Vehicle Tracking in Occlusion and Clutter

by

KURTIS NORMAN MCBRIDE

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2007
© Kurtis McBride 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Vehicle tracking in environments containing occlusion and clutter is an active research area. The problem of tracking vehicles through such environments presents a variety of challenges. These challenges include vehicle track initialization, tracking an unknown number of targets and the variations in real-world lighting, scene conditions and camera vantage. Scene clutter and target occlusion present additional challenges. A stochastic framework is proposed which allows for vehicles tracks to be identified from a sequence of images. The work focuses on the identification of vehicle tracks present in transportation scenes, namely, vehicle movements at intersections. The framework combines background subtraction and motion history based approaches to deal with the segmentation problem. The tracking problem is solved using a Monte Carlo Markov Chain Data Association (MCMCDA) method. The method includes a novel concept of including the notion of discrete, independent regions in the MCMC scoring function. Results are presented which show that the framework is capable of tracking vehicles in scenes containing multiple vehicles that occlude one another, and that are occluded by foreground scene objects.

Acknowledgments

I would like to express my sincerest appreciation to Professor David Clausi and Professor Paul Fieguth for their support and guidance in both scholastic and personal matters over the course of my Masters research.

I would also like to acknowledge the support of the Ontario Centres of Excellence (OCE), formerly Communications and Information Technology Ontario (CITO), and BA Consulting Group for their financial support of this research.

Dedication

This thesis is dedicated to the belief that a person should be measured less by what they have already achieved and more by what they aspire to become.

Contents

List of Figures

List of Tables

Glossary of Terms

t	current time index
T	total time window of interest
i,j	pixel indexes
$b_{i,j}$	Pixel-based Gaussian mixture background scene model at pixel i,j
$q_{i,j}$	Number of modes in the Gaussian mixture model centered on pixel i,j
$\pi_{k,i,j}$	Weight of mode q in the background model at pixel i,j
$m_{k,i,j}$	The mean of mode q in the background model at pixel i,j
$\sigma_{k,i,j}$	The variance of mode q in the background model at pixel i,j
$\phi_{i,j}$	Discrete state (foreground/baground) of pixel i,j
$v_{i,j}$	Velocity at pixel i,j
$W(i,j)$	Neighbourhood search window centred on pixel i,j
J	Energy minimization function
N_k	Number of samples in k^{th} cluster
$x_{n,k}$	n^{th} sample in k^{th} cluster
Υ	Set of particles representing a time dynamic stochastic process
υ_n	Single particle in the set Υ
N_Υ	Number of particles in the set Υ
Γ	Set of weights for each of the particles in the set Υ
γ_n	Single weight in the set Γ
$\mathcal{F}_T$	Set of all images in an image sequence
$\mathbf{F}(t)$	Single image in the set $\mathcal{F}_T$
$f_{i,j}(t)$	Single pixel in the image $\mathbf{F}(t)$
$\mathcal{S}_T$	Set of all information region sets in the image sequence $\mathcal{F}_T$
$\mathbf{S}(t)$	Set of information regions at time t
$s_n(t)$	Single information region in the set $\mathbf{S}(t)$
$N_{S(t)}$	Number of information regions in the set $\mathbf{S}(t)$
$\mathbf{\Psi}(t)$	Association between information regions
$\mathcal{R}_T$	Set of all target sets in the image sequence $\mathcal{F}_T$
$\mathbf{R}(t)$	Set of targets at time t

$r_n(t)$	Single target in the set $\mathbf{R}(t)$
$N_{R(t)}$	Number of targets in the set $\mathbf{R}(t)$
Ω_T	Set of all target tracks in the set $\mathcal{F}_T$
τ_n	Single target track in the set Ω_T
N_τ	Number of target tracks in the set Ω_T
$N_{a,t}$	Number of new targets at time t
$N_{z,t}$	Number of terminated targets at time t
$N_{d,t}$	Number of detected targets at time t
$N_{f,t}$	Number of falsely identified targets at time t
$G(\sigma)$	Gaussian image kernel with variance σ
$D(t,\sigma)$	Difference of Gaussian image pyramid
N_Δ	Number of levels in an image pyramid
$\mathcal{E}_t$	Joint association between $\mathcal{R}_t$ and Ω_t
$\boldsymbol{R}$	Set of all image regions in $\mathcal{F}_T$
$\mathbf{r}_n$	Single region in the set $\boldsymbol{R}$
ω	Discrete partition of selected from $\mathcal{E}_t$
K_t	Number of target tracks at time t
p_z	Probability of target track termination
p_d	Probability of target track detection
λ_f	False target track detection rate
λ_a	Target track birth rate
ξ	Sampler move selection probability

Chapter 1

Introduction

1.1 Problem Context

The work presented in this thesis was motivated in part by a need identified in the urban transportation planning industry to reduce the cost of collecting transportation data by replacing manual methods with computer vision tracking algorithms. In order to understand and investigate the operation of transportation networks, the urban transportation planning industry engages in traffic monitoring activities. These engagements involve manually capturing information about the operation of a given transportation environment. Examples include intersections, drive-throughs and highways, as shown in Figure 1.1. The use of computer vision tracking technology to address this identified need has the potential to greatly reduce the costs traditionally associated with manual counting methods. In addition to the potential for cost savings which arise from using computer vision tracking technology to collect traffic data, the technology is capable of providing more accurate traffic data as compared to manual methods. The traffic engineering community generally recognizes that the long duration, high vehicle volumes and mundane nature of manually collecting traffic data result in inconsistent and inaccurate data.

The objective of this thesis is to propose and develop a stochastic framework which will allow target tracks to be identified from a sequence of images. The stochastic framework is based on a data-oriented combinatorial optimization method for solving the Monte Carlo Markov Chain data association problem [42] and Bayesian recursion [47]. The method takes a deferred logic approach, meaning that the association of targets between frames considers all available data for a given time

Figure 1.1: Example Transportation Scenes: (a) Highway, (b) Intersection, (c) Drive-through.

period. This allows the initialization of a new track to be based on the entire history of observations for a given track. The framework presented is computationally efficient, and the Monte Carlo nature lends itself to application in parallel computing. In particular, this thesis will focus on the identification of tracks which pertain to real-world transportation scenes. The framework was formulated to operate under conditions where vehicles exist under conditions of occlusion and scene clutter. Although the thesis will focus primarily on the identification of vehicle movements, the stochastic framework presented is flexible enough to track a wide range of visually identifiable events, such as people in urban and retail environments [20].

1.2 Problem Overview

The topic of target tracking is one which has received a great deal of recent discussion in the computer vision literature. Popular methods include kernel-based approaches [8], [10], [28], particle filtering based approaches [13], [15], [21] and approaches based on data association [1], [12], [41]. These methods are discussed in more detail in Section 2.3, Section 2.5 and Section 2.6 respectively.

Kernel-based methods can be effective in relatively simple tracking applications; however, they tend to break down when the object being tracked has a tendency to deform rapidly between frames, or to become occluded over the course of the target's existence in a given scene. Depending on the type of kernel metric selected, the effects of shadowing can also cause kernel based tracking methods to break down.

Particle-based tracking methods have the advantage that they can simultaneously

represent multiple hypotheses about a target's state. Depending on the rate of decay that is selected in the algorithm, particle methods also have the ability to keep a *memory* about the evolving state of an object over subsequent images. This can be useful for resolving targets through situations of occlusion. The disadvantage of these methods is that they require relatively accurate target initialization and termination, which makes tracking an unknown number of targets in a complex scene difficult. To approximate a number of possible hypotheses about a given target location successfully, depending on the model used and method of parametrization, these methods can require the use of many particles. This makes them computationally intense, meaning that real-time performance suffers.

Methods based on data association are effective at target tracking if the associations are performed over multiple image frames in sequence, although this can lead to computational infeasibility if the association is conducted using a brute force approach. These computational issues are further exacerbated as the number of targets that are being tracked increases. Methods such as gating [11] can be employed in order to reduce the computational requirements. Methods based on a Monte Carlo approaches to data association can also greatly reduce the computational requirements [33]. Monte Carlo data association will be the focus of the tracking algorithm presented in this work.

1.3 Thesis Contribution and Organization

The primary contribution of this thesis is the presentation of a stochastic framework for generalized vehicle tracking based on Monte Carlo Markov Chain Data Association (MCMCDA). The framework can automatically initialize and terminate an unknown number of tracks in scenes containing regions of occlusion and noise in the form of scene clutter. Scene clutter can give rise to false alarms about the presence of targets in a scene. The implementation that is discussed also utilizes scale-space [24] and Delaunay clustering [35], which, together with MCMCDA, represent a unique approach to vehicle tracking.

This thesis is organized into three main discussion chapters. First, a background chapter (Section 2), provides insights into the various mathematical concepts that contributed to the formulation of this thesis. Second, a tracking chapter (Section 3), discusses the proposed tracking framework and its implementation considerations. Finally, an application chapter (Section 4) highlights how the proposed framework can be applied to vehicle tracking.

Chapter 2

Background

2.1 General Multi-Target Tracking

Multi-target object tracking is an important, unsolved problem in many areas of computer vision. Object tracking has application in a range of industries and applications. Examples include surveillance [33], vehicle tracking [29] and pedestrian tracking [43]. In this chapter, many of the concepts which are central to multi-target tracking will be discussed.

The literature about target tracking focuses on two major areas: target detection [14] and target-track association [32]. Target detection is generally conducted in one of two ways – either using a physical model of the target of interest [5] or using some notion to distinguish foreground from background [23]. Both approaches have their own inherent advantages and disadvantages. Background subtraction is the focus of Section 2.2. Motion estimation is discussed in Section 2.3. Various clustering strategies are discussed in Section 2.4. Probability density propagation is the focus of Section 2.5. Target-track data association methods are discussed in Section 2.6.

Methods based on physical models of a target can be effective; however, the models are difficult to apply to generic target tracking problems as the specific size and structure of the target being tracked may not be known. Additionally, since an image in an image sequence is a two-dimensional projection of a scene that is actually three-dimensional, it is often difficult to know which aspect of a given physical model to employ to a segmentation of the given object. In environments where the position and angle of the camera can be controlled or estimated, and

where the type of object being tracked is known, physical model-based methods are effective as the computational complexity involved with matching target segments to models can be constrained. In generalized target tracking applications, to assert knowledge about the relative angle between the camera and the road plane, or the specific structure of a given target being tracked is not always practical.

Once a target has been detected using either background subtraction or a physical model-based approach, a feature vector can be generated about the detected target. Feature vectors from subsequent frames can then be evaluated using a data association approach which chains together feature vectors to form tracks. A variety of data association methods exist, such as JPDA [11] or Markov chain based data association [16]. One of the difficulties faced with generalized object tracking is that often the number of targets being tracked at any given time is not known.

Figure 2.1 illustrates the allowable vehicle movements that exist for a single vehicle at a T-intersection. The figure illustrates a T-intersection where collecting information about six of the possible vehicle movements is desired: left-out, right-out, left-in, left-out and the two through movements. An understanding of the number of times that each of these movements occurs is important when urban planners are making operational design decisions about a given intersection. The various movements that are illustrated show the vehicle tracks which are identified a priori, and are desired to match against vehicle tracks that can be observed in the scene.

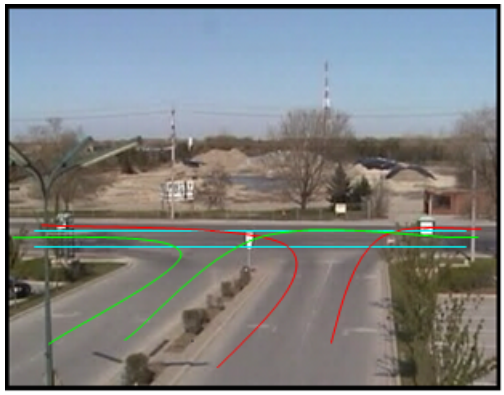

Figure 2.1: Potential Vehicle Movements at a T-Intersection: Left-out and right-out movements are shown in red. Left-in and left-out movements are show in green. The two possible through movements are shown in blue.

2.2 Background Subtraction

A key element of many target tracking algorithms is accurate background subtraction. Background subtraction [26] is used primarily to identify image regions that contain foreground information. Foreground regions of an image include all regions that are made up of non-static elements of the scene (i.e., vehicles). A key element of background subtraction involves maintaining an accurate background model of an image sequence, as the background in real-world scenes tends to be dynamic. The background generally changes due to variation in lighting conditions, camera vibration and environmental conditions. Depending on how a background model is maintained, objects that persist in the scene can also change the background model.

Background subtraction is a commonly used method for isolating pixels which correspond to foreground objects of interest [52]. This inherently assumes that a foreground object of interest is sufficiently different from the background. Assuming that this assumption is valid and that an accurate notion of what makes up the background elements in a scene can be maintained, isolating foreground elements from the rest of the scene is possible. In the literature about background subtraction there are two main approaches that are discussed: parametric approaches (typically based on assumptions about the Gaussianity of the background at a given pixel [37]) and non-parametric approaches (typically based on nearest neighbor methods [19]). In both cases, a pixel is considered to be part of the foreground if it is beyond a certain threshold distance away from the model at any given time, as discussed in [52]. The parameterized version of this concept is conveyed in (2.4).

A variety of challenges exist when trying to accurately maintain a model of the background in a sequence of images. For the purposes of this work, an assumption is made that all image sequences are captured using a static camera position. In practical applications this may not always be the case; therefore, an ideal background subtraction model should handle translation and rotation in camera position. In addition, the background may consist of multiple scene elements at a particular pixel, meaning that the background is not uni-modal. An example of this would be a green bush blowing in the wind in front of a grey road. A background model that is able to model multiple modes per pixel would result in both the grey and green scene elements being considered background, while other coloured pixels would be considered foreground. Finally, the ideal background model is dynamic and able to handle slow variations in the background conditions, as well as be able to identify when static background objects which may have been present at initialization are

removed from the scene. Figure 2.2 illustrates the histogram of a pixel centered on a swaying bush taken over ten seconds of video. The small black boxes in frame 1 and frame 2 of the figure are centered on the pixel in question.

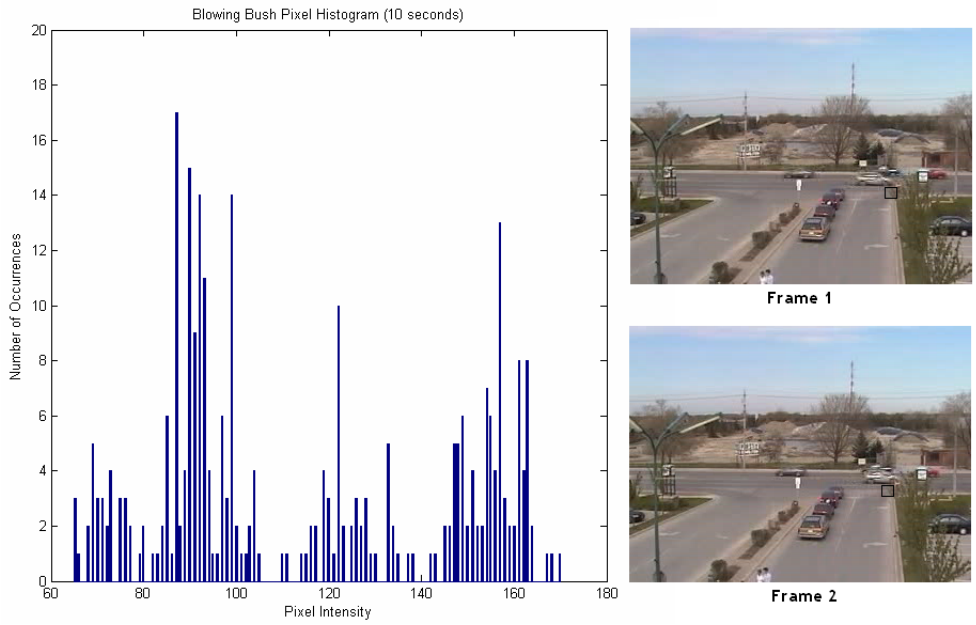

Figure 2.2: Histogram of Bush Blowing in the Wind: The small black box in Frame 1 and Frame 2 is centers on the pixel that the histogram describes. The histogram shows the pixel values present over a 10 second period of video.

