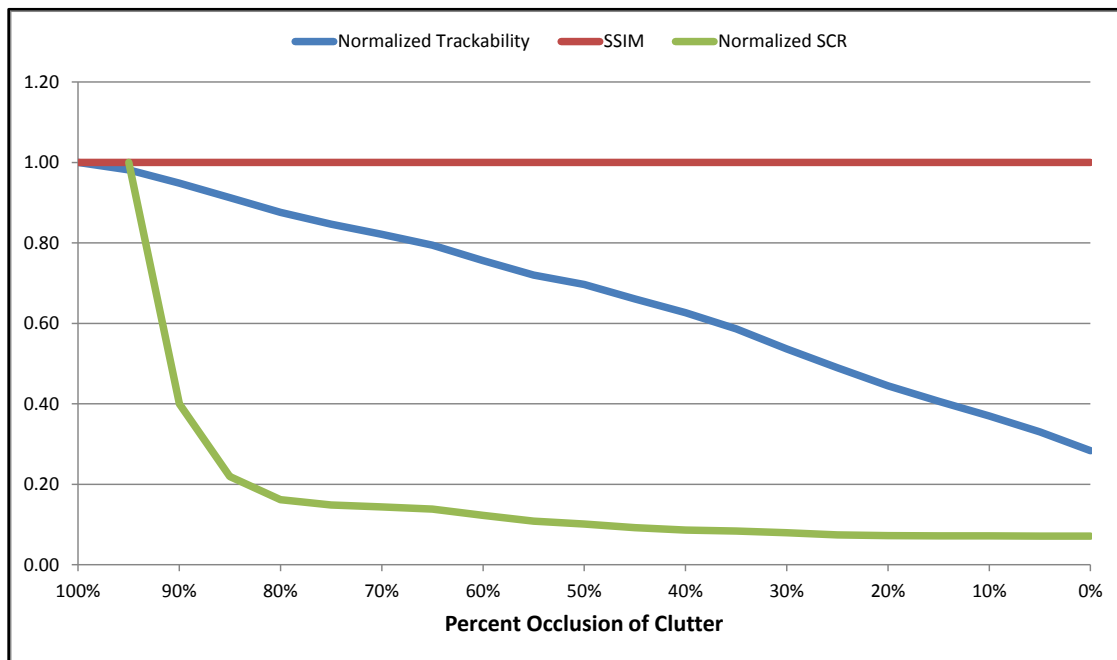


Figure 32 –Trackability Measure versus SSIM and SCR in varying clutter occlusion

Here we show the trackability measure as it compares to SSIM and SCR when the presence of clutter identical to the target is gradually revealed. The clutter starts by being fully hidden (occluded). The clutter object then increases in resemblance to the target in the image. It can be seen that SSIM does not account for clutter existence in the image, while SCR is significantly impacted by a small amount of the clutter's exposure.

clutter object is occluded gradually from 100% (no clutter) to 0% (perfectly matching object to the target of interest). Then, we record the trackability measure (normalized, so that it can be compared) with SSIM and normalized SCR. Here, we assume a perfect template and no noise. SSIM does not consider clutter in its measure and thus fails to show the effect of gradually revealing an identical object in the neighborhood of the target of interest, as illustrated in Figure 32. Note that the y-axis in all figures represents the normalized measures for comparison.

After approximately 30% of the clutter is revealed (70% still occluded), the SCR has a more or less flat response, indicating that an object that matches only a small portion of the actual template presents the same tracking challenge as an identical copy of the target in the vicinity. Trackability shows a basically linear decrease in difficulty with the linear increase in clutter (matching the template).

