SELECTED TECHNIQUES FOR VEHICLE TRACKING AND ASSESSMENT IN WIDE AREA MOTION IMAGERY

A Thesis

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

KOYELI GANGULI

Dr. K. Palaniappan, Thesis Supervisor

Dr. F. Bunyak, Thesis Co-Supervisor

DEC 2010

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

SELECTED TECHNIQUES FOR VEHICLE TRACKING AND ASSESSMENT IN WIDE AREA MOTION IMAGERY

presented by Koyeli Ganguli,

a candidate for the degree of Master of Science,

and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. K. Palaniappan

Dr. F. Bunyak

Dr. J. Uhlmann

Dr. T. Han

ACKNOWLEDGEMENTS

There are several people who played an integral part in the research that led to this thesis. First and foremost, I would like to express my sincere gratitude to my advisor, Dr. K. Palaniappan, for introducing me to this project almost two years ago. His motivation, valuable guidance and encouragement have made me stay focused. I would like to thank Dr. Filiz Bunyak for her immense help throughout this research. Her strong guidance, advice and mentoring has played an integral part in this research without which this research would not have been in its final shape as it is today. I would also like to thank the other committee members Dr. Uhlmann and Dr. Han for kindly accepting to serve on my committee and for their advice and recommendations.

Special thanks to all my friends at the Palaniappan Lab for all their help and support and also to my other friends at University of Missouri. They have made these two years at the graduate school a very enjoyable and an enriching experience for me which I can relish for the rest of my life.

I would also like to take this opportunity to thank my parents without their inputs and guidance I would not have achieved whatever little I have today. Special thanks to my husband, Shramik Sengupta for providing me suggestions and ideas in writing this thesis. I am really grateful to him for making time from his busy schedule to proofread my thesis and providing his valuable feedback. His constant help, encouragement and motivation has been a very big driving factor.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Tracking of vehicles in wide area motion imagery (WAMI) is very important for civilian and military surveillance. Tracking in a dataset that is characterized by very large format video with an extremely wide field -of-view (covering few to tens of square miles), and with very minimal ground resolution (images taken at about 4000ft to 5000ft above ground) and with low frame rates (1-10 frames/ sec), is a very challenging job. Currently, analysts spend many hours manually tracking vehicles using this data. Efforts are underway to automate this tracking process, starting from region of interest selection to generating the track produced by the vehicle. This research describes some of the techniques and approaches taken towards developing a low frame rate automatic and assisted vehicle tracking system and also develops a performance evaluation system for low frame rate tracker. One approach that is taken on this challenging dataset is extracting roads from these images using the geo-registered property of the data. This makes the car detection algorithms using Bayesian approach run considerably faster and efficiently. Also, car tracking algorithms can use this apriori knowledge of roads. The car tracking algorithm using Camshift has been further modified/improved, customizing it to track cars better in this dataset. A performance evaluation system developed in this research can be used for measuring the performance improvement of the tracker as it advances over the coming years. It can also be used for parameter tuning. This performance evaluation system can be used for testing the tracker performance using two approaches, the approach using gaps and approach using tracklets. Both of these frameworks are developed using information theoretics measures and non-information theoretic measures.

CHAPTER 1

INTRODUCTION

1.1 Motivation

As video surveillance systems continue to cover larger areas at higher resolutions, the need for automated tracking systems becomes increasingly strong, so that the full use of the entire collected imagery can be accomplished by narrowing human analysis to those frames where trackers have detected interesting features.[2]. Wide area persistent airborne video enables continuous coverage of geographical regions on the order of few to tens of square miles. Manual tracking of vehicles requires long hours of toil and also suffers from human errors. Efforts are underway towards automating the tracking of vehicles.

1.2 Problem Definition and Goals

Very large format video or WAMI is obtained by an airborne camera with persistent observation. It is characterized by an extremely wide field of view, low frame rates between one to ten frames per second, very low ground resolution, continually varying relative pose between the sensor and the scene, urban scene complexity, large data volumes, artifacts arising from sensor geometry, low contrast, dense traffic or clutter, and frequent occlusions in urban regions with tall structures. All of these

present a new set of challenges for video object tracking. This kind of spatiotemporal coverage pattern cannot be accomplished using satellite imaging or would be more complex using a collection of distributed airborne camera arrays and sensor networks [3]. As a step towards developing a fully automatic tracker for this challenging dataset, we seek to develop a semi automatic tracker. In this research we improve, modify, customize different existing algorithms and test our own approaches on this dataset. Finally, we focus on developing a system which will assess the performance of the tracker based on information theory.

1.3 Thesis Organization

The aim of chapter 2 is to analyze the difficulties encountered in handling this data and identify the potential hurdles. In this chapter, first, the imaging system used in capturing the data is explained, this gives an idea about the uniqueness of the data. The second topic in this chapter discusses the challenges imposed by the dataset. Finally chapter 2 concludes by applying some popular computer vision algorithms like Mixture of Gaussians, Canny edge detection etc. The analysis of the performance (to these algorithms) gives further insight into the data characteristics and helps to build a better tracker. In CHAPTER 3, the geo-registration property feature of the data is explored. How this feature can be used to build a computationally less intensive tracking system is discussed. In CHAPTER 4 an improved technique of car detection using Bayesian networks is discussed. This technique is made computationally less intensive by using apriori information. In CHAPTER 5 car tracking is done with an apriori knowledge. Also, an existing car

tracking algorithm is modified and improved to suit the challenging dataset. In CHAPTER 6, a process to assess the performance of the tracker is presented. This is a very crucial step to not only understand the current performance of the tracker, but also to give an idea how much the tracker improved as it is modified over the coming years. The core components of the tracker performance evaluation system will be given and the results of evaluation on the current state of the tracker provided. The thesis is finally concluded in CHAPTER 7 with a discussion of possible future work.

CHAPTER 2

IMAGING SYSTEM, DATA CHARACTERISTICS OR CHALLENGES AND SIMPLE TECHNIQUES APPLIED

In this chapter we analyze the difficulties in handling this dataset and identify the potential hurdles. To do so, this chapter focuses primarily on three topics. The first section gives the description of the imaging system used for the data collection. This knowledge helps us understand why the data has some of the unique characteristics that it does. We then discuss the challenges of working with such a data and finally we apply certain common/popular image analysis algorithms to the data. The performance of these algorithms, especially their failures, yield insights regarding the improvements we must make, and/or contingencies that any software that we develop must handle.

2.1 Imaging System

In this research, we use images acquired from an eight camera array. The camera array is built by the Persistence Surveillance Systems (PSS). Figure 2.1 shows the whole imaging system and the path of the plane as it travels to capture the data. Each camera in the array produces an 11megapixel, 8 bit, grayscale image at one to four frames per second

(4096x2672). They are geo-registered to a 16Kx16K image mosaic with a ground sampling distance of 25cm. The pictures taken by these airborne camera arrays are combined with computational photography techniques to integrate the information from these multiple cameras across time in a consistent manner. Persistent wide area airborne sensors (sensors carried within the planes) typically use a continuous circular flight-path in a fixed 3D plane, perpendicular to the local ground plane. The orientation of the camera array remain fixed to a point on the ground. The flight pattern, combined with the effective field-of-view of the camera array enables persistent coverage of tens of square miles for long periods of time. At the higher spatial resolution and temporal sampling rate of 4 hertz the data volume is about 4 terabytes/sec of grayscale SD (secure digital) video. Figure 2.1 shows the imaging system [1].

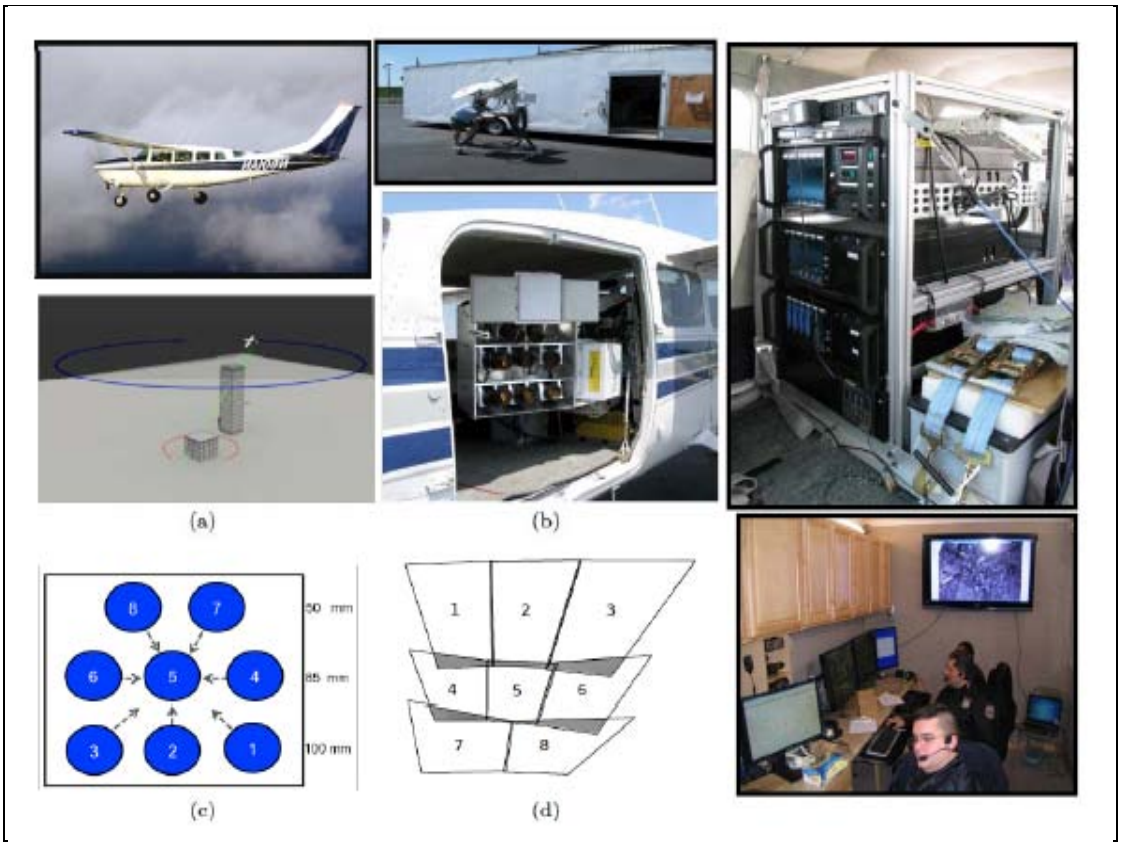


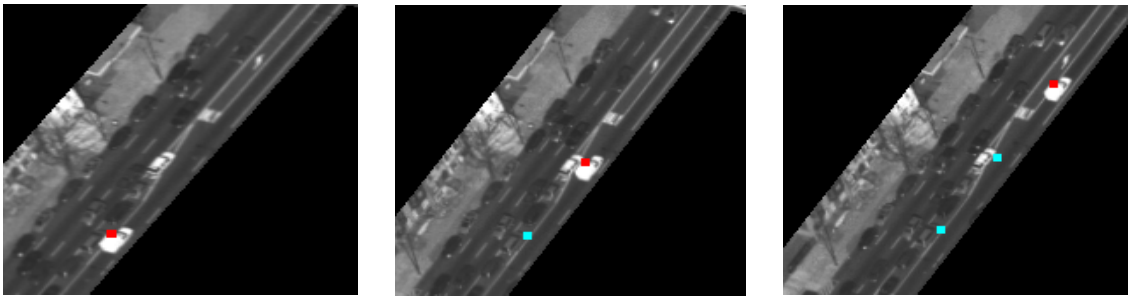
Figure 2.1: Imaging System (Image courtesy[1])

Figure 2.1(a) (top): shows the airplane that is involved in capturing images,

Figure 2.1(a) (bottom): shows the path of the airplane as it flies over a particular region. The image shows that projected path of the plane is bigger at higher altitudes and smaller at lower altitudes. Figure 2.1(b) (top): shows the trailer system that contains the imaging system to be loaded onto the plane. Figure 2.1(b) (bottom): shows the position of the cameras inside the plane. Figure 2.1(c) shows how the 8 cameras are oriented with respect to each other. Figure 2.1(d) shows the overlap between the field of view of the cameras. Figure 2-1(e) (top) shows the real time downlink system. Figure 2-1(e) (bottom) shows the crew at work in the base station.

2.2. Data Characteristics and Challenges

The video data obtained by the camera sensors within the plane has the following characteristics that make handling a difficult task.



Position of the car at time t-1 Position of the car at time t Position of the car at time t+1

Figure 2.2: Considerable displacement of car (low temporal sampling frame rate)

Red dot identifies the position of the car at that instant of time,

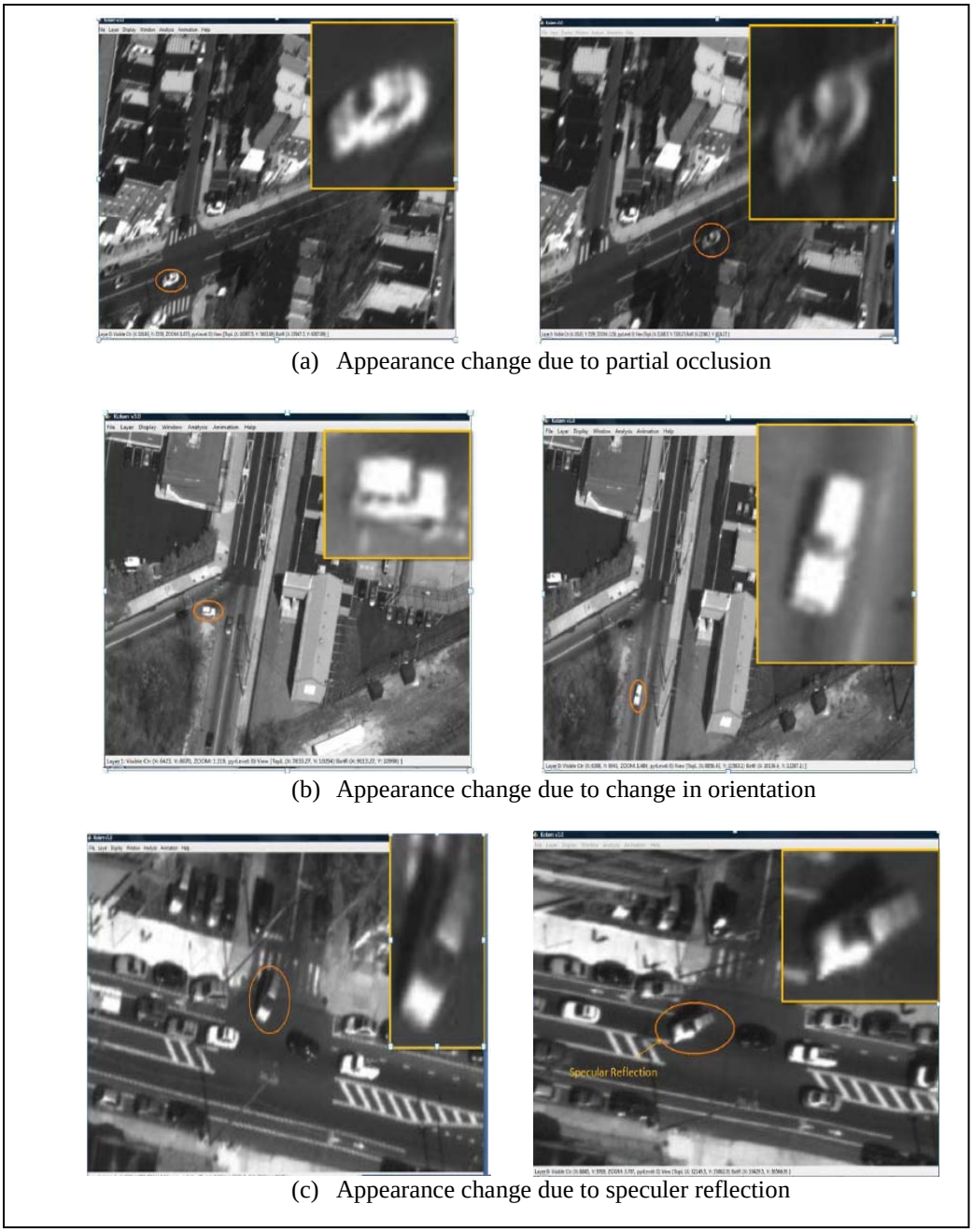
Cyan dot signifies the position of the car at the very previous time interval

The video has a very low frame rate about 1 frame per second. During these time intervals between 2 consecutive images, objects of interest (cars) may move a large distance. This makes it difficult for any algorithms to associate a particular car in one frame as the same car in the next frame. This is because the algorithms will try to locate the car in the next frame in a region closer to the previous frame.

Figure 2.2 shows the displacement of a particular car in consecutive frames.

Target appearance changes are observed due to partial occlusions, target orientations and camera viewing angle. Unlike in satellite images, the visual imagery of the objects obtained from orbiting airborne platforms have considerable changes in appearance within a short time due to viewing angle changes. Figure 2.3 shows changes

in appearance of targets due to partial occlusion, changes in orientation and due to specular reflection.



(a) Appearance change due to partial occlusion

(b) Appearance change due to change in orientation

(c) Appearance change due to speculer reflection

Figure: 2.3: Target Appearance change [1]

Figure 2.4 shows the challenges in handling low contrast in the images. As can be seen from these images, low contrast makes the car almost invisible in the background, (figure 2.4 a). In figure 2.4(b), the car is against a slightly better contrast. When algorithms are applied on a good contrast, it will be able to differentiate between the background and the target object (car), and thus the probability of detecting the car will be much better. So when developing algorithms to work on this dataset, the challenges of working with low contrast images must be kept in mind.

Figure 2.4: Low Contrast Challenges

Camera stitches are visible in the image, the registration between them are not accurate. This makes the appearance of the target object and also of the background change from frame to frame. As a result, automatic algorithms fail to detect the target object in the next frame. Figure 2.5 shows an example

Figure 2.5: Camera stitch and Improper Registration in image (making car detection difficult)

As can be seen, handling the Persistence Surveillance Dataset is extremely challenging. In order to get a further insight into this data, some popular image processing algorithms are applied on it. The performance of these algorithms, especially their failures insights regarding the improvements we must make, and/or contingencies that any code that we develop must handle.

2.3 Applying Selected Image Processing Algorithms

Some popular Image Processing Algorithms were tried on the dataset. Given below are the results and analysis of them.

2.3.1 Applying Canny Edge Detector

Canny edge detector was applied on the image to extract the roads from the image. Different thresholds were tried. It was seen that by choosing a very low threshold, all the edges in the image were visible. Also, it was seen that by applying high threshold the roads were not visible in the image. Detecting the roads from that image was very

challenging, just by applying this algorithm. As the intensity of the edges in the image were very close to each other, an optimum threshold at which the roads can be made visible by suppressing the other edges in the image could not be obtained. Figure 2.6 shows the result after applying the canny detector.

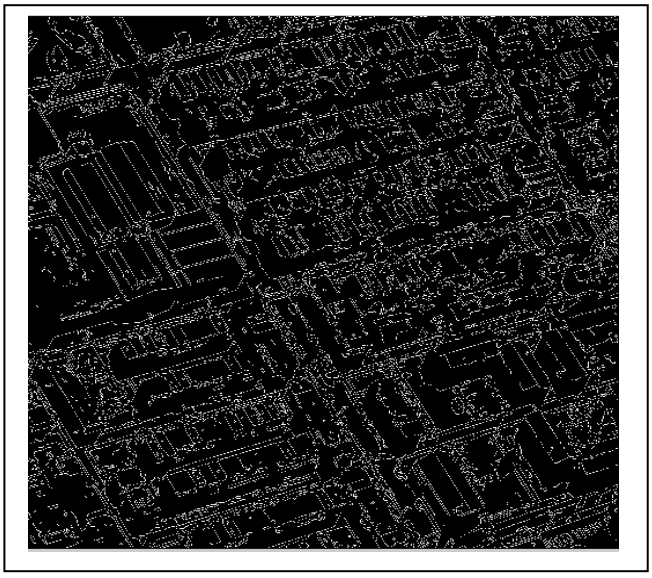


Figure 2.6: Result of Canny edge detector on the image

2.3.2 Applying Hough Transform in the image to detect lines

The Hough Transform was then tried on the image to detect the lines. The results are shown below:

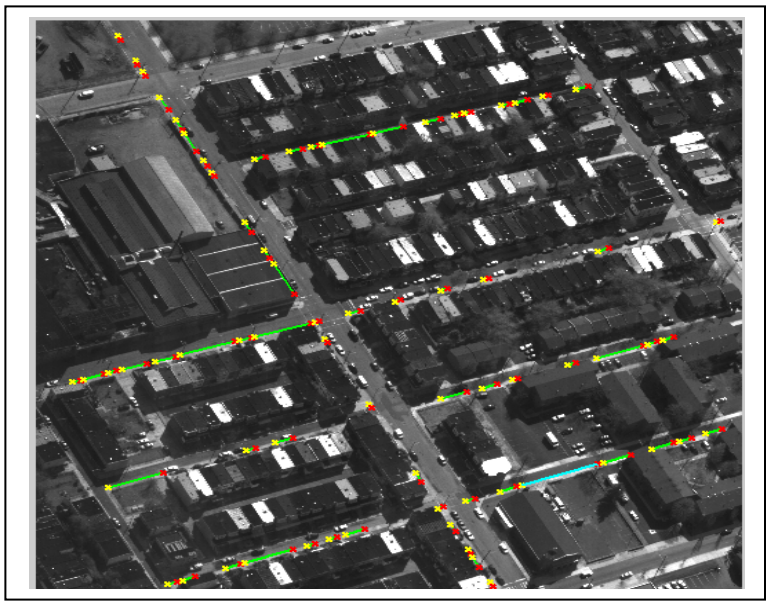


Figure 2.7: Result of Hough Transform on the image

As can be seen from the above result, the lines were detected along the edges of the curbs, but whenever there was a crosswalk or intersection, or places where the curbs did not exist, the lines were lost. Different values were tried for the rho resolution and the theta resolution, but they all failed to address the basic problem of discontinuity. The image had to be processed further in order to extract the roads from it. Application of just the Hough transform was not enough.

2.3.3 Applying MOG technique for car detection

Mixture of Gaussian is a robust technique for background subtraction. Mixture of Gaussian [4] was applied on the images and few of the results shown in figure 2.8.

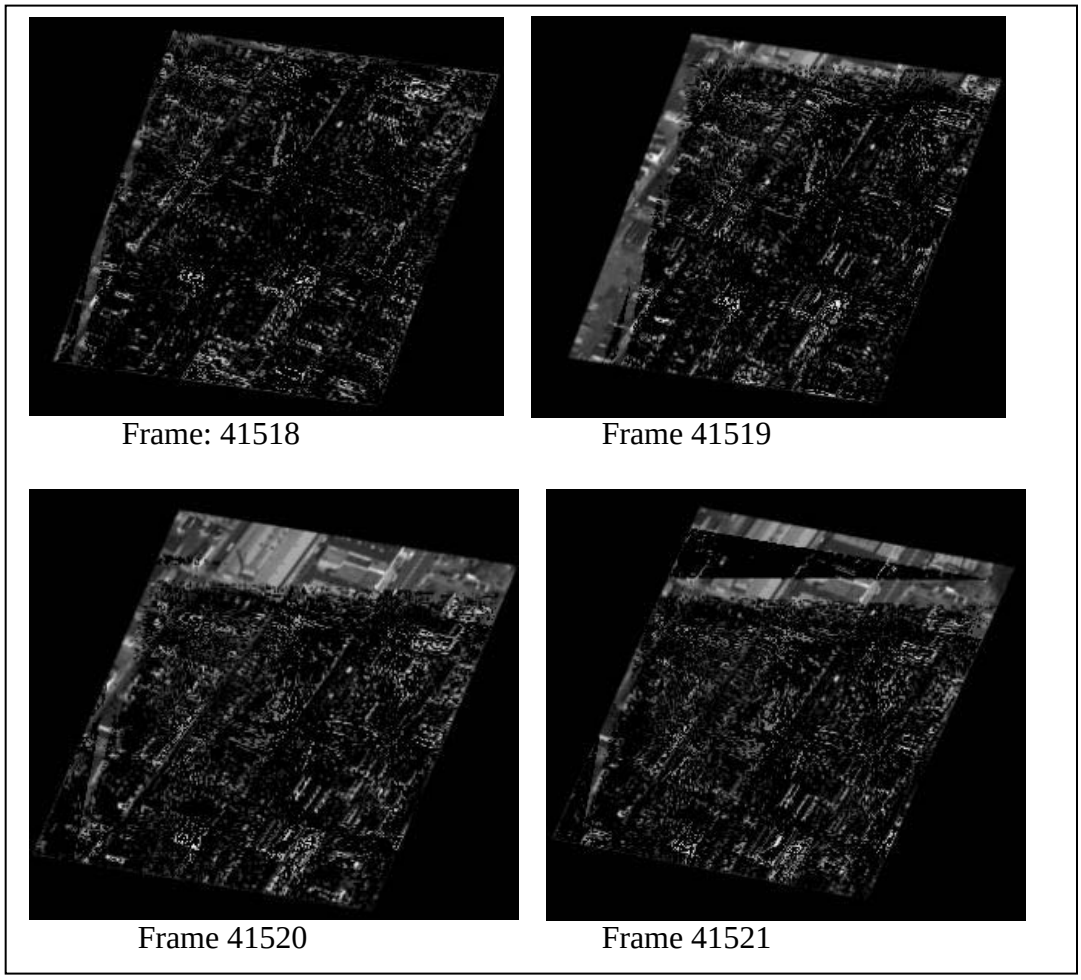


Figure 2.8: MOG results on 4 consecutive frames

As can be seen above the results, the cars cannot be detected after applying mixture of Gaussians. The reason can be stated as follows: Mixture of Gaussian is able to extract the foreground from static backgrounds. However, in this dataset, if we consider the roads and buildings as backgrounds and the cars as our object of interest, just by applying the mixture of Gaussians we will not be able to extract the cars from the image. As the backgrounds in this dataset are not static. Due to the circular motion of the camera, roads, buildings and other objects in the background appear to be moving along with the

movement of the foreground object (cars). Hence, by applying mixture of Gaussians these background objects cannot be eliminated.

2.4 Conclusion

In this chapter, we gained knowledge on how the data was captured. The challenges in handling this dataset were also explored. Many different common/ popular image processing algorithms were applied on the dataset. The performance of these algorithms leads to the conclusion that none of these methods work. The motion of the backgrounds in the image makes it impossible to extract the foreground by any static background subtraction techniques like w4, wallflower, MOG etc. The low resolution, considerable amount of rectangular shaped objects in the image makes it very difficult to apply some common edge detection techniques like canny edge detection and Hough transform.

Hence, when we develop any algorithm, the challenges of working with this dataset must be kept in mind. In order to track cars in this dataset, one of the main things that should be kept in mind is that the algorithms should be made robust and computationally less intensive. To achieve this, the problem is broken down into smaller modules and then each of these modules is addressed. The three major modules that I have worked on are the road extraction, car detection and then the car tracking. In order to evaluate the performance of the tracker, I have also developed an evaluation tool for tracker performance evaluation.

CHAPTER 3

GEOREGISTRATION AND ROAD NETWORK ANALYSIS FOR WAMI

3.1 Introduction

As explained in the previous chapter, the nature of the data is such that achieving our objective (tracking cars) is very challenging. In order for a tracking algorithm to be robust enough for this dataset, it is important that it approach the problem intelligently. It should take into consideration factors such as, the places to look for the cars are only the roads. This will cut down on the processing time as it will not look for cars where it cannot exist, for eg: structures in the image that are not roads. For this we perform pre-processing. In this preprocessing step, the data can be filtered to identify only the region where it is most probable to find cars. For instance, if the roads can be extracted from the image and then the car tracking algorithm be applied on the streets, then it may increase the efficiency and robustness of the algorithm. This will prevent our tracker from misunderstanding objects which have similar shapes and appearances as cars to be cars. Also, since the tracker will be working on a subset of the data, it will be less

computationally intensive. In this chapter, we will explore this approach. In other words we will first extract the roads, then we will use the extracted roads to detect and finally track cars in it.

The extraction of roads is helped by the fact that the PSS image is a geo-registered image, so every point in the image has a known latitude and longitude. In this road network extraction method we use this latitude/longitude information to extract out the road network with the help of maps obtained from TIGER database[5] or Google Earth. I will discuss the two different approaches taken to accomplish the road extraction. Briefly explain the advantages of both the approaches and finally, the scope of improvement of this work.

3.2 Literature Review

The extraction of roads in a complex urban scene is a very challenging issue in photogrammetry and computer vision [6]. Many different groups have worked on road extraction. Jiuxiang Hu et al.[7] use a two step process of road network extraction from aerial images. First step is the shape classification of a local homogeneous region around a pixel and the second step is the road tree pruning step based on the area-to-perimeter ratio of the footprint to prune the paths that leak into the surroundings. Katherine Treash et al [8] use steps like edge detection, edge thinning and edge linking for automatic road detection in grayscale aerial images. Koutaki et al. [9] uses ribbon snake method for road extraction. However, these approaches require a lot of computation and extracted roads require a lot of post editing. Hence, applying these techniques to the PSS imagery will be very difficult as it is very complex. The geo-registration feature of the PSS images can be

used to detect the roads in the image. This technique is very fast and is extremely computationally less intensive. This is the major plus point of this technique, as the road detection process is just a module in the bigger goal of car tracking in these PSS images.