To deal with the issue of multiple elements of background being present at a particular pixel, some have suggested the use of Gaussian mixture models [50]. The remainder of this section is the formulation of a pixel-based background subtraction model based on assigning a Gaussian mixture model to each pixel in the scene. Under the assumption that the camera view is non-translating and non-rotating, the background model parameters can be estimated using a set of initial frames:

$$b_{i,j}(t) \sim \sum_{k=1}^{q_{i,j}(t)} \pi_{k,i,j}(t)\mathcal{N}(m_{k,i,j}(t), \sigma_{k,i,j}(t)), \tag{2.1}$$

subject to

$$\sum_{k=1}^{q_{i,j}(t)} \pi_{k,i,j}(t) = 1 \; \forall \; i, j, \tag{2.2}$$

where $b_{i,j}(t)$ represents the Gaussian mixture-based background model at time t, centered on pixel (i, j), $q_{i,j}(t)$ is the number of modes in the mixture model, $\pi_{k,i,j}(t)$ is the weight of mode k in the model for pixel (i, j), and $\mathcal{N}(m_{k,i,j}(t), \sigma_{k,i,j}(t))$ is the normalized pdf for background mode k, at pixel (i, j) and time t. The mean of the model is represented by $m_{k,i,j}(t)$ and the variance is represented by $\sigma_{k,i,j}(t)$. The combination of these parameters represents a robust way to model the background of a scene when there is slight variation in the elements which make up the background over time.

The number of modes, $q_{i,j}(t)$, in the mixture model can either be asserted by forcing $\pi_{q,i,j}(t)$ to be some value to effectively limit the number of modes, or alternatively, can be estimated from the data using a method such as competitive clustering [13].

The conditional probability density function for the observed background is denoted as follows:

$$p(f_{i,j}(t)|b_{i,j}(t)) = \sum_{k=1}^{q_{i,j}(t)} \pi_{k,i,j}(t)\mathcal{N}(f_{i,j}(t)|m_{k,i,j}(t), \sigma_{k,i,j}(t)), \tag{2.3}$$

where $f_{i,j}(t)$ represents a pixel from the video frame at time t. Background subtraction will be used to indicate the presence of background or foreground at a particular pixel. An experimentally determined threshold, T_b, is employed as follows:

$$\phi_{i,j}(t) = \left\{ \begin{array}{ll} \textit{background} & if \;\; \min_q \left(\frac{|f_{i,j}(t) - m_{q,i,j}|}{\sigma_{q,i,j}} \right) < T_b \\ \textit{foreground} & else \end{array} \right\}, \tag{2.4}$$

where $\phi_{i,j}(t)$ represents the discrete state of the pixel $f_{i,j}(t)$ in the image at time t, which describes whether the pixel is part of the foreground or background. Once a model describing the background has been asserted, there are two stages that must be considered to use the model: model initialization and model maintenance. Initialization is the process of estimating the mixture model parameters for the background when the algorithm is run the first time. Maintenance is used to ensure

that the model parameters that are selected in the initialization step remain valid as the scene evolves over time.

Initialization of the model can be done in a variety of ways. An example of a background initialization method is to use samples from a given number of frames at the start of the image sequence to estimate the model parameters. Investigation has shown that using a trimmed Gaussian [45] approach to background model estimation has proven to be effective as it is able to remove the effect of statistical outliers that arise from the presence of foreground objects in the model initialization step.

Maintenance is necessary to keep the background model up-to-date and to account for slow variations in scene conditions, such as lighting or the position of objects that are stationary for long periods of time. A method discussed in [52] for conducting model maintenance is to use the following update model:

$$b_{i,j}(t) = \left\{ \begin{array}{cc} (1-\alpha)b_{i,j}(t-1) + \alpha f_{i,j}(t) & for\ \phi_{i,j}(t) = background \\ b_{i,j}(t-1) & for\ \phi_{i,j}(t) = foreground \end{array} \right\}. \qquad (2.5)$$

This update equation does not account for changes in the number of modes in the model parameters. This could occur if, for example, the scene conditions change from still to windy, meaning that a given pixel close to a swaying bush would change from uni-modal to multi-modal. To deal with this case, one approach is to periodically, or continuously, re-initialize the background model.

2.3 Motion Estimation

In this section the idea of block-based motion estimation is discussed. Motion estimation is an important topic in tracking as it allows dynamic motion model parameters to be accurately estimated. The advantages and disadvantages of this method will be examined in this section. The purpose of the method is to obtain information about the motion present in a given image sequence. Directly measured process dynamics of an image can be included in the target tracking model.

Block-based motion estimation is discussed in the video compression literature [31]. The method is implemented such that a neighbourhood window of pixels in a given

image, centered on a specific pixel, is searched over a larger neighbourhood window of pixels in the previous image, centered on the same pixel. The method makes the assumption that from frame-to-frame in a video sequence pixel intensity values do not change and that the video source is stationary. Using some choice of difference metric, commonly a squared or absolute difference approach, the displacement which produces the minimum difference is deemed to be the motion vector for the center pixel. One downside to this difference-based approach is that accurate motion measurements from homogenous regions of an image are difficult to obtain.

To conduct this motion estimation using a difference measure, either pixel intensity can be used directly, as shown in (2.6), or a discrete notion of image segmentation, such as that produced by background-foreground subtraction thresholds, can be employed, as shown in (2.7). For the sake of this discussion, notation is presented for both image intensity and pixel state. Pixel velocities can be found by minimizing the following objectives between subsequent image frames, subject to the window displacement, d:

$$v_{i,j}(t) = \arg\min_{d} \sum_{w \epsilon W(i,j)} \|f_w(t) - f_{w+d}(t-1)\|^2, \tag{2.6}$$

$$v_{i,j}(t) = \arg\min_{d} \sum_{w \epsilon W(i,j)} \|\phi_w(t) \cup \phi_{w+d}(t-1)\|^2, \tag{2.7}$$

where $W(i,j)$ represents the neighbourhood search window over which to evaluate the difference between the search kernel centered on w and offset by d.

Executing block-based motion estimation on each pixel in the image can be computationally expensive since for each pixel in the image a search window must be traversed to determine the displacement with the minimum difference. This operation yields a run time of $O(n^2 \cdot m^2 \cdot k^2)$ where n^2 is the size of the image, m^2 is the size of the search window and k^2 is the size of the kernel window that is used in the template match. To reduce the run time, the algorithm is run on a subset of pixels in the image and the results are interpolated over the entire image. Using this approach leaves room to use larger search windows. A larger search window is advantageous to model more of the image structure, whereas a smaller search windows increases match ambiguity. A problem that arises from a window that is

too large is that too much structure is included, and if the structure deforms too much between images in sequence due to rotation and scaling an accurate match cannot be made.

2.4 Clustering Techniques

In this section the problem of grouping related data, commonly referred to as clustering, is examined. In the context of this thesis, clustering is used primarily in model initialization and image segmentation. An important property of any clustering algorithm, specifically those used in target segmentation, is that the algorithm produce consistent clusters between frames as the clusters tend to evolve organically. This is especially important in tracking applications, since the clusters found in target segmentation are used directly in tracking and must produce consistent parameterizations from frame to frame. Any clustering algorithm used for segmentation should also be robust to noisy data.

In this section, two parametric clustering techniques will be examined: K-Means and competitive clustering, as well as non-parametric K-Nearest Neighbor.

2.4.1 K-Means Clustering

The K-Means clustering algorithm [38], [44] is used to partition a set of samples into disjoint subsets, known as classes, which contain samples such that the sum-of-squares criterion (2.8) is minimized. The criterion is presented below:

$$J = \sum_{k=1}^{K} \sum_{n=1}^{N_k} (\hat{x}_{n,k} - \mu_k)^2, \tag{2.8}$$

where $\hat{x}_{n,k}$ represents one of N_k samples associated with class k, and μ_k is the mean value of the N_k samples contained in class k. The major advantage of this algorithm is the ease of implementation. K-Means requires that the number of classes, K, be known. This property of the algorithm makes K-Means ill-suited for clustering applications where K is not know a priori. Extensions to the algorithm, such as competitive clustering [22], can be applied which aid in evaluating K from the data. Competitive clustering is also sensitive to initialization conditions and as such does not always converge to stable classes, especially if data points are in close proximity or the number of classes is asserted incorrectly.

K-Means is implemented using a re-estimation procedure as follows. Initially, samples are assigned at random to the k classes and the mean of each class is computed. When the algorithm iterates, each sample is assigned to the cluster with the mean closest to that point based on the distance metric that is in use. A typical distance metric is Euclidian distance. With samples reassigned to classes, new means are calculated. The algorithm then iterates until a designated stopping criterion is met. A typical stopping criterion occurs when there is no further change in the assignment of samples to clusters, meaning that the algorithm must be iterated a minimum of two times.

2.4.2 Competitive Clustering

In this section, competitive clustering is discussed. Competitive clustering is a clustering method based in part on concepts from fuzzy logic [49] and in part on concepts from information theory [38]. The general principal of competitive clustering involves iteratively minimizing the following objective function, subject to a set of sample points denoted $X = \{x_1, ..x_n\}$:

$$J = J_1 + \alpha J_2, \tag{2.9}$$

where the term J_1 is at a minimum when there are as many clusters as there are points in X, and the term J_2 is at a minimum when all the samples in X are contained in one cluster [22]. The term α controls the relative impact of each term on the overall objective function. The value of α is initially set large to encourage the formation of small clusters and is allowed to decay in subsequent iterations to encourage clusters to merge with one another. At each iteration, points in X are assigned to a cluster in a similar fashion to K-Means clustering. At each iteration, empty clusters are removed from the set of clusters. This process is repeated until the algorithm converges based on a stopping criterion.

k-Means Clustering Algorithm
Assign $X = \{x_1, x_2, ..., x_N\}$ randomly to $C = \{c_1, c_2, ..., c_K\}$
$t = 0$
Do
$\quad \mu_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \hat{x}_{n,k}$
$\quad$ Assign x_n to c_k if $\arg\min_k \lvert x_n - \mu_k \rvert$
$\quad t = t + 1$
Loop Until $C_t = C_{t-1}$

Many variations to the general form of the objective function presented above exist in the literature. An example of a variation can be found in [13]. Some variations present more robust implementations, but each involves minimizing the objective function with respect to a cluster parameter vector $\Theta = \{\theta_1, ..., \theta_K\}$, where K represents the number of clusters. The basic implementation of competitive clustering follows:

$$J = \sum_{k=1}^{K} \sum_{n=1}^{N} u_{k,n}^2 d_{k,n}^2 - \alpha \sum_{k=1}^{K} \left[\sum_{n=1}^{N} u_{k,n} \right]^2, \tag{2.10}$$

subject to

$$\sum_{k=1}^{K} u_{k,n} = 1, \ for \ n \ \epsilon \{1, 2, ..., N\}, \tag{2.11}$$

where N is the total number of samples, $u_{k,n}$ represents the degree of membership of sample point x_n in prototype θ_k, and $d_{k,n}$ is the distance between the point x_n and the prototype θ_k. Variations of this form include objective functions which account for the stochastic degree of membership, $p(d_{k,n})$, of point x_n in cluster θ_k in the first term, and a measure of the fuzzy *typicality*, denoted $f(u_{k,n})$ in the second term [22]. Figure 2.3 illustrates the first four iterations in a competitive clustering execution.

This algorithm functions well to find clusters under noisy data and is computationally efficient. The major downside of this algorithm is similar to that of K-Means, in that, due to the random nature of the initial conditions, the clusters produced by the algorithm are not stable when the algorithm is executed repeatedly on the same data set, nor are the results stable between images when slight variations in the features of an image occur. This is due to the fact that the algorithm converges to local minima based on the initial conditions. This makes the algorithm ill-suited for tracking applications, where consistent target segmentation is critical for accurate tracking.

2.4.3 Nearest Neighbour

The nearest neighbour algorithm [34] is known as a non-parametric algorithm since samples are used directly in the class definition and sample membership assignment. This is contrary to model based methods that fit predefined model parameters to the samples. K-Nearest Neighbour [38] is a variation on the Nearest Neighbour

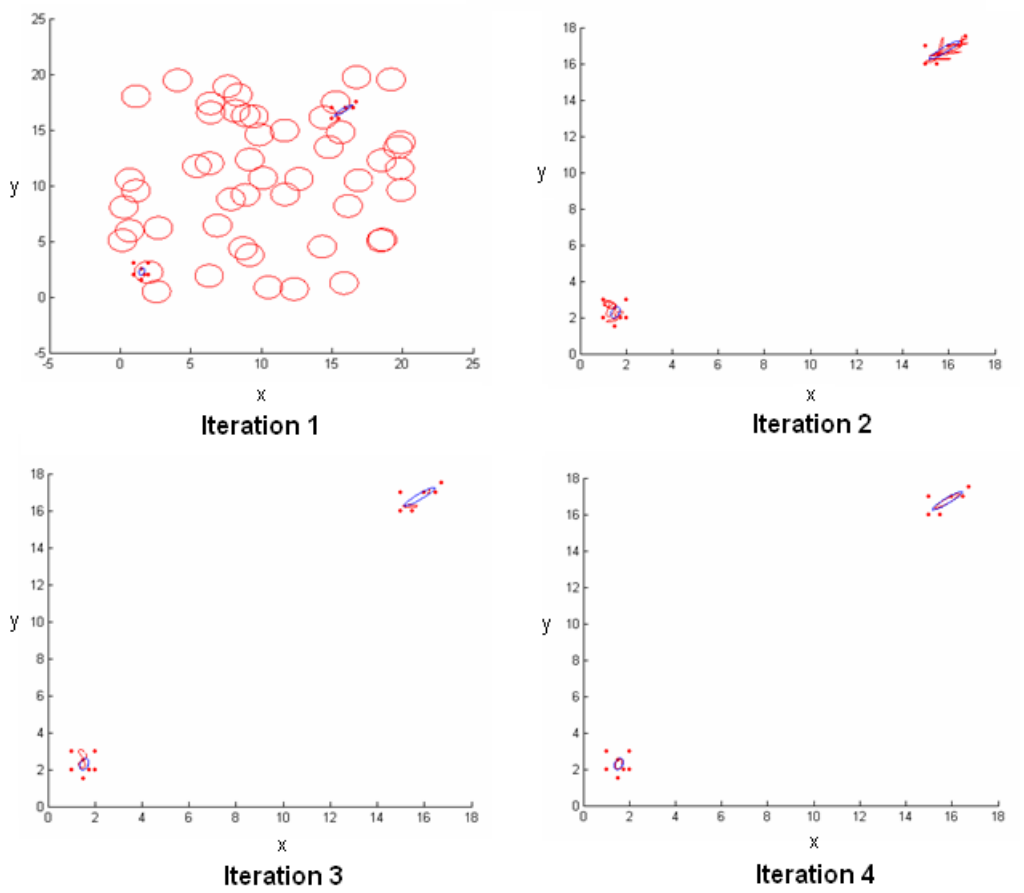

Figure 2.3: Competitive Clustering Results: Initial and Final Iteration of Competitive Clustering. The first step (top-left) shows the large number of randomly seeded clusters that is initially considered, and the final step (bottom-right) shows the outcome of iterating the algorithm on the data until the convergence criterion is met.

algorithm, where the k^{th} nearest point is used to conduct class assignment. This approach is more robust to noise as long as K is chosen to be sufficiently large and there are sufficient data points in the class to support the choice of a large K.

Nearest Neighbour Clustering Algorithm
$n = 1$
$k = 1$
$d_{min} = D_{MIN}$
Assign x_n to c_k
Do
Select $x_{n+1} = \arg\min_j \lvert x_j \epsilon X - x_n \rvert$
If $x_{n+1} - x_n > d_{min}$ Then $k = k + 1$
Assign x_{n+1} to c_k
$n = n + 1$
Loop Until $n = N$

The nearest neighbour clustering algorithm is fairly straightforward to implement and converges rapidly. The algorithm works by iterating though the data and performing the following steps: Assign data point x_i to cluster C_k where $k = 1$. Find the data point, $x_j \epsilon X$ that is closest to x_i. If $dist(x_i, x_j) \leq d_{min}$, then assign x_j to C_k. d_{min} is the heuristic minimum distance to consider two data points in the same cluster. In the case where $dist(x_i, x_j) > d_{min}$ a new class is created, by setting $k = k + 1$, and point x_j is assigned to that class. The algorithm is then iterated until all sample points have been visited and the algorithm converges.

One advantage of this algorithm is that it can be implemented to perform in real-time and is robust to noisy data. One drawback of this algorithm is that it requires a heuristic minimum distant threshold to define the connectivity of a sample with a class. In practical applications, this threshold may change as a function a position in the scene, or as a function of time. This means global distance thresholds can be ineffective.

2.5 Connected Components Clustering

Connected components clustering constructs an undirected graph by labeling the pixels in an image based on pixel *connectivity* [27]. Different measures of connectivity can be defined; however, the two most common structuring elements used

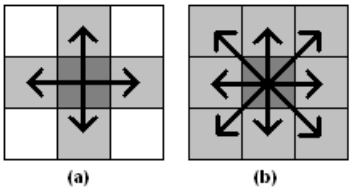

Figure 2.4: Connected Components Clustering: (a) Illustrates four pixel connectivity and (b), shows eight pixel connectivity.

to measure connectivity are four-pixel connectivity and eight-pixel connectivity. Figure 2.4 shows the two types of structuring elements.