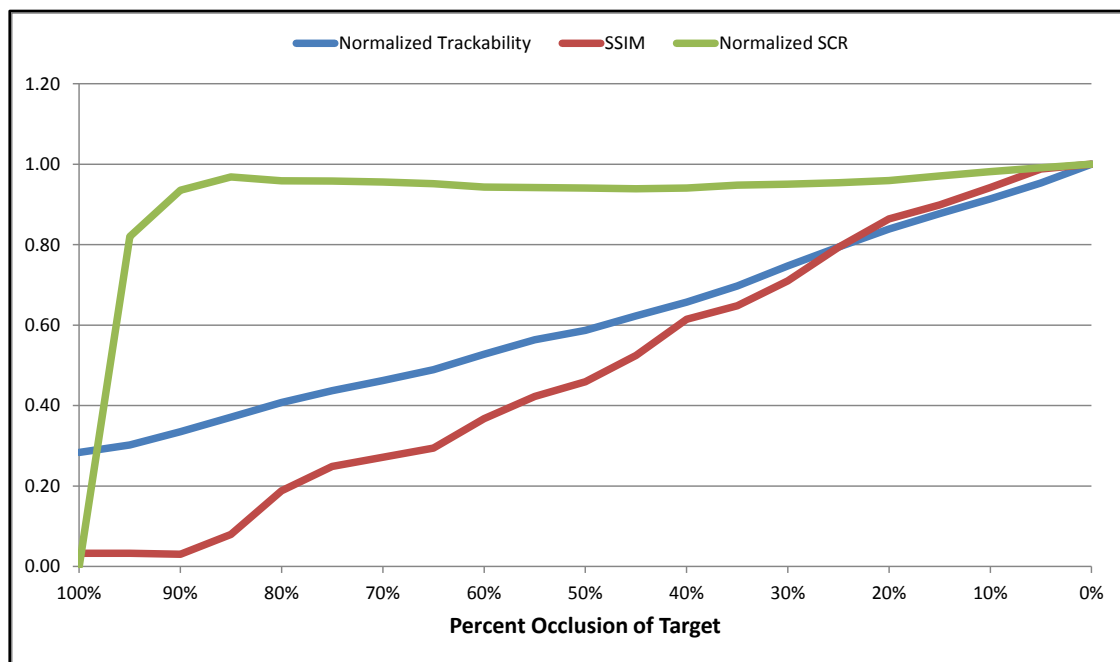


Figure 33 –Trackability Measure versus SSIM and SCR in varying target occlusion

Here we show the trackability measure as it compares to SSIM and SCR when the target is not present in the current frame (100% occlusion) and then the target is gradually revealed. It can be seen that the SSIM, as it solely investigates the target, gradually increases as the target is revealed. On the other hand, SCR almost plateaus once 15% of the target is revealed.

3.4.5.2 The effect of occlusion of the target

Here, we start with 100% occlusion of the target (impossible to track) and progress to 0% occlusion of the target (easy to track). In this experiment, we assume no clutter, a perfect template, and no noise. Next, we record trackability, SSIM, and SCR. Aside from 100% occlusion of the target, the response of SCR is fairly flat, as can be seen in Figure 33. For example, 50% occlusion of the target and 0% occlusion yield roughly the same SCR. Both SSIM and trackability gradually increase as the target is revealed indicating that as more of the target is visible, it becomes easier to track.

3.4.5.3 The effect of noise

Using normalized intensity values for the image (0 to 1), we increase the variance of Gaussian-distributed white noise (zero mean) from 0 to 1. Trackability, SSIM, and SCR are recorded. Here,

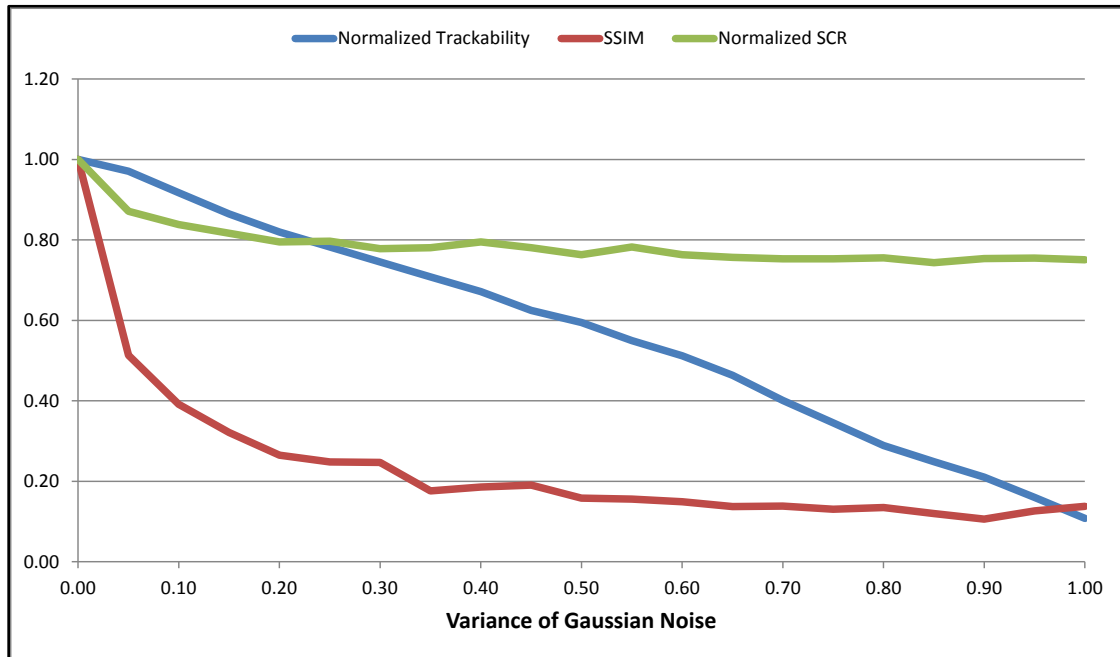


Figure 34 – Comparison of the Trackability Measure to SSIM and SCR in varying noise levels

Here we show trackability as it compares to SSIM and SCR when the noise level in the image increases. A Gaussian noise is added, increasing the variance from 0 to 1. SCR is only slightly impacted by the added noise. While all measures decrease, only the trackability measure follows a linear trend with noise increase.

all three measures respond in relationship with the noise variance, with only trackability having a roughly linear response to noise variance. These results are illustrated in Figure 34.