3.3 Overview of the Approaches Taken

The technique taken to extract the roads from the PSS imagery is novel but simple. Since these images are geo-registered, every point within the image is defined by a set of latitude and longitude points (available upto 13 decimal places). The basic concept for the 2 approaches remain the same, but the way they can be applied differs quite a bit. In the approach using the Google Earth, the starting and ending points of a particular road segment is specified. Then it is extracted from the Google Earth image by processing for it. In the approach using the TIGER database, a bounding box encompassing a particular region is specified. The bounding box is defined by the upper left hand lat/lon co-ordinates and the lower right hand co-ordinates. These co-ordinates are passed into the TIGER database and processed to get all the roads within that bounding box. I will explain the details of applying these approaches, their difficulties and advantages next, starting with the approach using Google Earth images.

3.4 Extracting Roads using Google Earth Imagery

Google Earth images contains the most comprehensive geo-spatial database information from around the globe. This approach used the following steps:

In Google earth instead of specifying 2 sets of lat/lon co-ordinates (i.e the start and end point of a road segment), provision was there to specify only 1 set of co-ordinate. So we specified any one of the lat/lon co-ordinates upto 2 significant figures. This returned a very large area. Then the road which was present in the PSS frame was searched manually inside this image. Once the road was obtained, they were marked out using a suitable color. Then this image from Google earth was saved to be processed and extract the roads. Figure3.18 shows a Google earth image, with the roads highlighted for our purpose.

Figure 3.1: Google Earth Image with road highlighted in red:

Latitude: 39°59'22.30"N, Longitude: 75° 9'16.19"W

3.4.1 Advantages of this approach

Google earth is a very trusted database to download cartographic information from places all around the globe. Roads extracted from this database can be assumed to be free of errors. Since Google earth maps are updated at fairly frequent intervals, the problem of

dealing with older maps will not be there. The approach will work not only for the places in USA but all around the globe.

3.4.2 Difficulties with this approach

This approach suffered from a number of difficulties. They can be enumerated as follows:

It would be better if we specify a region (defined by a bounding box) and get all the roads within that region at once. This way we prevent ourselves from searching a large region for a particular road and then feeding it into our tracking system (one at a time). But in order to specify a region defined by a bounding box, atleast 2 sets of lat/lon co-ordinates needed to be specified to Google Earth. But, in Google earth we could specify only one lat/lon co-ordinates and not 2 sets of co-ordinates at once.

The second difficulty was with respect to registering the PSS data with Google Earth to extract the roads from PSS images. In order to extract the roads highlighted in Google Earth from PSS images. We first need to register Google Earth with the PSS images, so that both the images belong to the same co-ordinate system. But registering Google Earth images to PSS was a big challenge. It is because, in Google Earth, images are not ortho-rectified (according to Google earth community). But our images are ortho-rectified. Figure 3.2 gives an example.

Figure: 3.2 Problems of registering PSS data with Google earth(due to difference in views) –Top: PSS data from frame 45700, Google Earth image around 40.004009, -75.158926

In order to solve the problem we tried to extract the roads from google maps, as Google maps were ortho-rectified. But this posed another problem as the maps in the Google map image could not be save as image file.

In order to mitigate the difficulties of the above approach another approach was tried using the TIGER MAPS database [5]. This is explained below in the next section.

3.5 Extracting Roads Using the Tiger Maps Database

This approach involved using the TIGER MAPS Database [5] network. The TIGER/Line files are extracts of selected geographic and cartographic information from the Census Bureau's TIGER (Topologically Integrated Geographic Encoding and Referencing) database. From this database, cartographic maps in the form of shape files, road vectors can be obtained. The steps taken for this are as follows:

All the cartographic information, from this database were first downloaded and placed in our local system, so that henceforth no remote connection to the TIGER database server will be necessary.

The Matlab Mapping Toolbox [10] was used to work with this dataset. This toolbox provides tools and utilities for analyzing geographic data and creating map displays. So with this Matlab utility the bounding box of the Region of Interest was provided (i.e. the top left hand corner co-ordinates and the bottom right hand corner co-ordinates). A call to the database was made to extract the road vectors within that bounding box region. Figure 3.3 shows an example of a road vector produced.

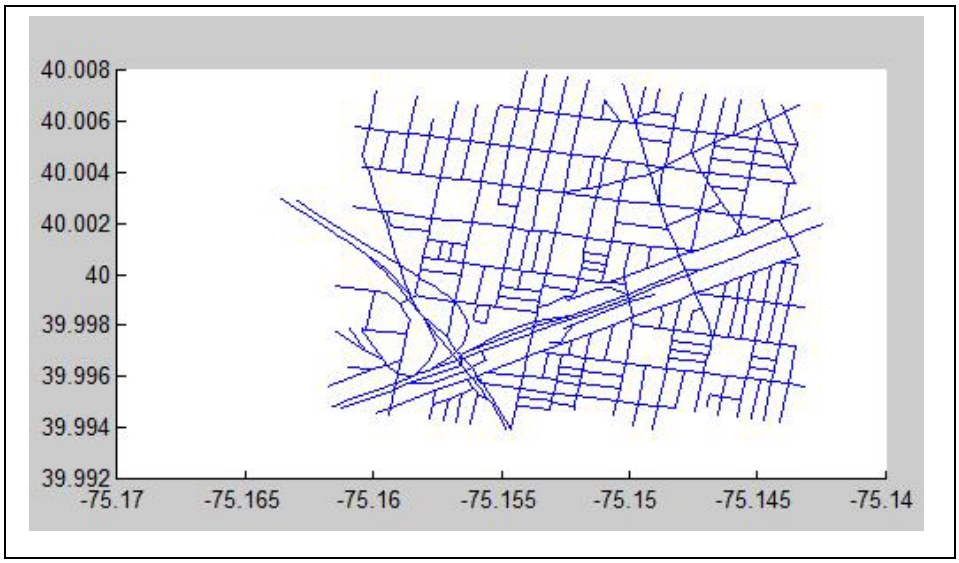


Figure 3.3: Road vector obtained for bounding box `[-75.1603034 39.9947854; -75.1435616 40.0065253]`

Once the road vector was obtained it was superimposed on the image to extract the roads from the image by it. Figure 3.4 shows the road network obtained in figure 3.3 superimposed on the PSS image. In our applications we have made use of the angle of these road vectors with the x-axis. For eg: for our car detection exercise, instead of searching for a car in all the 360 degree directions, we look for the car only at the directions similar to the road vectors at the point. This approach thus lessened our computation and helped our application to run faster.

Figure 3.4: Road vectors obtained from TIGER database superimposed on PSS image

3.5.1 Advantages of this approach

This approach is a way of extracting the roads from the images using the geospatial information of images. It does not involve much computation. It is very fast and easy, with the use of the Matlab mapping toolbox, this requires minimal computation. It just required the image to be geo-registered, since it is a geo-registered based approach

3.5.2 Disadvantages of this approach

Nonetheless, this approach suffered from a few faults.

First of all, this approach of using the TIGER database would work with only the roads in US. Since, our project currently focuses on USA this posed no problem, but if in future this project is extended to other countries, then this approach would not work, as maps/vector data/shape files are not available for other countries in this database

The other important disadvantage of this approach is that sometimes, the road network extracted from this database, when superimposed with the PSS images introduces a varying offset of 120 pixels-301 pixels in a subimage of 1500 x 1500 pixels. The road vectors fall on the allies or sometimes even on buildings and houses as shown in the figure below. Figure 3.5 shows 2 such scenarios.

Figure 3.5: Confusion introduced by Tiger Maps: Top: to which road does the line highlighted in blue fall

In some cases, Tiger Maps and Google Maps/Google Earth when superimposed do not match, whereas in some cases they do. Figure 3.6 shows the mismatch

Figure 3.6: Tiger Vectors overlaid on Google Earth Images

Figure 3.6(a) shows mismatch when TIGER maps superimposed on Google maps/Google Earth, figure 3.6(b) shows TIGER maps almost matching up with actual roads in PSS images.

From the above figure, it can be concluded that the TIGER maps work for some scenarios and do not work for some. When they do, they are very easy to implement and computationally not at all intensive. If the errors that this database has for certain scenarios can be minimized, or the registration of the PSS image to the co-ordinate of the earth is made more perfect,then this technique will serve as a robust, fast, easy, efficient way of road detection from geo-registered images.

3.6 Conclusion

In this chapter a simple technique to obtain road masks from geo-registered images. The beauty of the technique lies in the fact that it is, an extremely computationally less intensive technique. This was one of the main reasons which prompted us to explore such techniques, primarily because road extraction is a step towards tracking cars in the highly complex WAMI data with limited resolution. Two different approaches were investigated and applied. The first approach using the Google Earth, could not be used because of the unavailability of the Google maps as images or the non-orthorectified view of Google Earth images. If any one of these can be provided then the approach would work. For the second approach using the TIGER database [5], this approach can be made more robust by using a more accurate database, but for which the underlying principle of operation is the same as the TIGER database. Another way to make this technique more robust, is to make the geo-registration of the PSS images more accurate.

CHAPTER 4

CAR DETECTION USING BAYESIAN NETWORK S

4.1 Introduction

In the last chapter we extracted road network from the images. Now, in this chapter we develop algorithm to detect cars on the extracted roads. Car detection is a necessary prerequisite towards car tracking. Hence, our approach to track cars in the Persistence Surveillance dataset is to first identify those present in the areas delineated as roads by the earlier algorithm. The apriori information incorporated into the car detection algorithm makes it run much faster and more efficiently. It also helps to narrow the search area, thereby making the algorithm more robust, by looking for cars only on the roads, screening out other objects like buildings etc which sometimes look like cars. Features that are used for car detection are the boundary of the car and the boundary of its front windshield. It is observed that these features are dependant on the intensity of the car, (i.e. whether the car is a dark car or a gray car or a black car). These features are represented as a Bayesian network, which is used to integrate all the features [11]. The Bayesian network is a probabilistic graphical model that represents a set of random

variables and their conditional dependencies via a directed acyclic graph. The road network information of the images are used to align the cars in the images in the same direction as the orientation in the template. We are able to run the car detection algorithm more efficiently and in an extremely computationally less intensive manner by incorporating the road network information in the car detection algorithm. The details are given within this chapter.

4.2 Challenges

As explained in Chapter 2, the Persistence Surveillance Dataset, has grayscale images and poses a number of challenges. Some of the challenges that are especially relevant to identifying cars include:

As the viewpoint is constrained, this makes the car have different appearances in different angles. Also there is change in appearance due to different degrees of partial, specular reflection and changes in orientation. As the image resolution is low, not many details are available. The car is typically 30-40 px in length. This makes it difficult to distinguish one car from another and/or a car from a small building. Cars in this image are of varying intensities, ranging from very dark to very light. The images have low contrast, this makes some cars' intensity very close to the road and hence indistinguishable from the road.

4.3 Literature Review

Car detection in computer vision has been tried out by different groups. Papageorgiou and Poggio [12] uses a learning based approach by deriving a model of an object class by training a SVM classifier using a large set of positive and negative samples. Schlosser et al. [13] do car extraction by matching the model top down to the image and evaluating the support found in the image.

Schneiderman and Kanade [14] use decision rule to determine if the object is present at that specific orientation. The decision rules use the statistics of object appearance and non-object visual appearance. Then they represent each set of statistics using a product of histograms. Rajagopalan and Chellappa [15] use HOS for pattern classification. They model the unknown distribution of the image patterns of vehicles by learning HOS information about the vehicle class from sample images. For most of these works the resolution of the vehicle is high and there is little angle of view change. Thus, in order to make the car detection algorithm work in low resolutions and high view change as is the case with our dataset, we make use of its features.

4.4 Approach Taken

This work builds on the work of Zhao et al. [11], there is significant computation optimization that has been done here. Instead of trying to find cars in all (360 degrees) directions, we try to find the cars along the orientations of road. This saved a lot of computation. Zhao et al. in their work, estimates the road by clustering straight lines in an image considering the fact that most urban lines in an image are aligned with the direction of roads [11] and they use Hough transform for this purpose. But, this method is

much more error prone, as the lines detected by the Hough transform may not be totally accurate. Also, a lot of computation and preprocessing has to be done to extract the lines by Hough Transform approach. So, in our work, we have applied a different technique of road extraction, we have used the information from the TIGER/LINE [5] database. This database has all the road information of US cities. We could use this information because the images in our dataset are geo-registered (mapped to the co-ordinates of the earth). This approach is much better than the approach given in the Nevatia [11] paper because of the following reasons, first it saves a lot of computation. Also, this technique is much more error free.

The features that are used to detect the car are the four boundaries of the car and the boundary of its windshield. It is seen that these features are affected by the intensity of the car. This knowledge is then incorporated into the structure of the Bayesian Network (BN), which is used to combine all features. A training data is provided to BN to learn the parameters of the model. Finally, the decision whether a car is detected or not is made using a Bayesian minimum risk classifier.

4.5 Feature Extraction

4.5.1 Road Network Extraction

The Persistence Surveillance dataset images are geo-registered, meaning that each point within the image is a pair of latitude/longitude co-ordinate. The TIGER/LINE database is used to extract the roads as is explained in Chapter 3. The road networks detected by the TIGER/LINE database [5] is shown below in Figure 4.1. The red lines are the road

networks are obtained from the TIGER/LINE database, they are superimposed on the image to show the match with the actual roads.

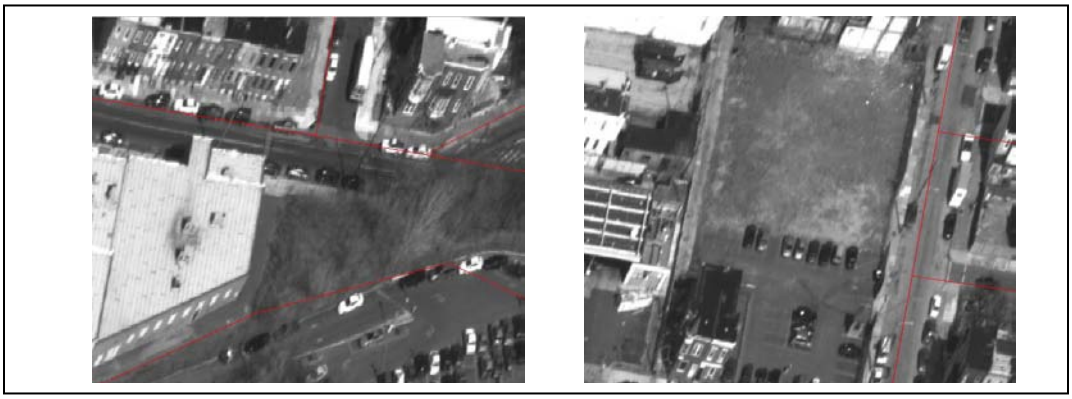
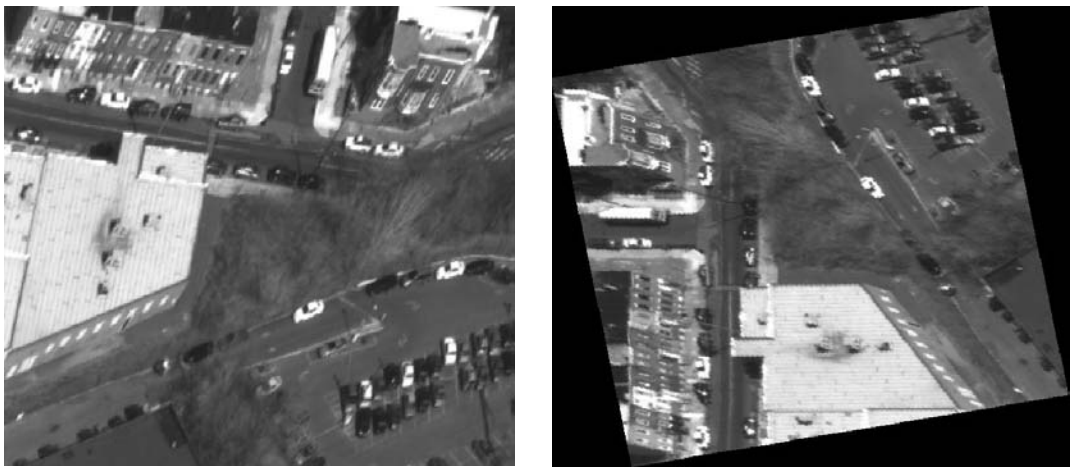
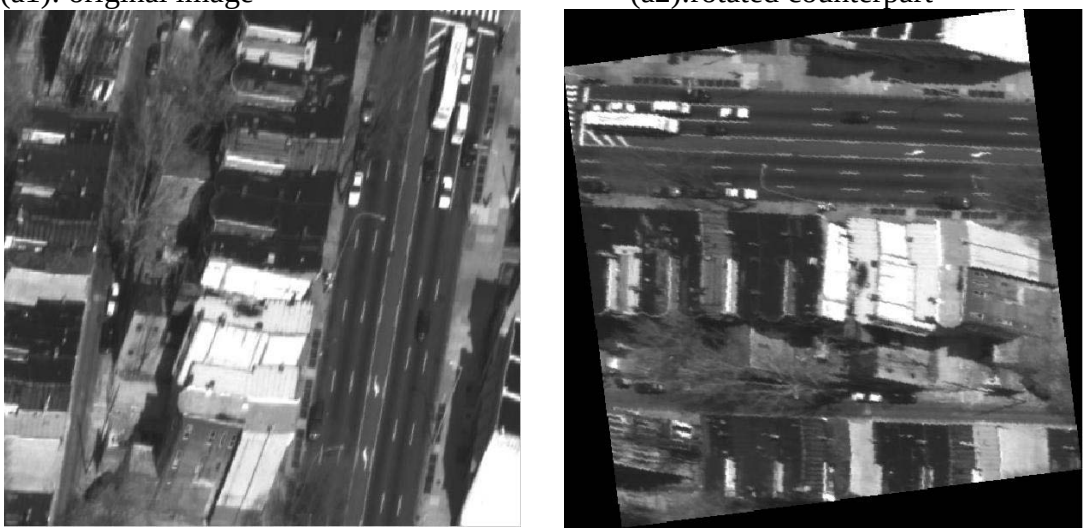


Figure 4.1: Road vectors from TIGER database superimposed on the image

After extracting the roads as lines, the slopes of the lines are computed. Then these slopes are clustered. If the slope of the different lines varies by less than 5 degrees, it is considered to be of the same slope. The resultant slope of those lines which varies from each other by less than or equal to 5 degrees are taken as their average. Only if the slope of two different lines varies by more than 5 degrees, it is considered to be lines with 2 different slopes. Once the slope of the lines are obtained the image patch is rotated to make the direction of interest horizontal in the image. Typically, one image is rotated with respect to the number of slopes of lines obtained from that image. Each rotated image is treated separately, the car detection algorithm using Bayesian Networks is applied on each of them. Figure 4.2 shows an image and then its rotated counterpart.



(a1): original image (a2):rotated counterpart



(b1): Original Image (b2): Rotated counterpart

Figure 4.2: Image and its rotated counterpart

(a1 and b1 are 2 different images a2 and b2 are their rotated counterparts respectively)

4.5.2 Features Used for Detection

The different features used are as follows:

Boundary of the car: The boundary of the car is mostly rectangular. Figure 4.3(a) shows a typical car template and its boundaries.

Boundary of the front windshield: The boundary of only the front windshield is used, because this front windshield remains relatively constant with respect to the shape, size and location in the car. It is always assumed to be rectangular. Figure 4.3 (b) shows a typical window templates.

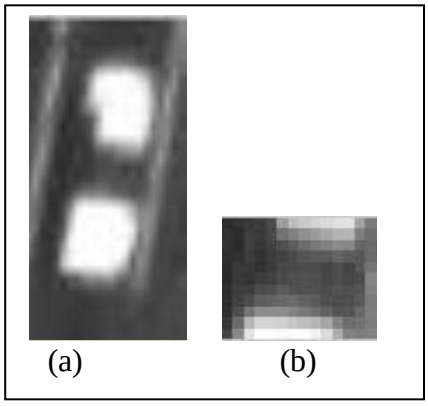
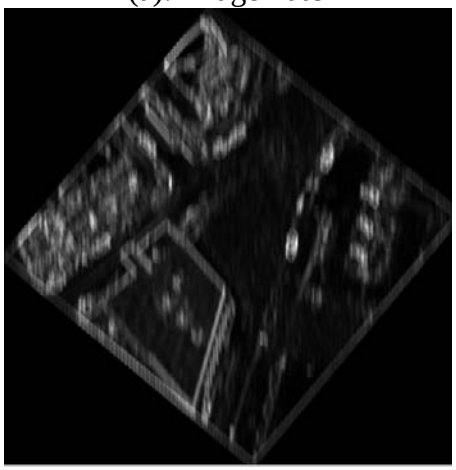


Figure 4.3: Typical car template and its window

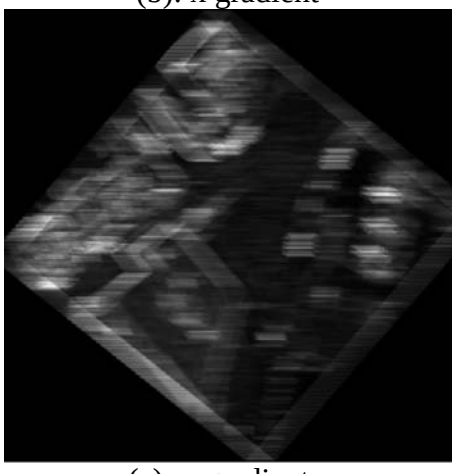
As it is seen, the features used are mostly gradient features that are linear or almost linear in shape. To extract the edges, gradient filters are used. There are three important reasons why this is done: First due to the small size of the car and possible occlusions in the image, edges of the car may be lost or fragmented. Hence, it may be difficult to extract them in a robust manner. Second some features are not always linear, so for these features, it will not be sufficient to extract edges. Third, the response of the gradient filter has better resolution in amplitude than the binary valued edge detector [11]. Hence, the features are represented as gradient (vertical or horizontal) masks. These masks are convolved with the images to get the value of corresponding features.

The figure 4.4 shows an image patch and the result of applying the x/y gradient map on the image.

(a): Image Patch



(b): x gradient



(c): y gradient

Figure 4.4: Image patch and its x/y gradient map

4.5.3 Model Based Feature Prediction

To detect cars using the above gradient, their shapes need to be known.

The user of this system is asked to randomly pick a car from this dataset and crop its boundaries and the boundary of its window. From this model, the shape of the boundaries of the car and the window are determined, i.e the 2d location of the edges of the car and its window. Specifically, the shape of the left/right, top/bottom, edges of the car boundary need to be known, relative to the center of the car. Also, the shape of the left/right top/bottom of the front windshield need to be known relative to the center of the window. The figure 4.5 shows the 4 different features (top, bottom, right, left edges) of the car and the 4 different features (top, bottom, right, left edges) of the car-window obtained from the x and y gradient, by shifting the image, according to the size and the orientation of the car.

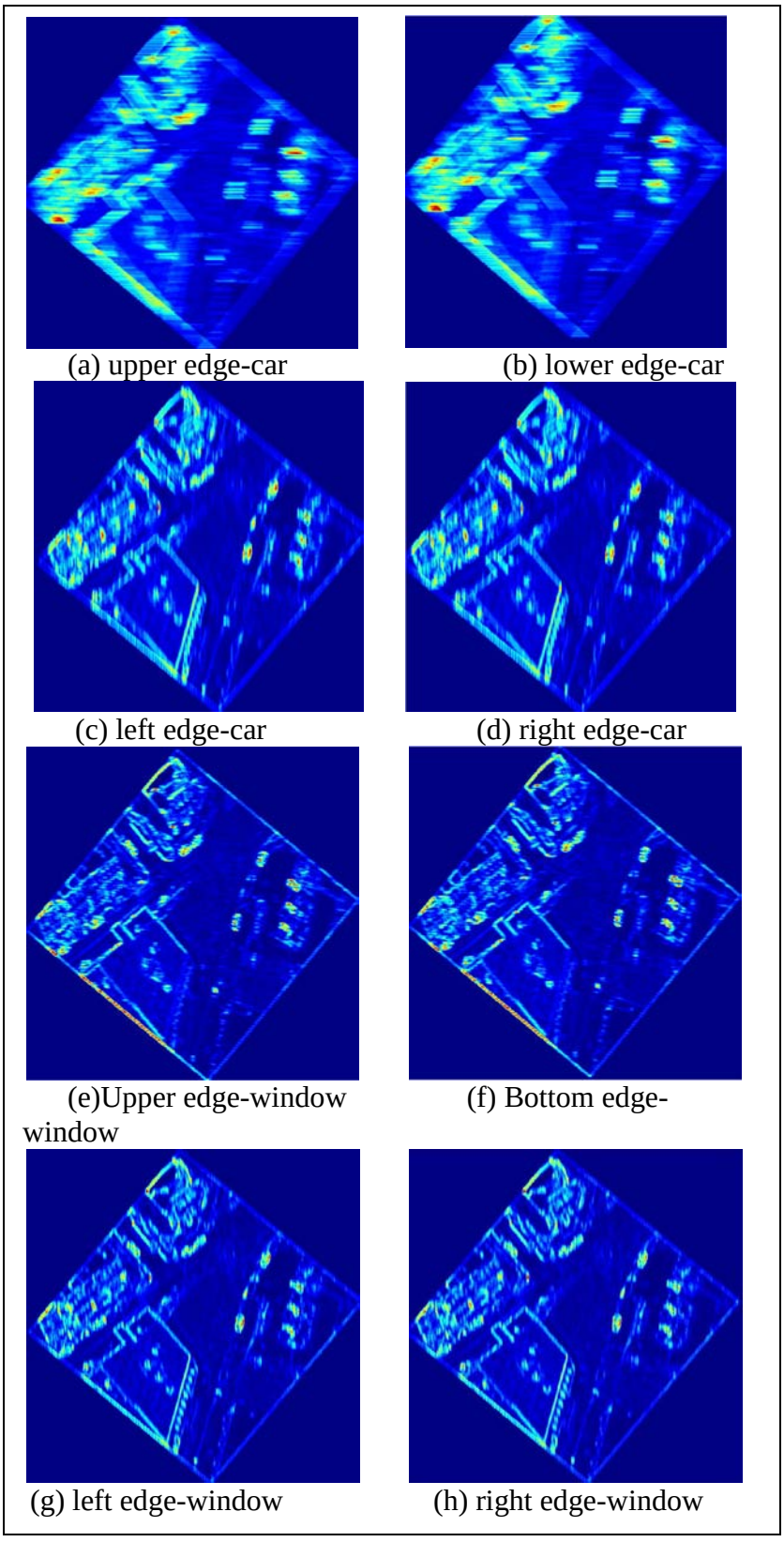


Figure 4.5: 8 Features of the car and its window

4.6 Multi-feature Integration

4.6.1 Parameterization of the features

It is observed that the features of the car are dependent on its intensity. In other words, the values of the four boundaries of the car as well as the four boundaries of the window are affected by its intensity. Hence, these factors are quantized into discrete values for simplicity. The intensity is quantized into three values dark, gray and white. The quantization is based on simple thresholding method. The figure 4.6 shows an image patch and the following three images show the pixels classified as gray, dark and white, highlighted in yellow, red and blue respectively.

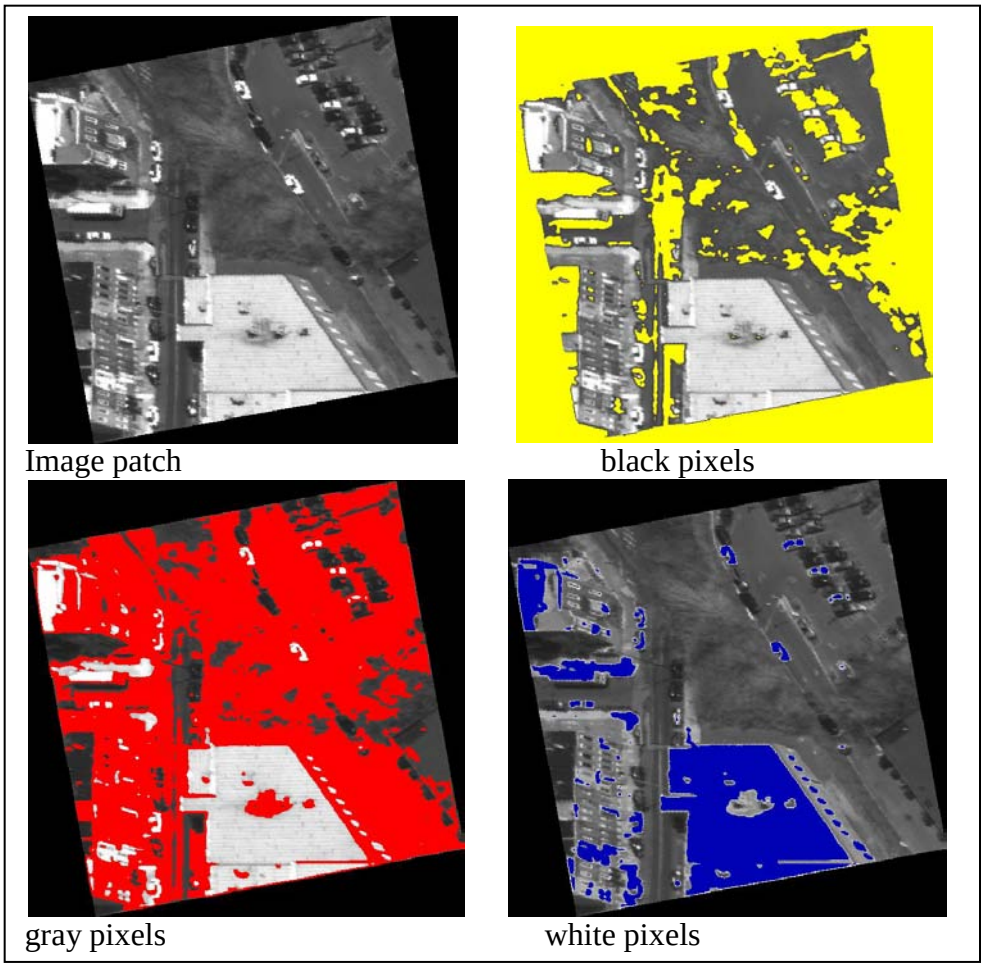


Figure 4.6: Classification of pixels in the image patch

4.6.2 Integration through a Bayesian Network

With all the available evidences, there are 8 features to use, they are the 4 boundaries of the car and the 4 boundaries of its front wind shield. These are then combined to give the final decision (probability) as to whether a car is detected or not. Bayesian network provide an optimal way to integrate multiple features for a decision, if their conditional distributions are known [11]. The structure of the Bayesian network used to integrate all the available evidences is shown in figure 4.7 is.