Two pixels are said to be part of the same connected component cluster if there exists a path between them. Allowable paths are defined by the structuring elements. Typically, connected component labeling is performed on binary images, although measures of connectivity involving grey scale and colour images are possible. In the binary case, the connected components labeling algorithm traverses a row in an image until a pixel is found where $f_{i,j} = 1$. When this is found the neighbours defined by the structuring element are evaluated. If all neighbors are 0, a new label $p_{i,j}$ is assigned to the pixel. If pixel (i, j) only has one neighbor where $f_{i',j'} = 1$, assign $p_{i,j} = p_{i',j'}$. Finally, if more than one neighbor is equal to 1, we assign one of the labels to $p_{i,j}$ and make note of the equivalences of the connected labels.

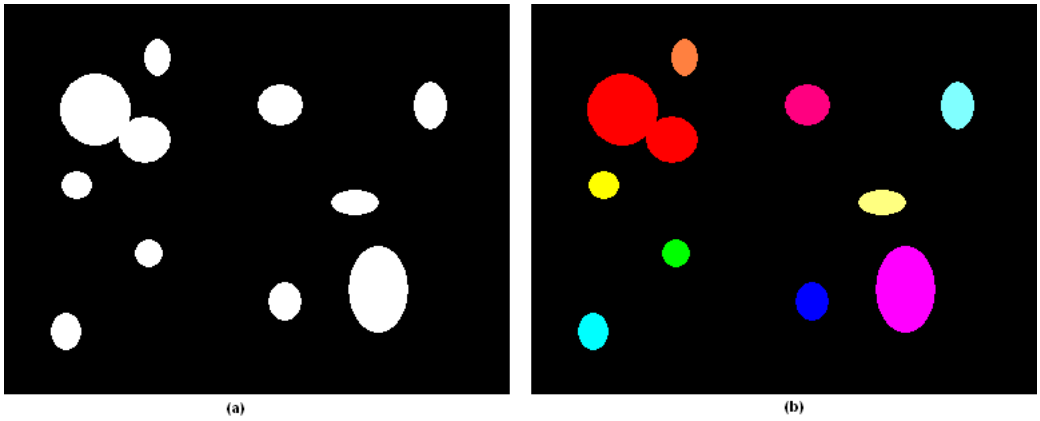

Figure 2.5: Connected Components Clustering Result: Result (a) Shows a binary image mask, and (b) shows the results of connected components clustering.

After completing the algorithm, the equivalent labels are sorted into clusters and a unique label is assigned to each. Figure 2.5 illustrates the results of running the connected components clustering algorithm. The different clusters are indicated with different colours.

2.6 Mathematical Morphology

Mathematical morphology operators are derived from set theory. Morphology is used in the analysis of binary and grey-level images and common usages include edge detection, noise removal, image enhancement and image segmentation [40].

The two most basic operations in mathematical morphology are erosion (2.12) and dilation (2.13). Both operators take two inputs: an image to be eroded or dilated, denoted A, and a structuring element, denoted B,

$$A \oplus B = \{d | B_d \cap A \neq \emptyset\}, \tag{2.12}$$

$$A \ominus B = \{d | B_d \subset A\}, \tag{2.13}$$

where d is the offset of structuring element. Binary morphology can be considered a special case of gray-level morphology, where the image only has two possible values. For a gray-scale image, structuring elements are applied directly to pixel intensity values. For binary images, $f_{i,j} = 1$ is usually taken to represent foreground, while $f_{i,j} = 0$ usually denotes background. Morphological structuring elements can be defined arbitrarily and can be considered as a set of point coordinates. Typically, the point coordinates are centered on the origin.

Erosion and dilation morphology operations work by applying the structuring element, B, at all points in the input image, A, where $f_{i,j} = 1$, and examining the intersection between the translated point coordinates of B and the coordinates of A. For instance, in the case of dilation morphology, the resulting image will consist of a new set of pixels comprised of the union of the structuring element and the input image.

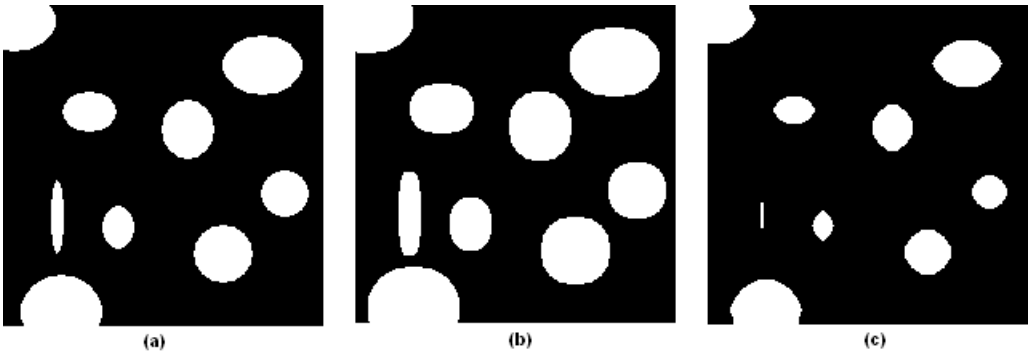

Figure 2.6: Result (a) shows the original binary image image (200x200 pixels), (b) shows the results of applying a dilation operation with a 7x7 pixel structuring element and (c) shows the results of applying a erosion operation with a 7x7 pixel structuring element.

2.7 Density Propagation

In this section, the basic ideas behind particle filtering and Kalman filtering are discussed. The discussion is broken into two sections: a discussion about the Kalman filter and a discussion about particle filtering based on conditional density propagation. In this work, Kalman filtering, discussed in Section 2.7.1, is used to estimate the state of a given track over time; however, this could also be accomplished using conditional density propagation, discussed in Section 2.7.2.

2.7.1 The Kalman Filter

The Kalman filter is a recursive solution to the discrete-time linear data filtering problem [47]. It addresses the problem of trying to estimate the state of the dynamic discrete-time process, x_t, governed by the linear stochastic equation:

$$x_t = Ax_{t-1} + Bu_{t-1} + w_t, \tag{2.14}$$

where x_{t-1} describes the state of the target at the previous discrete time, u_{t-1} describes an input or control and w is the process noise. In addition, a measurement process $\hat{x}_t$ is denoted:

$$\hat{x}_t = Hx_t + v_t, \tag{2.15}$$

where v is the measurement noise.

The basic premise behind the Kalman filter is that at each discrete-time step a *predict* and *update* process is employed. The predict step is a result of the known dynamics of the system, whereas the update step is a result of the available measurements from that step. The following set of equations describes the prediction step of the Kalman filter:

$$x_t^{'} = Ax_{t-1} + Bu_{t-1} \tag{2.16}$$

$$P_t^{'} = AP_{t-1}A^T + Q \tag{2.17}$$

where Q is the covariance of the process noise w and P_t is the state covariance of x_t. The next set of equations illustrates the second stage, update, of the Kalman filter:

$$K_t = P_t^{'}H^T(HP_t^{'}H^T + R)^{-1} \tag{2.18}$$

$$x_t = x_t^{'} + K_t(\hat{x}_t - Hx_t^{'}) \tag{2.19}$$

$$P_t = (1 - K_tH)P_t^{'} \tag{2.20}$$

where K_t is known as the Kalman gain. The recursive nature of the Kalman filter makes it a much more attractive, feasible implementation of linear discrete state estimation than the Wiener filter; however, like the Wiener filter, the basic Kalman filter assumes the state variable obeys linear dynamics and is Gaussian. This assumption may not always represent the underlying truth of an evolving state, especially in real-world environments. The extended Kalman filter [39] aims to address the issue of non-linear dynamics by first linearizing the problem.

2.7.2 Conditional Density Propagation

Another approach to state propagation is particle filtering [2]. Particle filtering is a more general approach to state estimation and is based on conditional density propagation. In this approach, the assumption about linear process dynamics and Gaussian distributed state variables is not part of the particle filtering formulation. The discussion that follows describes how particle filters work and contrasts the method to the optimal Kalman filter.

The basic premise behind particle filtering and more specifically, the conditional density propagation approach, is Bayesian recursion. Following the notation presented for the Kalman filter, Bayesian recursion can be denoted as follows:

$$p(x_t|\hat{X}_t) = \frac{p(\hat{x}_t|x_t, \hat{X}_{t-1})p(x_t|\hat{X}_{t-1})}{p(\hat{x}_t|\hat{X}_{t-1})}, \tag{2.21}$$

where x_t is the state of the target at time t, $\hat{X}_t$ represents all observations of x_t over all time and $\hat{x}_t$ is the estimate of the state of the target at time t. As with the Kalman filter, conditional density propagation can be implemented as an approximation to Bayesian recursion in a two step process, the first involving a prediction step based on the process dynamics of the system, and the second involving an update step which works by conditioning the model based on the observational density. These two concepts are presented and discussed below. The main difference between the particle filtering method that is described and the Kalman filter is that the particle filter represents a time-dynamic stochastic process with a set of discrete particles, $\Upsilon_t = \{\upsilon_{t,1}, .., \upsilon_{t,N_\Upsilon}\}$. Υ_t tends toward the distribution of x_t as $N_\Upsilon \to \infty$, where N_Υ represents the number of particles in the set.

Process Dynamics: The process dynamic step in a conditional density propagation implementation is meant to shift existing particles in Υ_t to a new state. In this formulation, the process dynamics of the scene do not necessarily have to be linear. The process dynamic step of conditional density propagation can be mathematically denoted as follows [30]:

$$p(x_t|\hat{X}_{t-1}) = \int_{t-1}^{t} p(x_t|x_{t-1})p(x_{t-1}|\hat{X}_{t-1})dx \tag{2.22}$$

The above notation can be considered as any time-dependent perturbation of a stochastic system. In the case of a tracked target this perturbation is due to the target's physical dynamics.

Observational Density: Once the particles in Υ_{t-1} have been predicted forward they are denoted Υ'_{t-1}. At this point the observational density step is conducted. This step is meant to condition Υ_{t-1} based on the actual measured state, $\hat{x}_t$. A variety of approaches for this conditioning are discussed in the literature. The approaches vary depending on the particles used to represent the state that is being propagated; however, if the particles which make up the state can be considered as Gaussian, a common approach is to condition each particle based on a notion of

distance between the particle and the nearest measurement. Typically, a minimum probability is set to account for cases of missing measurements. This minimum probability approached is accomplished by truncating the probability measured from the Gaussian distribution at an arbitrary minimum threshold, as illustrated in Figure 2.7.

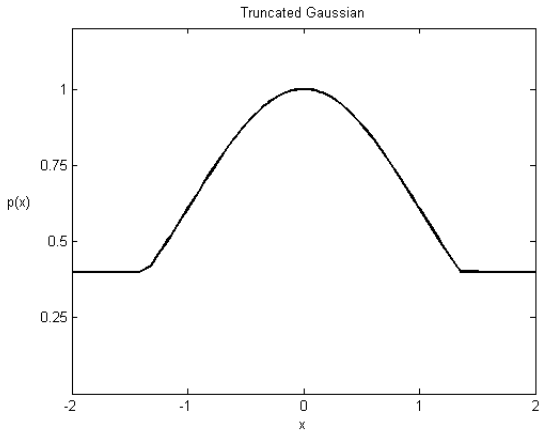

Figure 2.7: Truncated Gaussian used in Observational Density

Once each particle in Υ'_{t-1} has been conditioned based on the measurements, a set $\Gamma_t = \{\gamma_{t,1}, .., \gamma_{t,N_\Upsilon}\}$ results. This set represents the conditional probability of each particle in Υ'_{t-1} and is used to re-sample Υ_t. This results in particles with strong likelihood to be more likely to survive re-sampling stage, while particles with weaker likelihood are less likely to be propagated. The minimum probability, illustrated in Figure 2.7, ensures that when the particles are re-sampled, particles that are missing corresponding measurements still have a chance to propagate. The conditional density step can be mathematically approximated under the Markov assumption as follows:

$$p(x_t|\hat{X}_t) \approx p(\hat{x}_t|x_t)p(x_t|\hat{X}_{t-1}) \tag{2.23}$$

A possible extension to conditional density propagation involved having new particles flow into the recursive implementation. In order to facilitate this, $N_\Upsilon - M_\Upsilon$ particles would be re-sampled in the observational density step, where N_Υ is the total number of particles allowed to propagate, and M_Υ is the number of new particles injected into the model. This is a useful strategy when it is desired to have new model characteristic flow into the density model. An example of this would be

if a new target appeared onto a scene and it was desired to add that target to the scene model for tracking.

2.8 Probabilistic Data Association

Data association techniques are of great importance in tracking applications. A variety of methods are discussed in this section. The two classes of data association are sample based and model based. An example of a sample based filter is the Nearest Neighbour Standard Filter [48]. An example of a model based filter is the Probabilistic Data Association (PDA) filter [7]. Evaluating PDA involves assigning a probability that a particular sample can be associated with a particular target. This is accomplished by relating a sample to a prior model which describes a given target based on some distance metric.

An extension of the PDA filter, called the Joint Probabilistic Data Association (JPDA) filter [51], allows for observed samples to be considered jointly for association with multiple targets. One of the major advantages of the JPDA is its ability to track targets through occlusion and clutter due to its ability to simultaneously represent all possible combinations of measurements. It is also able to handle targets which exist in close proximity to one another. JPDA will be discussed in detail in this section. A major problem of JPDA is that it requires considerable computation to compute all possible associations. Methods have been proposed which limit the computational complexity of evaluating the joint association. One such method is called gating [17], which is illustrated in Figure 2.8.

The primary role of gating in data association problems is to reduce the overall computational complexity. This is accomplished by drastically reducing the number of measurements which must be considered for association with a particular class. Gating is conducted by demanding that a measurement must fall within a minimum distance of a given target. The example points x and y, shown in Figure 2.8 would be associated against class A, and points y and z would be evaluated against class B. In the case where classes are well separated in feature space, gating can greatly reduce the required computation.

Since a given measurement is gated based on its distance to a target, an appropriate distance measure must be defined. If it is assumed that the probability density function of target can be represented as a Gaussian, then the distance can be

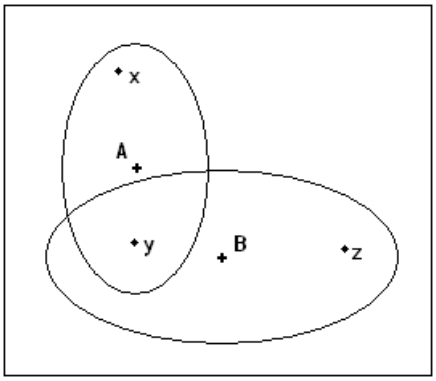

Figure 2.8: The points x and y would be associated against class A and points y and z would be evaluated against class B.

computed as the Mahalanobis distance between measurement and target. The Mahalanobis distance [38] can be expressed as:

$$d_{i,j} = (z_i - \mu_j)\Sigma_j^{-1}(z_i - \mu_j)^T \tag{2.24}$$

where $d_{i,j}$ is the distance between the sample z_i and the j^{th} target and μ_j and Σ_j are the mean and covariance of class j. In order for a measurement to be considered in the association problem of target j, the distance must be less than the gate threshold, denoted as T_g.

One of the shortcomings of using traditional JPDA is that its computation complexity increases exponentially as either the number of targets or number of measurements increases. This increased computational complexity is greatly mitigated by the introduction of gating and under the assumption that targets are well separated in feature space. The literature also discusses methods to further reduce the computation load of JPDA, for example by constructing *feasibility* matrices to limit the number of evaluations that are required [18].

Another way to reduce the computational complexity of evaluating the likelihood of the various feasible solutions would be to employ notions from dynamic programming [4] and graph theory [9]. It is possible to use dynamic programming because the probabilities of the composite events which make up the joint associative probabilities, are independent of their membership in a particular configuration of

measurement. As such, the probability of each composite configuration of measurements must only be evaluated once, regardless of the number of times it appears. This can be accomplished using a tree or graph structured implementation.

2.9 Conclusion

In this chapter, various topics that represent different aspects of common, end-to-end tracking methods have been discussed. The algorithms that were covered included foreground identification methods, use to isolate objects in a scene, motion estimation methods, used to measure the dynamic aspect of an object in a scene, clustering techniques that can be used to combine measured information about objects in a scene, density propagation methods that can be used to propagate the state of an object through time, and data association methods than can be used to uniquely identify persistent objects in a scene.

Many of the methods that were discussed in this chapter provide the basis for the work presented in the remainder of this work. The material that was covered, provides an overview of the fundamental mathematical concepts required to understand the next chapter. In the next chapter, a framework is proposed for represented objects in a scene using a graph-based structure. A Monte Carlo Markov Chain method for tracking is also developed that take advantage of this graph structure, as well as many of the concepts presented in this chapter. The tracking method also combines additional feature extraction and motion estimation methods.