3.4.5.4 *The effect of error on the motion model*

The formulation of both SSIM and SCR does not account for a motion model. Consequently, from the results presented here it can be seen that both SSIM and SCR are unaffected by errors in the motion model. SSIM only measures the target as it is impacted by video quality, while SCR considers other objects in the vicinity of the target, or clutter. As the motion model error increases, only trackability decreases in proportion to the amount of error in the motion model. These results are illustrated in Figure 35.

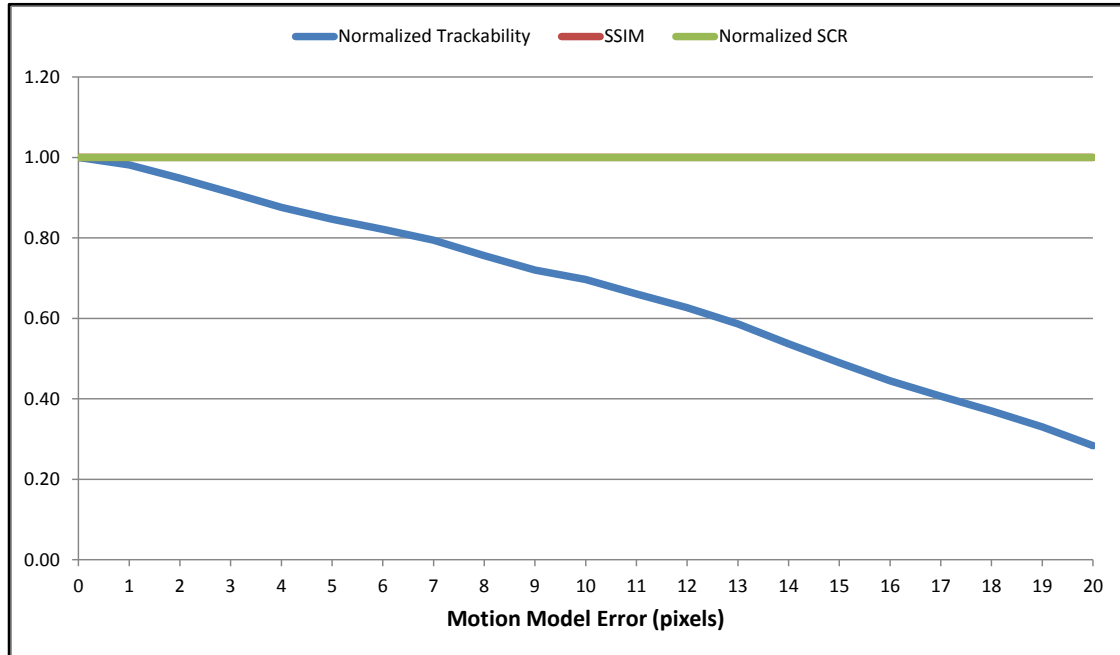


Figure 35 –Trackability versus SSIM and SCR in varying motion model error

Here we show trackability as it compares to SSIM and SCR when the error of the motion model is changed from zero pixels (easy to track) to 20 pixels (harder to track). Note that both SSIM and SCR completely overlap at 1. This is due to lack of motion model sensitivity in either measure. Only the trackability measure accounts for error increase in the motion model.

5.4.4 Trackability Results for Yuma, AZ Data

The results for the trackability measure are given for a sequence of wide area surveillance images taken from an airborne platform; the Yuma, AZ data we presented in earlier chapters (see section 3.4). We compare the performance of our ASIFT² (automated SIFT tracker) and the MS²T (morphological scale-space tracker) to the trackability measures Q_{ST} , and Q_V . Table 8 and Table 9 tabulate this performance. Note that “% frames tracked” refers to the percentage of frames in which the tracked position is inside the target. The RMSE measure is the root mean squared error in units of pixel width from the ground truth center positions.

Table 8 - The overall tracker performance versus signal-template match metrics.
The frame rate $r_F = 1$ Hz, $Q_{ST} = \overline{MI(\{S, T\} | C)}$. All mutual information quantities shown below are averages and are expressed in units of bits.