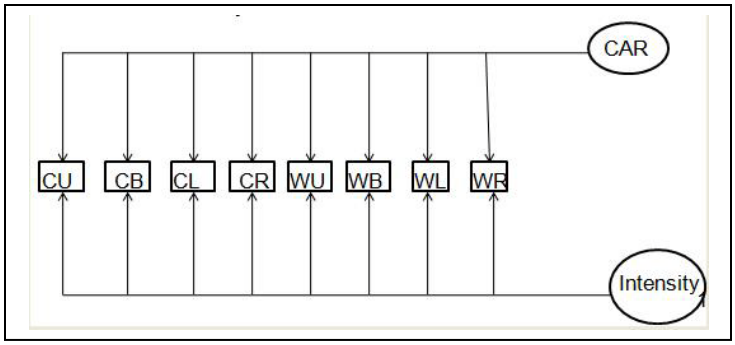


Figure 4.7: Bayesian Network

The factor intensity is the parent node. The posterior probability, P(car|F,i) of the node 'car' (where F = [f1, f2,..fn] are all the available features) is the one to be evaluated. The values of the evidence nodes are measured from the image. Equal probability is given to the finding of a car or a no-car in the image, and the weight is given as 50%. The following steps are taken for finding the values of the evidence nodes in the image. :

First the image is smoothed, to give better results. For smoothing, Opening with Reconstruction and Closing with Reconstruction are performed on the image. The Opening with Reconstruction is an erosion operation followed by reconstruction. The Closing with Reconstruction is dilation operation followed by reconstruction.

Second, the location of the edges of the car has already been determined with respect to the center of the car, as described in the model based feature prediction module.

Third, the gradient of the image is taken in the X and Y direction. This is done by convoluting the image with gradient filters. In this case, a simple gradient filter is used in the X and the Y direction whose sum would add up to 1. The probability of finding the vertical and horizontal edges are then obtained by convoluting the derived image with another filter of same size as the object (in this case, either the windshield or the car).

Finally, the probabilities of observing the vertical and horizontal edges are shifted to find the probabilities of the upper/bottom/right/left edge of the car with respect to its center (or to find the probabilities of the upper/bottom/right/left edge of the window with respect to its center).

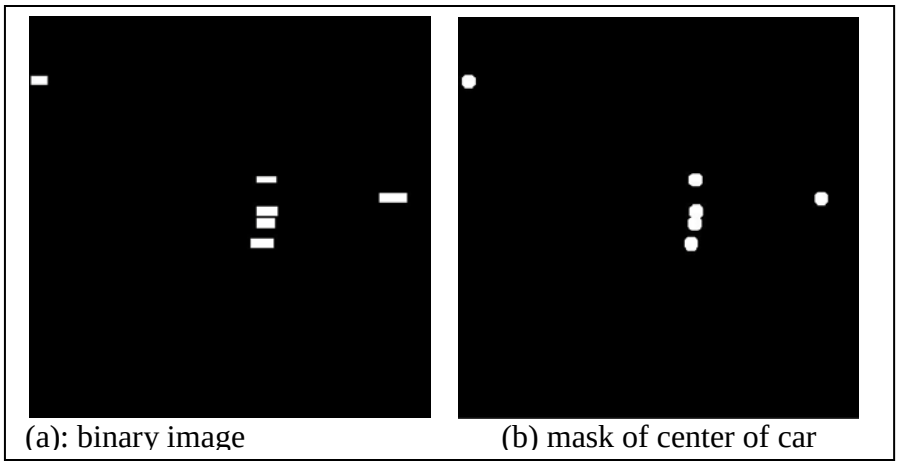
The above steps, gives the values of the evidence nodes from the image at its each (x,y) location.

To obtain the parameter node intensity' each pixel in the image is classified as dark, gray and white pixel (as described in the feature parameterization module, section 4.6.1). Each of the evidence nodes has a conditional probability associated with it that is indexed by the value combination of its parents. For example the conditional probability of the node BU is in the form P(car|BU, i). It is worth mentioning that the Bayesian Network assumes conditional independency, i.e the values of the feature nodes are assumed independent when all their parent nodes are fixed. In this case, a parameter node is introduced to achieve conditional independency.

4.6.3 Learning the parameters of the Bayesian

Learning the parameters of the Bayesian Network is very essential, because then the data can be modeled with much finer resolution. Since, there are no hidden nodes in the network, learning the conditional probability involves just calculating the histogram. By the conditional independence assumption, the conditional probability of each evidence node can be learnt independently. The following steps are involved in generating the distribution of each feature with respect to the intensity of the car:

First a training image is chosen and 15 cars are chosen from the image, to serve as the dark, gray and white cars. These are the positive examples. The other parts of the image serve as the negative example. Then, three binary images are formed from the training image, (one which has the patches of the 15 cars marked out). These correspond to the binary image of the dark, gray and white car respectively. The white region in the binary images correspond to the patch of the gray cars and the black region corresponds to the non-gray car region. Once the binary images are formed, the center of the cars are obtained from the center of the white regions within the binary image. These serve as the mask. Figure 4.8 shows a typical binary image of a gray car and the mask.



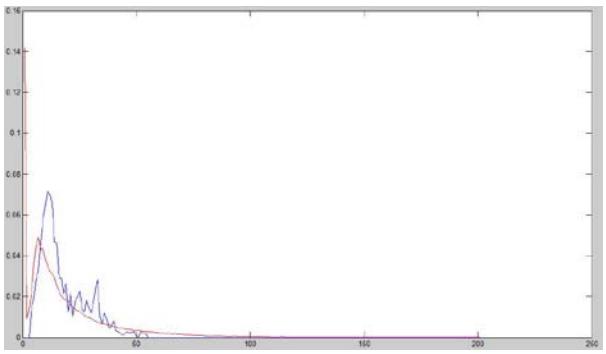
(a): binary image (b) mask of center of car

Figure 4.8: Formation of a typical mask of the center of car

The next step is to compute the probability of observing the different features of the car, with respect to its center. This is done in the same way as described in (the finding the values (probabilities) of the evidence nodes in the image) in section 4.6.2

Then, with these probabilities the distributions (i.e histograms of the feature and no feature of gray, black and white cars are plotted) using the mask obtained from the training set. (obtained from step 3 in this section).

It is seen that the results are better if the histograms obtained are normalized and then smoothed. In this research, a simple smoothing is performed with a window span of 10. Figure 4.9 shows 2 different histograms one with smoothing and the other without smoothing. Along the x-axis are the values of the different features, for our convenience, we have scaled all the feature values to a range within 0-200. Along the y-axis are the probabilities of finding the different features.



(a)

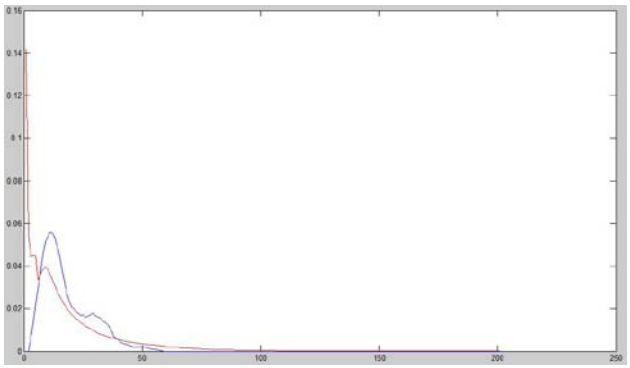


Figure: histogram with smoothing

Figure 4.9: Histograms:(a) without smoothing; (b) with smoothing:

(Histogram color :red~no object, blue~object)

Clearly as can be seen from 4.9, the smoothed histogram, provides a better judgement, (a clear demarcation) as to which node corresponds to car and no-car.

Hence, the above steps thus result in generating distributions with each feature and intensity(color of the car). In this research we thus have 24 distributions corresponding to the 8 features of the car and 3 intensities (dark, gray, white).

4.7 Detection

After the Bayesian Network is constructed and learnt, it is used to compute the probability of the presence of the car at a given point. First the feature maps, FM1, FM2,

.. FMn are computed (as explained in section 6.6.3). To evaluate the P(car|F, i) at image location (x,y), the offset of a feature j (j = 1, 2,…N) in the location from (x,y) is computed by the feature prediction module as (dxj, dyj). The features are then retrieved from the corresponding feature maps so that Fj=FM(x+dxj, y+dyj).

Then,

$P(car|F, i) = alpha*(\pi_{i=1}^{N} P(Fi|car, i))\, P(car)$

Here alpha can be calculated from :

$$1/((\pi_{i=1}^{N} P(Fi|car, i))P(car) + \pi_{i=1}^{N} P(Fi|nocar, i)P(nocar))$$

P(car) and P(nocar) in an image is obtained by eye estimation, and is taken as 0.5 each (i.e. there is equal probability of finding the car and no car).

Then, P(nocar|F, i) is simply taken as 1 - P(car|F, i)

Thus after this operation, two matrices are obtained which are same size as the image, one with the probability of detecting a car and the other gives the negative probability (probability of not detecting a car). Then a comparison is done, wherever, P(car|F, i) > P(nocar|F, i), a detection is said to have been made and the pixel is colored.

4.8 Results and Discussions

Figure 4.10 shows the results.

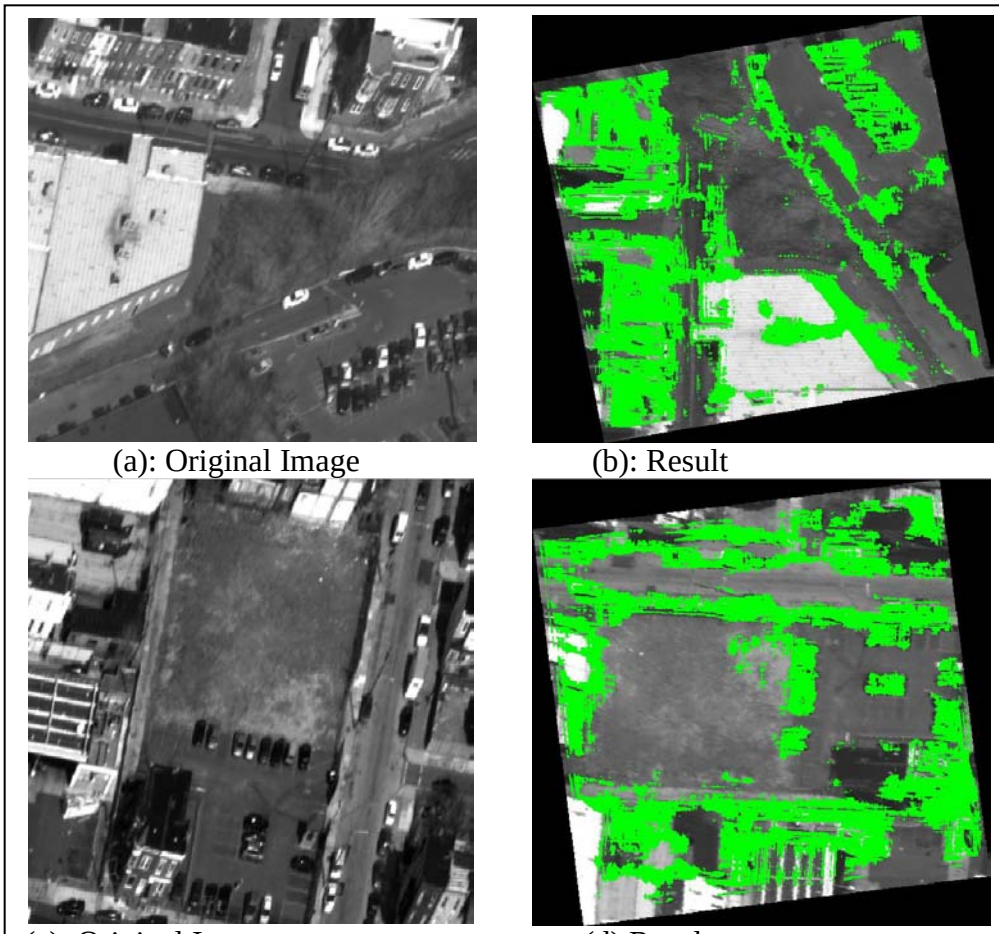


Figure 4.10: Sample Results

A car detection system for aerial images using the Bayesian approach has been described. The road networks from the TIGER/LINE database [5] have been used to optimize the performance of the system and reduce errors from line detection algorithms [11]. Response of gradient mask filters as features are used, to account for the low resolution and noise in aerial images. The BN is used to combine the multiple features with learning. The algorithm is applied on images having different views of roads, different types of background and against different image qualities. Posed by the challenges in this imaging system, the algorithm has done a good job. As of now, this algorithm is based only on the vertical view of the car and the dimensions of the car obtained from the small training set. Hence, it is seen that the algorithm falsely identifies the windows of the buildings in the perspective view image as cars, (because of its dimensions and orientations). This algorithm can be improved and extended with the following modifications.

Incorporating different views of the car, for eg: oblique views and views from various other angles, increasing the size of the training data from 15 to many hundreds. This will take into account the appearance change of the car. Also, incorporating cues from the shadow of the car will help in car detection.

CHAPTER 5

USING CAMSHIFT FOR VEHICLE TRACKING IN WAMI

5.1 Introduction

In the previous chapters, we applied different techniques which laid the ground work to ameliorate some of the inherent complexities in this highly complex dataset. This makes the problem (tracking) simpler in this highly complicated PSS dataset. In this chapter, we apply a novel technique to track cars. Choosing an approach that will work for this dataset requires detailed knowledge of expertise in the field of Computer Vision. Hence, the algorithm to be developed or applied in this dataset must be fast and efficient. Camshift algorithm satisfies some of the criteria necessary to be applied to this dataset. This exercise focus on two objectives, first I will show the application of Camshift coupled with a novel apriori knowledge, the road information. Explain the results, advantages, disadvantages and the scope of camshift for car tracking. Show the results of using camshift with an apriori information, the road information. Finally, enhance camshift by improving it to suit our dataset.

5.2 Related Work

Tracking of cars is an ongoing research. Many groups have tried out different techniques for tracking cars in aerial images. As reported by Hinz [16], the pioneering work in this field was done by Nagel and co-workers (based on optical flow) [17] [18]. Their technique explored the use of stationary cameras for traffic applications. [19-24] use techniques for car tracking that require high frame rate and oblique views. These do not work with our data, as we deal with ortho-rectified view. Works by Hinz and co-workers [6] [25] developed algorithm on images that handle images taken by photogrammetric cameras with a high resolution of 5-15 cm on the ground. This approach cannot be used in our dataset, because since they have got a very high resolution, they are able to consider the minutest details of the vehicle, but for our case, the vehicle has very limited resolution. Ernst et al. [26] developed a matured monitoring system for real time traffic data acquisition. Their system consisted of an infrared and optical sensor mounted on slow moving air vehicles. The traffic parameter estimation is based on vehicle tracking in consecutive image frames collected with a much higher (25 Hz) frame rate than ours. Due to our low frame rate (as low as 1 frame per sec) such estimations are not possible.

5.3 Reasons for choosing Camshift

In order to find a fast, efficient algorithm with minimal computation cost, we focused our attention on the robust statistics and the probability distributions. The advantages of using Camshift as reported by Bradski et al [27] are given here. Robust statistics are those that tend to ignore outliers in the data (points that are far away from the region of interest). Thus robust algorithms help compensate for noise and distracters in the vision data.

Hence, a robust non-parametric technique was chosen for climbing density gradients to find the mode of the probability distributions. The popular mean shift algorithm operates on probability distributions and is designed for static distributions. But, distributions derived from video image sequences change over time, so the mean shift algorithm has to be modified to adapt dynamically to the probability distribution it is tracking [27]. Camshift is a popular algorithm for visual tracking providing speed and robustness with minimal training and computational cost. Unlike mean shift, Camshift is designed for dynamically changing distributions [27]. These typically occur when the objects in the video sequences are being tracked and the object moves so that the size and location of the probability distribution changes in time. For such dynamically changing distributions the mean shift algorithm alone would fail as a tracker. A window size that works at one distribution scale is not suitable for another distribution scale as the object moves towards and away from the camera [27]. Small fixed size windows may get lost for large object translations like buses, trailers etc and large fixed windows may include distracters. Camshift is able to continuously deal with image problems like irregular object motion, image noise, and distracters due to the following characteristics it posses [27].

Camshift continuously rescales itself in a way that naturally fits the structure of the data. An object's apparent velocity and acceleration scale with its distance to the camera, which in turn, scales the size of its color distribution in the image plane.

Camshift's windowed distribution and gradient climbing enables it to ignore distribution outliers.

Camshift's robust ability to ignore outliers also allows it to be robust against distracters. Once Camshift is locked onto the mode of a color distribution, it will tend to

ignore other nearby but non-connected color distributions. Thus, when Camshift is tracking a face, the presence of other faces or hand movements in the vicinity will not cause it to loose the original face unless the other faces or hand movements substantially occlude the original face [27].

Since, our dataset is a low frame rate video with lots of distractions, irregular object motion due to perspective, image noise and occlusions, Camshift seemed to be a promising method.

5.4 Brief on Camshift

Gary R. Bardski and co-workers developed and used camshift on the HSV (Hue Saturation Value) color system [27]. Camshift operates on the probability distribution image derived from histograms. Models are first created by taking 1D histograms from the H (hue) channel in HSV space. Gary R. Bardski applied camshift to face tracking via a flesh color model. First the flesh areas were sampled by prompting the users to center their face in an onscreen box. The hues derived from the flesh pixels in the image were sampled from H channel and binned into 1D histograms. During operation, the stored flesh color histogram was used as a model, or lookup table, to convert incoming video pixels to a corresponding probability of flesh image. Below are given the steps on the calculation of Camshift:

1. Set the calculation region of the probability distribution to the whole image.
2. Choose the initial location of the 2D mean shift search window.
3. Calculate the color probability distribution in the 2D region centered at the search window location in an area slightly larger than the mean shift window size.

4. Mean shift to convergence or for a set number of iterations. Store the zeroth moment (area or size) and mean location.

5. For the next video frame, center the search window at the mean location stored in Step 4, and set the window size to a function of the zeroth moment found there.

Go to Step 3.

For each frame, the mean shift algorithm will tend to converge to the mode of the distribution. Therefore, CAMSHIFT for video will tend to track the center (mode) of color objects moving in a video scene [27]. Figure 5.1 shows Camshift locked into the mode of a flesh color probability distribution (mode center and area are marked on the original image).

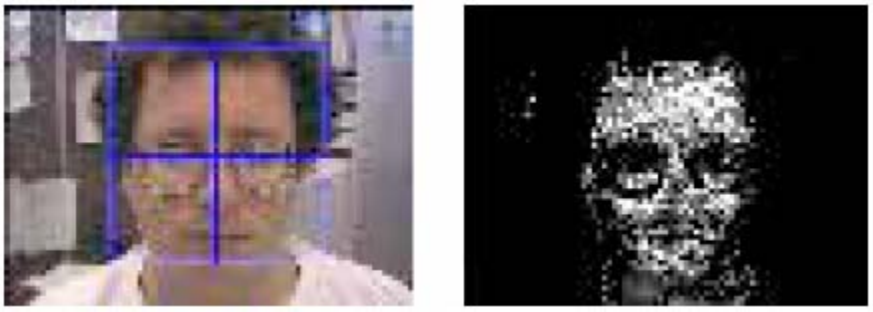
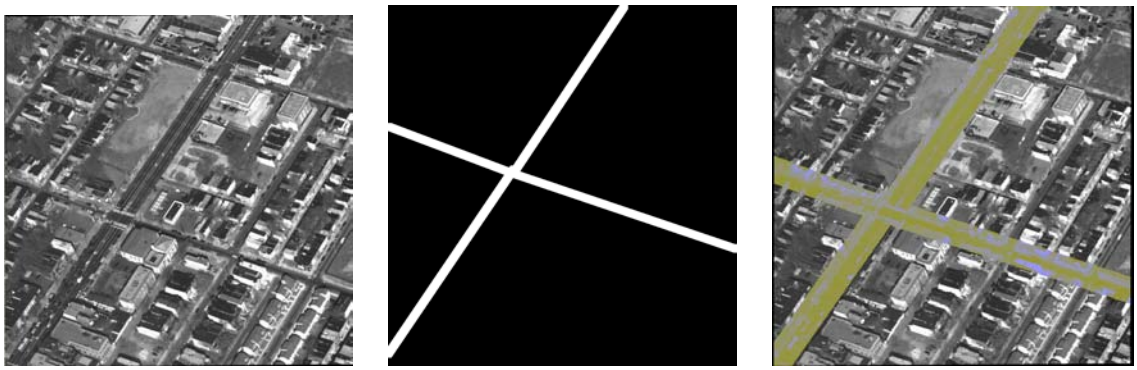


Fig 5.1: Camshift locked into the mode of flesh color. Image courtesy [27]

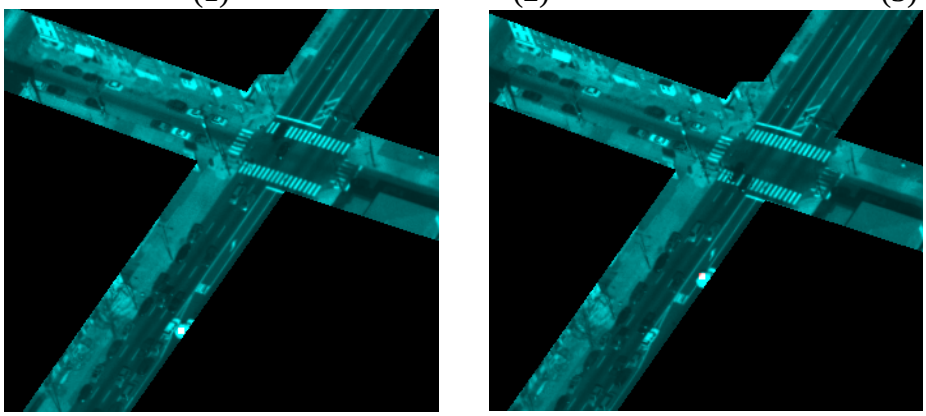
5.5 Applying Camshift with an apriori knowledge

One of the important challenges in our dataset is that it is a grey scale image. Since the video is taken from above 35000ft altitude, the resolution of the cars in the image is minimal. The background noise is also very large, with the surrounding buildings and other distractions on the side of the street. Being gray scale images, there is not much color difference between the car that is being tracked and the surrounding buildings and

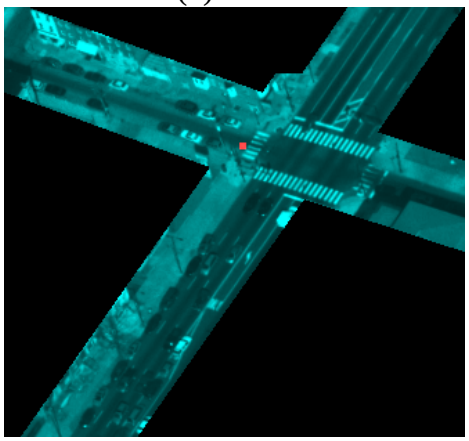
the roads. The background changes very fast, as the frame rate of the dataset is 1 -3 frames per sec. Hence, in order to make this algorithm more robust in this dataset, we used an apriori information. The apriori information in this case was the road vector maps discussed in Chapter 3. By using this map information, the road was extracted from the image as road mask. On this road mask, the camshift algorithm was performed. Figure 4.2 shows the mask of the road, the masked image and the result of applying the camshift algorithm in the masked image.



(1) (2) (3)



(4) (5)



(6)

Figure: (1) shows an unprocessed image. (2) shows a mask of the road (3) shows a road mask blended with the whole image (4, 5) shows the tracking result on using camshift on the masked road(red dot is the tracker): Tracker is able to successfully track the car, as it lies on the ambulance (6) shows the tracker lost the car in the crosswalk

Figure 5.2: Original Camshift applied with an apriori knowledge in PSS data

From figure 5.2, it can be seen, that as soon as the car reaches the crosswalk, the tracker looses the car. In a nutshell, reasons for this behavior is explained in the disadvantages part beginning with the advantages of this approach.

5.5.1 Advantages of this approach

This approach had a very big advantage, which was to eliminate the nearby noise from the area close to the road. Even though Camshift automatically adjusted its window, still due to the non-car objects being very close to the road, the tracker often lost the car. By creating the mask of the road and then performing the camshift on this mask, very easily this can be avoided. Also, since camshift uses histogram feature distribution, it is invariant to object orientation and possible shape deformation. It is also very fast, since the feature of each search window point is directly used in the probability map computation, no further transformation of block processing is required.

5.5.2 Disadvantages of this approach

Although this approach eliminated objects outside the street. But, it failed when the car reached a cross-walk or an intersection. It was seen that when the car reached the intersection the tracker attached itself to the crosswalk, in other words lost the car because the color of the car was very similar to the color of the crosswalk. The reason can be attributed to the fact that, once the user points out the object of interest to be tracked, then camshift creates a model of the desired hue using a color histogram. This hue is then sampled and binned into 1D histogram and stored for future use. During the operation, this histogram serves a look up table to convert the incoming pixels to the probability of the object to be tracked. Hence if the color matches, in our case, color of the car and color of the crosswalk, the tracker thinks it to be the car, and attaches itself to it. No influence

of the neighbourhood pixel were taken into consideration. In the case of tracking face in a color video, the algorithm worked [27] because the color of the face is a unique flesh color and it is invariably different to the color of the shirt or pants or any other non-face object in the image. Figure 5.3 shows the process of matching a feature point (X,Y) to the model of the car.

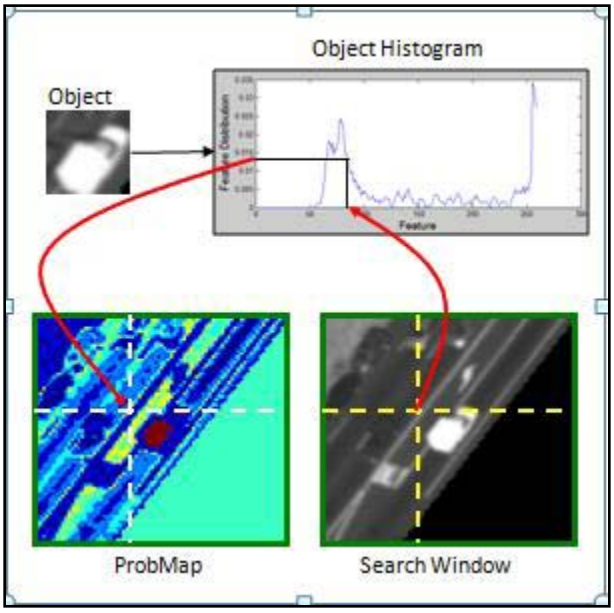


Figure 5.3: Processing done in our dataset with original Camshift

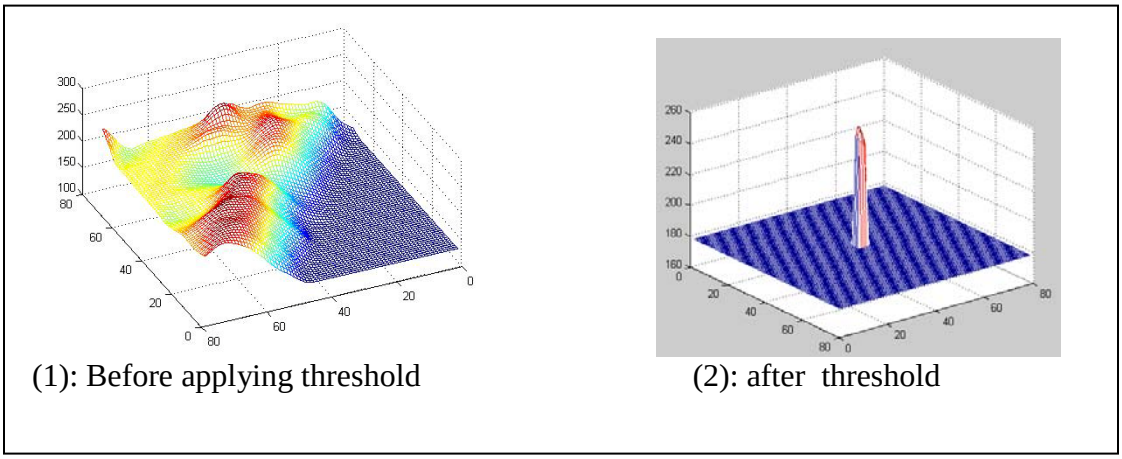
Thus attempts were made in our case to improve this algorithm by taking into consideration the influence of the neighbouring pixels. In my next section I will describe some of the approaches taken, to take into account the influence of the neighbouring pixels.