Chapter 3

Target Tracking Algorithm

The objective of this chapter is to present the proposed target tracking algorithm in some detail. A high level system model of the algorithm is initially presented which aims to provide context to the remainder of the chapter. Discussion about various features and methods of extraction that will be employed in the tracking algorithm follows. The tracking framework that is described in this chapter has been designed to function in environments containing an unknown number of target initializations and terminations under conditions of occlusion and clutter.

This chapter is broken into six sections. Section 3.1 provides an overview and discussion about the tracking algorithm that has been developed in this thesis. Section 3.2 presents notation that is used to discuss the remainder of the chapter. Section 3.3 presents a method for identifying features and extracting information about them. Section 3.4 discusses a method for estimating the motion of the identified features. Section 3.5 deals with segmenting targets, Section 3.6 develops a target tracking model and suitable scoring function and Section 3.7 discusses how to identify target tracks using data association.

3.1 Tracking Algorithm Overview

In this section a target tracking algorithm is developed to address the following problem statement: It is desired to develop a multi-target tracking method for determining the paths that vehicles take through an arbitrary transportation scenes. In the scenes, the number of vehicles is not known a priori and is subject to change over time. For this application, it can be assumed that the transportation scene in

question comes from a non-translating, non-rotating camera source. In addition, vehicles in the scene can vary in size, speed of travel and orientation relative to the camera.

The method that is developed in this chapter extracts the position of targets from images in sequence, estimates the motion of these targets and finally combines the located targets into tracks. The algorithm that is developed has particular application to vehicle tracking but can be extended to other object tracking problems. Figure 3.1 illustrates the system diagram for the algorithm. The algorithm that is developed employs a method of scale-space extrema [25] localization and Delaunay clustering [35] to achieve target segmentation. Motion estimation is achieved using a hierarchical motion estimation approach. Target-track association is accomplished using a MCMCDA [46] approach.

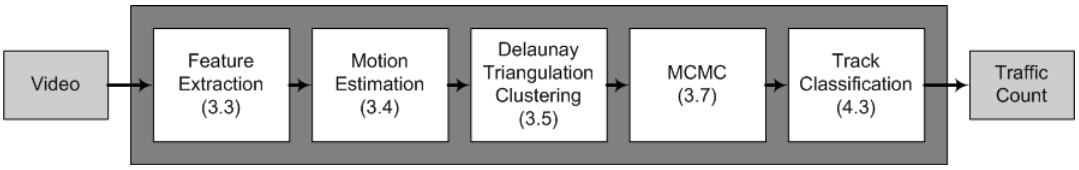

Figure 3.1: Overview of Proposed Tracking Algorithm. Section numbers are indicated inside the brackets.

Feature extraction is used to extract information from a sequence of images that can be used for tracking. In this work, we address feature extraction by first constructing a scale space for the given image. We then identify scale-space extremum points from the constructed scale space and use background subtraction to identify which extremum from the image belong to foreground targets of interest. Background subtraction is possible since the video sequence is known to be from a static (non-translating, non-rotating) camera source. This work also employs feature based motion estimation on scale space extrema points which have been identified as belonging to foreground. This combined motion history, background subtraction and feature information provides the basis for target segmentation.

Upon completion of the feature extraction step, the scale space extrema points are associated with one another into targets. These clusters represent the targets we desire to track – in this case, vehicles. The clustering step presents a variety of challenges. First, the number of clusters is not known for a given frame in the image sequence, nor is the number of clusters necessarily a function of the number

of clusters present in the previous frame. Additionally, clusters can appear and disappear due to occlusion, target creation, target termination and false alarms. Due to the fact that vehicles can vary in shape, size and aspect relative to camera position, the nature of the clusters in a given image is also unknown. Clusters do not necessary adhere to a particular size or shape. Finally, clusters can exist in close proximity to one another and can split and merge from frame to frame.

A good clustering algorithm for this application should be able to identify an unknown number of clusters, of unknown shape and size which exist in close proximity to one another. It is also desirable to have this algorithm operate on low information content, and to be robust to clutter in an image. The algorithm should also produce stable clusters when executed repeatedly on a single image and should converge to similar clusters between images in sequence if information in the scene evolves organically – that is they change only slightly in terms of size, shape and orientation between frames.

The final step in the tracking algorithm is to determine the various tracks that a particular object takes through a scene. This step, known as the data association step, must be able to ascertain a single track that is indicative of each of the targets from the multiple measurements that have been taken over time in the scene. A well-suited algorithm for the identification of vehicle tracks will be able to perform effectively under conditions of strong occlusion, in which case measurements may be missing from the track, as well as under conditions of clutter, meaning there will be environmental and camera noise present in the scene. Given the application domain, the algorithm should also be able to function in as near real-time as possible. Another advantageous property of the tracking algorithm is that it should be stochastically based. This will allow for multiple hypotheses to be stored about a given track, increasing the probability that over time the correct vehicle tracks will be identified.

While developing this tracking algorithm, two criteria were used to guide the choice of algorithmic approach and implementation strategy. These were real time performance and the ability to implement the algorithms in parallel. Although these two topics will not be a considerable focus of this thesis they are important factors to consider in the event the algorithm is used in a commercial application.

3.2 Tracking Algorithm Notation

In this section, the mathematical notation used throughout the remainder of this chapter is presented. The notation is meant to provide the foundation upon which the tracking algorithm is based. A solid understanding of this notation and what it is aiming to represent is therefore vital to a firm understanding of the remaining aspects of this work.

3.2.1 Image Sequence

The image sequence $\mathcal{F}_T$, contains a set of images:

$$\mathcal{F}_T = \{\mathbf{F}(0), \mathbf{F}(1), ..., \mathbf{F}(T)\}, \tag{3.1}$$

where $\mathbf{F}(t)$ denotes the image at time $t \in \{0, 1, .., T\}$. A specific pixel in $\mathbf{F}(t)$ can be defined as $f_{i,j}(t)$. The term $f_{i,j}(t)$ is a vector, representing the various colour channels that exist for a given pixel. Under the assumption that the sequence of frames is in the RGB colour space then:

$$f_{i,j}(t) = \begin{bmatrix} r_{i,j}(t) \\ g_{i,j}(t) \\ b_{i,j}(t) \end{bmatrix}. \tag{3.2}$$

This notation could be extended to include a three-dimensional representation of an image. Three-dimensional image modeling would reduce the potential for occlusions to be present in the scene. This type of image representation would be denoted by $f_{i,j,k}(t)$; however, for the remainder of this work two-dimensional image notation will be employed. For the purposes of this work certain assertions about $\mathcal{F}_T$ will be made, although these assertions are not explicitly required. The main assertion is that $\mathcal{F}_T$ is derived from a non-translating, non-rotating image sequence (i.e. the camera generating the image sequence is stationary). In addition, the size of the objects being tracked are smaller than the entire field of the image sequence is asserted. Finally, the assertion is made that the background present in the image sequence changes slowly over time. The preceding assertions are all indicative of an outdoor scene being filmed from a stationary camera at a distance, such as those found permanently mounted at intersections.

To facilitate discussion about the implementation of this algorithm, a corresponding measurement set is also defined:

$$\hat{\mathcal{F}}_T = \left\{ \hat{\mathbf{F}}(0), \hat{\mathbf{F}}(1), ..., \hat{\mathbf{F}}(T) \right\}, \tag{3.3}$$

where $\hat{\mathbf{F}}(t) = \mathbf{F}(t) + \mathbf{W}_F(t)$, $\mathbf{W}_F(t)$ being the measurement noise. Similarly, there exists a pixel based measurement representation, defined as $\hat{f}_{i,j}(t) = f_{i,j}(t) + w_{i,j}(t)$

3.2.2 Image Feature Graph

To greatly reduce the computational requirements of this implementation, a change of basis is proposed which abstracts information content found at the pixel scale into features, known from this point on as *information regions*. To accomplish this change of basis the information in $\mathcal{F}_T$ is clustered and measured. Techniques for this clustering will be discussed in Section 3.5.

Information regions can be thought of as components of the targets being tracked, meaning a set of these information regions constitutes a target. A time-series of the sets of information regions is defined as

$$\mathcal{S}_T = \{\mathbf{S}(0), \mathbf{S}(1), ..., \mathbf{S}(T)\}. \tag{3.4}$$

As with the pixel based notation, a corresponding measurement set $\hat{\mathcal{S}}_T$ is also defined. At every discrete time step of the tracking algorithm, the information regions can be denoted

$$\mathbf{S}(t) = \{s_1, s_2, ..., s_{N_{S(t)}}\}, \tag{3.5}$$

where the information region is distributed as $s_k \sim \mathcal{N}(\mu_k(t), \Sigma_k(t))$ and N_S is the number of information regions in the set. The time index is not included unless ambiguity exists, in which case the sub-targets are denoted as $s_k(t)$. Corresponding measurement notation is also defined as $\hat{\mathbf{S}}(t) = \{\hat{s}_1, \hat{s}_2, ..., \hat{s}_{N_{S(t)}}\}$.

In addition to the set describing the information regions, a set is introduced which describes the associations between information regions within a given image:

$$\mathbf{\Psi}(t) = \{\psi_{s_i(t),s_j(t)}\}, \tag{3.6}$$

where $\psi_{s_i(t),s_j(t)} = p(|s_i(t) - s_j(t)|)$, $i, j = 1, .., N_{S(t)}$ and $i \neq j$.

3.2.3 Targets

Given the set of information regions $\mathcal{S}_T$, there exists a set of targets $\mathcal{R}_T$ (i.e. vehicles),

$$\mathcal{R}_T = \{\mathbf{R}(0), \mathbf{R}(1), \ldots, \mathbf{R}(T)\}, \tag{3.7}$$

where the set $\mathbf{R}(t)$ represents the targets that are present at time t, and can be expressed

$$\mathbf{R}(t) = \{r_1, r_2, \ldots, r_{N_{R(t)}}\}, \tag{3.8}$$

where r_k is the state vector that describes a given target and $N_{R(t)}$, denotes the total number of targets present at time $t \, \epsilon \, \{0, 1, \ldots, T\}$. The connection between $\mathcal{S}_T$ and $\mathcal{R}_T$ can be thought of as $\{s_1 \cup s_2 \cup .. \cup s_{N_{S(t)}}\} = \{r_1 \cup r_2 \cup .. \cup r_{N_{R(t)}}\}$.

3.2.4 Tracks

Given the set of targets, $\mathcal{R}_T$, there exists a set of target tracks Ω_T (e.g. vehicle trajectory over time) that describes all target tracks for $t = 0, 1, \ldots, T$:

$$\Omega_T = \{\tau_1, \tau_2, \ldots, \tau_{N_\tau}\}, \tag{3.9}$$

where N_τ denotes the total number of target tracks over all time. Each target in $\mathcal{R}_T$ arrives on the scene at time t_a and exits the scene at time t_b; therefore, track i can be expressed in terms of targets as

$$\tau_i = \{r_j(t_a), r_j(t_{a+1}), \ldots, r_j(t_b)\}, \tag{3.10}$$

where $t_a \leq t_b \leq T$. $N_{a,t}$ is the number of new targets at time t, $N_{z,t}$ is the number of terminated targets, $N_{d,t}$ is the number of detected targets and $N_{f,t}$ is the number of falsely identified targets.

3.2.5 Notation Overview

The notation that has been defined in this section defines a structure that can be used to construct targets and tracks. Figure 3.2 illustrates the structure that has been defined in this section.

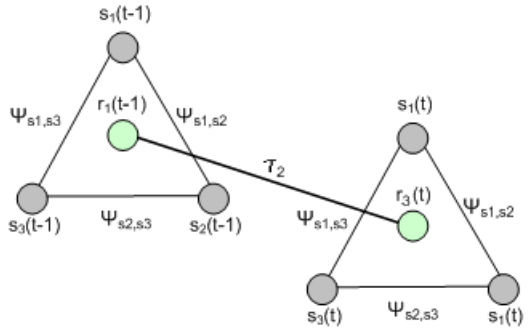

Figure 3.2: Target-Track Structure: The structure illustrated is meant to show how all the terms defined in Section 3.2 relate to one another.

3.3 Scale Space Feature Extraction

To implement a successful vehicle tracking algorithm, the ability to produce stable, consistent image features in successive frames of an image sequence is of paramount importance. In essence, the objective of this step of the algorithm is to segment a given image in an image sequence and represent those segments stochastically. Put succinctly, an appropriate method for image feature localization must have the following two properties: (1) the method must be able to consistently parameterize features and (2) the method must generate parameters that are stable and robust to slight variation in scale, rotation and aspect. Throughout the investigative efforts conducted as part of this work, a variety of methods for identifying stable image features were examined. These methods included K-Means clustering [44], expectation maximization (EM) [6] and competitive clustering [13].

Each of these methods presented unique challenges that made them unsuitable for use in the image feature extraction step of object tracking. K-Means clustering was initially examined as a means to construct information regions, $\mathcal{S}_T$. This method demanded that the number of clusters be known a priori. This method was abandoned early as an option, due to the fact that the number of image features is not known. EM and modified K-Means methods, such as competitive clustering, were also abandoned since these methods were unable to converge to stable clusters when run repeatedly on a single image, and moreover the methods were unable to converge to stable clusters when executed on successive images in a video sequence. The major reason for these shortcomings is that both EM and competitive clustering are data-oriented stochastic methods and the problem we wish to solve involves

identifying features which are often made up of only a small number of data points (i.e. pixels). This meant that the parameters produced by these methods were unstable.

Upon the realization that data oriented clustering methods would not yield consistent, stable image features it was decided to examine a method based on the identification of scale space extrema points. This concept is covered extensively in [24] and [25]. Scale space extrema points are capable of representing image features in a stable and consistent fashion. For the purpose of this work, the method will be used to localize image features $\mathcal{S}_T$, in the image sequence $\mathcal{F}_T$, which will then subsequently be used for target segmentation, $\mathcal{R}_T$.

To facilitate searching for scale space extrema points, a scale-space must be constructed. A scale space is constructed by applying a continuous function, $L(\sigma)$, to an image, which results in a structure that is capable of simultaneously representing features of an image across all scales. In order to construct a scale space in a computationally efficient manner, the scale space is discretized and a cascading filtering approach is employed to build a difference-of-Gaussian structure. In this application a cascading filter is accomplished by iteratively applying a Gaussian blur to an initial candidate image, based on a scale parameter σ. This difference of Gaussian structure is built by first convolving a Gaussian kernel with an input image, $\mathbf{F}(t)$, to produce

$$L(t, \sigma) = G(\sigma) * \mathbf{F}(t), \tag{3.11}$$

where σ is varied over all desired scales in the discretized space and $G(\sigma)$ is the two-dimensional Gaussian kernel function:

$$G(\sigma|(x, y)) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}, \tag{3.12}$$

where (x, y) are the image coordinates. The next step in identifying scale space extremum is to evaluate the difference-of-Gaussian, computed as the difference between adjacent scales of L. A scaling factor $k \epsilon \Re$ is introduced to describe the difference in standard deviation used between these scales. The difference of Gaussian is computed as:

$$D(t, \sigma) = L(t, k\sigma) - L(t, \sigma). \tag{3.13}$$

Once the set of difference-of-Gaussian images has been constructed, the second step in the identification of scale space extremum points is extremum localization. Scale-space extremum points are defined as points in space that either have a higher or lower value than all the points within a defined local radius for both the scale of the point and the scales directly above and below the point. Extremum points with a low scale space response can be removed from the set by evaluating the local image gradient that surrounds an extremum point. To reduce the computational complexity only points that are extremum in one scale are evaluated against corresponding points in scales above and below.

After an extremum point has been localized, and the points which do not meet a local gradient threshold are removed, the set $\mathbf{S}(t)$ is initialized from the remaining points. The set $\mathbf{S}(t)$ comprises $\{s_i\}$, constituting information regions. Each feature point s_i is made up of information about its position, local image intensity and process dynamics (motion gradient of the image feature). Initially, only the position and local image intensity information of the element are set, based on the identified scale-space extremum points and corresponding information.

3.4 Hierarchical Motion Estimation

Upon identification of image extremum points and subsequent construction of information regions, $\mathbf{S}(t)$, the next step in the proposed target tracking algorithm is to estimate the motion of the information regions. This motion information can be used to estimate the dynamics of the targets. Motion estimation is a very important topic in target tracking. The most common approach to motion estimation is block-based matching using a sum of absolute difference criterion:

$$SAD = \sum_{w \epsilon W(x,y)} |f_w(t) - f_{w+d}(t-1)|, \tag{3.14}$$

where $f_w(t)$ and $f_{w+d}(t-1)$ are regions of successive images in an image sequence, W is the area from which all search window are selected and (x, y) is the pixel index of the centroid of a given search window. In this work the use of a hierarchical approach to motion estimation is proposed. This approach can be thought of as an efficient approach to narrowing the search for the block which minimizes the sum of absolute difference criterion. A hierarchical approach to motion estimation is approached by first constructing two image pyramids for each of the two image

regions that it is desired to determine a matching region. Here, the image pyramid approach is especially attractive since the pyramid structure constructed previously for the identification of scale space extremum points can be reused.