Sequence #	MS ² T		ASIFT ²		$MI(S, T)$	$Q_{ST} = \overline{MI(\{S, T\} C)}$
	% frames tracked	Normalized Tracked RMSE	% frames tracked	Normalized Tracked RMSE		
1	92%	0.06	83%	0.68	2.9	3.8
2	100%	0.2	45%	0.37	3.1	2.9
3	90%	0.15	90%	0.65	3.1	2.8
4	90%	0.61	10%	0.49	2.3	2.7
5	100%	0.19	100%	0.85	2.6	3.3
6	100%	0.25	100%	0.38	3.5	3.1
7	100%	0.08	100%	0.36	3	2.7
8	100%	0.09	55%	0.31	3.6	3.7
9	100%	0.32	100%	0.14	3.3	3.6
10	100%	0.14	100%	0.31	3.1	3.9

Table 9 - The overall tracker performance vs. video quality metrics
Here, TM is calculated with $w=600$ to maximize RMSE to TM correlation

Sequence number	Normalized RMSE - MS^2T	Normalized RMSE - $ASIFT^2$	SNR	Frame Rate (Hz)	Target Size (pixels)	Bit depth - bits per pixel	Motion uncertainty (pixels)	Quality (res. & SNR)	Quality (motion model)	Trackability Measure
				r_F	N	B_p	σ_e	$BW(1 + SNR)$	$-B_p r_F \sigma_e$	TM
1	0.06	0.68	15.1	1	1000	6.7	453.6	8085.7	-3039.12	7326.61
2	0.2	0.37	13.9	1	1000	5.6	514.1	6569.8	-2878.96	5430.88
3	0.15	0.65	9.7	1	1000	6	407	6176.3	-2442.00	5414.30
4	0.61	0.49	10.7	1	672	5.9	470.1	4235.1	-2773.59	3081.55
5	0.19	0.85	10.8	1	1000	6.1	751	6538.5	-4581.10	3937.38
6	0.25	0.38	13.3	1	1232	6	751	8540.2	-4506.00	5894.24
7	0.08	0.36	8.3	1	918	5.7	423.3	5067.7	-2412.81	4274.87
8	0.09	0.31	12.1	1	1742	7.1	845.5	13818.6	-6003.05	10035.58
9	0.32	0.14	9.2	1	900	6	258	5446.4	-1548.00	6058.44
10	0.14	0.31	12.3	1	1872	7.1	218.8	14937.3	-1553.48	15723.86

In this chapter we presented a novel trackability measure. This measure is a proof of concept that can be applied in the future to determine the difficulty of a given tracking scenario in a video sequence. Table 9 shows the efficacy of our tracking method, MS^2T . MS^2T , with an average error of 0.209, outperforms $ASIFT^2$, with an average error of 0.454. This performance exceeds 200% in error reduction. MS^2T also performs well on the sequences that have low trackability. In Table 9, TM is calculated using the weighting parameter, w , of 600. The table shows that trackability (TM) and MS^2T as well as $ASIFT^2$ performances are anticorrelated, as expected (Spearman correlation coefficient of -0.41 and -0.54, respectively). Furthermore, we illustrate the relationship between the tracking results and the trackability measure in Figure 36.

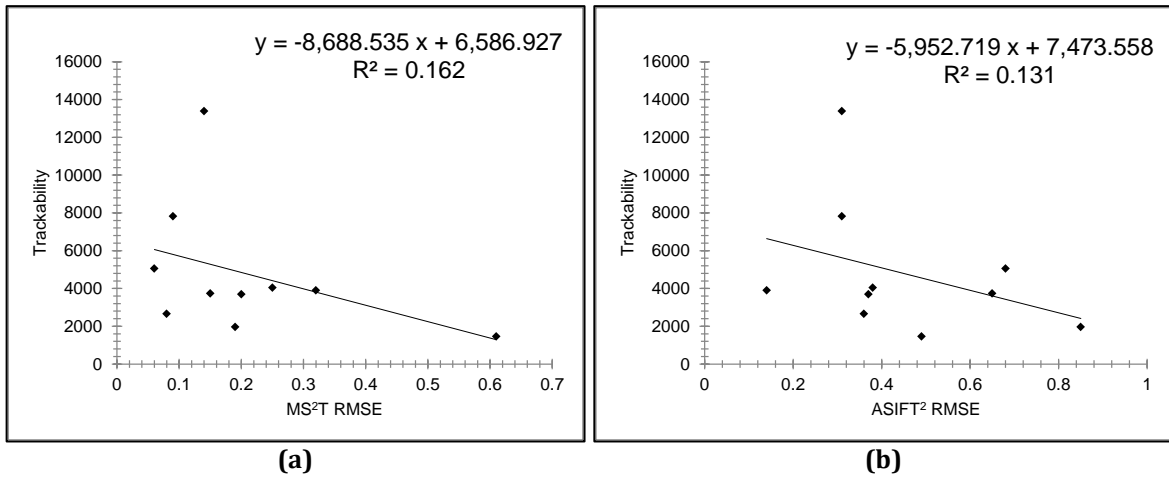


Figure 36 – Regression Analysis for RMSE of MS²T and ASIFT² vs. Trackability.