5.6 Modification of Camshift

In order to cope with the difficulties imposed by using camshift in our dataset, we decided to make some changes. This will help to track cars in a better manner in our dataset. Some of the modifications that were undertaken are given below:

5.6.1 Changing the algorithm underlying Camshift

Camshift is Continuously Adaptive Mean shift algorithm. To track the cars more efficiently in our dataset, we used an object location approach which was different from mean shift. The meanshift algorithm creates a confidence map in the new image which is created based on the color histogram of the object in the previous image. It then uses mean shift to find the confidence map near the object's old position. If the object that is being searched is found, then a peak is generated. As this dataset was gray scale in nature, when meanshift was applied as an underlying algorithm of Camshift, it generated a lot of peaks. For instance, when tracking a white color, the white color present in the search window (region around the car), such as those from the crosswalks etc also gave rise to the peaks. In order to detect the true peak, the one originally from the car, our object of interest, the following steps were taken: The probability map, which was fed into the mean shift program, was first thresholded. The maximum probability in the probability map was obtained. All the places inside the probability map that had 90% of this maximum value was set equal to 1 and the rest of the points were set to zero. Then after thresholding the mean shift was applied. Figure 5.4 shows the peaks obtained.



(1): Before applying threshold (2): after threshold

Figure 5.4: Peaks detected: (1) original camshift (2) after modifying underlying algorithm

5.6.2 Improving the probability map to capture local neighborhood information

The second important modification was in the generation of the probability map. In the camshift approach, as we have seen earlier, the histogram of the object was created and used as a look up table to match the incoming pixel to the probability of the object to be tracked. This had its disadvantages, as explained earlier. To overcome these disadvantages we use the full histogram of the search window was used as opposed to just one pixel intensity PDF estimate. Our algorithm incorporated the following steps as given in figure 5.5 and made use of the sliding window operation within the search window. First, the object was made to slide over the search window, and within the search window and within each block (same as the object size) the similarity and disimilarities were calculated between the object and block of the same size. This method thus took into consideration the influence of the neighborhood pixels. Also to make this new algorithm more robust a 2 frame velocity constant prediction was introduced. Various similarity/ dissimilarity measures were tested for this purpose.

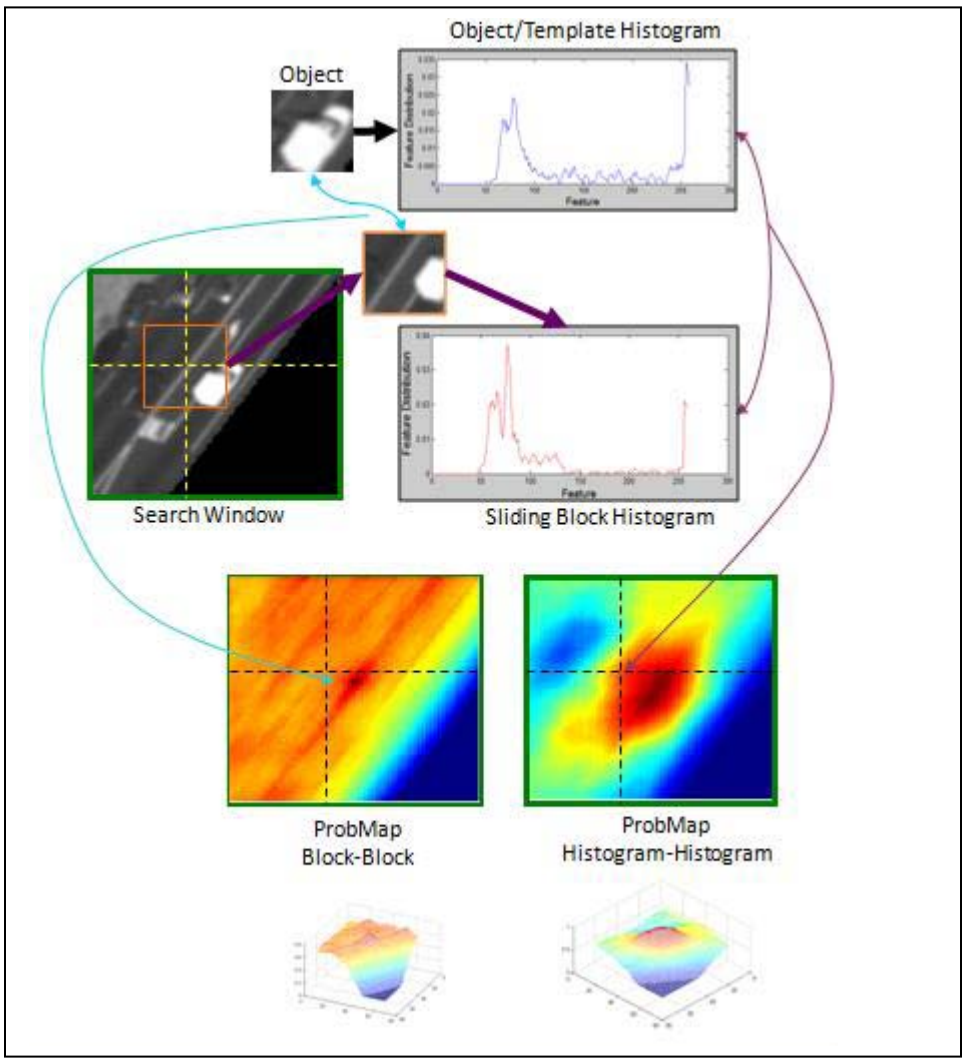


Figure 5.5 Overview of modified Camshift

Different similarity measures (intensity and histogram) were performed on the data to select the best similarity measure for computation.

Figure 5.6 shows the result of applying different measures.

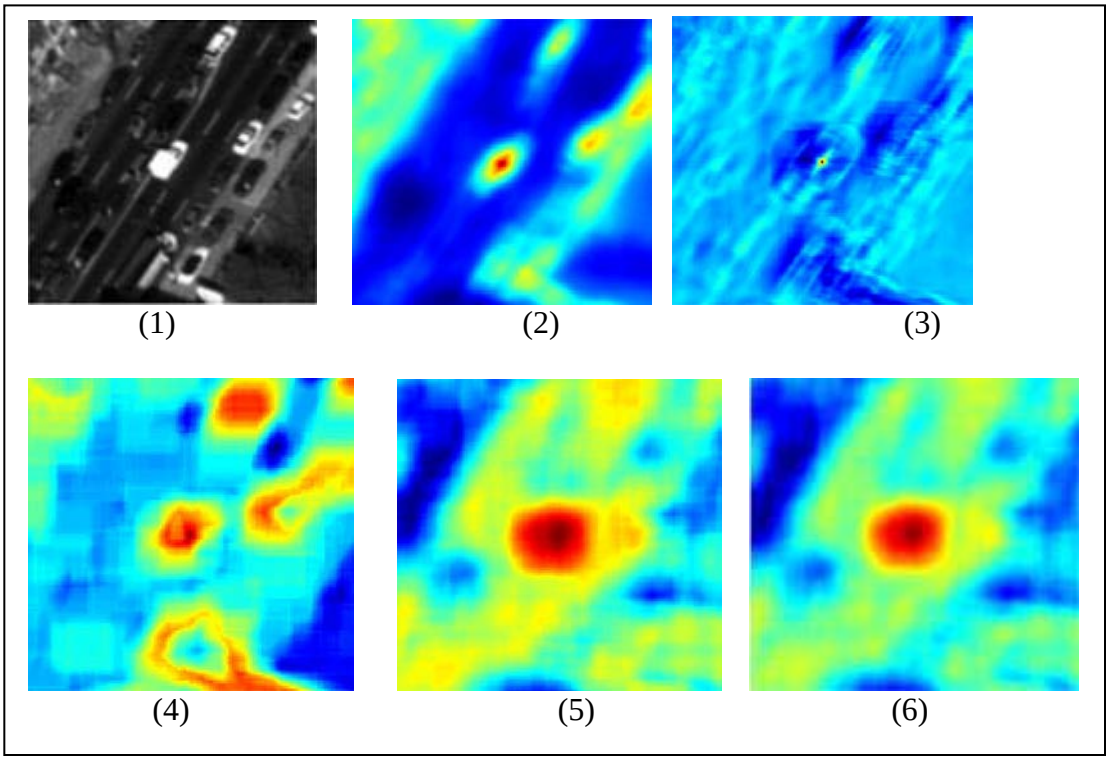


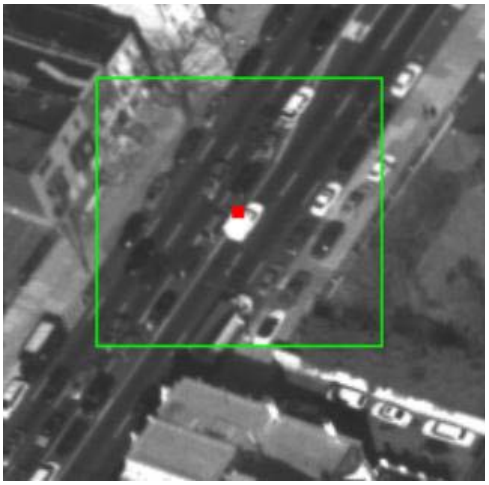
Figure5.6: (1): Search window from the original image :prominent white car is to be tracked. (2): applying intensity correlation (3) applying gradients of intensity normalized cross-correlation (4)chebyshev –histogram comparison results (5) Chisquare –histogram comparison results (6) Kolmogrov –historgram comparison results

The prominent red circle in image figure 5.6, indicates the location of the car. As we can see from the result above, for the block to block matching Figure 5.6(2) and Figure 5.6(3), capture the local information and spatial configuration but are sensitive to orientation changes and deformations. Block to block matching is also computationally intensive. For the histogram measures, Chebyshev Figure 5.6(4) captures the true location of the car, but also mistakenly identifies a few other regions as probable target object

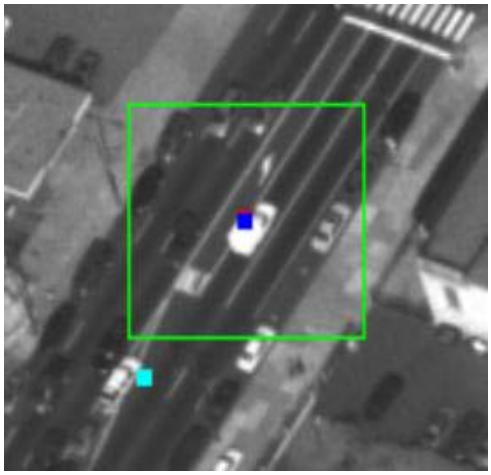
locations. A more clear-cut identification thus desired to exactly pin-point the location of the target object in the image. Chisquare Figure 5.6(5) and Kolmogrov Figure 5.6(6) pin-points the object location(as is the case with all histogram based comparisons), but loses information regarding the spatial orientation of the object. They are however, invariant to object and background orientations, and possible shape deformations. Hence, it can be said that choosing the correct comparison measure requires due consideration

5.7 Results and discussion

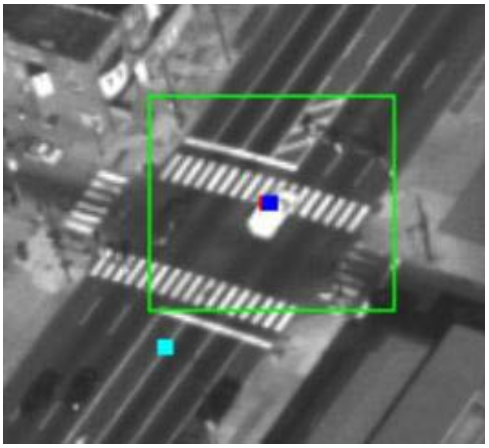
The Figure 5.7 shows the application of modified camshift with a two frame constant velocity prediction approach. It uses thresholding combined with meanshift, to detect the accurate peak. It also uses the full histogram instead of just 1 pixel intensity PDF (Probability Density Function) estimate. To capture the local information, it uses the sum of absolute differences (SAD) to converge to the best new estimate. Some of the frames are presented below:



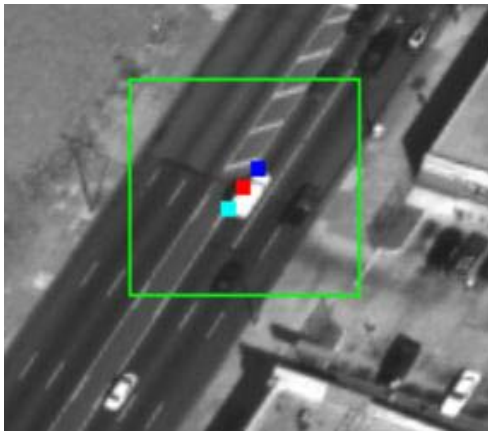
1. The green box shows the search window & red dot shows the center of the car. Since it is the first frame of the sequence, it does not have a predicted point



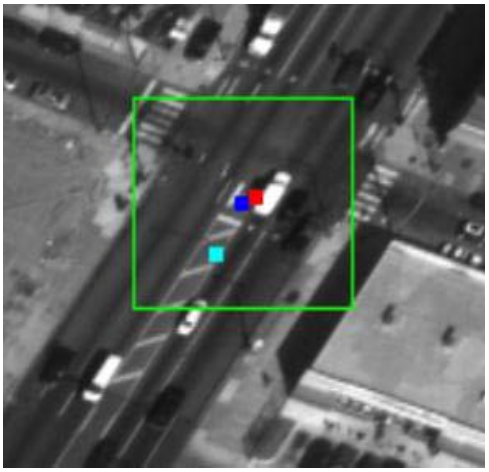
2. The green box shows the search window& the red dot shows the center of the car. The blue dot shows the predicted point, and the cyan dot shows the previously tracked point



3. Shows the car at an intersection, The tracker, with the new algorithm is able to track, the car properly



4. Shows a frame towards the end of the sequence. The appearance of the car has changed quite a bit, but the tracker is able to track it.



5. Shows the last frame being tracked

Figure 5.7: Modified Camshift

Thus it is seen that on using the modified camshift, (a new algorithm), the object can be tracked for much longer duration. The object does not get lost when it approaches the

crosswalk. The constant velocity prediction used helps to build a better search window around the object.

5.8 Conclusion and Future Work

A car tracking system using Camshift has been described. Two different methods has been tried. First the original Camshift algorithm [27] has been applied with an apriori knowledge, the map information obtained from TIGER MAPS [5]. The results obtained from this method are discussed. The reasons for its not so satisfactory performance is inherent in the Camshift algorithm itself. This is explained in details with reasons as to why it worked for the original problem [27] and why it did not work for us. The scope of modifying the Camshift algorithm to serve the purpose of tracking cars in our dataset is explained in details with various intermediary diagrams. The final results obtained by applying this modified Camshift to our PSS dataset is given.

This tracking work shown here is at its very primitive stage. It can be improved and extended to give much better results. For instance, in order to produce the probability map, different features can be fused together. As we had seen earlier, no similarity measure is perfect in its own way. A particular measure may perform well in certain scenarios, but not so well in others. So, if we can fuse these measures to produce a feature intensive map, then no information will be lost and a robust tracker can be obtained.

Also, in our modified Camshift approach, we currently use a two frame constant velocity prediction algorithm to predict the next position of the target object (car). We can improve upon this by introducing a Kalman filter or particle filter prediction algorithm so that it can also predict non-linear motion of the target object (like turns etc).

Nevertheless, our approach represents a substantial improvement over the previous state-

of-the-art Camshift algorithm, in that we are able to track the target object for a considerable time span, even when the color of the target object is very similar to the color of the background.

CHAPTER 6

QUANTITATIVE EVALUATION OF THE LOFT TRACKER

6.1 Introduction

Automated tracking of vehicles in wide area surveillance applications is an ongoing research. In order to determine the best tracking algorithm for a system, a good evaluation procedure is required. Many groups have proposed tracker performance metrics like track detection, track completeness, fragmentation rate, and ID change rate [28-32]. All of these existing evaluation approaches assess tracker performance through measures of correspondence between ground truth and system tracks. However, each of these metrics only addresses a part of the tracker behavior. Hence to give an overall score of tracker performance to these individual track pathologies, these metrics are often combined by an ad-hoc weighted sum. These weights are chosen according to the application and the user's unique needs. Choosing a weight that is justified for the algorithm and also satisfies all the needs of the user is a difficult task and introduces a lot of uncertainties, especially when the metrics are in different units and often highly correlated. Hence, an overall scoring function based upon information theoretic approach

has been proposed for multi-tracking systems by Kao et al [2]. Our tracking system is a low frame rate automatic and assisted tracking system of single objects. In this part of the research, the existing track methodologies and the information theoretic approach [2] will be discussed with respect to our automatic and assisted single object tracking system. The differences in the behavior of a multi-tracking system and our single tracking system will be highlighted. Two different techniques for our tracker performance evaluation using the information theoretic metric will be proposed and the results will be discussed.

6.2 Our System for Evaluation

Figure 6.1 gives an overview of our whole system for evaluation.

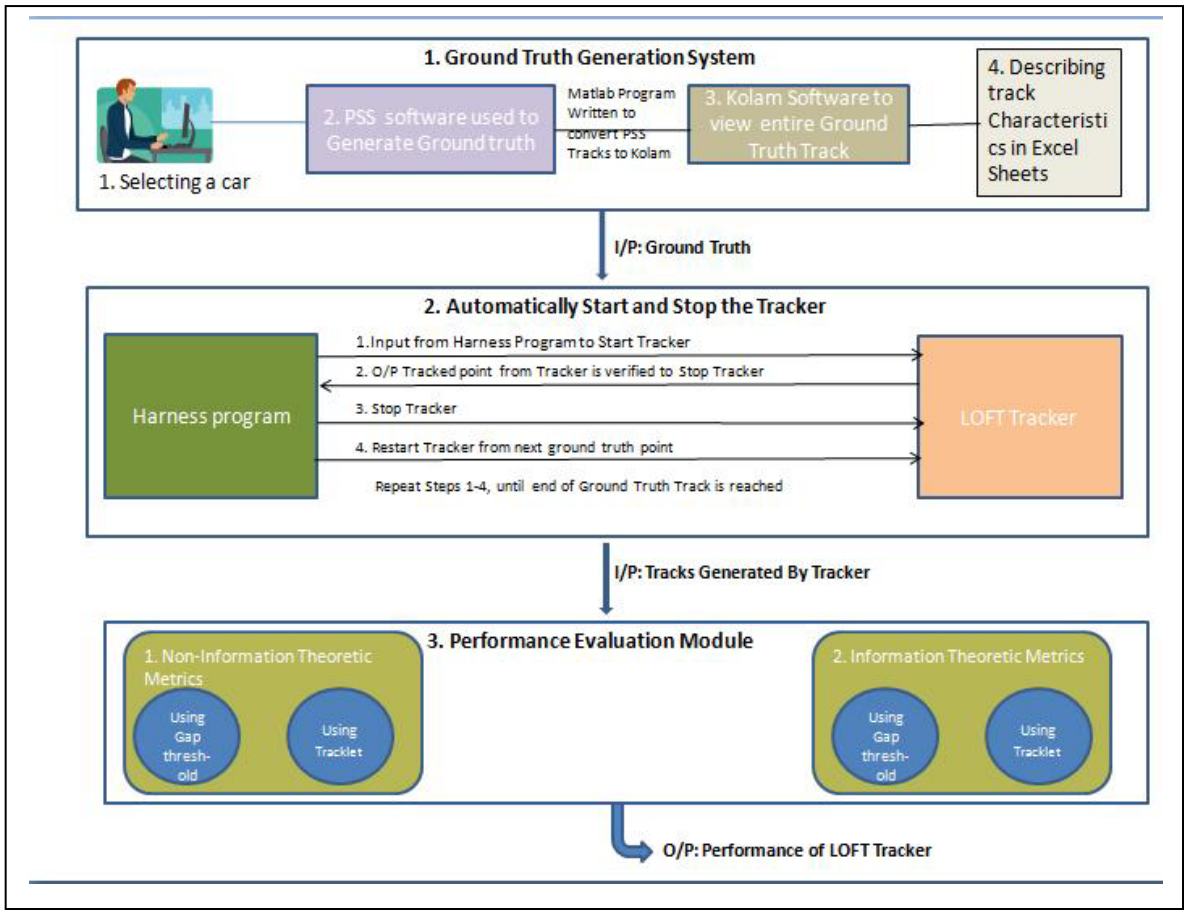


Figure 6.1: Overall system for Performance Evaluation

As can be seen from the diagram, the overall system consists of a number of different components, namely the ground truth generation system, module to automatically start and stop the tracker and the performance evaluation component/module.

6.2.1 Ground truth generation system

Performance of the tracks can be done in 2 ways. In one way no ground truth is required [33], and in the other method ground truth is required. We have evaluated the performance based on ground truth data. Hence for our system, the generation of ground truths plays a crucial role in evaluating the performance of the tracker. The ground truth data collected for this system is done by manually tracking vehicles for many hours with patience. It has been collected on similar characteristic datasets as described in chapter 2. As such they pose a lot of challenges, even for manual tracking of cars.

Challenges:

Some of the notable challenges for ground truth collection are as follows:

It may be recalled from chapter 2, that the data for the video surveillance systems is collected by an airborne system. Sometimes it was noted that in the event of capturing data, there were a lot of parallax errors introduced. This resulted in introducing, translational error, registration error etc. If the jitter happens in frame n then, it becomes very difficult to recognize the car from frame n-1 to frame n. The figure 6.2(a-j) shows tracks of car overlaid on PSS images. The translational errors, registration errors etc manifest as a distorted car track as shown in the images in figure 6.2.

Figure 6.2(a): Sample Track 1

Figure 6.2(b): Sample Track 2

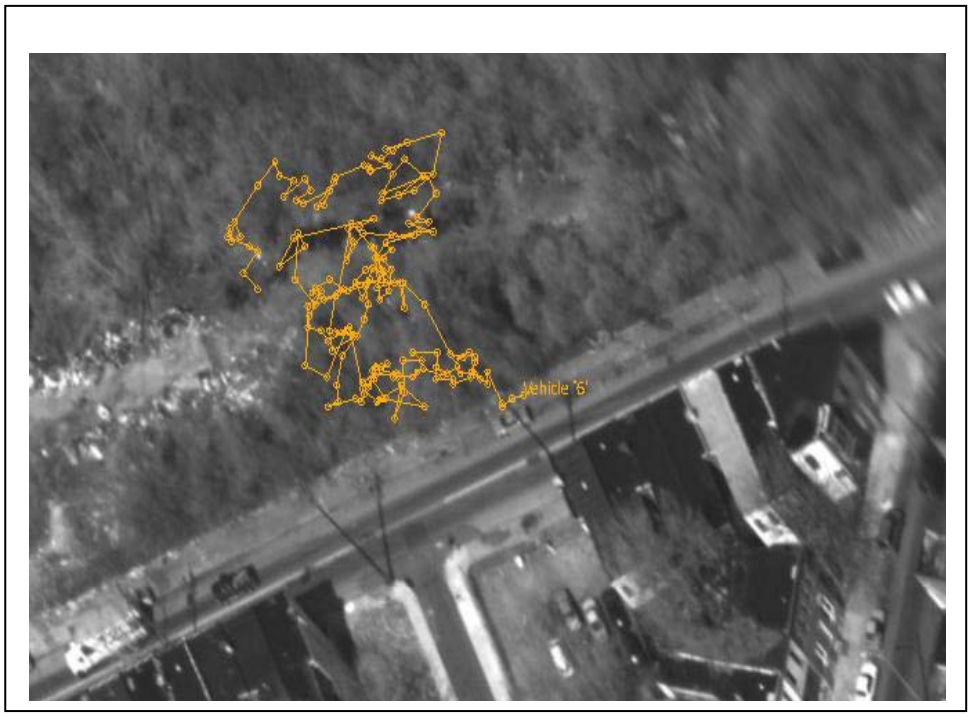


Figure 6.2(c): Sample Track 3

Figure 6.2(d): Sample Track 4

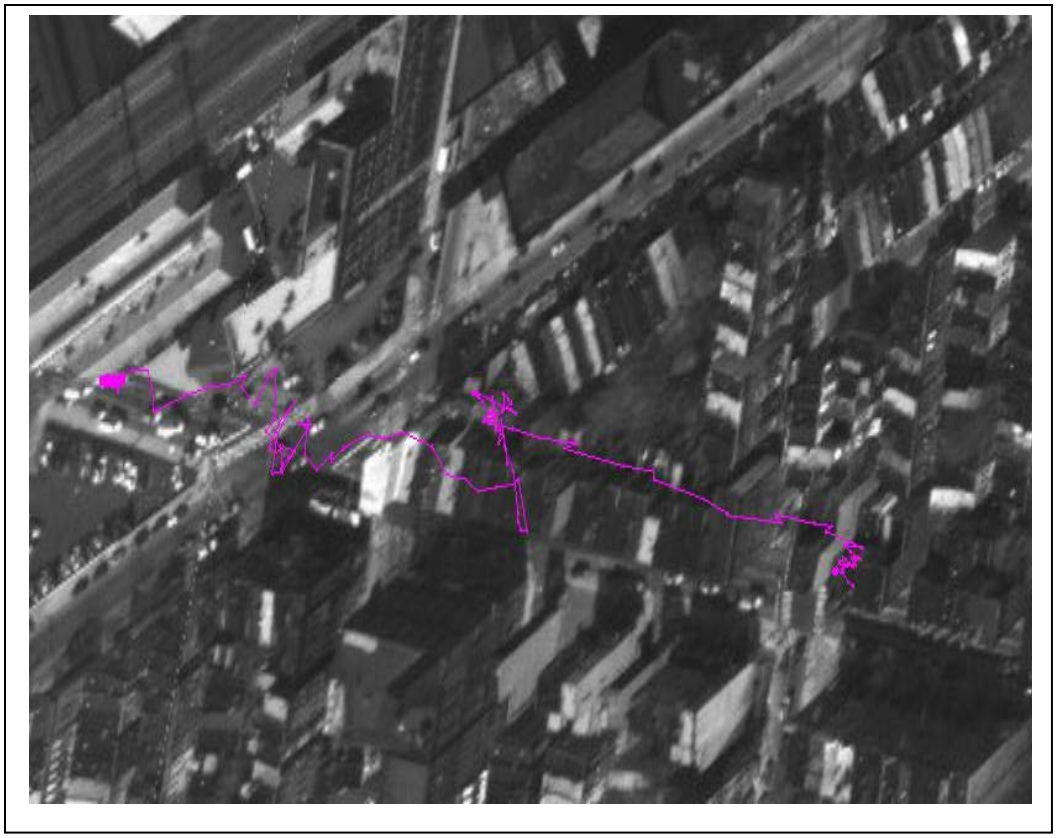


Figure 6.2(e): Sample Track 5

Figure 6.2(f): Sample Track 6

Figure 6.2(g): Sample Track 7

Figure 6.2(h): Sample Track 8

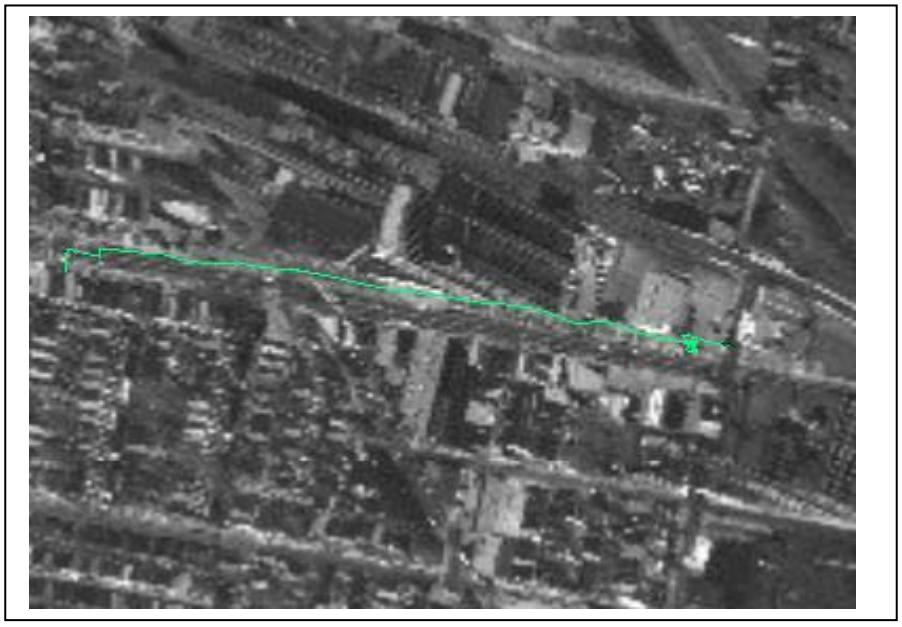


Figure 6.2(i): Sample Track 9

Figure 6.2(j): Sample Track 10

The persistence surveillance data does not have frame to frame stabilization, hence a complete track of the car looks significantly distorted, which sometimes raises confusion in the ground truth collection. Figure 6.2 gives examples.

