The Pennsylvania State University
The Graduate School

LIDAR-BASED MULTI-OBJECT TRACKING SYSTEM WITH DYNAMIC MODELING

A Thesis in
Electrical Engineering
by
Mengran Gou

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2012

Mengran Gou

The thesis of Mengran Gou was reviewed and approved* by the following:

Constantino Lagoa
Professor of Electrical Engineering
Thesis Advisor, Chair of Committee

Sean Brennan
Associate Professor of Mechanical Engineering
Thesis Co-Advisor

William E. Higgins
Distinguished Professor of Electrical Engineering
Committee Member

*Signatures are on file in the Graduate School.

Abstract

Laser-based detection of objects and features is becoming increasingly common due to the high accuracy of these sensors and their dropping costs. One of the biggest challenges in the use of laser-based measurements is to track the motion of multiple moving objects simultaneously and in situations where a moving object might be temporarily occluded, for example a vehicle moving behind another vehicle relative to the sensors position. In such situations, accurate motion predictions are essential.

This thesis develops motion-prediction models suitable for laser-based multi-object tracking systems. In the literature, a number of different approaches have been studied. However, most of them cover a limited set of models or the model is too conservative to take into account any previous information of the object. In this thesis, an online dynamic model is applied to approximate the motion of the object. The advantages of this model include: a) It does not need the prior knowledge of the object's motion dynamics; b) It can deal with motion changes by updating periodically. These dynamic model predictions are evaluated by comparison to known, simulated data. Then, the approach is tested by using field-measured range data collected in urban scenes. The results show that the dynamic motion model-based multi-tracking system can track different dynamic motions and be robust to the motion changes, and can often predict the states of occluded objects.

Table of Contents

List of Figures vi

List of Tables ix

Acknowledgments x

Chapter 1
Introduction 1
1.1 Motivation . . . 1
1.2 Multi-object tracking system . . . 1
1.3 Thesis Organization . . . 2

Chapter 2
Related Work 3
2.1 Early days: Vision-based tracking systems . . . 3
2.2 The late 1990s: Introduction of laser rangefinders . . . 4
2.2.1 Historical development of laser-based tracking system . . . 4
2.2.2 Moving objects detection for laser-based tracking system . . . 4
2.2.2.1 Background subtraction . . . 5
2.2.2.2 Segmentation . . . 5
2.2.2.3 Occlusion . . . 7
2.2.2.4 Classification . . . 7
2.2.3 Filtering methods for laser-based tracking system . . . 9
2.2.3.1 Kalman Filter . . . 9
2.2.3.2 Particle Filter . . . 10
2.2.4 Data association approaches for laser-based tracking system . . . 12
2.2.4.1 Greedy Nearest Neighbor (GNN) Filter . . . 14
2.2.4.2 Joint probabilistic data association (JPDA) . . . 15
2.2.4.3 Multiple Hypothesis Tracking (MHT) . . . 16
2.2.5 Approaches for occlusion handling . . . 17
2.2.5.1 Explicit model with Kalman Filter . . . 18
2.2.5.2 Probability model with particle filter . . . 19
2.2.5.3 Interacting Multiple Models (IMM) . . . 19

2.2.5.4 Dynamic Model 21

Chapter 3
System Setup **23**
3.1 LIDAR system 23
3.2 Experimental Protocol 24
3.2.1 Park Avenue 24
3.2.2 Intersection between Curtin Road and Bigler Road 24

Chapter 4
Methodology **27**
4.1 Pre-processing 28
4.2 Segmentation and classification 30
4.2.1 Segmentation 30
4.2.2 Partially occluded object detection 31
4.2.3 Object classification and feature extraction 33
4.2.4 Line and "L-shape" classification 35
4.2.5 Robust line fitting 36
4.2.6 Weighted line fitting 37
4.2.7 Corner fitting 38
4.2.8 Feature points calculation 39
4.3 Data association 39
4.3.1 GNN data association 41
4.3.2 Feature point association 41
4.4 Model motion 44
4.4.1 Dynamic modeling for motion model 44
4.4.2 Implementation 45
4.4.3 Experimental results 46
4.4.3.1 Tracking of different dynamic motions 46
4.4.3.2 Prediction during occlusion 47

Chapter 5
Experimental Results **54**
5.1 Experimental results 54
5.1.1 Result one 54
5.1.2 Result two 55
5.2 Failure analysis 55
5.2.1 Failed pedestrian tracking result 56
5.2.2 Failed vehicle tracking result 56