The hierarchical motion estimation algorithm that was used for this work follows:

1) For a given s_i, in $\mathbf{S}(t)$, construct an image pyramid, Δ_i from the pixels that parameterize s_i. To accomplish this pyramid construction efficiently the same down sampling rate is used as the pyramid constructed for the identification of scale space extremum. This means the pyramid structure constructed in this step can be reused.

2) For a given s_i, construct a set of pyramids from the pixels centered on the feature point and found within a defined search window W_Λ, denoted $\Delta_{\Lambda,i}$. As with the previous step, the down sampling rate is selected such that previously constructed pyramids can be reused.

3) For the first iteration $m = N_\Delta$, where N_Δ is the number of levels in the image pyramid and m is the pyramid level index. $\overrightarrow{w}_{\Lambda,m}$ denotes where the minimum absolute difference search will be centred when the algorithm iterates to $m = m+1$, i.e.,

$$\overrightarrow{v}_{\Lambda,m+1} = \arg\min_d(\Delta_{i,m} - \Delta_{\Lambda,i+d,m}). \tag{3.15}$$

4) Step 3 is repeated until the bottom level of the pyramid, $m = 1$ is reached. The final result is denoted $\overrightarrow{v}_\Lambda$ and represents the position of the minimum difference between Δ_i and $\Delta_{\Lambda,i}$ at pyramid level $m = 1$.

Results are presented in Figure 3.3 which demonstration the effectiveness of the motion estimation approach in transportation images. Motion vectors are illustrated with red lines. The length of lines indicated the relative magnitude of the motion vector. One downside of the method is that targets that lack sufficient texture, or are too small relative to the image resolution, generate erroneous motion estimates.

Upon successful execution of this algorithm, the dynamics of $\mathbf{S}(t)$ are included in the information regions. The next step of the algorithm is to assemble the information regions into targets using the information about the position, colour and dynamics of the feature.

Figure 3.3: Results in (a) and (b) show two sample frames from a transportation scene with corresponding motion estimates.

3.5 Delaunay Triangulation Clustering

In this step of the tracking algorithm grouping information regions from $\mathbf{S}(t)$ in such a way as to create targets, denoted $\mathcal{R}(t)$, is desired. An ideal clustering algorithm must be able to create clusters when the number of such clusters is unknown. An ideal algorithm must also be able to produce clusters that are repeatable given a set of data. The algorithm should also be have the potential to run in realtime. In this work the use of a clustering algorithm based on Delaunay Triangulation is proposed.

Delaunay Triangulation is the dual of the Voronoi Diagram [3] which partitions a set of points in space into cells, one cell for each point. The cells are selected in such a way that the area contained in each cell is closer to the point contained in the cell than any other point in the space. In a Delaunay diagram the edges in the graph connect two points that have Voronoi cells sharing a common boundary. Under the assumption that information regions that make up targets are connected to one another, assume that feature points that are closest to one another are the only points that need be considered as part of the same object is reasonable. Figure 3.4 illustrates the resulting Delaunay graph constructed from $S(t)$.

A clustering algorithm based on Delaunay Triangulation is used [35] which is comprised of the following steps:

1) Construct the Delaunay graph out of the position components of $S(t)$.

2) For each vertex in the Delaunay graph, which can also be thought of as each information region in the image, the average length of the incident edges (i.e. the connections between graph vertices) is calculated, subject to

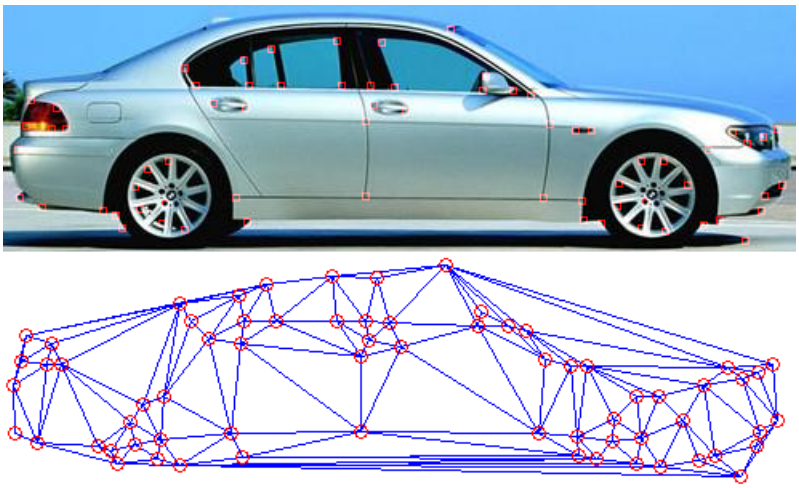

Figure 3.4: Delaunay Triangulation clustering of image features extracted.

$$\mu_{s_i} = \frac{1}{N_{s_i}} \sum_{j=1}^{N_{s_i}} |\psi_{s_i,s_j}|, \tag{3.16}$$

where N_{s_i} is the number of incident edges that exist at the point s_i and ψ_{s_i,s_j} is the association along the given edge.

3) Next the standard deviation of the incident edges is calculated subject to

$$\sigma_{s_i} = \sqrt{\frac{1}{N_{s_i}} \sum_{j=1}^{N_{s_i}} (\mu_{s_i} - |\psi_{s_i,s_j}|)^2}. \tag{3.17}$$

4) To consider both the local and global edge effects in the formulation, consider the global average standard deviation of the edge association,

$$\sigma_{\mathbf{S}} = \frac{1}{N_{\mathbf{S}}} \sum_{i=1}^{N_{\mathbf{S}}} \sigma_{s_i}. \tag{3.18}$$

5) The next step in the clustering algorithm is to remove all incident edges from the Delaunay graph which do not conform to the following criterion:

$$\psi_{s_i,s_j} > (\mu_{s_i} + \sigma_{\mathbf{S}}). \tag{3.19}$$

Once this removal step has been completed, the vertices that are still connected to one another along incident edges can be considered part of the same object.

6) The last step in the clustering algorithm is to compute the mean and covariance of all state values for all information regions in the graph that remain connected. The targets, denoted $R(t)$ are then filled with these mean values.

This data driven clustering algorithm was found to be effective at identifying targets in $S(t)$ when the targets are not in close proximity to one another. When targets do appear in close proximity, the method's effectiveness is diminished. This diminished performance can be improved by taking additional feature vector information into account when evaluating the distance along an incident edge. An example of an effective feature vector to include is the motion of the information regions in $S(t)$, since the information regions making up a target tend to have highly correlated motion.

3.6 Tracking Model

In this section, the problem of tracking an unknown number of targets is considered. Targets can appear and disappear at random and the scene can contain noisy measurements as well as an arbitrary number of target births, target terminations, false alarms and undetected targets. Each target's motion is defined by dynamic and measurement models:

$$r_t = A\Delta t r_{t-1} + \Delta t w_t, \tag{3.20}$$

$$\hat{r}_t = H r_t + v_t, \tag{3.21}$$

where ϖ_t and v_t are Gaussian noise with zero mean and where A is the state transition model, and H is the observation model which maps the state space into the observation space. Two versions of the motion model are presented. The first model would be appropriate for tracking moving vehicles since vehicles tend to travel in predictable straight line paths over a small number of frames. For the first model, a target's state can be described as

$$r_t = \begin{bmatrix} x_t \\ y_t \\ \delta x_t \\ \delta y_t \end{bmatrix}. \tag{3.22}$$

More complicated models which take into consideration the colour and structure of an object could also be considered. The motion model is as follows,

$$r_{t+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} r_t + \begin{bmatrix} \alpha\Delta t & 0 & 0 & 0 \\ 0 & \alpha\Delta t & 0 & 0 \\ 0 & 0 & \beta\Delta t & 0 \\ 0 & 0 & 0 & \beta\Delta t \end{bmatrix} w_t, \tag{3.23}$$

where α and β represent the noise components for position and velocity respectively. α and β can be selected based on the properties of the moving objects in a scene. Relatively higher values of α and β are used as the uncertainty about the dynamic behavior of a target increases. The second is a random walk model, which would be better suited to tracking pedestrians since pedestrians have a tendency to travel in unpredictable erratic paths as compared to vehicles. For this model a target's state can be described as,

$$r_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix}, \tag{3.24}$$

and the motion model is as follows,

$$r_{t+1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} r_t + \begin{bmatrix} \alpha\Delta t & 0 \\ 0 & \alpha\Delta t \end{bmatrix} w_t. \tag{3.25}$$

Now that the motion model for the targets has been defined, a formulation for target track prior that will be used for the scene is derived.

Assume that the scene given at time t is denoted by $\mathcal{E}_t$. $\mathcal{E}_t$ represents the joint probability distribution between all targets in $\mathcal{R}_t$ and tracks in Ω_t. This relationship can be mathematically denoted as

$$\mathcal{E}_t = \mathcal{R}_t \ominus \Omega_t. \tag{3.26}$$

In addition, assert that $\mathcal{E}_t$ can be subdivided into multiple regions, $\boldsymbol{R} = \{\mathbf{r}_1, \mathbf{r}_2, .., \mathbf{r}_{N_R}\}$. Each region, r in $\boldsymbol{R}$ is also assumed to be independent. In a scene comprised of a single camera, this subdivision can be accomplished by dividing a given image into multiple sections. This concept can also be extended to scenes constructed from multiple cameras views as well. Next, we define notation for a specific discrete partition that is selected from the joint association distribution, $\mathcal{E}_t$, which defines the track-target associations as ω. Given the independence assumption, it can be stated that:

$$p(\omega_t) = \prod_{\mathbf{r} \epsilon \boldsymbol{R}} p(\omega_{t,\mathbf{r}}). \tag{3.27}$$

Next, given that an arbitrary scene can be represented completely by knowing whether a given object point is detected or not, a false alarm or not, and either a birth, continuation or death, it can be asserted that the posterior for the scene can be represented as the product of the following probabilities: (1) the probability that $N_{f_{t,\mathbf{r}}}$ measurements from time t, region $\mathbf{r}$ and partition ω_t do not originate from targets, (2) the probability that $N_{d_{t,\mathbf{r}}}$ measurements do originate from detected targets (3) the probability that $N_{a_{t,\mathbf{r}}}$ new targets are identified at time t in region $\mathbf{r}$, and (4) the probability that $N_{z_{t,\mathbf{r}}}$ targets terminate. If we assume that all hypotheses about the values of $N_{f_{t,\mathbf{r}}}$, $N_{d_{t,\mathbf{r}}}$, $N_{a_{t,\mathbf{r}}}$ and $N_{z_{t,\mathbf{r}}}$ are equally likely given the number of targets and the number of measurements, it is appropriate to divide the aforementioned product of probabilities by the total number of possible partitions in $\mathcal{E}_T$ to yield the likelihood of a particular partition.

$$\begin{aligned} p(\omega_{t,r}) &= p(\omega_{t,r}, N_d, N_a, N_z, N_f, K_t) \\ &= p(\omega_{t,r}|N_d, N_a, N_z, N_f, K_t)p(N_d, N_a, N_z, N_f, K_t) \end{aligned} \tag{3.28}$$

$$\begin{aligned} p(N_d, N_a, N_z, N_f, K_t) &= p(N_d, N_a, N_z, N_f|K_t)p(K_t) \\ &= p(N_d|K_t)p(N_a|K_t)p(N_z|K_t)p(N_f|K_t)p(K_t) \\ &= p(N_d|K_t)p(N_a)p(N_z|K_t)p(N_f) \end{aligned} \tag{3.29}$$

In the interest of conciseness, the subscripts t, r are removed from the remaining formulation, but should be considered in any implementation. Next, it is necessary to propose models to describe each of the probabilities mentioned previously. For our formulation it is proposed that a discrete Poisson distribution be used to model

the arrival rate of new targets on the scene for each region, as well as for the rate of false alarm in the measurements for a given region. A Poisson distribution is proposed for these two scene properties because both occurrences happen independently of the number of previous occurrences in a transportation scene. Each of these models can be formulated as follows,

$$p(N_a) = \left(\frac{{\lambda_a}^{N_a} e^{-\lambda_a}}{N_a!} \right), \tag{3.30}$$

$$p(N_f) = \left(\frac{{\lambda_f}^{N_f} e^{-\lambda_f}}{N_f!} \right), \tag{3.31}$$

where λ_a is the arrival rate of targets per unit area (volume in the 3D case) of a region, per unit time, and λ_f is the false alarm rate for measurements per unit area of a region, per unit time. Figure 3.5 illustrates the Poisson Distribution that is used to model the target birth and target false alarm rates as the rate is varied.

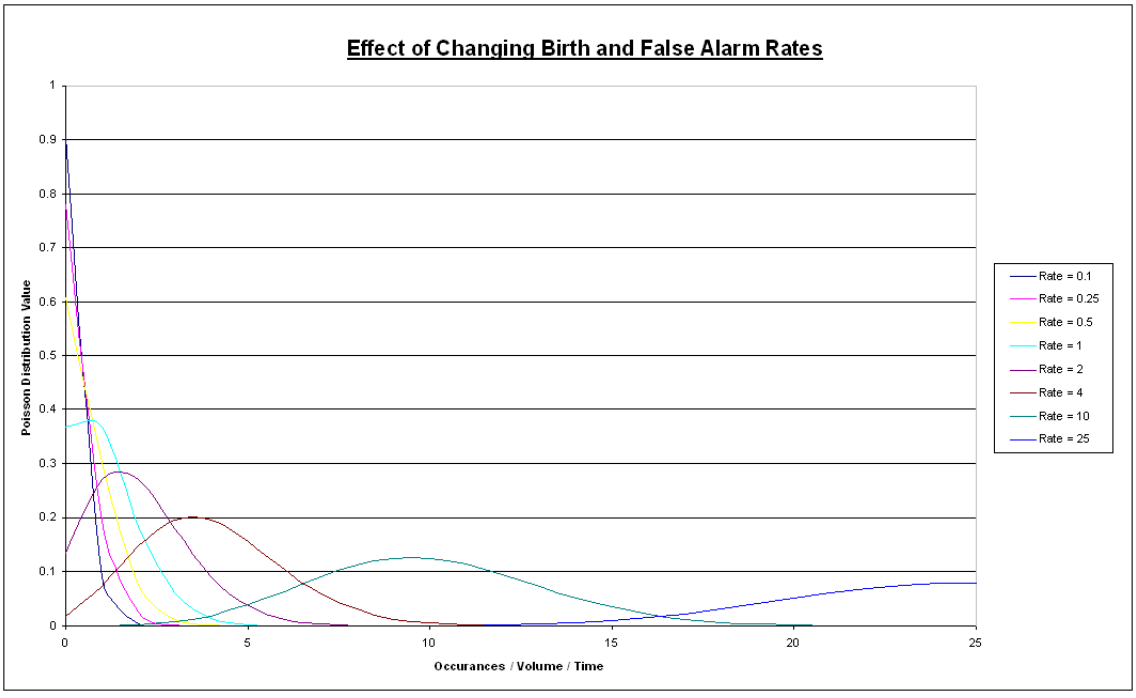

Figure 3.5: Poisson Distribution: Poisson distribution showing the effect of varying the birth and false alarm rates.

The rate of measurement detection and rate of target termination can be described by the discrete binomial distribution. A binomial distribution is appropriate for these scene properties because the likelihood of occurrence is a function of the number of targets in the scene:

$$p(N_z|K_t) = \left(\frac{K_{t-1}!}{(K_{t-1}-N_z)!N_z!}\right) p_z^{N_z}(1-p_z)^{K_{t-1}-N_z}, \tag{3.32}$$

$$p(N_d|K_t) = \left(\frac{K_t!}{(K_t-N_d)!N_d!}\right) p_d^{N_d}(1-p_d)^{K_t-N_d}, \tag{3.33}$$

where K_t is the number of targets present in a region at time t, p_d represents the probability that a target is detected for a given region and p_z represents the probability that a target terminates for a given region. Figure 3.6 illustrates the binomial distribution that is used to model the target detection and target termination probabilities as the probabilities are varied from 0 to 1.