Due to the drastic appearance change in the vehicles, it required immense concentration in manual tracking

Grey cars were by far the most difficult to track, because of easily losing them to the background

Components description:

As shown in figure 6.1, our ground truth generation system consists of many components

First, an analyst (person generating the ground truth) selects a car from the video data randomly.

Secondly, the person manually tracks the car using an editor specialized to do this job. This editor is known as the PSS software. There were several advantages of using the PSS software. Firstly, this software was equipped to handle the large (16k x 16k) data. This data was of a special format to enable quick online streaming of this large format imagery. The data format had an extension .PSS. Hence, the PSS editor capable of handling this data was used. This editor generated track files (.trk files). These.trk files had the center co-ordinates of the car in each frame as it was getting tracked in latitude/longitude co-ordinates.

As a general rule of manual tracking, the car being tracked was followed through the road. If the car got occluded behind buildings or trees etc, the point would be observed for the next 10 frames to see if the car was again visible in the image. If the car became visible again just after 10 frames, then the car would be tracked till it disappeared

from the scene and was no longer visible. For those 10 frames, when it was occluded, the point where it was last seen was marked till it was seen again.

The other scenario that could arise is, if the car did not become visible even after 10 frames of occlusion, then the point where the car was last seen would be considered as the end of the ground truth track.

The third component in the generation of the ground module is the ability to view the tracks from its start point to its end point as continuous lines on the PSS image. Also, to view multiple tracks overlaid on the same image at once. This will give better idea of the tracks, we can also analyze the tracks by overlaying different tracks on the image and concentrate on interesting features. This feature of overlaying single track or multiple tracks as continuous lines was not present in the PSS editor. So we took help of another software Kolam [34], [35]. In order to do this, the tracks generated by the PSS software had to be converted to the format supported by Kolam. A separate Matlab program was written to accomplish this task as shown in the Figure 6.1.

The fourth component in this module, is related to documentation. Each of the ground truth tracks that were captured in this process was characterized. The below table shows the parameters used to characterize the tracks.

File name		k_46574_1.trk			
Overall assessment of Track Complexity(Low, Medium, High)		Medium			
Sequence(frame nos)		46574_0	46787_0		
Comments: (Overall information about the track)					
Vehicle					
Appearance					
Color		[D]ark	[G]ray	**[W]hite**	
Type/Size		[S]edan	[V]an	[T]ruck	[B]us
Appearance change (definition in % per category needed)	OK	[Low]	[M]edium	[H]igh	
Location in the image	OK	[M]iddle	[B]order		
Coments:					
Dynamics					
Acceleration - number and strength	OK				
Deaccelerationn - number and strength	OK	Yes			
Turn Directions	OK	[L]eft	[R]ight		
Comments: car went for a lot of turns					
Traffic					
Density	OK	[Low]	[M]edium	[H]igh	
Comments:In this track sequence went from medium-high					
Road Conditions					
Road Type	OK	Side[S]treet	Main [R]oad	[H]ighway	
Road Surface	OK				
Curved Road	OK, Just Yes/No				
Crosswalk	OK	Yes			
Comments:					
Buildings					
Building Heights	OK	[Low]	[M]edium	[H]igh	
Density	OK	[Low]	[M]edium	[H]igh	
Comments:					
Occlusion & Visibility					
Occlusion Type	Building	Vegetation	Shadow	Bridge	Mixed
Occlusion Frequency		[Low]	[M]edium	[H]igh	
Comments:occlusion was due to veg and buildings					
Imaging					
Altitude	Once per track	[Low]	[M]edium	[H]igh	
Registration Quality (frame-to-frame jitter)	Choice: Image stabilization problem	[Low]	[M]edium	[H]igh	

Figure 6.3(a): Sample Table Characterizing a ground truth track

In the above table, a ground truth k_46574.trk is documented. In this table the parameters used to characterize tracks are shown, and the characteristics relevant to this track are highlighted in red.

This documentation will serve as a database for the future as well. As the tracker gets developed over the coming years, this documentation can be used to select a particular ground truth track based on its property. For example to see, how the tracker performs with turns, a ground truth track with lots of turns can be selected and tested with the tracker.

With the above procedure, different ground truth data were captured from 3 datasets. The below table gives a summary of the ground truth tracks that were captured by us.

Dataset Name	Total No. Of Tracks	Average Length of Tracks
Philadelphia	13	163
Juarez	30	355
Nascar	28	277

Table 1: Entire Set of Manual Ground Truth Data Collected

In this research, a portion of the ground truth data from Table 1 is used. The characteristics of the ground truth data used in the experiments in this chapter is given in the below table.

Dataset Name	Total No. Of Tracks	Average Length of Tracks
Philadelphia	10	122

Table 2: Ground Truth data used in experiments

	Vehicle							Traffic	Roads				Buildings		Occlusions		Imaging	
	Appearance				Dynamics													
	Color	Type/Size	Appearance change	Image region	Acceleration	Deacceleration	Turn Directions	Density	Road Type	Road Surface	Curved Road	Crosswalk	Building Heights	Density	Occlusion Type	Occlusion Frequency	Altitude	Stabilization Quality
car 1	[D]ark	[S]edan	[H]igh	[M]iddle				[H]igh	Main [R]oad			[Y]es	[H]igh	[H]igh	Building	[H]igh		
car 2	[G]ray	[S]edan	[H]igh	[M]iddle	[Y]es	[Y]es		[H]igh	Main [R]oad				[M]edium	[M]edium	Bridge	[Low]		
car 3	[D]ark	[S]edan	[M]edium	[M]iddle				[H]igh	Main [R]oad				[M]edium	[M]edium	Building	[M]edium		
car 4	[W]hite	[B]us	[Low]	[B]order									[M]edium	[H]igh	Building	[M]edium		
car 5	[W]hite	[V]an						[H]igh	Side[S]treet				[M]edium					
car 6	[G]ray	[S]edan	[Low]	[M]iddle											Vegetation	[H]igh		[Low]
car 7	[D]ark	[S]edan	[Low]		[Y]es				Side[S]treet				[H]igh	[H]igh	Building	[M]edium		
car 8	[W]hite	[S]edan	[H]igh	[B]order														
car 9	[W]hite	[S]edan	[Low]	[B]order		[Y]es			Main [R]oad			[Y]es				[H]igh		
car 10	White	Van	High	[B]order				Medium	Main [R]oad									

Figure 6.3(b): Characteristics of 10 tracks

After capturing the ground truth data, they are then fed into the next module which starts and stops the tracker automatically. The below section describes that.

6.2.2 Harness/wrapper program to automatically start/stop the tracker

Automatic continuous tracking of any and all vehicles is very challenging due to the low temporal sampling rates and artifacts arising from sensor geometry, low contrast, dense traffic or clutter. Our tracking system is a low frame rate automatic and assisted tracking [3] system of single objects. Being an assisted tracker, after tracking for some frames automatically, the tracker loses the car. With the help of human intervention, the tracker is restarted once more from the point where it lost the car. This helps to get a complete overview of how the tracker performs for the entire ground truth. As the scenes around the car change considerably from frame to frame, with this start and restart strategy it becomes apparent how the tracker handles each scenes. Also, this method helps to use the entire ground truth and rates the performance of the tracker for the entire ground truth instead for a particular segment. In order to automate the process of starting the tracker, restarting, and stopping the tracker a separate Matlab program was written. This program was known as the harness program, as it served as a harness for the tracker. The harness program was then run with the LOFT tracker in debug mode. In the debug mode, the tracker is able to check for each tracked point if it satisfies a stopping condition (condition in which the tracker will be made to stop tracking the car). The tracker in that case will return the frames it tracked successfully upto that point and stopped. In order to start the tracker, the tracker needs an input file which contains the frame number and the co-ordinates of the center of the car to start. When the tracker starts for the first time it is provided to the tracker. There after when the tracker restarts every time, this input file is generated by the harness program from the ground truth data. The algorithm for the harness program is given below:

Function start_tracker

a) *Initial count of frames tracked by tracker = 0*

b) *While no. of frames tracked by the tracker < no of ground truth points (do)*

   a. *Start tracker* (from the first point in the ground truth data)

   b. *Get the frame no and co-ordinates of the frame tracked by tracker*

   c. *check if it satisfies stopping condition*

   d. *If the frame tracked by the tracker satisfies the stopping condition*

      i. *Then stop the tracker*

      ii. *Return the number of frames tracked by the tracker to harness program.*

      iii. *Increment counter at (a) with an offset = no. of frames tracked by the tracker*

   e. *Restart tracking (2 different procedures followed for restarting tracker depending on the approach , gap threshold based approach or tracklet based approach, details given in next subsection" procedure torestart tracker")*

   f. *Else continue tracking*

   g. *End if*

   h. *End while loop*

   i. *Save track points obtained for each segment from the tracker*

   j. *End program*

The input to the above program is the following:

- Folder where the ground truth files are kept
- Ground truth file, the track file
- Parameter file
- Path where the PSS images are kept
- Name of output folder

The output is the set of segments of the system track, containing the center of the car, as generated by the system track

Condition used to stop the tracker:

There are 2 reasons for which the tracker will be stopped:

a) If the frame no returned by the tracker is the last frame in the sequence of the ground truth data, or

b) Check the Euclidean distance between the ground truth data at the frame and the co-ordinate returned by the tracker at that frame

    a. If the Euclidean distance returned by the tracker is greater than 60 pixels then stop the tracker.

        Figure 6.4 shows the condition to stop the tracker. The green circle shows the ground truth point, giving the actual position of the car in frame n. The red circle shows the tracker point at frame n. The Euclidean distance between the two points is computed, if it becomes greater than 60 pixels then the tracker is stopped at that point. Figure 6.4(a) shows the computed Euclidean distance to be greater than 60 pixels and so the tracker would be stopped

at that position. Then restarted again from that position. Figure 6.4(b) shows the computed Euclidean distance to be less than 60 between the ground truth point and the tracker point at that frame and so the tracker continues tracking the car. Figure 6.4(c) shows the threshold beyond which the system track will be considered a failure. So if the system track point goes beyond the ribbon, then it will be considered a failure.

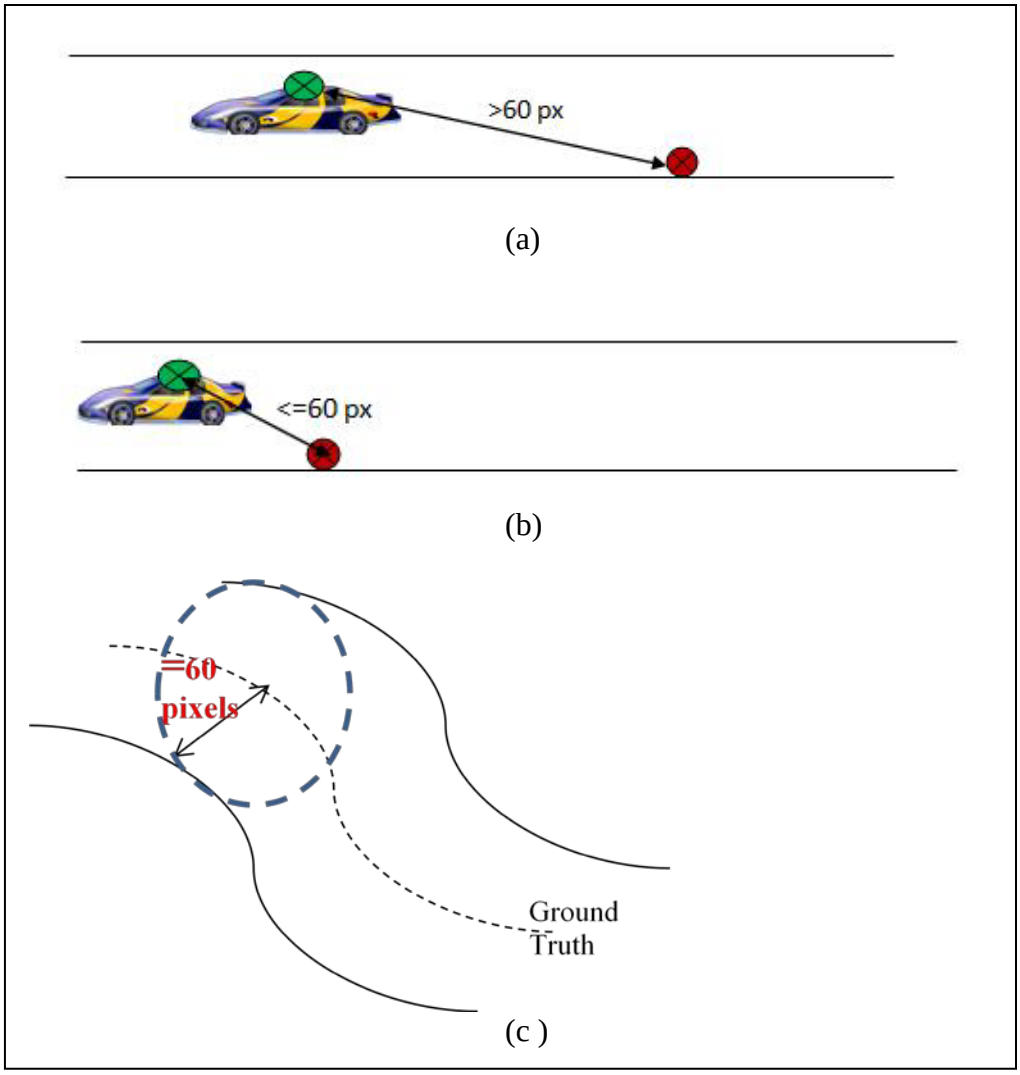


Figure 6.4: Condition to stop the tracker

Procedure to restart the tracker

In the above algorithm, a method is followed to restart the tracker. As mentioned in the earlier parts of this section, the tracker needs an input file or parameter file, which provides the co-ordinate of the frame and the bounding box of the car from where to start the tracker. The Loft tracker needs 2 consecutive frame numbers and 2 respective bounding boxes of the car for these two frames to start the tracker. For the very first time, the input file is provided with the ground truth track, and the tracker uses it to start

tracking. However, when the tracker is needed to restart, the co-ordinate of the frame and the frame number has to be generated by the harness program and provided to the tracker as an input (.txt1) file.

Generating the frame number for restarting the tracker:

The restarting frame number of the tracker is very much dependant on the approach used for evaluating the tracker. As given in the overall system diagram, figure 6.1, 2 different approaches will be developed for performance evaluation. The first one is a non-tracklet based approach, i.e. for the evaluation using gap threshold parameter, and the second one is a tracklet based approach. The details of these approaches will be given in section 6.2.3.

For the first approach, (non-tracklet based approach, i.e gap threshold), the tracker will be restarted from the same frame where it stopped.

For the second approach (tracklet based approach), the tracker will be restarted from the start frame of the next tracklet. So, in this method the tracklet size has to be fixed prior to starting the harness program and the entire ground truth data is divided in chunks, equal in size to the tracklet length. Figure 6.5 shows the restarting of the tracker using the two approaches.

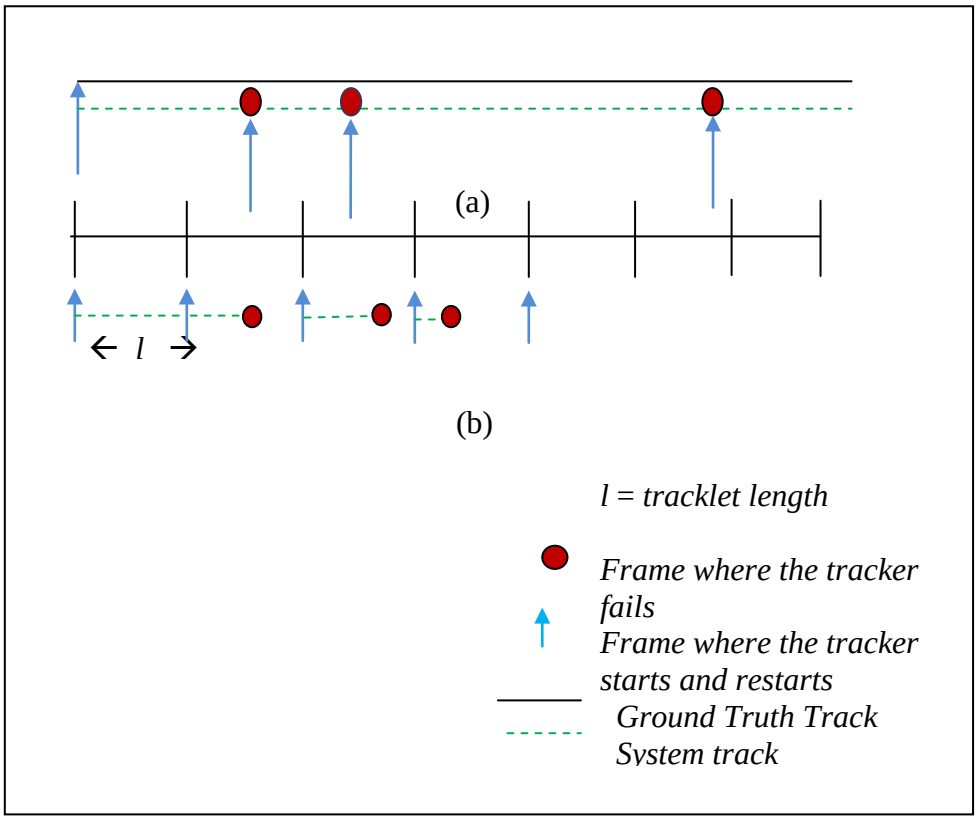


Figure 6.5: Procedure to start and restart the tracker after it fails using 2 approaches (a)Approach using arbitrary starting (for evaluation using gap threshold) (b)Approach using fixed Tracklet length

Generating the bounding box for restarting the tracker

For restarting the tracker using any approach, the angle of orientation of the car with respect to the start frame is considered. So for this the following steps are taken:

In order to find out the orientation of the car at the frame n, the angle of the car with respect to the x-axis between the n-1 and n frame is calculated, then the same is done between the nth frame and n+1th frame and finally the average of the angles of these frames are taken. This angle is then subtracted from the initial angle at the start of the tracker to get the net angle of orientation of the car.

The bounding box co-ordinates at the 4 corners(at the start frame), are rotated with respect to the center of the bounding box and the net angle obtained above.

Finally, the actual image co-ordinate at frame n(for the 4 corners), are obtained by adding this rotated co-ordinated to the center of the car, (which is obtained from ground truth).

This procedure of starting and stopping the tracker generates segments of the tracks. Thus, the complete track consists of multiple segments. This complete track information, is then fed to the tracker evaluation system to determine the evaluation statistics and performance measures of the track generated. The next section discusses the different performance evaluation metrics starting with the existing metrics.

6.2.3: Performance Evaluation module

Our low frame rate tracker is in its early stage of development. A simple experiment was conducted on the tracker to see for over how many consecutive frames the tracker could correctly track the target object without failing. The table below gives the average, minimum and maximum track length (in number of frames), that the tracker could track continuously. As already discussed in sub section 6.2.2, the tracker is deemed to fail, if its distance from the target car exceeds 60 pixels. This experiment was conducted over the set of ground truth data as given in Table 2.

Average Track Length	**Minimum Track length**	**Maximum Track Length**
7.5811	2	38

Table 3: Results of tracking

The distribution of the track lengths obtained from the tracker was plotted and is shown in the figure 6.6.

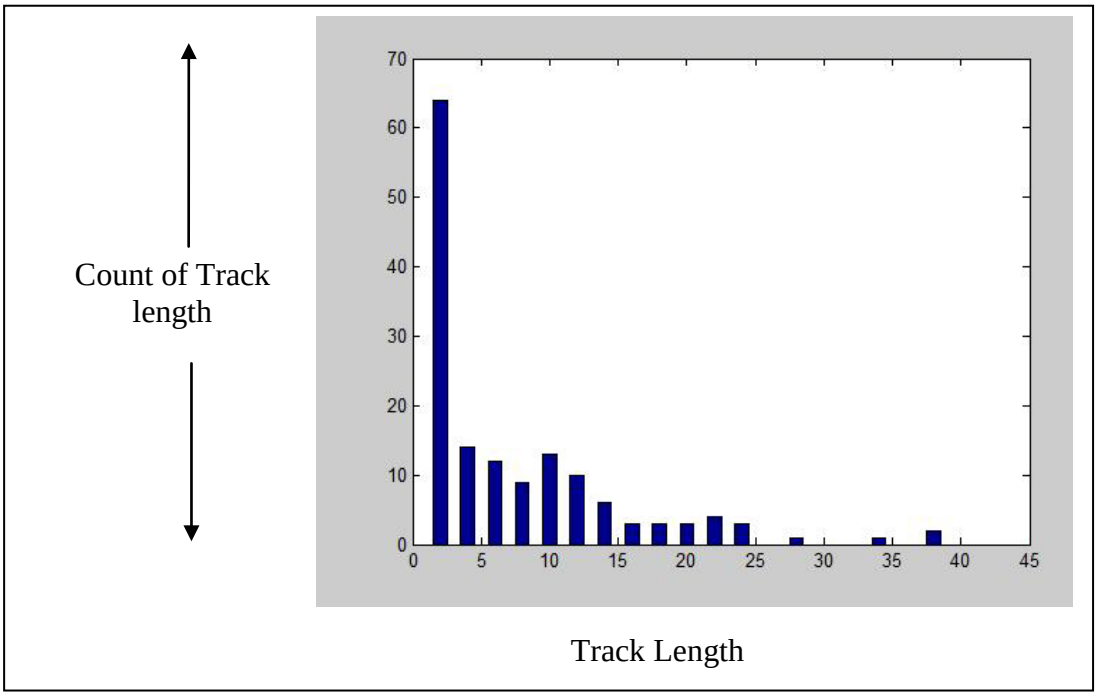


Figure 6.6(a): Distribution of track length obtained from low frame rate tracker

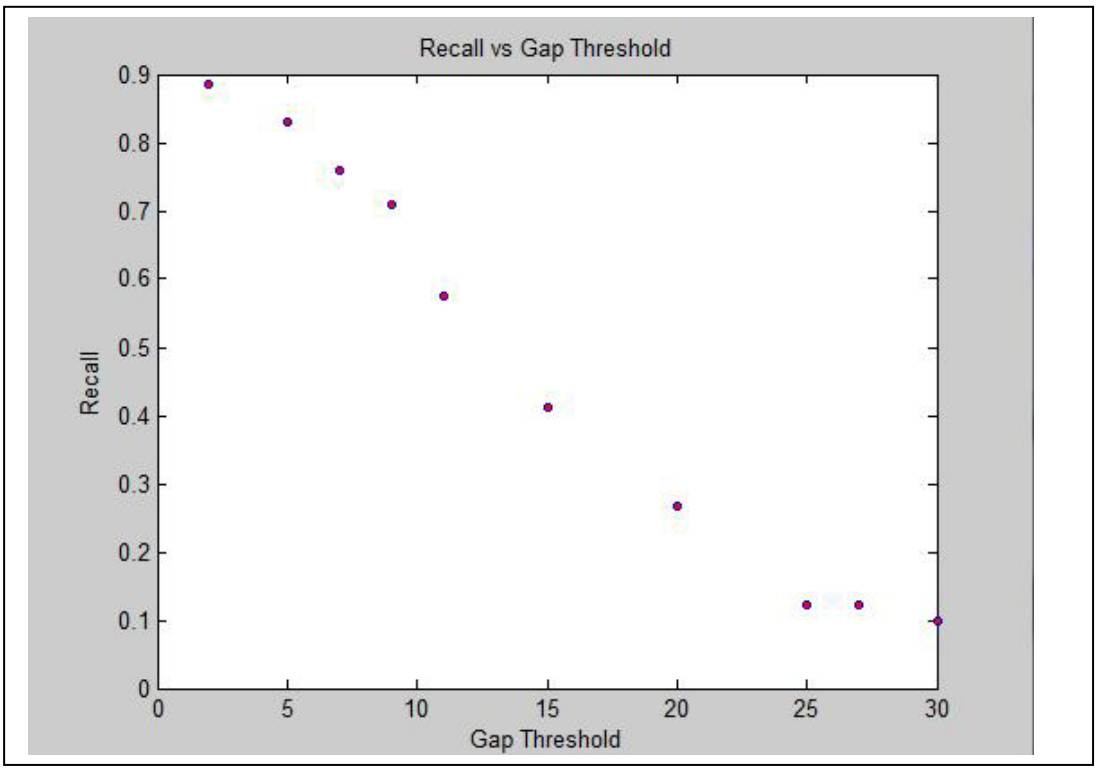


Figure 6.6(b): Recall rate vs Gap Threshold (after arbitrarily starting from where it fails)

As can be seen from this experiment, as of now the average track length that the tracker is able to track consecutively without failing is 7.6. The tracker thus has considerable scope of improvements. As the tracker develops over the years, such a simple experiment will not be very beneficial in evaluating it. Also, in order to understand the overall performance of the tracker more detailed analysis of the tracker performance is required. This will give a much better idea as to which components in the tracker need to be improved. Hence the development of a good performance evaluation tool is very essential.

Our performance evaluation module is broadly classified into 2 categories. In one category we have used non-information theoretic measures and in the other we have used the more recent information theoretic measures, largely based of the work of Kao et al. [2]. The input to our performance evaluation module is the different track metrics saved in the harness program from the LOFT tracker. The output of this module is the performance of our LOFT tracker in the form of graphs.

The novelty of our performance evaluation module lies in the fact, that we have used 2 different approaches for each of the non-information theory category and the information theory category. The description of these 2 approaches is given below.

Approach using gap threshold: The tracker starts from the first frame in the sequence of ground truth and tracks the car, until it fails at a certain point in the sequence. Wherever, the tracker fails to track the car, the tracker is stopped and then restarted again from the point, i.e. the point at which the tracker failed to track (figure 6.7(a)). Hence in this approach, the tracker is restarted arbitrarily. In this case, the segments returned by the tracker are of varying lengths, depending on how long the tracker could track (without

failing) after it started or restarted from a particular point. Hence, some segments can be as small as 1 frame long and some can be quite large.

Advantages: This is a more lenient approach. If we consider a particular segment, and the tracker could tracker only a part of this segment. The tracker will get some score for tracking a part of the segment. The tracker will not be penalized for not able to track the whole segment.

Disadvantages: No control is there on the point where the tracker will be restarted. The tracker is restarted arbitrarily, from the point where it fails. Hence, if we want to start the tracker only for a particular segment, to test how the tracker fares in a certain scenario, this approach cannot be used for that.

Evaluation Pre-requisites for this approach:

A parameter known as ‘gap threshold’ need to be fixed before evaluating the tracker using this approach. It is needed to fix the gap threshold prior to running any evaluation program to evaluate the tracker . The gap threshold is a constant value (for a particular set of tracks).In order to correctly evaluate the performance, it is needed to fix which segments should be considered as gap. Gaps are sections in the track where the tracker could not track. Gap threshold is measured as the number of frames and if the tracker has a segment that is less than or equal to the gap threshold, then that segment will be considered as a gap. (see figure 6.7 a0). It shows that depending on the value of the gap threshold, the number of gaps in system tracks change. Figure 6.7(a1) shows the segments obtained from the batch harness program using the approach of aribitrary starting of the tracker. The example determines gaps in the system track for gap threshold

equal to 5. If the tracker has a segment less than or equal to 5 (highlighted in figure as red), then they are considered as gaps in the system track. Consecutive gaps are merged as shown in figure 6.7(a2). The segments that are not considered as gaps are highlighted in green. Gaps are thus the portions in the system track for which the tracker could not track successfully. Quite naturally, the lesser the gaps in the output of the tracker the better the tracker performs.

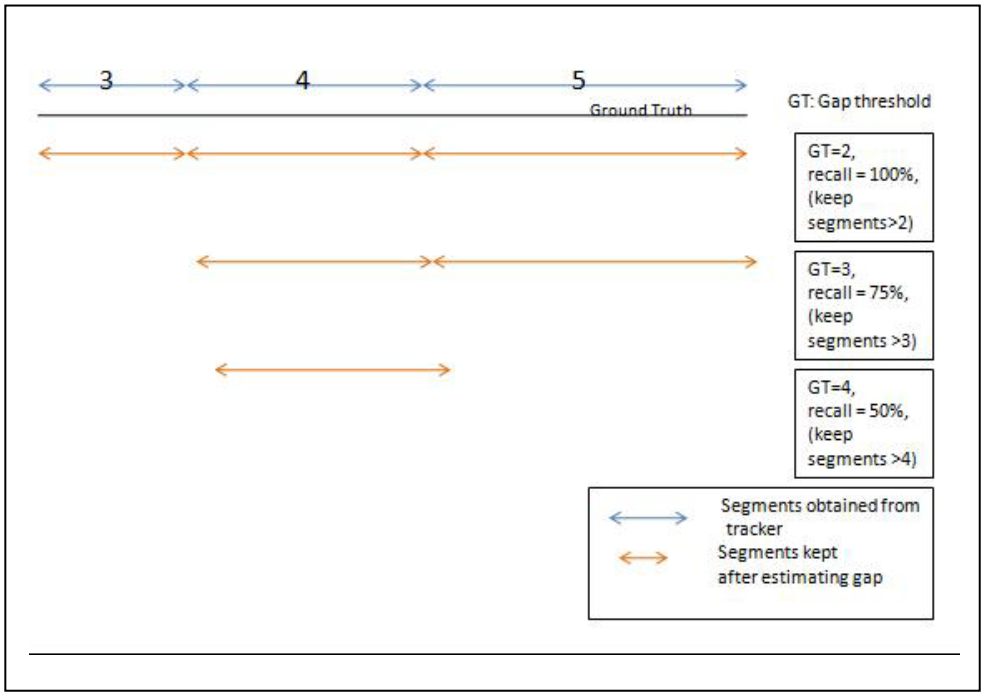


Figure 6.7(a0): Determining Gaps in System Track using different Gap Thresholds

Approach using Tracklets: In this approach the entire track (ground truth sequence) is divided up into small chunks called tracklets. In this approach the tracker is started and restarted from the first point of a tracklet starting from the first tracklet. If it fails at a particular point within the tracklet, then it is not tracked further within that tracklet (figure 6.7(b1, b2)).