Chapter 6
Conclusion **61**

Bibliography **62**

List of Figures

1.1 Summary of multi-object tracking system 2

2.1 Moving objects detection by grid map (from [7]) 5
2.2 Example of angular resolution limition 6
2.3 Examples for DBSCAN. ε is the radius of the circle and N is 2. Points p and q are density-connected and all the red and yellow points form a cluster. Point B is noise. 7
2.4 Example of partial occlusion. Vehicle A is occluded by pedestrian B, so it is divided into two parts: 1 and 2 in the LIDAR data. 8
2.5 Voting scheme based classification, where:+ $\rightarrow$ pedestrian; $*$ $\rightarrow$ car; $\circ$ $\rightarrow$ truck; $\times$ $\rightarrow$ post; $\square$ $\rightarrow$ wall; $\diamond$ $\rightarrow$ bush [30] 8
2.6 Rule based classification [36] 9
2.7 SLAM without DATMO [27] 11
2.8 SLAM with DATMO [27] 11
2.9 SLAM and DATMO with advanced algorithm [28] 11
2.10 Trajectory of people obtained by tracking with Kalman Filter [24] 12
2.11 Comparison of Kalman Filter (right) and Particle Filter (left). The arrow indicates the trajectory of the object. The Kalman Filter predicts the object into the obstacle. [45] 12
2.12 Simultaneous robot localization and people-tracking [6] 13
2.13 Tracking results for laser-based tracking systems using MCMCDA [59] 13
2.14 People tracking results in a railway station [61] 14
2.15 Example of Mahalanobis Metric 15
2.16 Trajectory of tracking four people by sample-based JPDA: the left scene is the true, the right is the estimated trajectory [62] 16
2.17 Results of racking four people: the left one is based on correlated samples and the right one is based on independent samples [46] 16
2.18 Occlusion of an object may cause temporary loss-of-tracking of the occluded moving object [61] 17
2.19 Tracking by constant velocity model. Object 10 is tracked through occlusion, where blue dots represent the trajectories of the moving objects, blue lines stationary obstacle.[40] 19
2.20 Tracking people during occlusion using Brownian model (left) and plan-based model (right) [70]. 20
2.21 Car tracking over missing data by IMM approach [22] 21
2.22 Velocity of the temporary stop target. Solid line represents the IMM with stationary model and dash line the CV model. [42] 21

2.23 Trajectory of people tracked by IMM with adaptive TPM. Green line represents the ground truth, red line the trajectory with adaptive TPM and blue line the trajectory without TPM. [81] . . . 22
2.24 Example of dynamic modeling for robust visual tracking [83] . . . 22

3.1 LIDAR system . . . 24
3.2 The experiment location on Park Avenue . . . 25
3.3 Picture of the experiment location on Park Avenue . . . 25
3.4 The experiment location of the intersection . . . 26
3.5 Picture of the experiment location of the intersection . . . 26

4.1 Flow Chart of entire process . . . 27
4.2 Example of pre-processing . . . 29
4.3 Stationary map area verses threshold value of Park Ave. data set . . . 29
4.4 Grid-map of one experiment location . . . 31
4.5 Flow chart of the classification step . . . 32
4.6 Comparison of different segment methods, DBSCAN (up) groups the segments 11 to 14 obtained by distance threshold segmentation (down) and eliminate the noise segment 19. . . . 33
4.7 Example of the segmentation results of a dark object . . . 34
4.8 Example of middle occluded object . . . 34
4.9 Illustration of occlusion. A, C is totally visible, E has one endpoint occluded, B,D have two endpoints occluded and F is occluded in the middle part. The vertical side of D and the horizontal side of A is marked as occluded due to the angular resolution limitation. . . . 35
4.10 Illustration of line/corner classification. Left one represents the L shape, middle one the line shape and right one the line shape . . . 36
4.11 Example of Robust Line Fitting . . . 37
4.12 Example of corner fitting . . . 39
4.13 Examples of feature points calculation. The black crossings represent the feature points. . . . 40
4.14 Flow char of data association step. . . . 42
4.15 Example of GNN data association . . . 43
4.16 Example of data association . . . 43
4.17 Example of data association . . . 44
4.18 Lengths of the sides . . . 44
4.19 The constant velocity motion . . . 47
4.20 The constant acceleration motion . . . 47
4.21 The turning motion . . . 48
4.22 The forward-backward motion . . . 48
4.23 The moving-stationary-moving motion . . . 49
4.24 The lane change motion . . . 49
4.25 The lane change (hard) motion . . . 49
4.26 Three objects pass by each other. The arrows represent the directions of the movements. . . . 50
4.27 Five objects pass by each other. The arrows represent the directions of the movements. . . . 50

4.28 Object is occluded by an obstacle. The object moves from the right to the left. The width of the obstacle is 4m and size of the training data set is 49. 51
4.29 Object is occluded by an obstacle (less training data). The object moves from the right to the left. The width of the obstacle is 4m and size of the training data set is 34. 51
4.30 Zoom in on Figure 4.29 52
4.31 Object is occluded by an obstacle (long occluded time). The object moves from the right to the left. The width of the obstacle is 5m and size of the training data set is 49. 52
4.32 Zoom in on Figure 4.31 53
4.33 Two pedestrians go crossing after an obstacle. Both of them move from the bottom to the top. 53

5.1 Snapshot of the tracking processing 55
5.2 Example of multi-object tracking result 57
5.3 Example of multi-object tracking result 58
5.4 Trajectory of a failed tracking pedestrian 59
5.5 Speed of a failed tracking pedestrian 59
5.6 Speed of a failed tracking vehicle 59
5.7 Speed of a failed tracking vehicle 60

List of Tables

3.1 List of instruments 23

4.1 Statistic information of the stationary map 30

Acknowledgments

First and foremost, I would like to thank my advisors Constantino Lagoa and Sean Brennan for helping me with support, guidance and priceless advise. I thank my committee member William E. Higgins for his insightful comments.