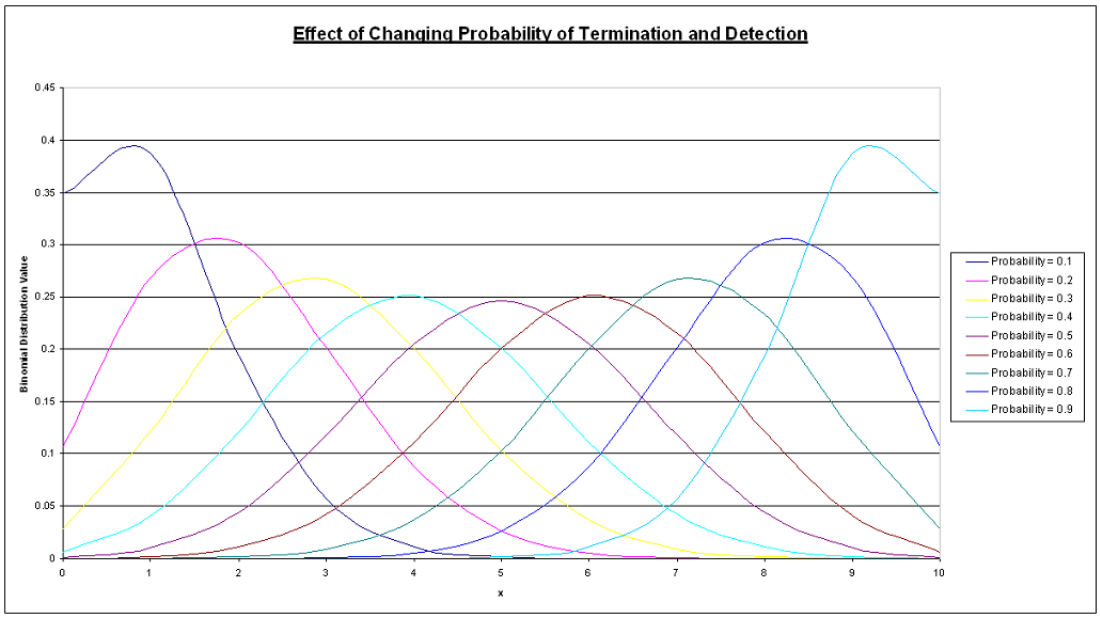

Figure 3.6: Binomial distribution: Binomial Distribution showing the effect of varying the detection and termination probabilities.

Expressed combinatorially, and under the assumption that all partitions in $\mathcal{E}_T$ are uniformly likely, the total number of possible associations found in $\mathcal{E}_T$ given the region configuration, can be shown to be

$$p(\omega_{t,r}|N_d, N_a, N_z, N_f, K_t) = \left(\left(\frac{(N_f + N_d)!}{N_f! N_d!} \right) \left(\frac{K_t!}{(K_t - N_d)! N_a!} \right) \left(\frac{K_{t-1}!}{N_z!} \right) \right)^{-1}, \tag{3.34}$$

where $K_t - N_d = N_u$ is the number of undetected targets at time t. When all of these models are combined the posterior for a given partition can be expressed as

$$p(\omega_{t,r}) = \left(\frac{e^{-(\lambda_f + \lambda_a)}}{(N_f + N_d)!} \right) p_d^{N_d} (1 - p_d)^{K_t - N_d} p_z^{N_z} (1 - p_z)^{K_{t-1} - N_z} {\lambda_a}^{N_a} {\lambda_f}^{N_f}. \tag{3.35}$$

Next, it is desired to integrate the prior into an appropriate likelihood function that can be used in evaluation by a sequential Bayesian estimator. We begin with Bayes' rule:

$$p(\omega|\hat{R}) = \frac{p(\omega) p(\hat{R}|\omega)}{p(\hat{R})} \tag{3.36}$$

which is equivalent to

$$p(\omega|\hat{R}) \propto p(\omega) p(\hat{R}|\omega). \tag{3.37}$$

The subscripts t and r are reintroduced and it is shown that the posterior we must maximize in order to determine the path that a set of targets has taken in a given scene over time:

$$\begin{aligned} p(\omega|\hat{R}) = \frac{1}{Z} \prod_{t \epsilon T} \prod_{r \epsilon \Re} \left(\frac{e^{-(\lambda_{f_{t,r}} + \lambda_{a_{t,r}})}}{(N_{f_{t,r}} + N_{d_{t,r}})!} \right) p_{d_{t,r}}^{N_{d_{t,r}}} (1 - p_{d_{t,r}})^{K_{t,r} - N_{d_{t,r}}} \\ \times p_{z_{t,r}}^{N_{z_{t,r}}} (1 - p_{z_{t,r}})^{K_{t-1,r} - N_{z_{t,r}}} {\lambda_{a_{t,r}}}^{N_{a_{t,r}}} {\lambda_{f_{t,r}}}^{N_{f_{t,r}}} \prod_{\tau \epsilon \mathcal{E} \setminus \{\tau_0\}} \prod_{i=1}^{i_\tau} \mathcal{N}(\tau(t_i)|\widehat{x}, \widetilde{p}). \end{aligned} \tag{3.38}$$

To maximize the posterior presented, it is necessary to develop a method of evaluation that negates the need to exhaustively evaluate all the possible partitions $p(\omega|\hat{R})$ for $\omega \epsilon \mathcal{E}_T$, as this would represent a computational infeasibility for realtime operation. Monte Carlo methods afford an attractive option as they can operate in real time and they are parallelizable across multiple CPUs. Such a method is discussed in some detail in Section 3.7.

3.7 Monte Carlo Markov Chain Data Association

In this section, a Markov Chain Monte Carlo (MCMC) [46] sampler for solving the multiple-target, multi-region tracking formulation is presented. The MCMC approach that is discussed employs a combinatorial optimization approach in order to determine an ideal partition, ω, from the joint association $\mathcal{E}_T$. A partition can be thought of as the discrete state that defines a possible scene. Further discussion about the method can be found in [16] and [42]. A more generic name for MCMC is simulated annealing. MCMC is a general method for generating samples from a distribution $p(\omega|\hat{R}_T)$ by constructing a Markov chain whose state is ω. In order to determine the maximum likelihood for the distribution, we evaluate the current partition, ω, against a proposed partition, ω'. The proposed partition is accepted with probability $\mathfrak{A}(\omega, \omega')$ if

$$\mathfrak{A}(\mathcal{E}, \mathcal{E}') = min\left(1, \frac{P(\mathcal{E}|\hat{\mathcal{S}}_T)}{P(\mathcal{E}'|\hat{\mathcal{S}}_T)}\right) \geqslant U(0,1), \tag{3.39}$$

where $\geqslant U(0,1)$ defines a uniform random number between zero and one. If the proposed partition is rejected, the sampler stays at ω. As long as we ensure that the Markov chain is irreducible, meaning it is possible to get to any state from any state, and aperiodic, meaning that it is always possible to move from one state to another in one step, then the solution will converge due to the ergodic theorem.

To make the algorithm more efficient, two additional assumptions are made: (1) the maximum velocity of a given target is less than $\bar{v}$ and (2) the number of consecutive missing observations of any track is less than $\bar{d}$. Both assumptions can be asserted reasonably for a given scene. Using these two assumptions, a structure is presented that can be used in formulation of the partition moves. The structure, $L_{\hat{r}^i(t)}$, represents all targets in $\mathcal{R}_T$, over all time, that fall within a radial distance of $||\hat{r}^i(t) - \hat{r}^j_{t+d}|| \leq \bar{v} * \bar{d}$.

To generate partitions which can be evaluated against the posterior, a move proposal distribution, ξ, consisting of eight types of moves is employed. Figure 3.7 illustrates the eight moves that are described [42].

1) **Track Birth (ξ_1):** For a birth move we increase the number of tracks in the proposed partition from K_t to $K'_t = K_t + 1$. Next we select a time, t_0 uniform at random from the observation window that is used as the start of the track. We

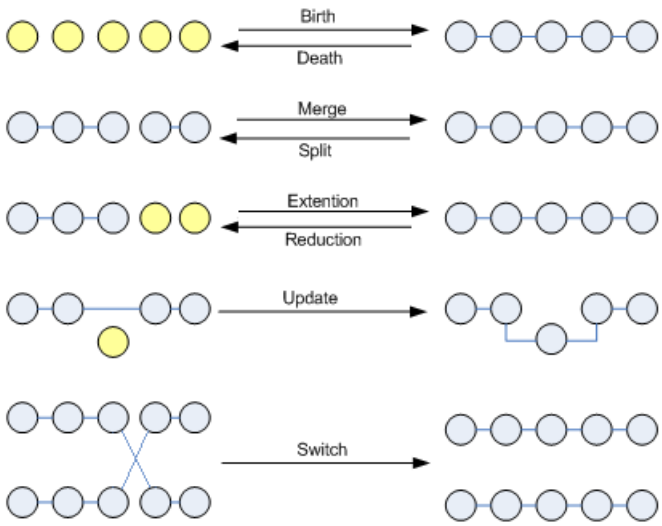

Figure 3.7: Illustration of MCMC Moves

select a target measurement point, γ_0, from $\hat{r}_{t_0}$. The proposed move is rejected if γ_0 either belongs to another track or does not have any neighboring points within L_{γ_0}. If the move is accepted we add γ_0 to $\tau_{K'}$, select γ_1 uniform at random from L_{γ_0} and add γ_1 to the track $\tau_{K'}$. This process is repeated with termination probability φ for subsequent γ_i, or until L_{γ_i} is empty.

2) **Track Death (ξ_2):** For a delete move we select a track, τ_k, at random from Ω and remove it, assuming that there is at least one track in ω, otherwise the move is rejected.

3) **Track Split (ξ_3):** For a split move, we select a track, τ_k, uniform at random from Ω. If τ_k has fewer than four points, the move is rejected. We next select a point from τ_k and split the selected track into two tracks, τ_{k_1} and τ_{k_2}, with the constraint that both tracks must contain at least two measurements.

4) **Track Merge (ξ_4):** For a merge move, two tracks, τ_{k_1} and τ_{k_2}, are selected at random from the set Ω. We define γ_1 as the end point for τ_{k_1} and γ_2 as the start point for τ_{k_2}. The move is rejected if γ_2 is not contained in the structure L_{γ_1} and accepted otherwise.

5) **Track Extend (ξ_5):** For an extend move we select a track, τ_k, uniform at random from $\mathcal{E}$ and assign new measurements to the track in the same fashion as described in the birth move.

6) **Track Reduce (ξ_6):** For the reduce move, a track, τ_k, is selected uniform at random from Ω. The move is rejected if the track is less than three points long. Next, a time, t_k, is selected from $2, ..., |\tau_k| - 1$, and all points in τ_k after t_k are removed.

7) **Track Update (ξ_7):** The track update move is a track reduction move, followed by a track extend move.

8) **Track Switch (ξ_8):** For a track switch move, two tracks, τ_{k_1} and τ_{k_2}, are selected uniform at random from Ω. We select γ_1 from τ_{k_1} and γ_2 from τ_{k_2}. The move is rejected if γ_2 is not contained in the structure L_{γ_1} and accepted otherwise. If the move is accepted all points in τ_{k_1} that fall after γ_1 are added to τ_{k_2}, and all points in τ_{k_2} that fall after γ_2 are added to τ_{k_1}.

Each of these moves can either be selected uniformly at random or can be assigned prior probabilities subject to priori information about the scene, subject to

$$\sum_{i=1}^{8} p(\xi_i) = 1. \tag{3.40}$$

In order to optimize the execution of an implementation, constraints can be applied to the moves in terms of when they are allowed to occur. In the case where no targets or tracks are present, only the birth move may occur. In the case where only a single track exists all moves except track merge and switch moves may occur.

In order to extend this method to support real-time tracking a sliding window approach is proposed. The sliding window approach works by allowing the MCMC moves to be performed iteratively on the measurements in the most recent N_{sl} images. The measurements just prior to the leading edge of the sliding window are taken to be truth and the remaining measurements are allowed to be iteratively put into different partitions until convergence is met. This process is then repeated at the next discrete time step with the leading edge of the sliding window incremented by one.

In this section a method for computing discrete-time target associations has been discussed. The framework described in this chapter can track an unknown number of targets which initiate and terminate at random. In Chapter 4, results will be shown that illustrate the method to be robust under conditions of occlusion and clutter since the scoring function (3.38) that was developed can incorporate false

alarms and undetected targets under association. In addition, the model will be shown to be flexible due to its ability to incorporate information scene specific information as well as define different model parameters per region of the scene. The Monte Carlo nature of the algorithm makes it an ideal candidate for parallelization in cluster computing environments.

3.8 Conclusions

In this chapter a detailed overview of the tracking algorithm developed over the course of this work. A high level system model was presented and was followed by discussions about the algorithm notation, the feature extraction method, the object segmentation method and the target-track association algorithm. The framework is suitable for tracking targets that vary in scale and orientation due to the use of Scale-Space, and provides the basis for the application of object models to the results of the Delaunay clustering step.

Throughout the course of investigation a number of areas were identified where future research could be conducted. One such area for future consideration would be the addition of the capability to model a non-stationary background. This would be of value in considering cases where either the camera moves do to wind, is bumped, or where the camera is mounted on a moving object such as a car. This addition would occur prior to the feature extraction step and the remaining methods presented in this thesis would still apply.

Finally, an area that was identified for further investigation was the addition of scene specific MCMC moves. An example of a move that was considered involved modification of the merge move to provide for the ability to merge a track that was contained within another track into a single track. A graphical representation of this concept is presented in Figure 3.8.

Figure 3.8: Proposed Modified Merge Move

Chapter 4

Experimental Results

In this chapter the stochastic framework that was proposed and discussed in Chapter 3 is applied to an intersection monitoring application. The application involves monitoring and counting the vehicle movements that can occur at an intersection. Examples of typical types of movements that can occur at an intersection include: 'Northbound Left Turn', 'Westbound Through', 'Eastbound Right Turn'.

The main users of this type of transportation data are city transportation and public works departments. Currently, these organizations predominantly collect turning movement volume information manually, using either paper and pen based method or using *low-tech* counting equipment. The ability to collect turning movement volume information using computer vision algorithms represents a considerably more cost effective method of collecting traffic data as compared with existing manual methods. The data also provide much more information than manual methods (i.e. speed, position, time-stamped event). The city transportation planning departments use the turning movement volume information as a means of optimizing intersection signal timings, and for deciding when roadways must be upgraded to meet mandated quality of service levels.

This chapter is broken into three sections. First, the problem of vehicle segmentation in transportation scenes is discussed. A discussion about the challenges inherent with conducting segmentation in different types of transportation scenes is also included in this section. Second, the problem of tracking the segmented vehicles using the stochastic framework presented in the preceding chapter is discussed. Results are presented which show the method to be robust to false detections (false alarms), missing measurements (occlusion), and tracks which exist in close proximity to one another. Finally, a method for classifying and counting the detected

tracks is discussed. Results are presented which show that the method is robust to similar tracks.

4.1 Segmentation

The segmentation approach employed in this work involves combining information obtained from evaluating the motion history, as well as the background-foreground characteristics of a given video frame. Background subtraction and motion history each present a number of challenges; however, when combined these challenges can be mitigated. These challenges, and their impact on the segmentation of vehicles from a transportation scene, will be the focus of discussion later in this chapter. Results are presented which show that combining background subtraction results with motion history results will improve segmentation. Once these two methods have been combined, a global Otsu threshold [36] is applied to the results in order to produce a binary segmented image. The Otsu method was selected because it assumes an image is bimodel (i.e. background/foreground) and finds the optimal threshold by minimizing the interclass variance. The final step that is employed involves labeling the objects in the scene using a connected components clustering approach.

Background subtraction, discussed in Section 2.2, presents two primary challenges when used as a segmentation method. The first occurs when the pixels which make up a foreground object are similar in pixel intensity to the background. This similarity results in the difference between the background model and the pixels making up the foreground object negligible. Under this circumstance, object segmentation is not possible. The second challenge that can arise from background subtraction deals with model initialization. If a foreground object persists in the same location in the scene during the period when the background model is being initialized, a false foreground artifact will appear at such a time as the persistent object moves, or leaves the scene. Dynamically updating the background model can mitigate this issue. Figure 4.1 shows the results of a background subtraction, where the difference between the current frame of a video is compared to a reference background scene model.

The next step in the proposed segmentation method is to evaluate the motion history of the current frame. As with background subtraction, motion history can present some challenges when it is used for segmentation. The first challenge is due to the fact that motion history is evaluated by subtracting subsequent image frames

Figure 4.1: Background Subtraction: (a) Video frame taken of a highway overpass, (b) Background model obtained by taking the average pixel value over 100 frames, and (c) Absolute pixel difference between (a) and (b).

in a video sequence. As a result, the speed in which vehicles in the scene are moving will impact the results of evaluating the motion history. Slow moving vehicles leave a limited motion history signature, whereas rapidly moving vehicles can produce a shadowing or streaking effect. The issue is especially prevalent when vehicles are moving at different speeds in the transportation scene, and as a results a global optimal frame rate for conducting motion history cannot be used. Figure 4.2 shows the results of a motion history image set. Each image in the set shows the absolute pixel difference between the current frame and the frames at incremental units of time previous to the current frame.