This figure illustrates the trackability measure as it relates to the RMSE for our trackers: MS²T and ASIFT². Both have an inverse relationship. As the RMSE increases, the trackability measure decreases.

5.5 Conclusions

This chapter developed a novel quality measure for tracking – trackability, where the trackability measure is computed in units of bits per second. The measure incorporates a comprehensive set of parameters that we believe directly impact tracking. These parameters are split into two measures – signal to template matching and video quality. In our synthetic results we illustrated how the measure is impacted by the signal to template matching by analyzing the results of occluding the target. Then we illustrated the impact of video quality on the measure by varying the noise level, the motion model, and the clutter in the vicinity of the target. In all synthetic results, our trackability measure had a direct relationship with the variance in the above parameters.

Additionally, when compared to our trackers from Chapter 3, the tracker performance (for the normalized RMSE of MS²T and ASIFT²) parallels the trackability measure, TM , with a Spearman correlation coefficient of -0.41 and -0.54. The negative correlation or anticorrelation is due to the fact that as mutual information increases, the error decreases. Given (55), a maximum-magnitude anticorrelation of -0.41 (with the RMSE of the MS²T) was achieved with weights w in

the neighborhood of $w = 600$. In future work, we plan to examine $Q_{ST} = \overline{MI(S, T) - MI(T, C)} r_F$ as an alternative to $Q_{ST} = \overline{MI(\{S, T\} | C)} r_F$.

The quality measure provided by the trackability analysis does not take into account or balance the negative aspects of increased computational expense. For example, the video quality for a temporal sampling of 30 fps is greater (in this trackability measure) than that of the same sequence subsampled to 10 fps. In this case, given a slowly moving target, the 10 fps sampling rate could be sufficient for tracking, while the 30 fps video costs roughly three times more to process.

We must also note that the trackability measure is only indicating the tracking difficulty for a given video and not how well a given tracker will perform. Our aim was to establish a methodology to quantify the ability to track targets and a very preliminary investigation was done to compare our novel trackability measure to our trackers presented in Chapter 4. A rigorous investigation is needed to study a variety of trackers and how their performances correlate to the trackability measure.

This work, for the first time, develops a trackability measure that can quantify the difficulty of a given tracking scenario. It considers the difficulty of finding the target in an information theoretic approach, which takes clutter into account. Furthermore, it goes beyond existing image quality measures by incorporating the effects of video frame rate and target motion.

For future work, additional parameters for the trackability measure should be considered and implemented. Firstly, the SSIM index presented in this chapter can be utilized as SNR since it provides a measure of the full reference target as it compares to a degraded target. Also, in registered video sequences a motion model can be incorporated. In the case of unregistered

sequences the lack of a motion model can indicate further difficulty of tracking because one less parameter can be identified.

Specific Aim 3: To develop a measure of tracking difficulty for a given target and sequence

Here we presented our methodology to develop a novel “trackability” measure. This measure uses a theoretical information approach to evaluate the difficulty of matching a template to a target in the presence of clutter. The measure also takes into account resolution, noise, frame rate, registration error, and motion model prediction error. This trackability measure is the first such attempt to quantify the difficulty of a given tracking experiment [59].

CHAPTER 6.

CONCLUSION

The increased prevalence of video cameras has led to an elevated volume of video data that requires analysis. Employing a human observer to analyze and tag video sequences is expensive, limited in capability, and prone to error. Consequently, an automated method is essential for the analysis of video sequences. Additionally, all video data are not created equal. Depending on the video acquisition mechanism, the resulting data will have varying temporal and/or spatial resolution, differing wavelengths, and can be acquired from a variety of platforms, such as buildings, helicopters, UAVs, etc. Therefore, we require a methodology to distinguish between the varying types of video as they impact trackability.

The aim of this dissertation was to study tracking obstacles emerging from target tracking in surveillance video sequences. To achieve this goal, we developed several novel tracking algorithms to tackle a variety of video sequences. We also designed a trackability measure to analyze the difficulty of tracking targets in video sequences whose focus relates to persistent surveillance applications.