Advantages: This approach enables to have more control over the tracker. If a particular scenario is to be tested, then that segment can be chosen and the tracker can be made to start from the beginning of that segment.

Disadvantages : If the tracker fails at a particular point then that whole tracklet is considered as a gap(i.e the tracker could not track that portion). Figure 6.7(b1) shows a snapshot of the segments obtained from the tracker for tracklet size 4. Segments which are smaller than the tracklet size are highlighted in red. Figure 6.7(b2) shows the gaps by removing the segments that are smaller in size than the length of the tracklet. Thus, the tracker is not given score for tracking a part of a tracklet. This tests the tracker in more harsh conditions. This can be very useful for developing a robust tracker.

Evaluation Pre-requisites: The length of the tracklets (in frames) before starting the harness program. Once the tracklets are formed for each track, they are then fed into the evaluation program. To determine the gaps, a severe penalization approach is followed. If the tracker could not track the entire tracklet length, then that whole segment is considered a gap.

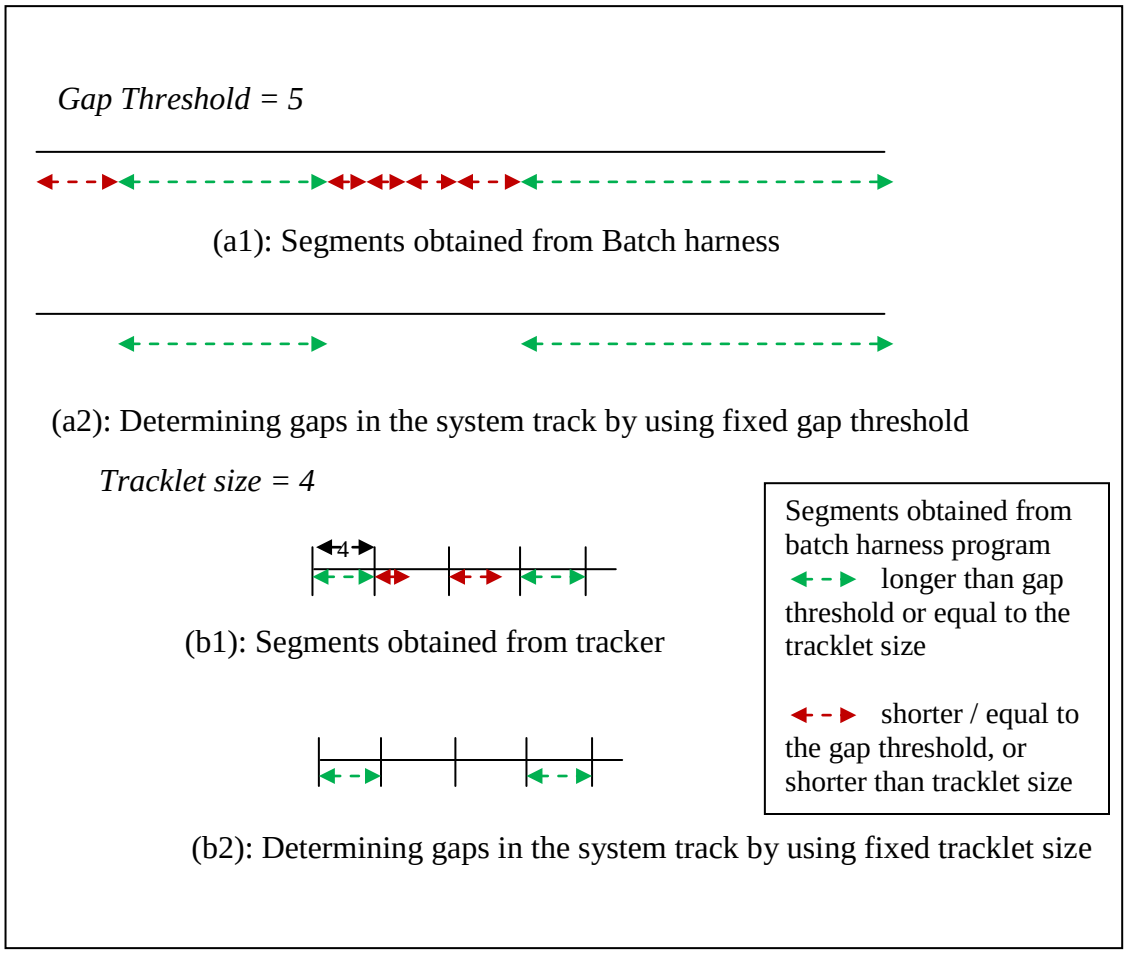


Figure 6.7: Determining Gaps

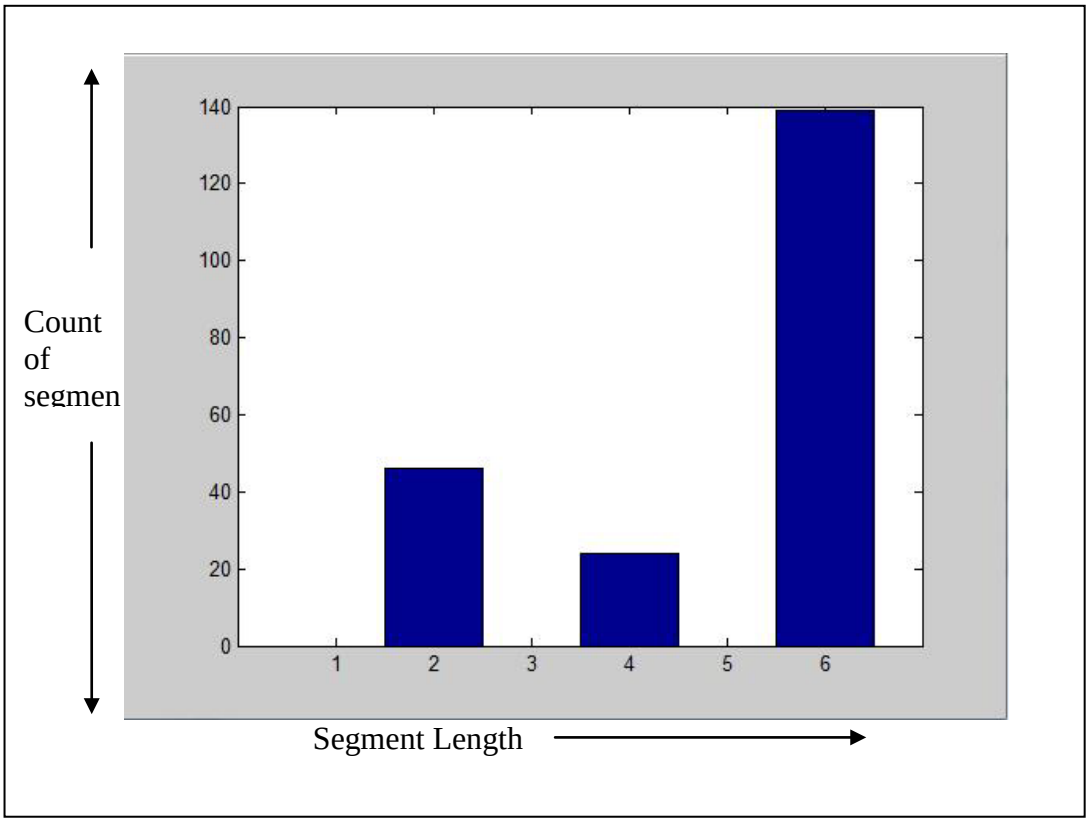


Figure 6.8: Histogram of segment size for tracklet 6 for 10 tracks

Tracklet = 6

Track Number	Average	Minimum	Maximum	Mode
1	3.78	2	6	2
2	4.94	2	6	6
3	4.53	2	6	6
4	4.86	2	6	6
5	5.75	4	6	6
6	5.07	2	6	6
7	3.89	2	6	4
8	5.83	2	6	6
9	4.69	2	6	6
10	5	2	6	6

Table 4(a): Average, Min, Mode of segment lengths obtained for tracklet length = 6 , each of the 10 tracks

In the above table, table 4(a), since the tracklet length = 6. The maximum length that the tracker can track is 6, because the entire ground truth is now divided into chunks of 6 frames and the system tracks are compared to these chunks of ground truth.

Track Number	Average	Minimum	Maximum	Mode
1	5	2	22	2
2	10.2	2	24	2
3	9	2	22	6
4	8.062	2	34	2
5	13	2	28	10
6	4.71	2	11	2
7	3.6	2	16	2
8	20.857	4	38	10
9	7.0667	2	20	2
10	7.1429	2	22	2

Table 4(b): Average, Min, Mode of segment lengths obtained by restarting wherever the tracker failed for each of the 10 tracks

6.3 Overview of Existing Track Metrics that do not use Information Theory

Multi-target tracking has been carried on for over half a century [2], and considerable effort has been devoted to find a suitable performance evaluation system for such trackers. Many groups [29], [28] use the ground truth as the true values and compare it against the measurement data. They use numeric scores to quantify the accuracy of detection or tracking algorithm. Black et al. [36] match up the ground truth tracks and system tracks through various surveillance metrics to characterize the tracking performance. Their evaluation is based on the distance between the centroids of the objects and the tracker. Bashir and Porikhli [31] evaluate the performance of the metrics based on the overlap between ground truth and system tracks. They build a correspondence matrix between multiple ground truth objects to multiple tracker result objects using three different methods of distance computation between trajectories. Brown et al [30] allow each system track to match to many ground truth tracks and then they use this information to identify the false positives and the true negatives. The above metrics are similar to detection ROC curve. In my next sub section I will define the existing tracking metrics.

6.3.1 Defining Existing Tracking Metrics

In this section, I will define the completeness metrics as given by Kao et al. C_t is the truth completeness and C_s is the track completeness. Equation(1) and (2) gives the completeness metrics for many-many association case. Equation (3) and (4) gives the completeness metrics for the one-one association case.

$$C_T^M = (\sum_{r \in T} \sum_{c \in S} l_{rc})(\sum_{r \in T} l_r)^{-1} \quad \text{.....(1)}$$

$$C_S^M = (\sum_{c \in S} \sum_{r \in T} l_{rc})(\sum_{c \in S} l_c)^{-1} \quad \text{.....(2)}$$

$$C_T^0 = (\sum_{r \in T} \max_{c \in S}(l_{rc}))(\sum_{r \in T} l_r)^{-1} \quad \text{.....(3)}$$

$$C_S^0 = (\sum_{c \in S} \max_{r \in T}(l_{rc}))(\sum_{c \in S} l_c)^{-1} \quad \text{.....(4)}$$

In the above equations, T is the set of all truth tracks, S is the set of all system tracks, l_{rc} is the length of the association between the truth track r and the system track c. l_r is the length of the truth track r and l_c is the length of the system track c.

C_T^M **:** Is the fraction of truth tracks observed by the system (probability of detection) , also known as Truth Completeness for many-many case (equation 1).

C_S^M : It is the fraction of the system tracks that correspond to the truth tracks, also known as track completeness for many-many case (equation 2).

C_T^0 **:** It is the ratio between the sum(maximum length of the segments) over the different ground truths to which the track coincides, to the total length of the ground truth, also known as Truth Completeness for one-one case (equation 3).

C_S^0 **:** It is the ratio between the sum of the maximum length of segments that correspond to the different ground truth (to which the track coincides) to the total length of the system track sequence, also known as track completeness for one-one case.(equation 4).

Track Incompleteness for many-many case = 1-Track Completeness = 1- C_S^M --(5)

Track Incompleteness for one-one case = 1-Track Completeness =1- C_S^0 ---- (6)

To demonstrate the computation of these metrics a simple example can be used. See figure 6.9.

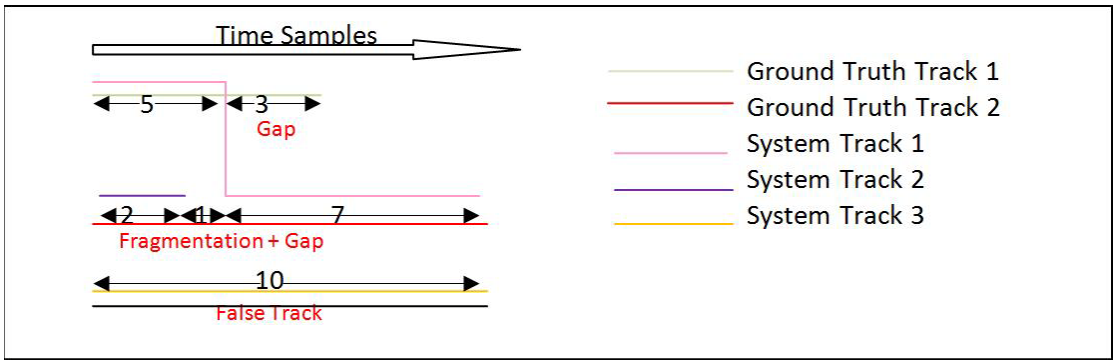


Figure 6.9: Multiple track pathologies (after [2])

From the above example (figure 6.9), the metrics calculation stands as:

Cmt = (5+2+7)/(8+10) = 77% (from equation 1)

Cms = (5+7+2)/(12+2+10) = 58% (from equation 2)

C0t = (5+7)/(8+10) = 66% (from equation 3)

C0s = (5+7)/(12+2+10)=50% (from equation 4)

6.4 Applying Existing Non-Information Theory Metrics to our LOFT tracker

As already discussed earlier, our tracker is a single object automatic and assisted tracker. Hence, the tracks that are generated from the tracker, consists of multiple segments. The size of the segments is equal to the number of frames that are successfully tracked from the start point of the tracker upto (but not including) the point where the tracker failed in

the ground truth sequence. When the tracker fails at a particular point the entire number of frames tracked from the start point to the end point (upto which the tracker could successfully track in the ground truth sequence) is given. Hence, for a given ground truth track, the system track generated by the tracker will consist of multiple segments. For evaluating the tracker performance, the entire set of segments for that particular ground truth is recorded and then analyzed.

6.4.1 Adapting the Track metrics discussed in section 6.3.1 to our LOFT tracker

C_T^M **:** For the case of our LOFT tracker, it is the ratio between the no. of actual segments obtained by running our tracker (after estimating the gap) with the total length of the ground truth track.

C_S^M : In our case, since ours is a single object tracking system, the system tracks observed by the tracker always correspond to the truth tracks.

C_T^0**:** In our case, the tracker coincides to one and only one ground truth, so it is the ratio between the maximum length of a system track to the total length of the ground truth track.

C_S^0**:** In our case, the tracker coincides to one and only one ground truth, so it is the ratio between the maximum length of a system track to the total length of the system track sequence.

The above metrics' does not give a comprehensive overall correspondence between the truth tracks and the system tracks.

Sample Computation Using fixed Gap Threshold for a Particular System Track

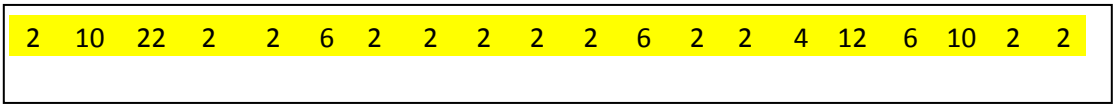


Figure 6.10: Length of Segments obtained from tracker

Sample Computation of Track Metrics by using figure 6.10 and gap threshold=2:

C_T^M = (10+22+6+6+4+12+6+10)/100 = 0.76 (applying equation 1 in section 6.3.1)

C_S^M = (10+22+6+6+4+12+6+10)/ (10+22+6+6+4+12+6+10) = 1 (applying equation 2 in section 6.3.1)

C_T^0 = 22/100 = 0.22 (applying equation 3 in section 6.3.1)

C_S^0 = 22/(10+22+6+6+4+12+6+10) = 0.2895 (applying equation 4 in section 6.3.1)

Track Incompleteness (many-many case) = 1- C_S^M = 0 (applying equation 5 in section 6.3.1)

Track Incompleteness (one-one case) = 1- C_S^0 =0.7105 (applying equation 6 in section 6.3.1)

No. of Gaps from figure 6.10 (after merging consecutive gaps) = 5

No. of Fragments (successfully tracked segments) from figure 6.10 = 8. The no. of gaps and the no. of fragments obtained from the system track is needed to plot the gap vs fragments plot.

Sample Computation Using fixed Tracklet size for a Particular System Track

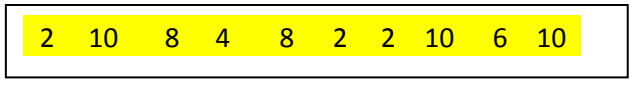


Figure 6.11: Length of Segments obtained from tracker for Tracklet size 10

Sample Computation of Track Metrics by using figure 6.11 and tracklet length = 10

C_T^M = (10+10+10)/100 = 0.3 (applying equation 1 in section 6.3.1)

C_S^M = (10+10+10)/ (10+10+10) = 1 (applying equation 2 in section 6.3.1)

C_T^0 = 10/100 = 0.1 (applying equation 3 in section 6.3.1)

C_S^0 = 10/30 = 0.33 (applying equation 4 in section 6.3.1)

No. of Gaps from figure 6.11 = 7

No. of Fragments (successfully tracked segments) from figure 6.11 = 3. The no. of gaps and the no. of fragments obtained from the system track is needed to plot the gap vs fragments plot.

Track Incompleteness (many-many case) = 1- C_S^M = 0 (applying equation 5 in section 6.3.1)

Track Incompleteness (one-one case) = 1- C_S^0 =0.6667 (applying equation 6 in section 6.3.1)

6.4.2 Results obtained from LOFT tracker - non-information theoretic metrics

Results obtained

To obtain the results the ground truth dataset described in Table 2 has been used.

Approach using Gap Threshold

For evaluating the track metrics in this approach, the following is done:

First a gap threshold is decided, this threshold is to decide which segments will be considered as gaps and which are not. In the event of tracking the whole ground truth sequence by the tracker, if there are consecutive gaps, they are merged into 1.

The figures below gives the results obtained on a set of ground truth data with a gap threshold of 2 frames. Figure 6.12(a), figure 6.12(b) below, shows the plot of the many-many completeness and the one-one completeness computed for each of the 10 system tracks. The x-axis represents the track incompleteness which is one minus track completeness and the y-axis the truth completeness. Figure 6.12(c) gives the gap vs fragment plot, the x-axis represents the no. of fragments obtained for each system track and the y-axis gives the no.of gaps in each of those system tracks for a gap threshold of 2.

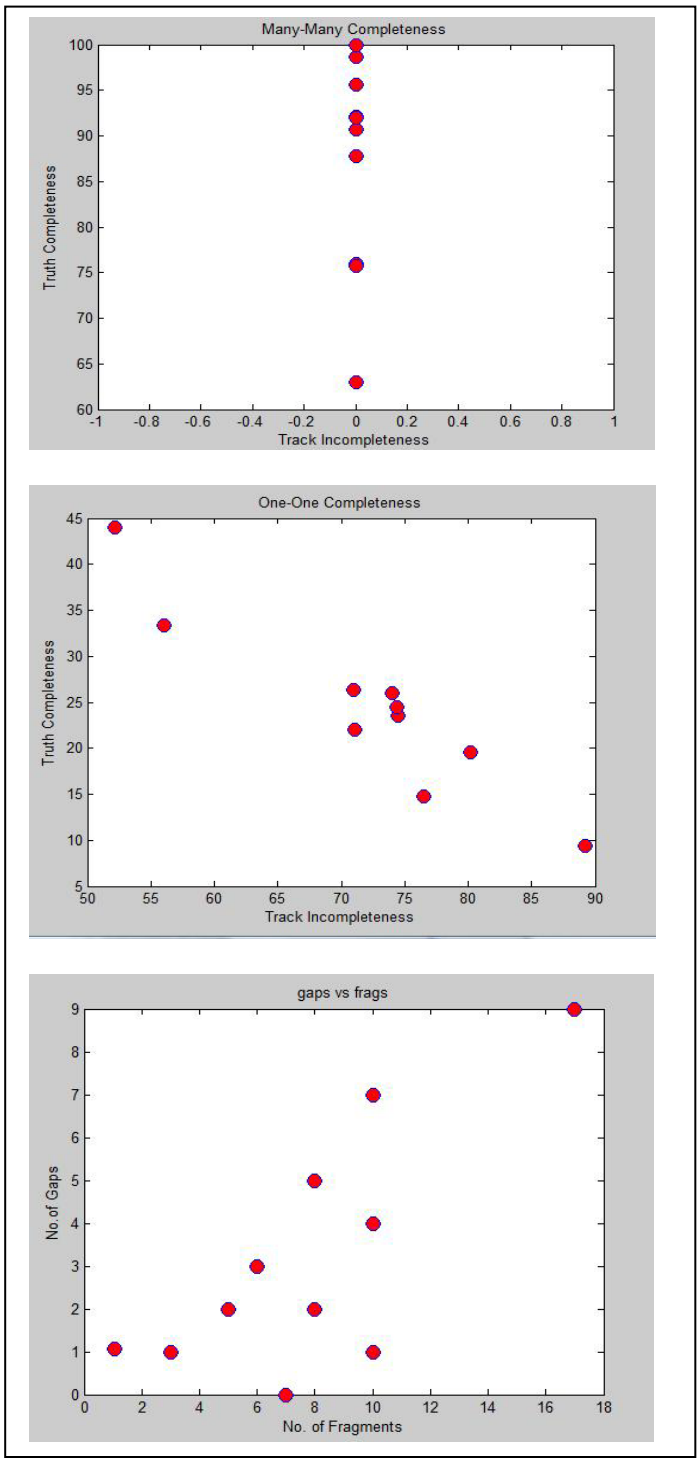


Figure 6.12[Top –Bottom](a,b,c): Many-Many Completeness, One-One Completeness, Gap vs Fragmentation. *Each of the red dots correspond to different tracks for a gap threshold = 2*

By varying the gap threshold, for 10 different thresholds (for the set of tracks from table 2), the probability of detection and the probability of false alarm can be plotted, for the one-one case and the many-many case.

Here truth completeness is the probability of detection and track incompleteness is the probability of false alarm. Figure 6.13(a) and figure 6.13(b) shows the plot for many-many and one-one case by varying the gap threshold . Table 5 and Table 6 shows the values used in the plot and the corresponding gap thresholds.

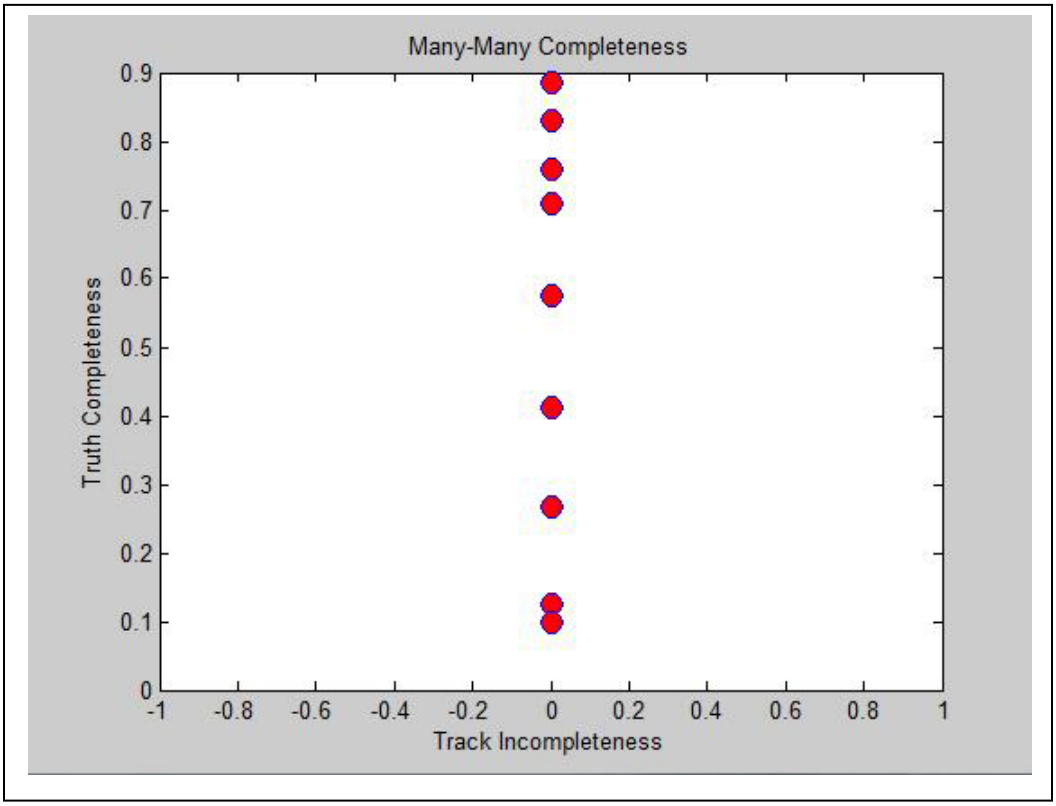


Figure 6.13(a): Many-Many completeness by varying gap threshold 10 times

mt	0	0	0	0	0	0	0	0	0	0
cm	0.885	0.830	0.760	0.709	0.574	0.412	0.268	0.123	0.123	0.098
Ga	2	5	7	9	11	15	20	25	27	30

Table 5: Many-Many Completenss and Gap Threshold, (mt:False alarm, cm:Probability of detection, Ga: Gap Threshold

In table 4, mt = probability of false alarm, cm= probability of detection and ga= gap threshold. The plot in figure 6.12(a) shows that the probability of detection for the many-many completeness decreases as the gap threshold is increased. Our system tracks always correspond to truth tracks hence for the many-many case where we are observing the the probability of false alarm which is 1-mt = 0 for each of the gap thresholds.

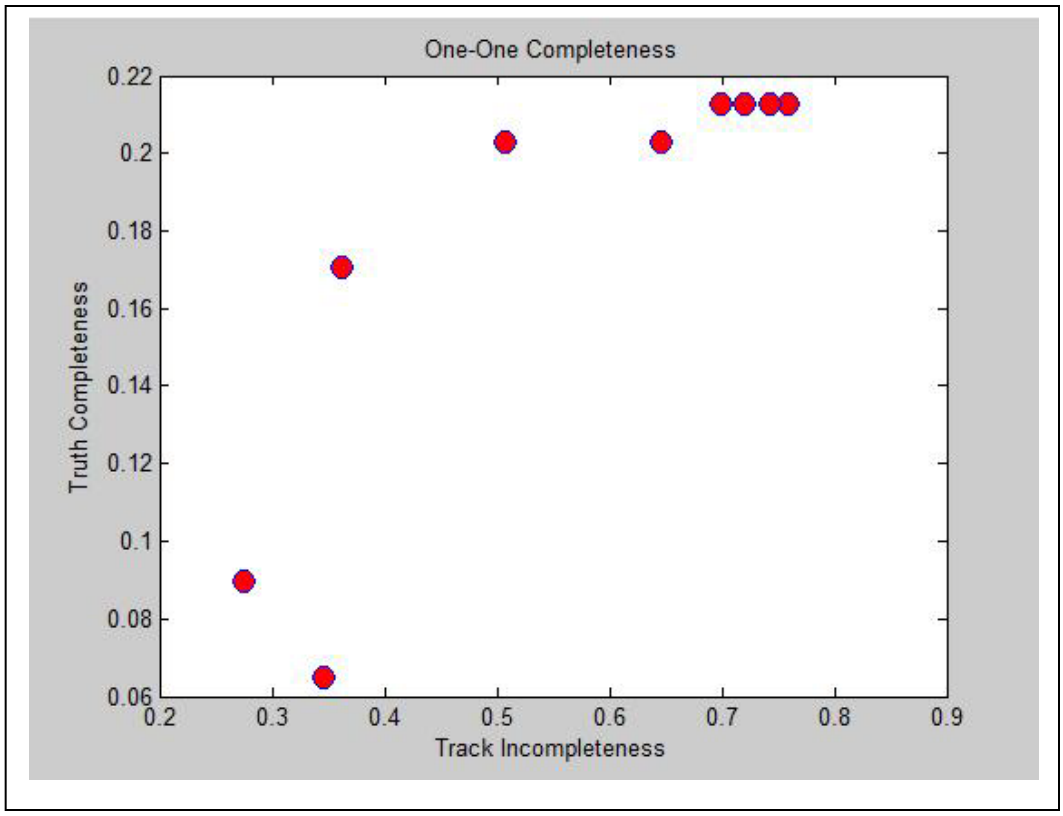


Figure 6.13(b): One-One completeness for 10 tracks by varying gap threshold 10 times

ot	0.759	0.743	0.719	0.7	0.646	0.646	0.507	0.275	0.275	0.345
ct	0.213	0.213	0.213	0.213	0.203	0.203	0.203	0.089	0.089	0.064
Ga	2	5	7	9	11	15	20	25	27	30

Table 6: One-One Completenss and Gap Threshold, (ot:False alarm, ct:Probability of detection, Ga: Gap Threshold)

The plot in figure 6.13(b) shows that the probability of detection for the one-one completeness decreases as the gap threshold is increased.