Upon completion of the background subtraction step and the motion history step, the final step in the segmentation method involves combining the previous results and applying a threshold in order to produce a binary image. For this work a global Otsu threshold is applied to the combined motion history and background subtraction results. Once a binary image, $\phi(t)$, is produced, a method which labels the connected components of an image is employed in order to segment the vehicles in the scene. Figure 4.3 shows the results of combining the background subtraction results with the motion history results. By means of Otsu global threshold and connected components clustering, the vehicles, $R(t)$, are subsequently identified.

Once the segmentation step has been completed, an additional layer of logic is applied. Using feedback about the location, size and motion of the vehicles present in the previous frame, the vehicles found in the segmentation step are either split, merged or left alone. This additional step improves the vehicle segmentation results, which in turn improves the tracking results.

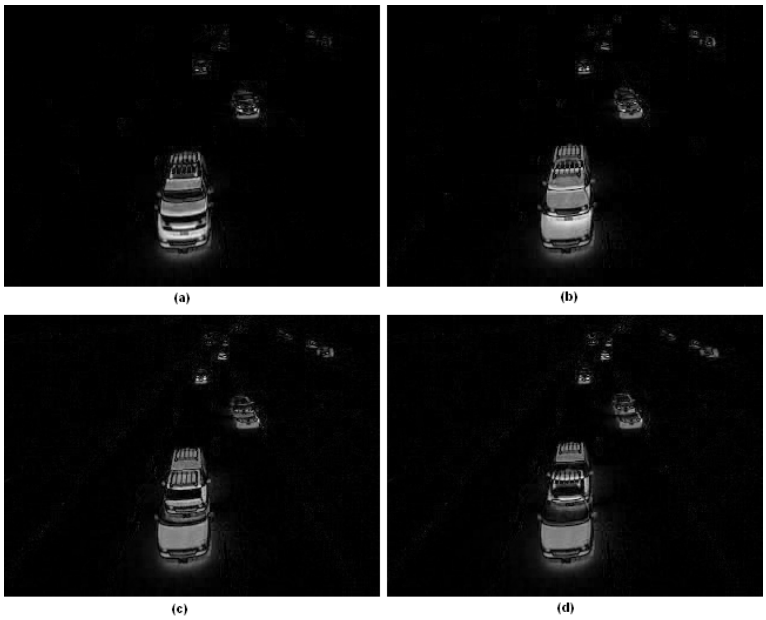

Figure 4.2: Motion History: (a) Frame difference between frame t and t-1, (b) Frame difference between frame t and t-2 , (c) Frame difference between frame t and t-3, and (d) Frame difference between frame t and t-4.

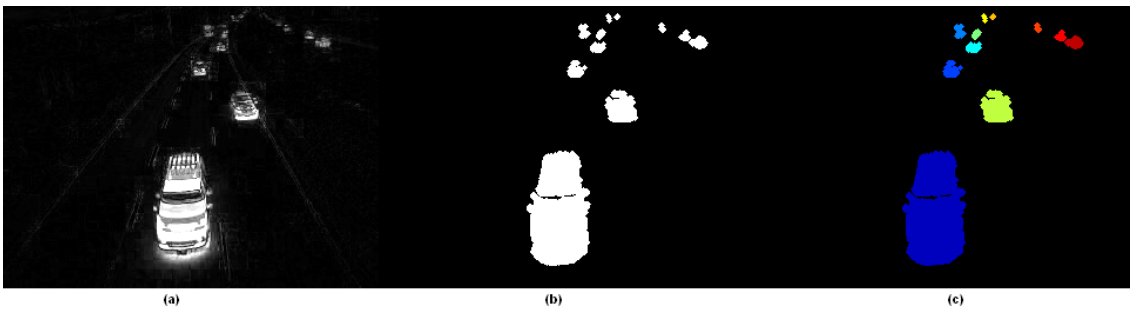

Figure 4.3: Vehicle Segmentation: (a) sum of background difference and first order motion history, (b) global Otsu threshold image with dilation morphology applied, and (c) connected components clustering.

4.2 Tracking

The next step in the vehicle counting process involves assembling the segmented vehicles found in sequential video frames into tracks (Discussed in Section 3.7). The discussion begins by presenting results from some generated data sets, and follows with illustrated, real world tracking examples. A potential user interface for collecting information that can be used in the tracking step is described and illustrated. Currently, video-based traffic counting technology is available which employs background subtraction methods to indicate when a vehicle is located in a detection zone. This traffic counting technology does not provide for the ability to relate detections to one another; therefore, conducting an intersection or roundabout count is not possible. Examples of firms that provide this technology are Iteris, Trafficon and Autoscope. The practical application of the tracking capability discussed in this section involves using cameras to conduct intersection counts. This capability represents a significant improvement over existing video-based traffic data collection technologies.

To address the solution of vehicle tracking at intersections and roundabouts, the Monte Carlo Markov Chain (MCMC), discussed in Section 3.7, method is used. The stochastic method was shown to perform well in scenes containing false measurements and in scenes containing missing measurements, also known as occlusions. The method also performs well when multiple tracks are in proximity to one another. Figure 4.4 shows the performance of MCMC under various conditions, including (a) only the true object positions, (b) the addition of ten percent false alarms, (c) in the presence of ten percent missing measurements and (d) a combination of (b) and (c). Target points in the figures are coloured the same if they belong to the same track. False alarms are indicated using white. There is no relationship between the colours selected for a given track in each of the four figures.

As discussed in Section 3.6, to construct a suitable scoring function, an understanding about prior statistics about the regions of the image which make up the transportation scene is beneficial. To obtain this prior information assumptions about the scene must be made. One of the primary motivations for the inclusion of regions in the MCMC scoring function was to facilitate an intuitive user interface for the input of prior information. The objective in breaking the scene into regions is to identify areas of the scene which have different birth rates, false alarm rates, termination probabilities and detection probabilities as described in section. In application terms, identifying areas of the transportation scene where vehicles are

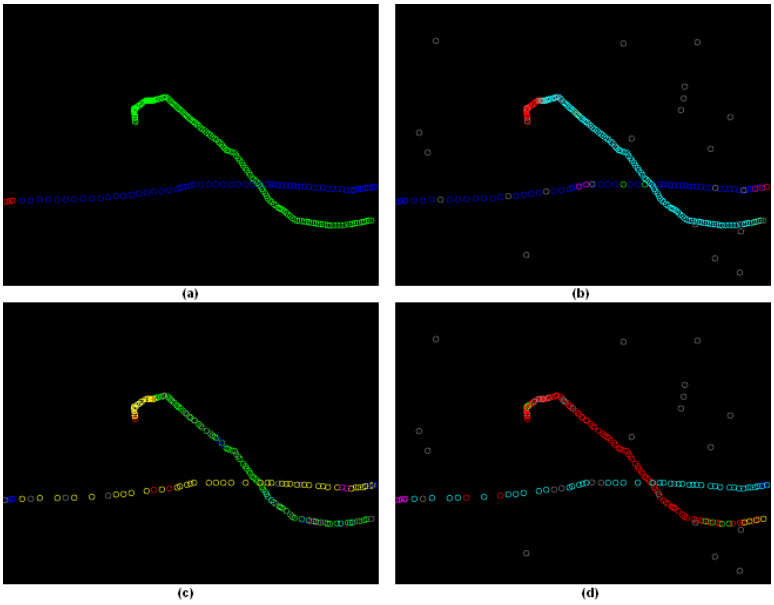

Figure 4.4: MCMC Tracking Results: This plot shows the results obtained with MCMC from contrived object data. (a) Shows tracking results with no false alarms and no missing measurements, and (b) tracking results with 10% false alarms and no missing measurements, (c) tracking results with no false alarms and 10% missing measurements, and (d) tracking results with 10% false alarms and 10% missing measurements.

more likely to emerge and leave from the scene, more likely to be occluded by foreground objects, or where vehicles are likely to be observed, is valuable. Figure 4.5 shows an example user interface that could be used for capturing information about the regions of a transportation scene. Information about road location, the location of potential foreground occlusions and the location of the intersection approaches are noted in the user interface.

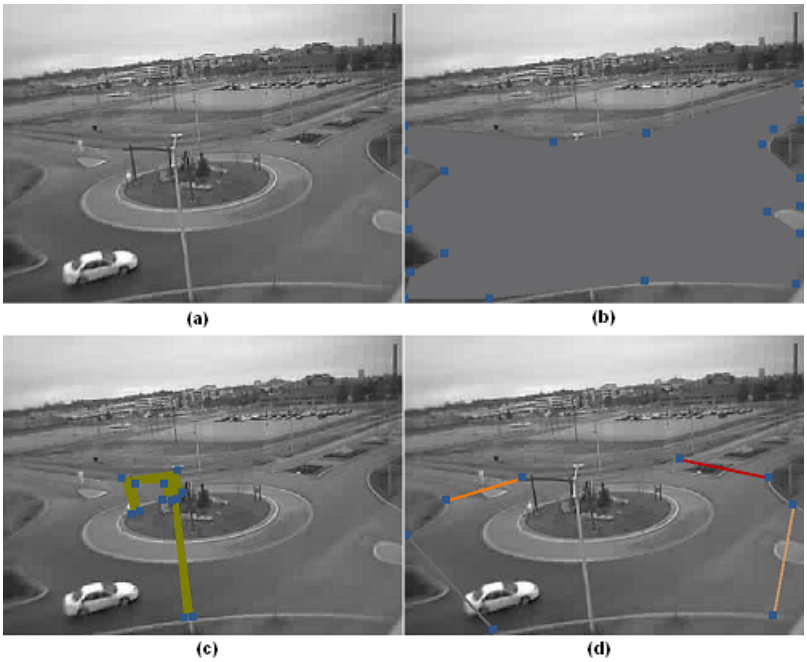

Figure 4.5: Transportation Scene Configuration User Interface: (a) Original roundabout scene image, (b) Road Location (ROI), (c) Location of potential occlusions and (d) Location of roundabout approach entrances/exits.

Once the regions of a transportation scene have been identified, termination probabilities, detection probabilities, birth rates and false alarms can be assigned to each region. To build a set of regions for a transportation scene, the three inputs from Figure 4.5 can be used, along with expert knowledge about the transportation scene. Under the assumption that vehicles can only enter/exit the scene from the edge of the image a region map can be constructed. Figure 4.6 shows the result of applying information gathered from the user interface with prior knowledge about the configuration of the transportation scene. The figure shows regions with higher assumed birth rates and termination probabilities in red. Regions that have relatively higher occlusion probabilities are shown in yellow and regions containing low

termination probability and high detection probability are shown in gray. Regions that are not colored are assigned a high false alarm rate.

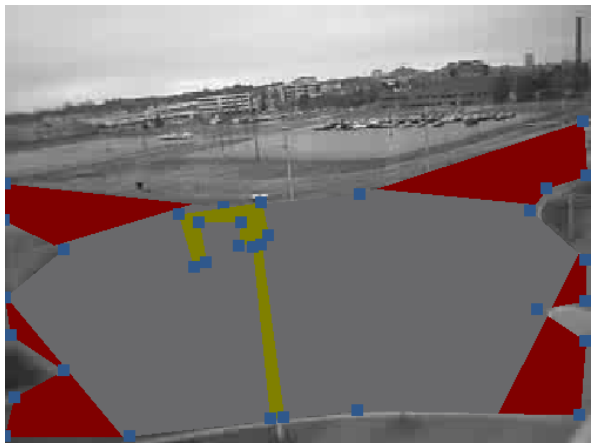

Figure 4.6: MCMC Region Map: Figure shows the location of birth/death regions (red), occlusion regions (yellow) and track continuation regions (gray). False alarm regions are represented by the area containing the original image.

To evaluate the posterior term of the scoring function (3.38), the observed vehicle positions must be scord against the estimated vehicle position. To accomplish this, a Kalman filter is used to smooth vehicle position using both the observed vehicle position and observed motion estimates. The observed vehicle position is then scored against the estimated vehicle position found using the Kalman filter. An important aspect of the Kalman filter, and by extension the evaluation of the MCMC scoring function (3.38), is the motion model that is used to update the vehicle's state. For this work, a first order linear motion model is used. Motion estimation is conducted using a hierarchical motion estimation method. The last step in the tracking approach is to iteratively apply the MCMC algorithm. At each iteration the proposed partition is scored against the current partition. The partition which scores the highest is passed onto the next iteration. This process is repeated until a stopping criterion is met. The stopping criterion was determined experimentally to be 500 iterations. The proposed tracking algorithms is then applied on each frame in the video sequence, resulting in vehicles being tracked through the transportation scene. Figure 4.7 shows the results of applying the aforementioned steps against a medium volume traffic circle. Results are shown that illustrate tracking through occlusion in proximity to other vehicles. Various intersection movements are shown in the figure.

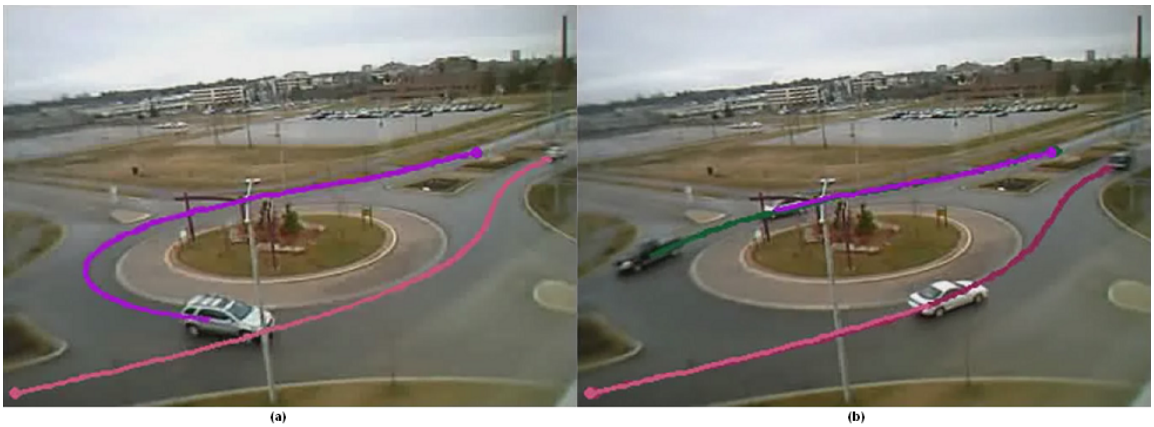

Figure 4.7: Real World Tracking Results: Figure shows tracking results of a roundabout. Each colored track represents a different vehicle.

The tracking method that is presented performs well in most transportation scenes. The primary scenario which presents challenges for this method occurs when two vehicles emerge on the scene connected to each other, persist in the scene connected to each other, and finally depart the scene connected to each other. This issue is caused as a result of deficiencies in the vehicle segmentation step, rather than the tracking method.

The segmentation method described in this chapter is essentially only extracting blobs from the image. No implicit notion of a vehicle is asserted in segmentation. As a result, when multiple objects emerge, persist and exit the scene together there is no way to differentiate them as separate vehicles. One potential solution to this problem would be to add the concept of a physical vehicle model to the segmentation step. Another approach that can be adopted is to ensure that a suitably high camera vantage is used when recording a transportation scene. This vantage ensures that there is sufficient spacing between the vehicles to segment them as individual entities.

4.3 Track Identification

The final step that is required to extract a full turning movement count from a video collected at a transportation scene, such as an intersection or a roundabout, is to determine how many of the tracks found in the scene, such as those shown

in Figure 4.7, crossed the approach entrance/exits, shown in Figure 4.5(d). In the case of an intersection a track must cross two of the approach entrance/exits to be counted. For example, if a track crosses the south approach and then the west approach, the vehicle will be counted as a northbound left turn. If a track crosses the north approach and then the west approach, the vehicle will be counted as a southbound right turn. Typically, in a transportation application, these counts are presented in fifteen minute time slices and the data are used to determine which hour of the study contains the peak vehicle volume for a given intersection. Figure 4.8 shows a typical turning movement output diagram that is used by municipal transportation planning agencies. The diagram shows vehicle volumes broken down into the twelve possible movements that can occur at an intersection.