Consequently, the first major contribution of this work is that it provides two automated methods to track multiple targets in persistent surveillance video sequences. In the first approach, we develop an automated tracker for registered (stationary camera) video sequences, while in the second approach we present three techniques for unregistered video sequences. The *Morphological Scale-Space Tracker (MS²T)* provides unique capabilities to track unregistered video sequences with low temporal sampling. The second major contribution is the introduction of a novel trackability measure that allows the user to quantify the difficulty of tracking in a variety of environments via an assortment of imaging sensors. Each aim of the dissertation was validated by

the appropriate set of experimental results as specified for each aim below. We believe that MS²T has viable extensions to object recognition, robotic navigation, and scene understanding work.

Aim 1 – Chapter 3

Our first specific aim was *to develop an automated tracking algorithm for a single moving target in surveillance video sequences by fusing a particle filter with the active contour.*

To address this aim we established a novel tracker for video surveillance applications presented in Chapter 3. The Snake Particle Filter (SPF) tracker unites the snake and the PF. The snake is used to establish the motion model and to determine particle weights, while the PF is employed to handle non-linear, non-white noise systems. In applying our tracking algorithm to two data sets, a low RMSE between the results of SPF and manual tracked targets was less than half a target in the horizontal and vertical directions.

Aim 2 – Chapter 4

Our specific aim was *to accomplish tracking with a morphological scale-space that allows tracking multiple targets in unregistered video sequences.*

Towards this aim, Chapter 4 illustrated three novel ideas towards successful tracking in unregistered video: the automated SIFT tracker, the morphological scale-space tracker (MS²T) and the incorporation of SIFT into the morphological scale-space tracker. We demonstrated the improvements in tracking as we developed each tracker. Additionally, we utilized our morphological scale-space concepts towards successfully detecting the target in every frame of unregistered video sequences. In applying our tracking algorithm to unregistered data, we demonstrated that by incorporating SIFT into our MS²T, we track 98% of the frames in the video

sequences on average with a normalized RMSE value of 0.28 of the tracked frames. It should be noted that percentage of tracked frames is a more significant measure than RMSE.

Aim 3 – Chapter 5

Our specific aim was *to develop a measure of tracking difficulty for a given target and sequence*.

To address this aim we presented a novel trackability measure in Chapter 5. The measure is a first known proof of concept that can be applied in the future to determine the ‘fit’ of a video sequence for tracking. This measure uses a theoretical information approach to evaluating the difficulty of matching a template to a target in the presence of clutter. The measure also takes into account resolution, noise, frame rate, registration error, and motion model prediction error. This trackability measure is the first such attempt to quantify the difficulty of a given tracking experiment. In this chapter we illustrated the direct impact of degradation in video quality, clutter, and occlusion of the target on the trackability measure.

Our research presented here can be further developed with the following future improvements.

In our first aim, the SPF can be extended to include automatic monitoring for incoming targets, and the characterization of target-to-target interaction. For our second aim, future investigation can include incorporating relative key-point locations to the target center to enhance the ASIFT². Knowing the location of the SIFT key-points will improve the SIFT tracker’s knowledge of the centers of targets.

Another idea that can be investigated is to use SIFT as a global tracker from one frame to the next. As a result, registration can be achieved and the global motion of the frame can be established. Knowing the motion of the frame will allow extracting the targets that are not moving with the general motion. Additionally, GPU implementations of both SIFT and the

connected filter should be investigated. The recent applications of GPU algorithm implementation have demonstrated a significant improvement in execution time. Sinha et al. demonstrated 20 times improvement over CPU implementation of SIFT key-points tracking in video [60]. Since many aspects of both algorithms could run in parallel, both algorithms will have shorter execution time if implemented using GPUs.

Lastly, our trackability measure is the first step in analyzing trackability of video sequences. The next step is to analyze large quantities of video sequences with varying qualities to understand the trackability of the various types of video. Additionally, the measure should be utilized to compare various tracking methodologies to study how tracking techniques perform relative to video sequences and the trackability measures.

REFERENCE

- [1] A. Rackham, *Alice in Wonderland Down: the Rabbit Hole*.
- [2] M. F. Abdelkader, R. Chellapa, Q. Zheng and A. L. Chan, "Integrated Motion Detection and Tracking for Visual Surveillance," in *Fourth IEEE International Conference on Computer Vision Systems*, New York, 2006.
- [3] W. Hu, T. Tan, L. Want and S. Maybank, "A Survey on Visual Surveillance of Object Motion and Behaviors," *IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews*, vol. 34, no. 3, pp. 334-352, August 2004.
- [4] M. S. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A tutorial on particle filter for online nonlinear/non-Gaussian bayesian tracking," *IEEE Trans. on Signal Processing*, vol. 50, pp. 174-188, 2002.
- [5] M. Kass, A. Witkin and D. Terzopoulos, "Snakes - Active Contour Models," *International Journal of Computer Vision*, vol. 1, pp. 321-331, 1987.
- [6] R. Collins, A. Lipton and T. Kanade, "Introduction to the special section on video surveillance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 745 - 746, 2000.
- [7] A. L. Gilbert, M. K. Giles, G. M. Flachs, R. B. Rogers and U. Y. Hsun, "A real-time video tracking system," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vols. PAMI-2, no. 1, pp. 47-56, 1980.
- [8] H. R. Myler, *Fundamentals of Machine Vision*, Bellingham, Washington: SPIE Press, 1999.
- [9] L. Stephan, J. E. Albus and J. R. Marcovici, "Fitts correlation tracker fidelity in presence of target translation, rotation, and size change," in *Proc. SPIE Acquisition, Tracking, and Pointing*