Approach Using the Tracklet

In this approach the tracklet length is fixed before starting the harness program and also the same tracklet length is used for evaluation. If the tracker is not able to track the whole tracklet then the whole trcaklet is considered as a gap ,thus, tracker is penalized severely for it. In the results obtained the tracklet length is fixed as 6 for each tracks.

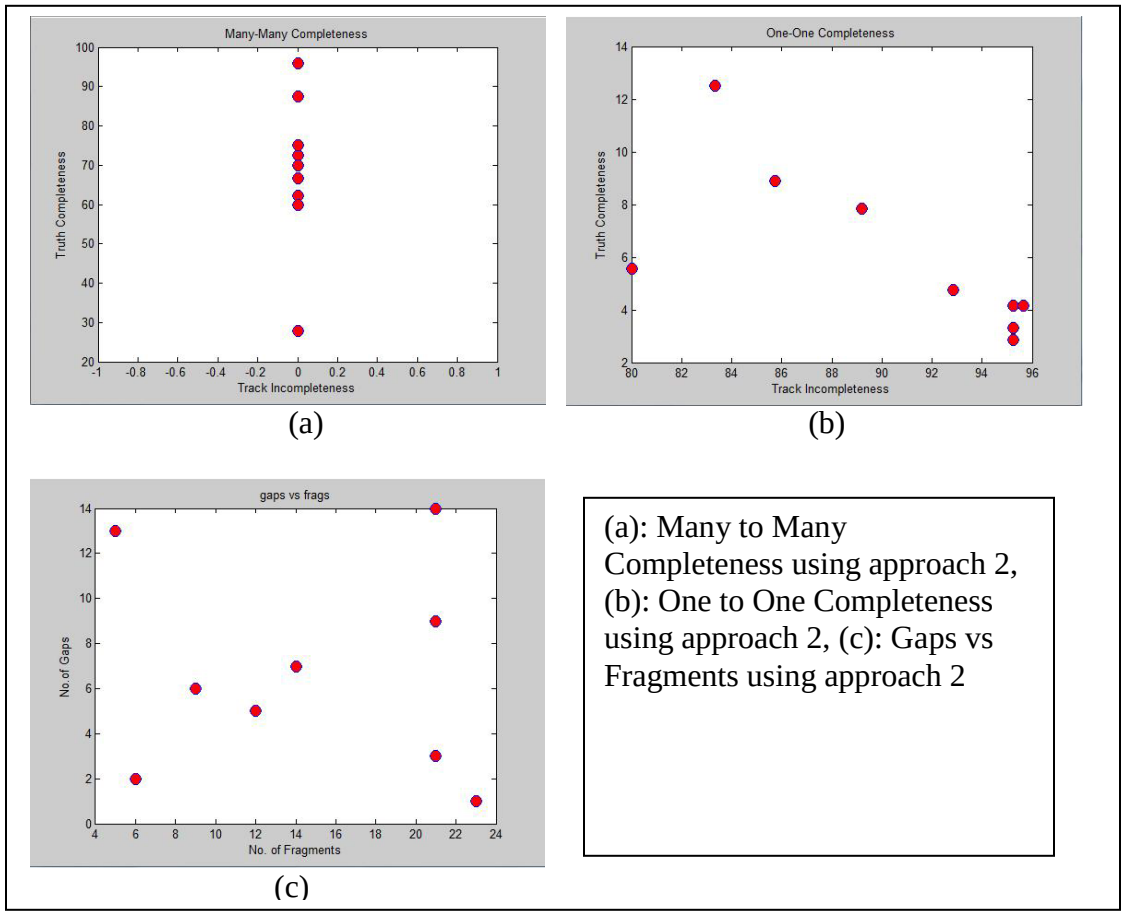


Figure 6.14: Non-information theoretic metrics plot for approach with tracklet *Each of the red dots correspond to different tracks for a tracklet length = 6*

Figure 6.15 shows the gap vs fragment plot for a high tracklet size

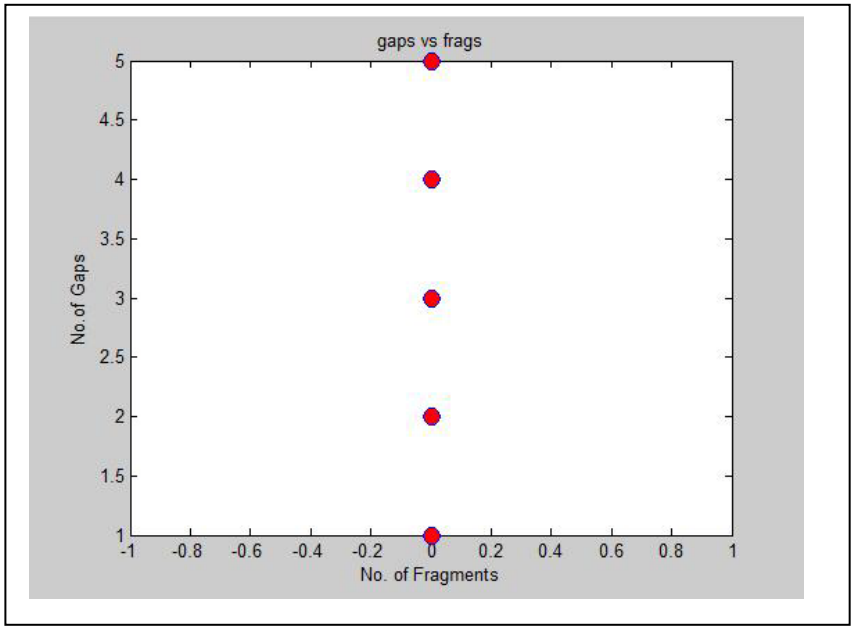


Figure 6.15: Gap vs fragment (i.e gaps vs successfully tracked segments) plot for tracklet length 39

By varying the tracklet size, for 10 different values (for the set of tracks from table 2), the probability of detection and the probability of false alarm can be plotted, for the one-one case and the many-many case.

Here truth completeness is the probability of detection and track incompleteness is the probability of false alarm. Figure 6.16(a) and figure 6.16(b) shows the plot for many-many case and one-one case by varying the tracklet size . Table 7 and Table 8 shows the values used in the plot and the corresponding tracklet size.

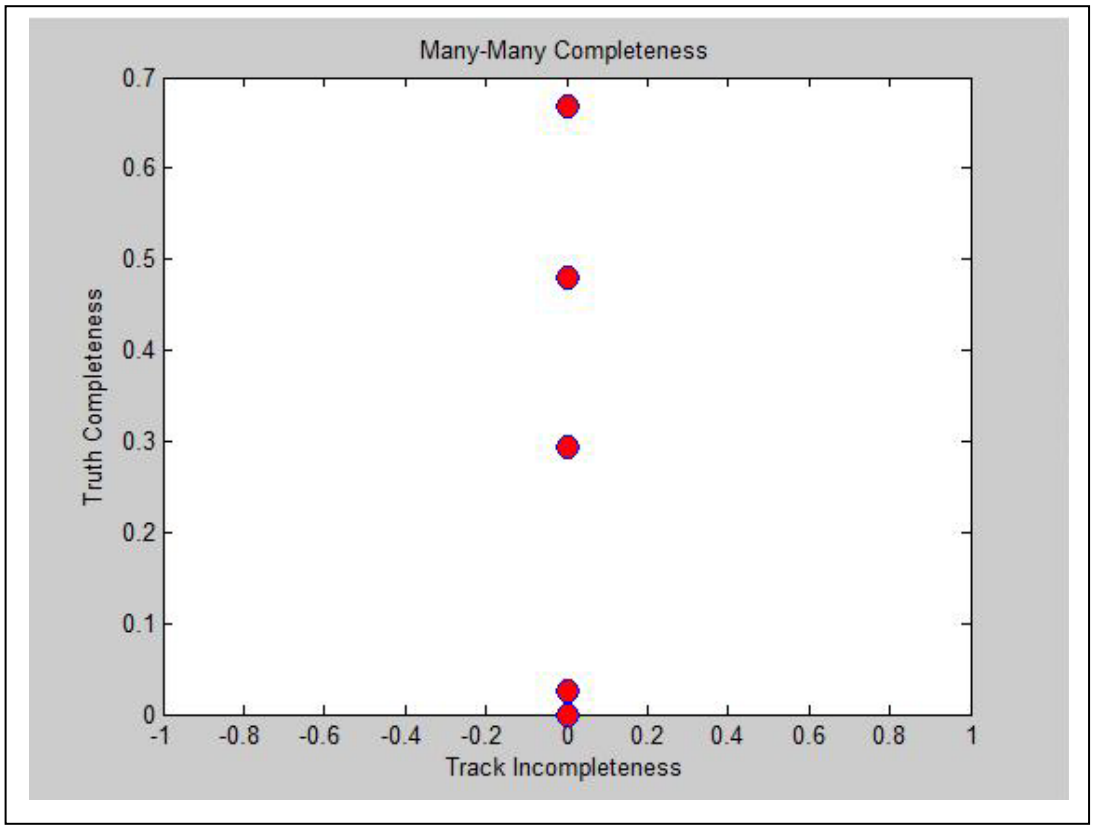


Figure 6.16(a): Many-Many completeness by varying tracklet size 10 times

mt	0	0	0	0	0	0	0	0	0	0
cm	0.668	0.48	0.294	0	0	0.026	0	0	0	0
Tk	6	10	16	23	27	30	35	39	43	50

Table 7: Many-Many Completenss and Tracklet Size, (mt:False alarm, cm:Probability of detection, Tk:Tracklet length

The above table gives the values of the probability of detection (Truth Completeness), false alarm (Track Incompleteness) and the gap threshold (Ga) used in the plot of figure 6.16(a). Our system tracks always correspond to truth tracks hence for the many-many case , the probability of false alarm which is 1-mt = 0 for each of the tracklet size. Also,

for larger tracklets since the tracker could not track any portion of ground truth data, any metrics for that is zero.

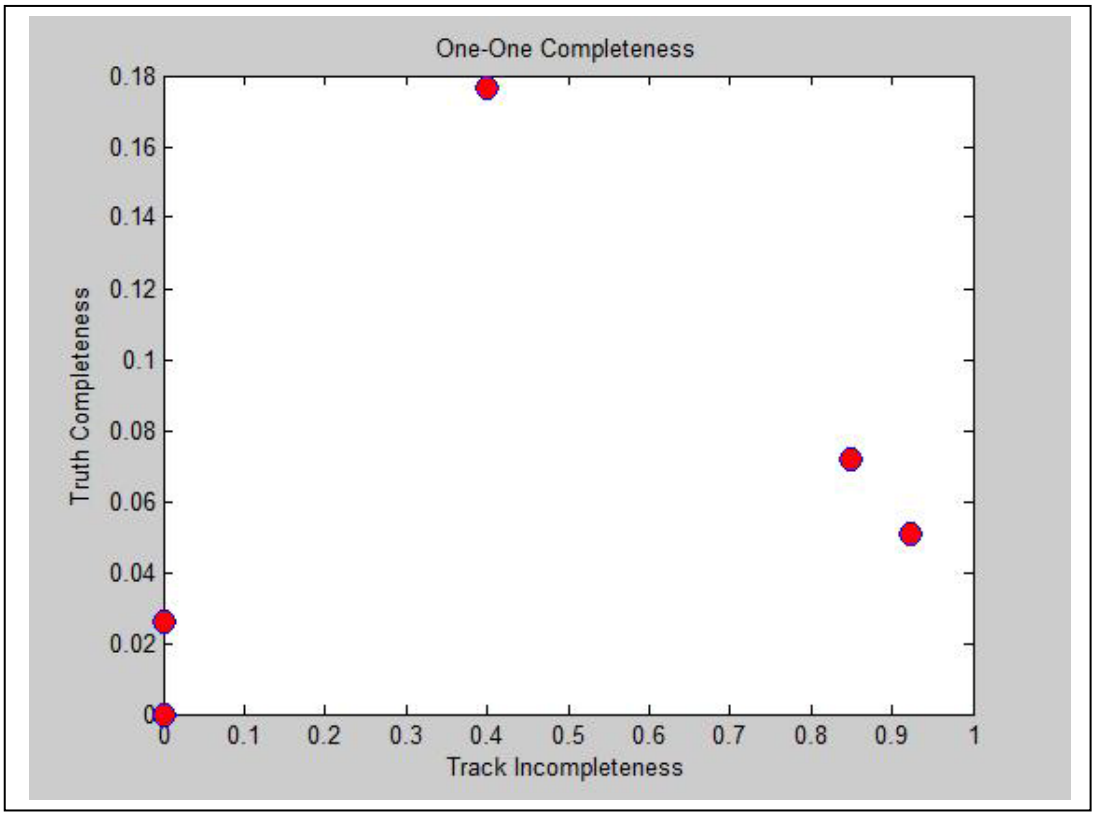


Figure 6.16(b): One-One completeness by varying tracklet size 10 times

ot	0.924	0.85	0.4	0	0	0	0	0	0	0
ct	0.051	0.072	0.176	0	0	0.026	0	0	0	0
Tk	6	10	16	23	27	30	35	39	43	50

Table 8: One-One Completenss and Tracklet Size, (ot:False alarm, ct:Probability of detection, Tk:Tracklet length

Discussion of the Results Obtained

Many-Many Completeness: As can be seen from the above figures, track Incompleteness is zero in our case, since Track Incompleteness is basically 1 – Track Completeness. Track Completeness from equation (1) (subsection 6.3.1) is the fraction of the system tracks that correspond to the truth tracks. In our single object tracking system, the system tracks observed by the tracker always correspond to the truth tracks. Hence Track Incompleteness is zero. The Cmt or Truth Completeness from equation (1) (subsection 6.3.1), is the fraction of truth tracks observed by the system (probability of detection), which is the ratio between the sum of the actual segments obtained by running our tracker (after estimating the gap) with the total length of the ground truth track.

One-One Completeness: As already seen in subsection 6.3.1, the Track Incompleteness in the one-one case is 1-Track Completeness. Track Completeness is the ratio between the sum (maximum length of the segments) over the different ground truths to which the track coincides, to the total length of the system track. In our case, the tracker coincides to one and only one ground truth, so it is the ratio between the maximum length of a system track to the total length of the system track sequence. Now the truth completeness is the ratio between the sum (maximum length of the segments) over the different ground truths to which the track coincides, to the total length of the ground truth. In our case, the tracker coincides to one and only one ground truth, so it is the ratio between the maximum length of a system track to the total length of the ground truth track.

Gap Vs Fragmentation: Tries to find out a relationship between the gaps and the number of fragments obtained for a particular gap threshold over the different tracks. But it fails to establish any, as the tracks vary in length.

For the approach with tracklets it can be seen in figure 6.15 that because of the severe penalization approach, there are no successfully tracked segments. The segments obtained from the tracker are all less than 39 frames and hence all of them are considered as gaps.

Apart from the way the two approaches (gap threshold and tracklet based) are computed and the overtly penalization case of a tracklet in high tracklets (resulting in large gaps), there is not much difference between the two.

As we can see, the completeness computed for the many-many case, does not penalize the track fragmentation. On the other hand, the completeness for the one-one case overly penalizes fragmentation. Hence, it can be said, that these completeness metrics are inadequate in evaluating a tracker, which will have track fragmentation as a very common track pathology. Thus, a good method is needed to combine to combine them into a holistic approach. Hence Kao et al [2] described a single metric based on information theory, which captures all the common track pathologies and also provides a comprehensive overall performance measure. Kao et al's group takes an approach by measuring directly the similarity between the ground truth and the observed tracks.

6.5 Overview of Information Theoretic Metrics for Performance Evaluation

Information theoretic measures based on Shannon's measure of information are used by different groups for evaluation of ROC curve data, in medical decision making for assessing the performance of non-binary diagnostics tests[37], in determining the quantitative estimates of the consequences of inspection error [38] and also for

performance evaluation of multi-object trackers [2]. The work of Kao et al is unique from the rest because they have proposed an overall scoring function over two dimensional metric space. We have adapted the work of Kao et al. from a multi-object tracker evaluation system to our single object tracker evaluation system and finally developed two tracker performance evaluation tools using two different approaches as explained in 6.2.3. In this section I will give an overview of the approach used by Kao et al, for evaluating the tracker performance.

As already stated above, Kao et al. used the information theoretic metrics to evaluate the tracker performance by interpreting the tracker data as messages received over a communications channel. The truth data is compared to as the original message. If the original data, in this case the truth dataset, is known, then the information theoretic metrics can be easily calculated.

Defination of terms used in equations below:

H(T): the total amount of truth track information

H(S): the total amount of system track information

I(T:S): the amount of matching information between the truth tracks and the system tracks, (also known as mutual information)

H(T|S): the amount of truth tracks information missed by the system tracks (also known as truth conditional entropy)

H(S|T): the amount of false information introduced by system tracks (also known as tracker conditional entropy)

A single score is obtained by summing the two conditional entropies. This score measures the amount of true information that a tracker misses plus the amount of false information that it generates.

In other words, Score (S1) = H(T|S) + H(S|T).

Again, H(T) = I(T;S) + H(T|S) and

H(S) = I(T;S) + H(S|T),

Normalizing the mutual information (I(T;S), and tracker conditional entropy (H(S|T)) with entropy of truth dataset we get:

f(T;S) = I(T;S)/H(T)

r(S|T) = H(S|T)/H(T)

The truth information completeness f(T;S) is a measure of the fraction of the truth information collected by a tracker. The false information r(S|T) is the ratio of false information to the truth information.

6.5.1: Computing the Association Matrix

As a step to determining the two different quantities, truth information completeness and the false information ratio, a joint probability density function between the truth and the system tracks is needed. The approach that is taken to generate the joint probability density function is to select a series of times to compare the tracker data to the ground truth. The best association between the system tracks and truth tracks is determined in each time sample. The number of times the tracks in the two sets associate or match and do not associate are counted. These counts are stored in a two dimensional matrix (known as the association matrix) and used to generate the probability density function. The first row(column) of the association matrix stores the number of times the truth(system) tracks

did not associate with any of the system(truth) tracks. Thereafter, each row(column) in this matrix, is for a particular truth(system) track. Figure 6.17 shows the association matrix representation

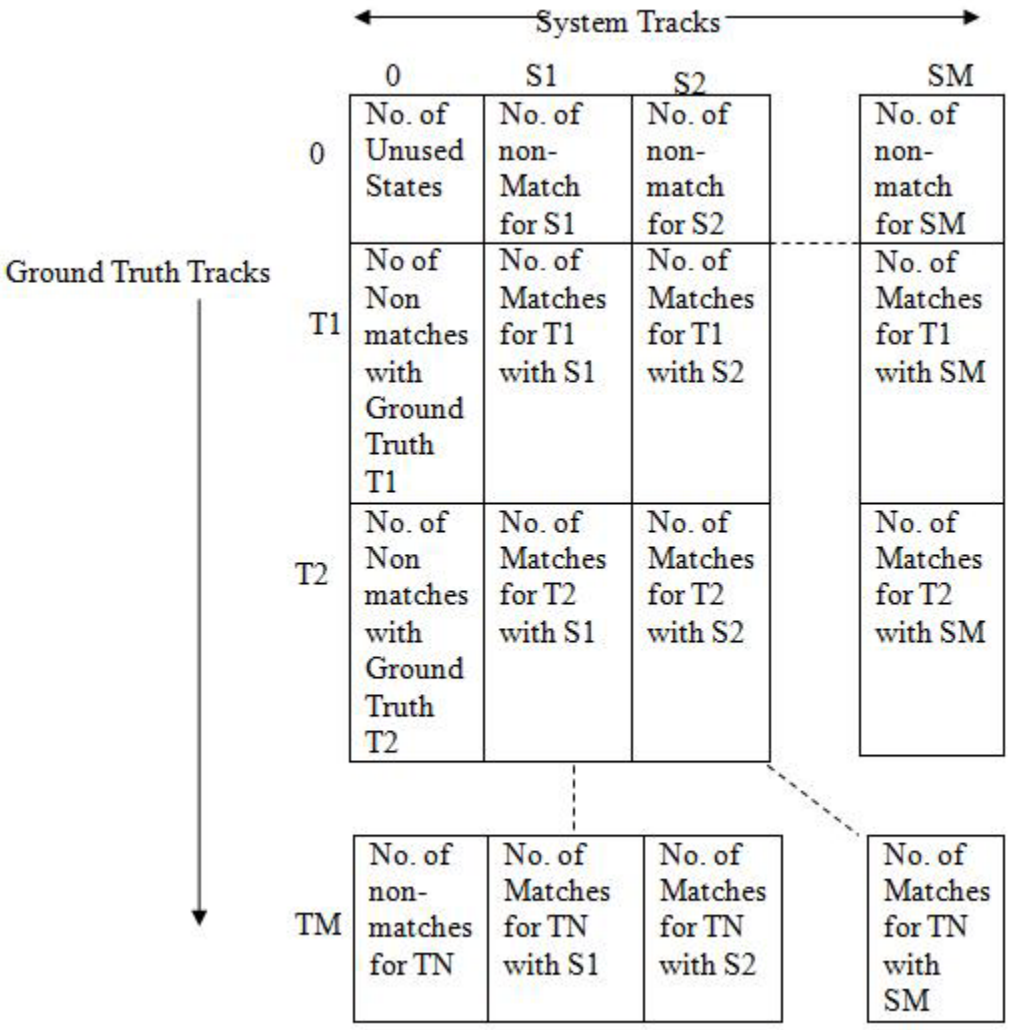


Figure 6.17: Association table used to generate joint probability (after [2]

Hence, these both sets of tracks (ground truth and system tracks) are now defined on a common state space.

6.5.2 Computation of the different probability terms and information metrics

In this subsection the different probability terms are obtained and finally from these probability terms the information metrics are obtained. First the joint probability density density function P(s, t) is obtained. For this, the association matrix is normalized by the total state estimate, in this case, sum of all the entries along the row and column. The system and truth track probability functions are P(s) and P(t). The conditional probability density functions are P(t|s) and P(s|t).

Evaluating other probability terms

P(t): This is obtained by summing along the rows of the normalized association matrix

P(s): This is obtained by summing along the columns of the normalized association matrix.

P(t|s) = P(s,t)/P(s)

P(s|t) = P(s,t)/P(t)

The information metrics is computed by using the below equations [39]:

$$H(T) = -\sum_t P(t)\log(P(t))$$

$$H(S) = -\sum_s P(S)\log(P(s))$$

$$H(S \mid T) = \sum_{s,t} P(s,t)\log(P(s \mid t))$$

$$H(T \mid S) = -\sum_{s,t} P(s,t)\log(P(t \mid s))$$

$$I(T;S) = \sum_{s,t} P(s,t)\log(\frac{P(s,t)}{P(s)P(t)})$$

Using the above information metrics, the truth information completeness and the false information ratio can be calculated. They are as follows:

Truth information completeness = I(S;T)/H(T)

False information ratio = H(S|T)/H(T)

6.5.3: Information Coverage Plot

Finally an information coverage plot between Truth information completeness (I(S;T)/H(T)) and False information ratio H(S|T)/H(T) is done. Figure 6.18(a).shows this information coverage plot. Point (0,1) is the point of perfect tracker performance. It is shown in the graph as a blue circle. Here, the false information generated is 0 and all the truth information is captured. By varying single tracker parameters, curves can be generated in the two dimensional space. Better trackers will produce curves towards the upper left corner of the graph. Figure 6.18(b) shows such a scenario, where 2 curves in red and green obtained by varying tracker parameters by 2 different tracking algorithms. Clearly, the red curve is a better tracker than the green one.

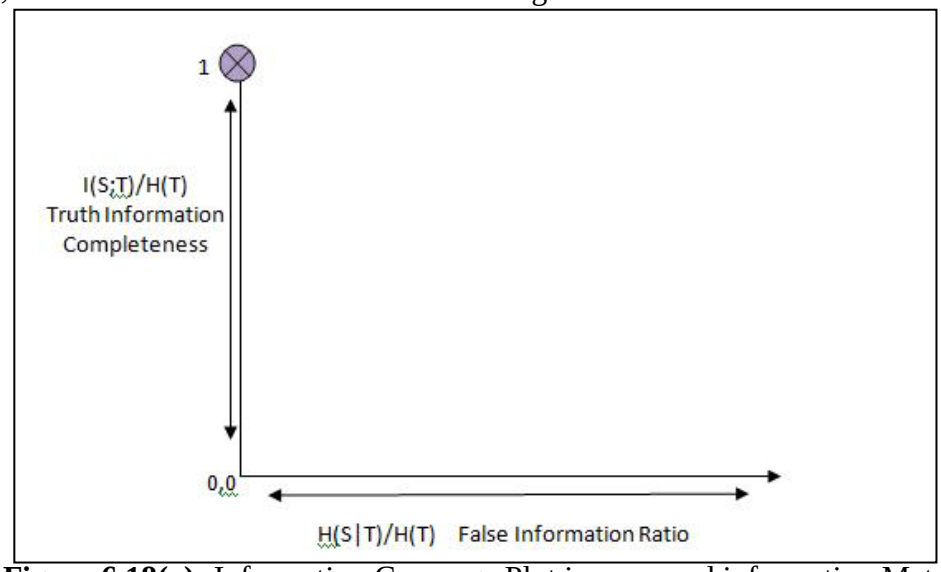


Figure 6.18(a): Information Coverage Plot in proposed information Metric Space (showing perfect tracker performance)

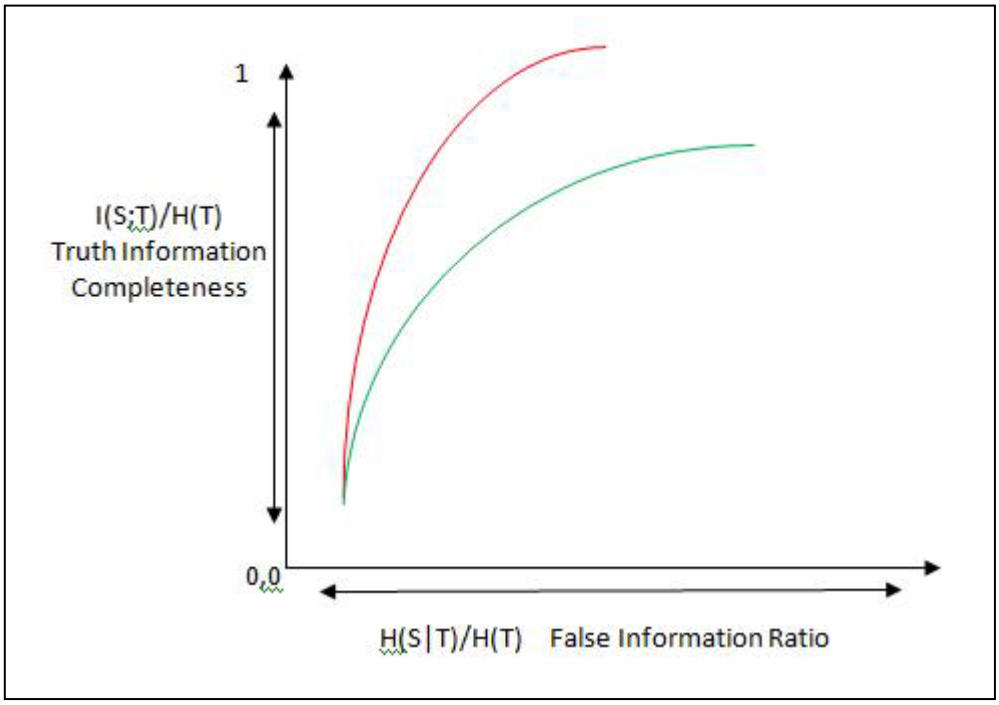


Figure 6.18(b): Information Coverage Plot in proposed information Metric Space (showing 2 curves for 2 different tracking algorithms obtained by varying single tracking parameter in the algorithms)

6.6 Using Information Theoretic metrics for our LOFT tracker

In this section, I will describe how the information theoretic method, described above, is applied to our tracker, which is a single object automatic and assisted tracker, (different from the multi-object automatic trackers used by Kao et al [2]. In the latter part of this chapter, I will also demonstrate our tracker performance using 2 different methods of applying the information theoretic approach. One of which is a tracklet based approach and the other is a gap threshold based approach. I will explain the differences between the two methods and show how our tracker fares with the two methods and explain why it behaves the way it does.

6.6.1 Adapting Information Theoretic Approach from multi-object tracking to our Single Object LOFT tracker

For our single object case, there is only one kind of association possible between the system tracks and truth tracks, since only one system track corresponds to a particular truth track. Hence, the number of times the tracks in the 2 sets (system and ground truth tracks) associate, is basically the number of frames for which the tracker could track successfully. The number of times the tracker could not associate is basically the gaps in our system tracks. For our tracker evaluation, the time sample that is used for generating the probability density function, are the segments for which the tracker could track. The gaps are generated, depending on the gap threshold or tracklet size, (i.e if the tracker could track equal to or less number of frames than the gap threshold then that segment is a gap). These counts are stored in a two dimensional matrix (known as the association matrix) and used to generate the probability density function. The first row(column) of the association matrix stores the number of times the truth(system) tracks did not associate with any of the system(truth) tracks. Thereafter, each row(column) in this matrix, is for a particular truth(system) track [2]. Figure 6.19 shows the association matrix representation.