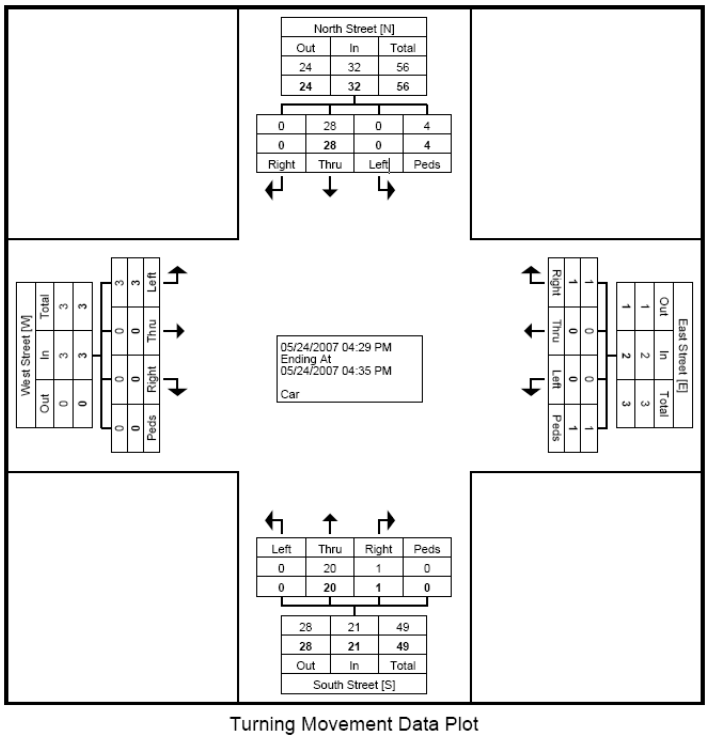

Figure 4.8: Turning Movement Diagram: Figure shows the typical output format that traffic planners and engineers use to present their manual and video-based intersection counts.

The intersection count diagram shown in Figure 4.8 was generated from data that were obtained by processing video of an intersection. In addition to intersections,

roundabouts and bi-directional counts are also of interest to transportation planning organizations. Results are presented in Table 4.1 that show the performance of the method when counting bi-directional vehicle movements from video taken from a highway overpass. The result was obtained from a five minute video clip of a highway overpass and compared to ground truth which was collected manually. Data sets for this work were provided by Miovision Technologies Incorporated and can be made available upon request (researchvideos@miovision.com). The method performs very well in this scenario, missing only one vehicle in each direction. Both missed detections were caused when two vehicles entered, persisted and exited the scene merged together.

Table 4.1: Bi-Directional Mid-Block Vehicle Counting Results

	Correctly Counted	Undetected	Error Rate
From North Through	38	1	2.6%
From South Through	29	1	3.4%
Total	67	2	3.0%

The counting of intersections by transportation planning organizations requires the use of field staff. The counting of roundabouts, also known as traffic circles, is also done using manual counting methods; however, unlike a traditional intersection, when vehicles enter a roundabout, it is unclear immediately as to which exit they will take. This makes roundabouts, especially high volume roundabouts, a more difficult transportation scene to count. A typical manual roundabout count requires approximately three times as many field staff hours to count as compared to a manual intersection count. Table 4.2 shows the results of applying the vehicle counting method to a medium volume roundabout.

Although the algorithm is able to track the majority of the objects through the occluded regions of the transportation scene, the movements that have to pass through at least one occlusion, shown in Figure 4.5(c) (From South Left, From South Through, From West Left) show the highest error rate. The cause of the errors in track classification were as follows: 56% of the errors where caused by occlusion and 44% of the errors were caused by vehicle segmentation issues, such as splitting and merging of objects. The majority of the incorrectly classified tracks only had a single break in the vehicle track. Figure 4.9 shows an example of this mode of failure.

A break in the track meant that one portion of the track crossed one of the approach lines, while the other portion of the track crossed the second approach line. Ap-

Table 4.2: Roundabout Vehicle Counting Results

	Correctly Counted	Undetected	Error Rate
From North Through	118	9	7.1%
From North Right	9	0	0.0%
From South Left	44	10	18.5%
From South Through	66	15	18.5%
From West Right	52	1	1.9%
From West Left	7	2	22.2%
Total	296	37	11.1%

Figure 4.9: Track Split Errors: This figure illustrates an example of a type of tracking error that causes a vehicle to not be counted.

proach lines are shown in Figure 4.5(d). This means that, even though the majority of the vehicle track was accurate, the vehicles were counted incorrectly. To correct for this issue, a heuristic measure could be added to the track results to correct for situations where a vehicle track terminated in an area of the image where this was an unlikely event.

Chapter 5

Conclusion

5.1 Concluding Remarks

The work presented in this thesis was motivated in large part by a practical need identified in the transportation planning industry to collect volume information from transportation scenes in an accurate and cost effective manor. The introduction of a computer vision based tracking approach for conducting this type of vehicle volume data collection represents a significant improvement over current manual data collection methods. Current manual methods are expensive, due to the labor involved, and plagued by inaccuracies caused by fatigue. The data which can be readily obtained using the methods described in this work will assist transportation planning organizations in building better transportation networks, reduce the costs involved with collection and improve data integrity.

The major contribution of this work was the presentation of a stochastic method for conducting vehicle tracking in scenes containing occlusion and the potential for false detections. The method uses a deferred logic approach based on a data oriented combinatorial optimization approach to solving Monte Carlo Markov Chain (MCMC) data association problems. The result of applying this method to the problem of vehicle tracking is the ability to track an unknown number of vehicles in a transportation scene, while requiring a minimal amount of prior information about the scene. The method is computationally efficient and lends itself toward large scale computational parallelization.

The development of a scoring function, presented in Section 4.2, that is able to account for local changes in detection and termination probabilities of the images

making up a video sequence is a novel approach in the literature. The primary motivation for the addition of the concept of regions to the scoring function, was to facilitate the construction of an intuitive user interface that can be used to provide relevant prior information about real-world transportation scenes. Results were presented in this work which show that the tracking method is robust in scenes containing false vehicle detections, as well as missing vehicle detections. Results were also presented which show that the segmentation method, tracking method and track classification method are effective at detecting vehicle movements and resulting counts in transportation scenes.

An area of research that has the potential to dramatically improve the results presented in this work, would be to integrate the vehicle segmentation step with the MCMC data association step. This would afford a unified stochastic model for both segmentation and tracking. The main focus of this work would involve determining a scoring function and a set of linear optimization moves that could simultaneously evaluate the fitness of all possible object segmentations and a track association over the entire life of an object, given a base set of local image descriptors (regions of an image with similar chromatic characteristics).

If for example, an object, such as a vehicle, could be described as being made up of a set of local image descriptors, then it would be possible to extend the moves in MCMC to included adding and removing objects from an image frame, adding and removing image descriptors from a object and splitting and merging objects present in the scene. The scoring function of MCMC would also have to be modified to account for the associations between image descriptors in a single frame, as well as the object associations between frames. This would affectively defer the segmentation of objects, similarly to the way the track association logic is deferred in this work. One obvious downside of this extension is that it would have the effect of greatly increasing the computational load of evaluating the scoring function.

5.2 Future Research Directions

A potential direction for future research is to modify the observation model to include prior information about the location of possible tracks in a transportation scene. If this approach is adopted, the advantage is that a built-in exclusion principal for associating targets to tracks is included in the observation model. In

addition, a dimensional reduction from 2-D to 1-D is achieved which has advantages in terms of computational performance. Figure 5.1 illustrates this concept by showing a left turn at an intersection.

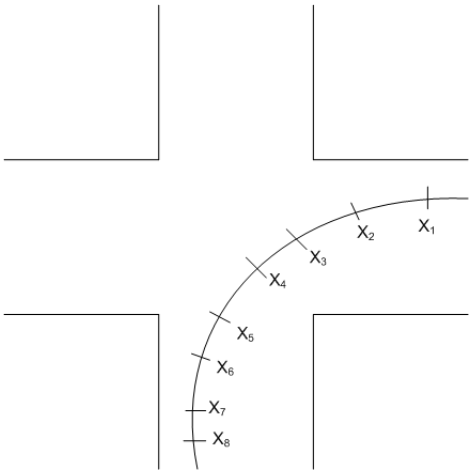

Figure 5.1: Left-Turn at an Intersection

A measurement model for the given track can be defined as $X = \{x_1, x_2, .., x_N\}$. As an object moves through along the track, the observation model will show a strong response in the location of the vehicle. Figure 5.2 illustrates a trivial example of a single vehicle moving along the track.

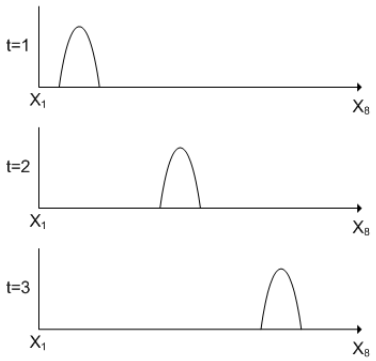

Figure 5.2: Left-Turn at an Intersection

Another advantage of this approach to modeling vehicle tracks in transportation scenes is that the model is well suited for the active contours tracking framework

presented in [2]. This framework has shown excellent performance in the tracking literature.

Bibliography

[1] K. M. Alexiev and P. D. Konstantinova. Hypothesis pruning in JPDA algorithm for multiple target tracking in clutter. *Bulgarian Ministry of Education and Science*, 1997.

[2] M. I. Andrew Blake. *Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion*. Springer, 2000.

[3] F. Aurenhammer and R. Klein. Voronoi diagrams. *Institut f'ur Grundlagen der Informationsverarbeitung Technische Universit at Graz*, September 1994.

[4] D. P. Bertsekas. *Dynamic Programming and Optimal Control. 2nd ed.* Athena Scientific, Belmont, MA, 2000.

[5] D. Bullock and J. Zelek. Towards real-time 3-d monocular visual tracking of human limbs in unconstrained environments. *IEEE Transactions on Image Processing*, Volume 13, NO. 6:836–847, June 2004.

[6] H. G. J. M. C. Carson, S. Belongie. Blobworld: image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 24, Issue 8:1026–1038, August 2002.

[7] G. H. C. Rasmussen. Probabilistic data association methods for tracking complex visual objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 23, Issue 6:560–576, 2001.

[8] A. E. Changjiang Yang, Ramani Duraiswami and L. Davis. Real time kernel based tracking in joint feature spatial spaces. *Perceptual Interface and Reality Labouratory, University of Maryland*, 2004.

[9] R. Diestel. *Graph Theory, Electronic Edition 2005*. Springer-Verlag Heidelberg, New York, 2005.

[10] V. R. Dorin Comaniciu and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 25, No. 5:564–577, May 2003.

[11] J. L. Fisher and D. P. Casasent. Fast JPDA multitarget tracking algorithm. *Carnegie Mellon University, Department of Electrical and Computer Engineering*, May 1987.

[12] R. Frezza and A. Chiuso. Learning and exploting invariants for multi-target tracking and data association. *Department of Information Engineering, University of Padova*, March 2005.

[13] H. Frigui. A robust clustering algorithm based on competitive agglomeration and soft rejection of outliers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 21, NO. 5:450–465, May 1999.

[14] C. Gentile. Segmentation for robust tracking in the presence of severe occlusion. *IEEE Transactions on Image Processing*, Volume 13, NO. 2:166–178, February 2004.

[15] M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. *University of Oxford, Oxford*, 1995.

[16] S. J. G. Jaco Vermaak and P. Perez. Monte-Carlo filtering for multi-target tracking and data association. *Signal Processing Laboratory, Cambridge University Engineering Department*, 2004.

[17] J. U. J.B. Collins. Efficient gating in data association with multivariate gaussian distributed states. *IEEE Transactions on Aerospace and Electronic Systems*, Volume 28, Issue 3:909–916, 1992.

[18] W. M. Y. Z. Z. R. Z. Jianzhou. A pruning technique of feasible matrixes of jpda algorithm. *Proceedings of the 3rd World Congress on Intelligent Control and Automation*, Volume 28:291–293, 2000.

[19] T. Kato and T. Wada. Integration between background subtraction and color detection based on nearest neighbor classifier: Instance based multimodal information integration. *Information Processing Societry of Japan Transactions on Computer Vision and Image Media*, Volume 45, No.13:110–117, 2004.

[20] D. F. Kiam Choo. People tracking using hybrid Monte Carlo filtering. *Proceedings Eigth IEEE International Conference on Computer Vision*, Volume 2:321–328, 2001.

[21] E. B. Koller-Meier and F. Ade. Tracking mulitple objects using the condensation algorithm. *Communication Technology Lab, Image Science, Swiss Federal Institute of Technology*, 2000.

[22] R. Krishnapuram and J. M. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, Volume 1:98–101, May 1993.

[23] L. S. Laurent Isenegger and N. Garcia. Moving objects segmentation based on automatic foreground/background identification of static elements. *Grupo de Tratamiento de Imagenes E.T.S. Ingenieros de Telecommunicacion, University Politecnica de Madrid Spain*, 2004.

[24] T. Lindenburg. Scale-space: A framework for handling image structures at multiple scales. *Proc. CERN School of Computing, Egmond aan Zee, The Netherlands*, September 1996.

[25] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[26] H. F. K. S. M. Seki, T. Wada. Background detection based on the cooccurrence of image variations. *Proceedings of CVPR*, Volume 2:65–72, 2003.

[27] K. Mehlhorn and S. Naher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, November 1999.

[28] H. T. Mei Han, Wei Xu and Y. Gong. An algorithm for multiple object trajectory tracking. *NEC Laboratories America, Cupertino*, 2004.

[29] E. B. Meier and F. Ade. Tracking cars in range images using the condensation algorithm. *Communications Technology Lab, Image Science, Swiss Federal Institute of Technology*, 1999.

[30] A. B. Michael Isard. Condensation conditional density propagation for visual tracking. *International Journal of Computer Vision*, Volume 29, No. 1:5–28, 1998.

[31] J. M. S. Min Tan and H. J. Siegel. Parallel implementations of block-based motion vector estimation for video compression on four parallel processing systems. *International Journal of Parallel Programming*, Volume 27, Number 3:195–225, 1999.

[32] S. Oh and S. Sastry. A polynomial time approximation algorithm for joint probabilistic data association. *Department of Electrical Engineering and Computer Science, University of California*, 2005.

[33] J. G. Oliver Frank, Juan Nieto and S. Scheding. Multiple target tracking using sequential Monte Carlo methods and statistical data association. *Swiss Federal Institute of Technology Zurich, Switzerland*, 2001.

[34] L. A. O.T. Ydz and H. Akn. Fast nearest neighbour testing algorithm for small feature sizes. *Electronics Letters*, Volume 40 No. 3, February 2004.

[35] Y. R. Padmavathi Mundur and Y. Yesha. Keyframe-based video summarization using Delaunay clustering. *Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County*, 2004.

[36] P.-C. C. Ping-Sung Liao, Tse-Shend Chen. A fast algorithm for multilevel thresholding. *Journal of Information Science and Engineering*, Volume 17:1713–727, 2001.

[37] M. P. R. Cucchiara, C. Grana and A. Prati. Detecting moving objects, ghosts and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 25 No. 10:1337–1342, October 2003.

[38] D. G. S. R. O. Duda, P. E. Hart. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[39] M. I. Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties. *Institute for Systems and Robotics, Lisboa Portugal*, february 2004.

[40] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, March 1984.

[41] I. H. Songhwai Oh, Shankar Sastry and K. Roy. A fully automated distributed multiple-target tracking and identity management algorithm. *University of California, Berkeley*, 2005. http://www.eecs.berkeley.edu/ sho/papers/aiaa05dmtim.pdf.

[42] S. R. Songhwai Oh and S. Sastry. Markov Chain Monte Carlo data association for general multiple target tracking problems. *Department of Electrical Engineering and Computer Science, University of California*, 2004.

[43] J. W. D. Stephen S. Intille and A. F. Bobick. Real-time closed-world tracking. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 697–703, June 1997.

[44] N. S. N. C. D. P. R. S. Tapas Kanungo, David M. Mount and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume. 24, NO. 7:881–892, July 2002.

[45] P. Torr. An assessment of information criteria for motion model selection. *Computer Vision and Pattern Recognition*, Issue 17-19:47–52, 1997.

[46] J. S. W. W. S. Kendall, F. Liang. *Markov Chain Monte Carlo: Innovations and Applications*. World Scientific, November 2005.

[47] G. Welch and G. Bishop. An Introduction to the Kalman Filter. *UNC-Chapel Hill*, April 2004.

[48] F. Yan, A. Kostin, W. Christmas, and J. Kittler. A novel data association algorithm for object tracking in clutter with application to tennis video analysis. *cvpr*, Volume 1:634–641, 2006.

[49] J. Yen and R. Langari. *The Textbook on Fuzzy Logic, Intelligent Control, and Fuzzy Information Systems*. Prentice Hall, 1999.

[50] J. Zhang. Nonlinear prediction for Gaussian mixture image models. *IEEE Transactions on Image Processing*, Volume 13, NO. 6:836–847, June 2004.

[51] W. M.-H. P. Y.-N. Y. Zhi-Sheng. Improved joint probabilistic data association algorithm. *Proceedings of the Fifth International Conference on Information Fusion*, Volume 2:1602–1604, 2002.

[52] Z. Zivkovic. *Motion Detection and Object Tracking*. PhD thesis, University of Twente, The Netherlands, 2003.