XVI, Orlando, FL, 2002.

- [10] J. P. Lewis, "Fast Template Matching," in *Canadian Image Processing and Pattern Recognition Society - Vision Interface 95*, Quebec City, Canada, 1995.
- [11] F. Leymarie and M. Levine, "Tracking deformable objects in the plane using an active contour model," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, no. 6, pp. 617-634, June 1993.
- [12] C. Xu and J. L. Prince, "Generalized Gradient Vector Flow External Forces for Active Contours," *An International Journal on Signal Processing*, vol. 71, no. 2, pp. 131-139, December 1998.
- [13] N. Ray and S. T. Acton, "Motion gradient vector flow: : an external force for tracking rolling leukocytes with shape and size constrained active contours," *IEEE Transactions on Medical Imaging*, vol. 23, no. 12, pp. 1466-1478, 2004.
- [14] J. Tang, G. Dong, N. Ray and S. Acton, "Evaluation of intravital tracking algorithms," in *Proc. 45th Midwest Symposium on Circuits and Systems*, Tulsa, Oklahoma, 2002.
- [15] B. Li and S. T. Acton, "Active contour external force using vector field convolution for image segmentation," vol. 16, no. 8, p. 2096–2106, 2007.
- [16] V. Caselles, R. Kimmel and G. Sapiro, "Geodesic Active Contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61-79, Feb./March 1997.
- [17] N. Paragios and R. Deriche, "Geodesic active regions and level set methods for motion estimation and tracking," *Computer Vision and Image Understanding*, vol. 97, pp. 259-282, 2005.
- [18] D. Comaniciu, V. Ramesh and P. Meer, "Kernel-Based Object Tracking," *IEEE Trans. on Pattern*

Analysis and Machine Intelligence, vol. 25, no. 5, pp. 564-577, May 2003.

- [19] D. Terzopoulos and R. Szeliski, "Tracking with Kalman Snakes," in *Active Vision*, A. Blake and A. Yuille, Eds., Cambridge, MA, MIT Press, 1992, pp. 3-20.
- [20] N. Peterfreund, "Robust Tracking of Position and Velocity with Kalman Snakes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 564-569, 1999.
- [21] T. H. Lam and R. S. Lee, "Visual Tracking by Using Kalman Gradient Vector Flow (KGVF) Snakes," in *Knowledge-Based Intelligent Information and Engineering Systems*, Berlin, Springer, 2004, pp. 557-563.
- [22] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425-437, 2002.
- [23] N. Gordon, D. Salmond and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEEE Proc. Radar and Signal Processing*, vol. 140, no. 2, pp. 107-113, 1993.
- [24] M. Isard and A. Blake, "CONDENSATION—Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5-28, 1998.
- [25] Y. Rathi, N. Vaswani, A. Tannenbaum and A. Yezzi, "Particle filtering for geometric active contours with application to tracking moving and deforming objects," in *Computer Vision and Pattern Recognition*, Atlanta, GA, 2005.
- [26] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, vol. I, Reading, MA: Addison-Wesley, 1992.
- [27] A. C. Bovik, *Handbook of Image and Video Processing*, 2nd Edition ed., A. C. Bovik, Ed., San