Computing the association matrix for approach using Gap Threshold

In our case, this two dimensional association matrix is formed by considering all the system tracks and the truth tracks. The columns are formed by placing the system track segments generated from ground truth track 1, 2 etc placed one after another. For our tracker, (because of the conditions, based on which it is made to start and stop by the harness program), it either has an association with its ground truth or a gap (not

associated with the ground truth, failure to track and hence stopped). Also, the tracker does not track beyond its ground truth. Thus the first row of this matrix is zeros, because the tracker is stopped the moment the Euclidean distance between ground truth point and system track point for that particular frame goes beyond 60 pixels or it reached the end of ground truth point. From the second row onwards, each row(column) in the matrix stores the association about the specific truth(system) track. The first column in each row (from the second row onwards), has the information on the length of the unassociated part (with a specific ground truth) of the system track, the gaps (i.e the number of frames for which the tracker could not successfully track). So for eg: the first column in second row represents the number of frames in ground truth sequence 1 that could not be successfully tracked. Since, in our case, only one system track is associated with a ground truth track, the 2 dimensional matrix gets an interesting form as shown in the figure below.

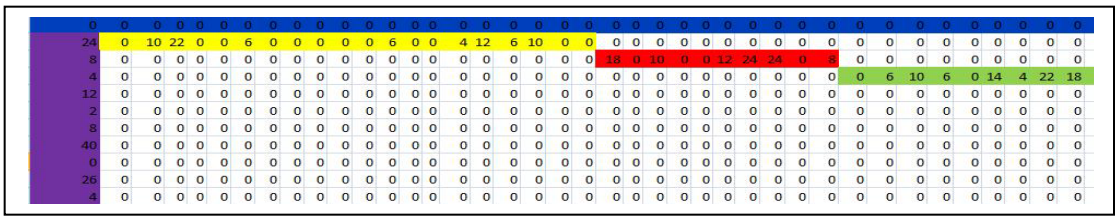


Figure 6.19: Association table for our single tracker case, Columns: System Track segments, Rows: Ground Truth

Discussion of association matrix obtained for our tracker (using gap threshold)

Figure 6.19 shows a snapshot of the association matrix. Along the columns of the association matrix are the different system track segments. The first row shows the segments that are not associated with any of the ground truth. In our case, this row is zero and is highlighted in deep blue. From the second row, starting from second column

onwards, it represents the association of each of the system tracks' segments with any ground truth data. In our case, since it is a single object tracker system, the system track segments are associated with only one ground truth sequence. In the above figure, system track segments for track 1 is associated with only ground truth sequence 1, highlighted in yellow. The 3rd row shows the association between system track segment 2 and any ground truth data. Again, in our case, it is associated with just ground truth 2, highlighted in red, the rest of the segments for this row is zero.

Plot of the information coverage for a fixed gap threshold

Thus, these both sets of tracks (ground truth and system tracks) are now defined on a common state space. Now to plot the information coverage plot, we use the formulae from subsection 6.5.2. The plot for our tracker is shown in figure 6.20.

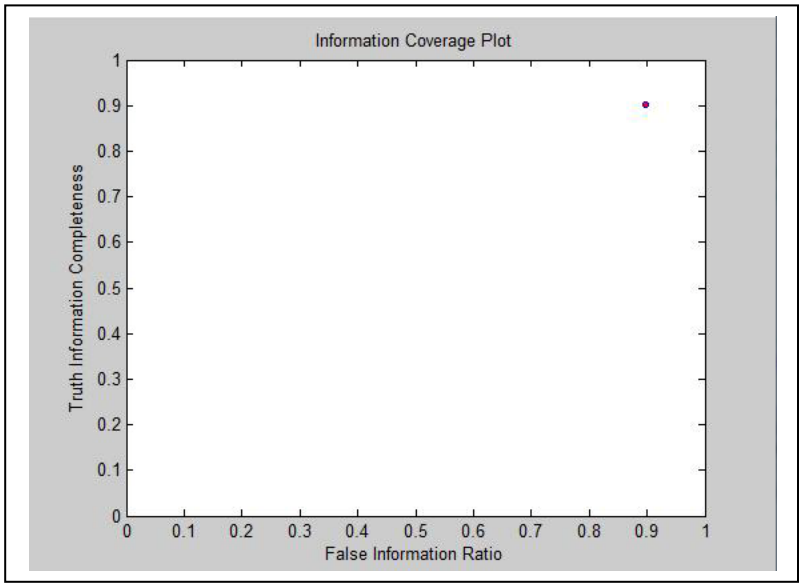


Figure 6.20: Information Coverage plot using a particular gap threshold for 10 tracks (Used Gap Threshold = 2)

By varying this gap threshold for a set of ground truth data, a curve can be traced, (in this research 10 different gap threshold values are used). These are plotted to show the performance of the tracker using this approach. See figure 6.21.

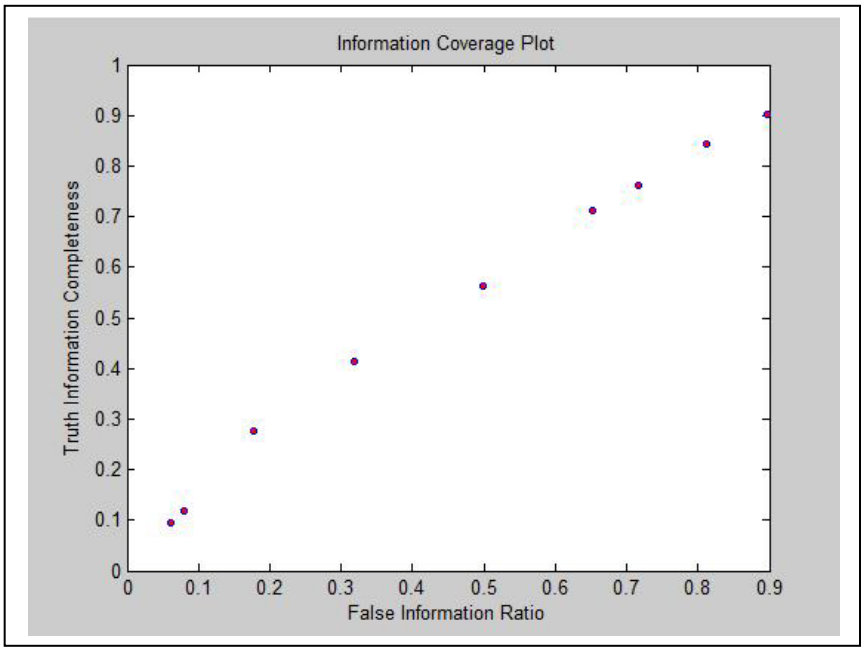


Figure 6.21: Information Coverage plot obtained by varying the gap threshold

The below table gives the values of the Truth Information Completeness (TI), false information ratio (FI) and the gap threshold (Ga) used in the plot of figure 6.21

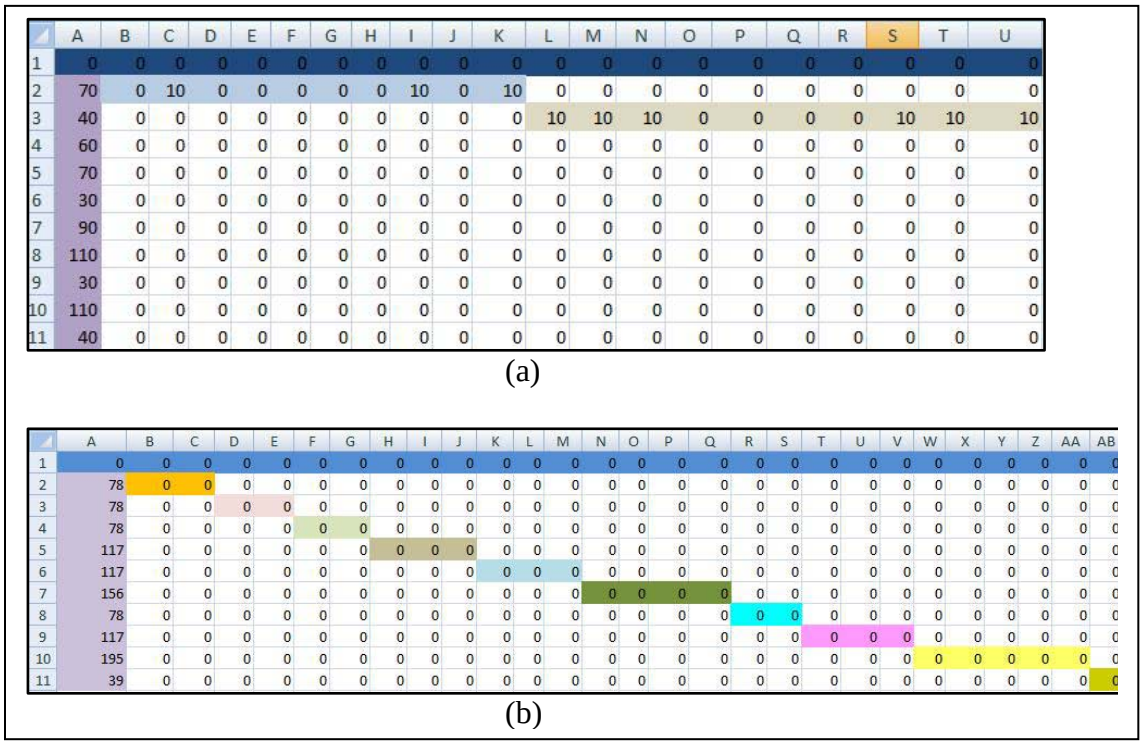
FI	0.896	0.812	0.716	0.653	0.499	0.318	0.178	0.079	0.079	0.061
TI	0.903	0.843	0.762	0.710	0.564	0.413	0.277	0.118	0.118	0.094
Ga	2	5	7	9	11	15	20	25	27	30

Table 9: Gap threshold and Information Coverage values

It can be seen from the plot and the table, that the best result is obtained with gap threshold = 11. In this set, it is the point, which is closest to the upper left hand corner of the curve.

Computing the association matrix for approach using tracklets

The steps to compute the association table remains the same as in the approach using gap threshold. However, it is seen that the gaps become more as the tracklet lengths increases. For example, for tracklets of length 39, the association table only has gaps. It could not track any segment for greater than 39 frames . This is because as explained in 6.2.3, we are penalizing the whole segment and considering it as zero, if it is not able to track the whole segment. With this high tracklet length, the tracker could not track continuously a stretch of 39 frames for any of the tracks, hence the association table consists of only gaps. Figure 6.22 shows the association table with two different tracklet lengths for the same set of tracks.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	70	0	10	0	0	0	0	0	10	0	10	0	0	0	0	0	0	0	0	0	0
3	40	0	0	0	0	0	0	0	0	0	0	10	10	10	0	0	0	0	10	10	10
4	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(a)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	117	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	117	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	156	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	117	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	195	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b)

Figure 6.22: Association tables with tracklet length (a) 10 and (b) 39

Consequently, the information theoretic metrics like the false information, mutual information, missed information all become zero for the case (b) in figure 6.19. The information coverage plot also becomes zero in this case. In this tracklet based approach the information coverage plot is done by varying the tracklet length for a set of tracks (10). 10 different tracklet lengths are used to trace a curve based on 10 different information coverage plots.

The figure 6.23 below shows the information coverage plot for approach using tracklets.

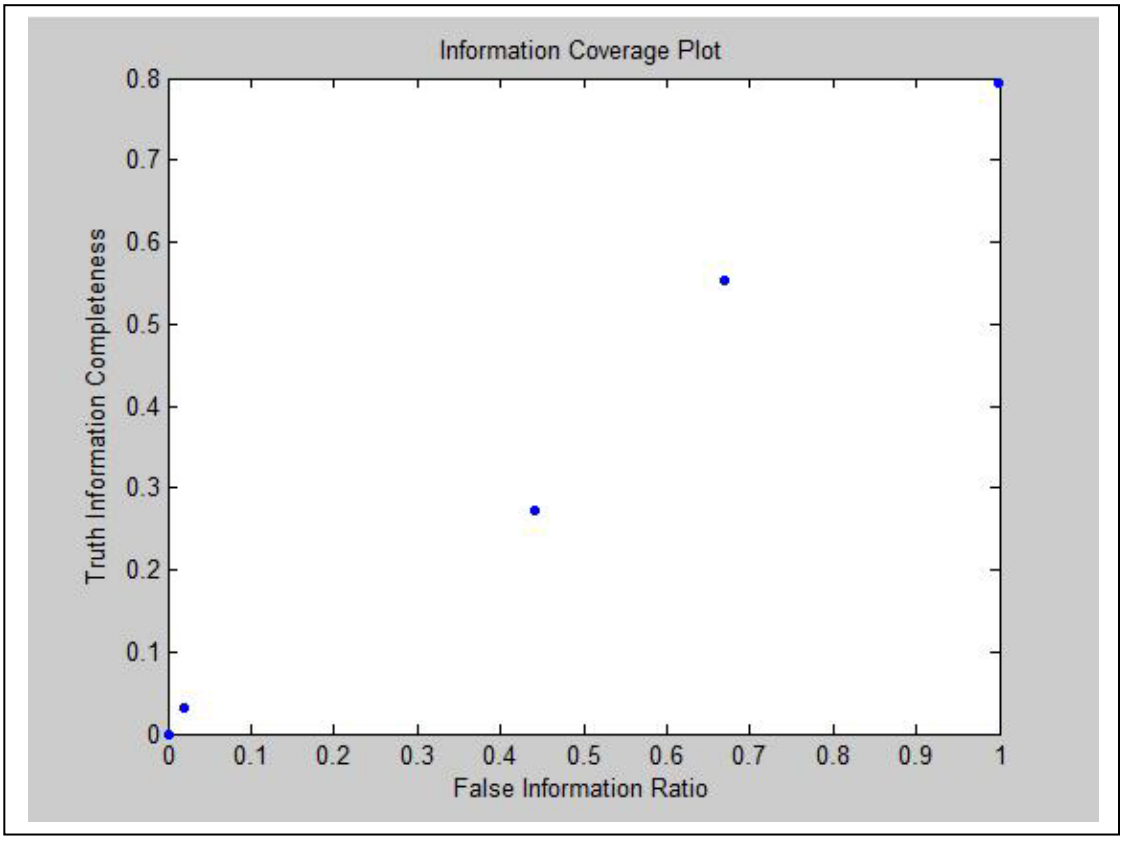


Figure 6.23: Information coverage Plot by varying Tracklet size

As can be seen from the graph in figure 6.23, for quite a few runs the information coverage plot is 0. This happens for the reason explained earlier, if the tracklet length exceeds 2, the tracker is not able to track continuously greater than 23 frames at a stretch

and hence, the information metrics' become 0. So only 4 points out of the 10 can be seen in the graph. The rest 6 become 0.

The below table gives the values of the Truth Information Completeness (TI), false information ratio (FI) and the tracklet length(Tk) used in the plot in figure 6.23

FI	0.998	0.670	0.442	0	0	0.019	0	0	0	0
TI	0.795	0.552	0.271	0	0	0.031	0	0	0	0
Tk	6	10	16	23	27	30	35	39	43	50

Table 10: Tracklet Length and Information Coverage values

As discussed in section 6.5.3, the optimum point is towards the upper left hand corner. However with this approach, the tracker fails to meet desired performance levels, the best performance of the tracker (out of these cases) is with tracklet size 10. The tracker is unable to track any point as soon as the tracklet size becomes equal and greater than 23. Out of the 2 approaches, one using gap threshold and the other using tracklet, the tracker performs better with the approach using gap threshold, which is a much lenient approach.

6.7 Conclusion

In this chapter, an overall performance evaluation system is explained. The various components of the evaluation system has been explained in detail. The 3 main components of this system namely the ground truth generation system, the harness program which automatically starts and stops the tracker and the performance evaluation module are tightly integrated with each other in series. The output from each module is fed as input to the next module. Algorithms developed and the programs written to

automate the process of performance evaluation of the tracker are clearly described in this chapter. The main advantages of the harness program is that it makes the task of starting and stopping the tracker a smooth one. The hurdles in each of the modules have been explained in detail. For example, the challenges while capturing the ground truth data have been explained. The ground truth data that has been collected has been well documented so that they can be used in the future as the tracker develops over the coming years. For the performance evaluation module, 2 different approaches have been used. One approach using the gap thresholds and other using the tracklets have been developed. The two approaches will help to develop a robust tracker. Both the approaches have been used for finding out how the tracker performs in a non-information theory case as well as with evaluators using information theory. The shortcomings of the non-information theory case has been explained. The comparison of the performance of the tracker with these two approaches have been given.

The evaluation tool using both the approaches will prove to be an invaluable tool in improving the performance of the tracker. As the tracker develops over the years, this tool can be used to measure, how much better it is performing with respect to the previous years. It will also prove to be an invaluable tool for other activities like parameter tuning, algorithm comparison etc.

CHAPTER 7

CONCLUSION

7.1 Conclusion

The objective of the work was to track the motion of cars in a Wide Area Motion Imagery (WAMI) that presented a number of challenges. Initially, various known computer vision algorithms were applied to the dataset to understand how each algorithm fared. This gave an idea as to in which direction the research should proceed. We chose to break down the overall car tracking problem into smaller modular problems like road network extraction, car detection and car tracking. These were presented in the subsequent chapters in detail.

We presented the road network extraction in Chapter 3, in Chapter 4, we presented a car detection algorithm and in Chapter 5 we presented an improved version of Camshift algorithm capable of tracking in grey scale images. This improved and modified algorithm uses a constant velocity prediction and takes into account influence of neighbouring pixels, which the original Camshift algorithm does not. It is adapted to work with this dataset. In each of the modules the challenges associated with them is explained in detail.

Finally, the research is concluded with the development of an evaluation system for our low frame rate tracker. Two systems using different approaches were developed for evaluating the tracker. One of the approaches is a severe tracker error penalization approach. This will test the tracker in extreme harsh cases. The other is a more lenient approach, but does not allow the user any control over starting and stopping the tracker. Whereas the former approach enables the user more control on starting and stopping the tracker. The performance of the tracker with both these approaches using information theoretic metrics and non-information theoretic metrics is given.

7.2 Future Work

Building a car tracker for this highly complicated and challenging dataset is an ongoing problem. In this research, various techniques have been developed and tried. The scope of improvements is outlined at the end of every chapter. Due to the lack of a good, reliable map database the map information is not yet integrated with the main system. The car tracking system can be improved further by handling occlusion events, which is a common phenomenon in this dataset. Right now only single tracking of cars is focused at, but this problem can be expanded to tracking of multiple cars. For the performance evaluation module, a limited evaluation of the low frame rate tracker has been done on selected tracks, for a thorough evaluation of the tracker , this evaluation program can be used on all tracks.

APPENDIX A

Summary of Ground Truth File for PSS Data Set

1. Name of data set: Juarez

Folder location: C:\pss_pss\jrz_2009_08_26

Non MUA ground truth:

a. Number of cars track: 15

b. Approximate total combined length of tracks (starts from 46904 to 47996): 1092 frames

MUA ground truth:

a. Number of cars track: 15

b. Approximate total combined length of tracks (starts from 46904 to 52414): 5510frames

Non MUA manual ground truth			MUA manual ground truth		
Car No.	Starting frame	End frame	Car No.	Starting frame	End frame
car 1	46904	46988	car 1	46904	47303
car 2	46989	47064	car2	46062	49767
car 3	47067	47250	car 3	49184	50279
car 4	47208	47419	car 4	48383	49467
car 5	47512	47694	car 5	48525	49544
car 6	47520	47760	car 6	48913	49695
car 7	47568	47760	car 7	48956	49532
car 8	47744	47897	car 8	48975	49685
car 9	47757	47787	car 9	49100	49538
car 10	47768	47809	car 10	49775	50677
car 11	47780	47815	car 11	51406	51744
car12	47818	47890	car12	51731	52286
car13	47887	47966	car13	51731	52414
car 14	47898	47930	car 14	49822	50441
car 15	47952	47996	car 15	46904	47294

2. Name of data set Philadelphia

Folder location: C:\pss_pss\Philadelphia

Non MUA ground truth:

a. Number of cars track: 9

b. Approximate total combined length of tracks (starts from 45700 to 46641): 941 frames

MUA ground truth:

a. Number of cars track: 3

b. Approximate total combined length of tracks (starts from 46574 to 47781): 1207 frames

Non MUA manual ground truth			MUA manual ground truth		
Car No.	Starting frame	End frame	Car No.	Starting frame	End frame
car 1	45700	46641	car 1	46574	46787
car 2	45700	45803	car2	46574	46625
car 3	45700	45791	car 3	46639	47781
car 4	45700	45830			
car 5	45700	45844			
car 6	45700	45883			
car 7	45702	45736			
car 8	45890	45999			
car 9	45925	46091			

3. Name of data set: Nascar

Folder location: C:\pss_pss\2008_05_25_racestart

Non MUA ground truth:

a. Number of cars track: 20

b. Approximate total combined length of tracks (starts from 62800 to 63324): 524 frames

MUA ground truth:

a. Number of cars track: 8

b. Approximate total combined length of tracks (starts from 62800 to 63900):: 1180 frames

Non MUA manual ground truth			MUA manual ground truth		
Car No.	Starting frame	End frame	Car No.	Starting frame	End frame
car 1	62800	63002	car 1	62800	63900
car 2	62800	63230	car2	62800	63478
car 3	62800	63061	car 3	62800	63775
car 4	62800	63075	car 4	62800	63900
car 5	62800	63073	car 5	62800	63638
car 6	62800	62994	car 6	62800	63782
car 7	62800	63016	car 7	62800	63812
car 8	62805	62961	car 8	62800	63681
car 9	62811	63033			
car 10	63250	63324			
car 11	62800	62867			
car12	62800	62984			
car13	62800	62886			
car 14	62800	62912			
car 15	62800	63066			
car 16	62800	62998			
car 17	62800	62849			
car 18	62800	62896			
car 19	62800	63012			
car 20	62800	62943			

BIBLIOGRAPHY

[1] K. Palaniappan, R. M. Rao, and G. Seetharaman, "Wide-Area Persistent Airborne Video," in *Distributed Video Sensor Networks*, 1st ed, B. Bhanu, C. V. Ravishankar, A. K. Roy-Chowdhury, H. Aghajan, and D. Terzopoulos, Eds., 2011.

[2] E. K. Kao, M. P. Daggett, and M. B. Hurley, "An Information Theoretic Approach For Tracker Performance Evaluation," *International Conference on Computer Vision (ICCV),* 2009.

[3] K. Palaniappan, F. Bunyak, P. Kumar, I. Ersoy, S. Jaeger, K. Ganguli, A. Haridas, J. Fraser, R. Rao, and G. Seetharaman, "Efficient feature extraction and likelihood fusion for vehicle tracking in low frame rate airborne video," in *13th Int'l Conference on Data Fusion*, 2010.

[4] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. Computer Vision and Pattern Recognition,* June 1999.

[5] US._Census_Bureau: Geography Division, December, 2008.

[6] S. Hinz, "Automatic road extraction in urban scenes – and beyond," *International Archives of Photogrammetry and Remote Sensing,* vol. 35, pp. 349-355, 2004.

[7] J. Hu, A. Razdan, J. C. Femiani, M. Cui, and P. Wonka, "Road Network Extraction and Intersection Detection From Aerial Images by Tracking Road Footprints," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 45, 2007.

[8] K. Treash and K. Amaratunga, "Automatic Road Detection in Grayscale Aerial Images," *Journal of Computing in Civil Engineering,* pp. 60 – 69, Jan 2000.

[9] G. Koutaki and K. Uchimura, "Automatic road extraction based on cross detection in suburb," *SPIE USE,* vol. 5, pp. 5299-36, 2003.

[10] Mapping_Toolbox.

[11] T. Zhao and R. Nevatia, "Car detection in low resolution aerial images," *Image and Vision Computing,* vol. 21, pp. 693-703, 2003.

[12] C. Papageorgiou and T. Poggio, "A Trainable System for Object Detection," *International Journal of Computer Vision,* vol. 38, pp. 15-33, 2000.

[13] C. Schlosser, J. Reitberger, and S. Hinz, "Automatic car detection in high-resolution urban scenes based on an adaptive 3Dmodel," in *Proc. IEEE/ISPRS Workshop on "Remote Sensing and Data Fusion over Urban Areas"*, 2003.

[14] H. Schneiderman and T. Kanade, "A Statistical Method for 3D Object Detection Applied to Faces and Cars," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2000, pp. 746-751.

[15] A. N. Rajagopalan and R. Chellappa, "Higher-order-statistics-based detection of vehicles in still images," *Journal of Optical Society America A*, vol. 18, December 2001.

[16] S. Hinz, D. Lenhart, and J. Leitloff, "Detection and Tracking of Vehicles in Low Frame Rate Aerial Image Sequences."

[17] L. Dreschler and H.-H. Nagel, "Volumetric model and trajectory of a moving car derived from monocular TV frame sequence of a street scene," *Computer Graphics and Image Processing*, vol. 20, pp. 199-228, 1982.

[18] M. Haag and H.-H. Nagel, "Combination of Edge Element and Optical Flow Estimates for 3D-Model-Based Vehicle Tracking in Traffic Sequences," *International Journal of Computer Vision*, vol. 35, pp. 295–319, 1999.

[19] Dubuisson-Jolly, M.-P., S. Lakshmanan, and A. Jain, "Vehicle Segmentation and Classification Using Deformable Templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 293–308, 1996.

[20] T. Tan, G. Sullivan, and K. Baker, "Model-Based Localisation and Recognition of Road Vehicles," *International Journal of Computer Vision*, vol. 27, pp. 5–25, 1998.

[21] A. Rajagopalan, P. Burlina, and R. Chellappa, "Higher Order Statistical Learning for Vehicle Detection in Images," in *IEEE International Conference on Computer Vision*, 1999, pp. 1204-1209.

[22] B. Meffert, R. Blaschek, U. Knauer, R. Reulke, A. Schischmanow, and F. Winkler, "Monitoring traffic by optical sensors," in *2nd International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2005, pp. 9-14.

[23] J. Kang, I. Cohen, M. G., and C. Yuan, "Detection and Tracking of Moving Objects from a Moving Platform in Presence of Strong Parallax," in *International Conference on Computer Vision*. vol. I, 2005 pp. 10-17.

[24] Q. Yu, I. Cohen, G. Medioni, and B. Wu, "Boosted Markov Chain Monte Carlo Data Association for Multiple Target Detection and Tracking," in *International Conference on Pattern Recognition*. vol. II, 2006, pp. 675-678.

[25] S. Punvatavungkour and R. Shibasaki, "Three line scanner imagery and on-street packed vehicle detection," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 35, 2004.

[26] I. Ernst, M. Hetscher, K. Thiessenhusen, M. Ruhe, A. Borner, and S. Zuev, "New approaches for real time traffic data acquisition with airborne systems," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVI, pp. 69-73, 2005.

[27] G. R. Bradski, "Computer Vision Face Tracking for use in a Perceptual User Interface," *Intel Technology Journal*, pp. 1-15, 1998.

[28] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti, and R. Bolle, "Appearance Models for Occlusion Handling," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* Kauai, HI, December, 2001.

[29] T. Ellis, "Performance Metrics and Methods for Tracking in Surveillance," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* Copenhagen, Denmark, June 2002.

[30] L. M. Brown, A. W. Senior, Y.-l. Tian, J. Connell, A. Hampapur, C.-F. Shu, H. Merkl, and M. Lu, "Performance Evaluation of Surveillance Systems Under Varying Conditions," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* Colorado, USA, Jan 2005.

[31] F. Bashir and F. Porikli., "Performance Evaluation of Object Detection and Tracking Systems," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* New York, USA, June 2006.

[32] F. Yin, D. Makris, and S. Velastin, "Performance Evaluation of Object Tracking Algorithms," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* Rio De Janeiro, Brazil, October, 2007.

[33] Ç. E. Erdem, B. Sankur, and A. M. Tekalp, "Performance Measures for Video Object Segmentation and Tracking," *IEEE Transactions on Image Processing*, vol. 13, pp. 937-951, July 2004.

[34] K. Palaniappan and J. Fraser, "Multiresolution tiling for interactive viewing of large datasets," in *Proceedings of the 17th International Conference on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology*, 81st AMS, Jan. 2001, pp. 338-342.

[35] J. B. Fraser and K. Palaniappan, "Kolam: A high-performance architecture for the visualization of extremely large datasets," Dept. of Computer Science, University of Missouri, Columbia 2002.

[36] J. Black, T. Ellis, and P. Rosin, "A novel method for video tracking performance evaluation," in *International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2003, pp. 125-132.

[37] C. E. Metz, D. J. Goodenough, and K. Rossman, "Evaluation of Receiver Operating Characteristic Curve Data in Terms of Information Theory, with Applications to Radiography, ," *Radiology*, vol. 109, pp. 297-303, 1973.

[38] T. Raz, "Information theoretic measures of inspection performance," *International Journal of Production Research*, vol. 29, pp. 913-926, 1991.

[39] T. M. Cover and J. A. Thomas, *Elements Of Information Theory*. New York, NY, USA: Wiley-Interscience, 1991.