- Diego, CA: Academic Press, 2005.
- [28] S. T. Acton and N. Ray, *Biomedical Image Analysis: Tracking*, A. C. Bovik, Ed., San Rafael, CA: Morgan and Claypool Publishers, 2006.
- [29] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. 1-25, March 1996.
- [30] February 2006. [Online]. Available: <http://www.ngsim.fhwa.dot.gov>.
- [31] B. Zitova and J. Flusser, "Image Registration Methods: a Survey," *Image and Vision Computing*, vol. 21, pp. 977-1000, 2003.
- [32] A. Aksel and S. T. Acton, "Target tracking using the snake particle filter," in *IEEE Southwest Symposium on Image Analysis & Interpretation (SSIAI)*, Austin, TX, 2010.
- [33] D. Marr, *Vision: a computational investigation into the human representation and processing of visual information*, San Francisco: MIT Press 2010, 1982.
- [34] T. Lindeberg, "Scale-space," in *Encyclopedia of computer science and engineering*, B. Wah, Ed., John Wiley and Sons IV, 2008, pp. 495-2504.
- [35] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.
- [36] H. Zhou, Y. Yuan and C. Shi, "Object tracking using SIFT features and mean shift," *Computer Vision and Image Understanding*, vol. 113, p. 345-352, 2009.
- [37] C. Harris and M. Stephens, "A Combined Corner and Edge Detection," in *Proceedings of The Fourth Alvey Vision Conference*, 1988.
- [38] Z. Zhang, R. Deriche, O. Faugeras and Q.-T. Luong, "A robust technique for matching two

- uncalibrated images through the recovery of the unknown epipolar geometry," *Artificial Intelligence*, vol. 78, no. 1-2, pp. 87-119, 1995.
- [39] S. Belongie, J. Malik and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Tran. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 24, pp. 509-522, April 2002.
- [40] J. J. Koenderink, "The Structure of Images," in *Biological Cybernetics*, vol. 50, 1984, pp. 363-370.
- [41] T. Lindeberg, "Scale-Space Theory: A Basic Tool for Analysing Structures at Different Scales," *Journal of Applied Statistics*, vol. 21, no. 2, pp. 225-270, 1994.
- [42] J. H. Bosworth and S. T. Acton, "Morphological Scale-Space in Image Processing," *Digital Signal Processing*, vol. 13, p. 338-367, 2003.
- [43] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 561-576, 2000.
- [44] S. Belongie, J. Malik and J. Puzicha, "Matching shapes," in *IEEE International Conference on Computer Vision*, 2001.
- [45] A. J. Lipton, H. Fujiyoshi and R. S. Patil, "Moving Target Classification and Tracking from Real-time Video," in *Fourth IEEE Workshop on Applications of Computer Vision*, Princeton, NJ, 1998.
- [46] S. Gupte, O. Masoud, R. F. K. Martin and N. P. Papanikolopoulos, "Detection and Classification of Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 37-47, March 2002.

- [47] A. Aksel and S. T. Acton, "Target Tracking Via the Morphological Scale-Space Tracker," *The Imaging Science Journal*, Submitted in 2014.
- [48] A. M. Eskicioglu and P. S. Fisher, "Image Quality Measures and Their Performance," *IEEE Transactions on Communications*, vol. 43, no. 12, pp. 2959-2965, 1995.
- [49] Z. Wang, L. Lu and A. C. Bovik, "Video quality assessment based on structural distortion measurement," *Signal Processing Image Communication*, vol. 19, no. 2, pp. 121-132, 2004.
- [50] Z. Wang, L. Lu and A. C. Bovik, "No-reference perceptual quality assessment of JPEG compressed images," in *Proc. of IEEE International Conference on Image Processing*, 2002.
- [51] X. Li, "Blind image quality assessment," in *Proc. of IEEE International Conference on Image Processing*, 2002.
- [52] R. A. Peters and R. N. Strickland, "Image Complexity Metrics for Automatic Target Recognizers," in *Automatic Target Recognizer System and Technology Conference*, Silver Spring, MD, 1990.
- [53] M. A. Phillips and S. R. F. Sims, "Signal-to-clutter measure for ATR performance comparison," in *Proc. of SPIE Automatic Target Recognition VII*, Orlando, FL, 1997.
- [54] W. M. Wells, P. Viola, H. Atsumi, S. Nakajima and R. Kikinis, "Multi-modal volume registration by maximization of mutual information," *Medical Image Analysis*, vol. 1, pp. 25-51, 1996.
- [55] C. E. Shannon and W. Weaver, *The mathematical theory of communication*, Urbana: University of Illinois Press, 1949.
- [56] W. McGill, "Multivariate information transmission," *IEEE Tran. on Information Theory*, vol. 4, pp. 93-111, 1954.
- [57] R. Fano, *Transmission of Information: a statistical theory of communications*, Cambridge: MIT

Press, 1961.

- [58] C. E. Shannon, "Communication in the presence of noise," *Proc. Instit. of Radio Engineers*, vol. 37, pp. 10-21, 1949.
- [59] S. Acton and A. Aksel, "An information theoretic trackability measure," in *SPIE Computational Imaging X*, Burlingame, California, 2012.
- [60] S. N. Sinha, J.-M. Frahm, M. Pollefeys and Y. Genc, "Feature tracking and matching in video using programmable graphics hardware," *Machine Vision and Applications*, vol. 22, pp. 207-217, 2011.