I would like to thank Pramod Vemulapalli and Wenqing Yao for their help on collecting data and inspiring discussion.

Thanks also go to the members of the LIDAR subgroup. The weekly meeting helps me a lot during the thesis work. I would like to specially acknowledge Kshitij Jerath and Emil Laftchiev.

I would like to thank Xin Lu and Guiyuan Han for their help on proof reading and useful suggestions.

Finally, I would like to thank my fiancee Wenjuan Qin for her endless support, encouragement and love.

Chapter 1

Introduction

1.1 Motivation

The problem of detecting and tracking multiple objects has been extensively studied for several decades; e.g., see [1]. Numerous radar-based tracking systems have been applied in military [2], ground-control systems in airport [3], meteorology [4], and other areas. Besides radar, many different types of sensors have been applied in tracking systems, for instance sonar, IR sensors, cameras, etc. In recent years, researchers have begun to study tracking systems based on laser rangefinders due to its portability and accuracy. Laser rangefinder tracking systems are now employed in many in-vehicle tracking systems for autonomous driving and collision alarms. In this thesis, a single stationary LIDAR-based multi-object tracking system with dynamic modeling technique is designed and evaluated.

1.2 Multi-object tracking system

The general data processing steps within a multi-object tracking system is summarized in Figure 1.1. During the detection step, the raw data obtained by the sensor is captured by the system, and the foreground data is partitioned from background data and separated into different segments. Once the foreground objects are detected and grouped, the problem of multi-object tracking becomes the problem of estimating the dynamic states of each object. Generally, this estimation consists of two parts: Data Association and Filtering [5]. Data association in the multi-object tracking problem seeks to match the observations from the sensor to corresponding existing tracked objects. Filtering is applied to improve the estimate of the state by combining recent observations with models of object behavior.

The selection of the motion model used in data filtering is one of the most crucial problems in multi-object tracking systems. This model is not only used for data filtering, but also for predicting the motion of any temporarily occluded targets. In real traffic scenes, although the

Figure 1.1. Summary of multi-object tracking system

object tends to keep the same motion for a short time interval, the motion model changes over time. For example, the model of a vehicle driving through a stop sign can be summarized as deceleration-stop-acceleration. Therefore, it is difficult for simplistic predefined motion models to approximate the dynamics of the moving objects accurately all the time.

Later chapters provide a more comprehensive discussion of the literature, but a preliminary review of methods illustrates the key challenges. In the most basic approach, some researchers assume that the objects can move in any direction with uniform probability, and thus they can apply Brownian motion models [6]. The drawback of this approach is that the predicted trajectory may spread out over a broad area because of the uniform distribution. In the approach called Interacting Multiple Models (IMM), different models are run in parallel and their outputs are then merged to estimate the states. Since the candidate models include different types of motion assumptions - constant velocity, constant acceleration, and turning for example - the IMM approach is general enough to cover different types of common motions. Nevertheless, IMM approach has several significant parameters to be set, e.g. the selection of the candidate models, and thus is not always practical.

To address this problem, this work uses dynamic models to describe the observed motion without any prior knowledge of object motions. The model is identified online by processing valid, recent observations and updated periodically. It is different than IMM because, instead of running candidate models in parallel, a model is identified online from previous measurements.

The results of this thesis show that the proposed model can provide good motion predictions, including the ability to adapt to changing motions. One crucial parameter in this approach, the order of the dynamic model, is determined by experiments.

1.3 Thesis Organization

The thesis is organized as follows: Chapter 2 briefly introduces the history of multi-object tracking systems and reviews the techniques used in those systems. Chapter 3 describes the instruments of the hardware system and the experiment protocol. The algorithms and methods applied in this thesis are presented in Chapter 4. In Chapter 5, the results and discussion are given to demonstrate the proposed methods. Chapter 6 concludes the thesis and provides ideas for future work.

Chapter 2

Related Work

The problems of Detecting and Tracking Moving Objects (DATMO) have become more and more crucial in advanced applications of vehicle automation, collision avoidance, and intelligent surveillance. The complicated motions and patterns of normal and abnormal traffic makes this a daunting problem. Especially under extremely crowded circumstances, for instance traffic during rush hours or situations where there are mixed pedestrians and vehicles, temporary occlusion and ambiguity caused by close targets are especially challenging for stable tracking. During recent decades, extensive studies on this subject have been conducted that have analyzed different sensors as well as introducing new and improving available algorithms [1, 5]. This chapter reviews the development of various approaches for DATMO problem.

2.1 Early days: Vision-based tracking systems

Before 2000, due to the relatively low spatial resolution and scanning speed of available distance sensors (sonar, radar and laser-based rangefinders), researchers tended to develop methods based on visual signals [8]. The visual information, including color and appearance, makes some detection and tracking applications easy to implement through the use of image flow. Several vision-based road following and lane detection systems have been developed [9, 10]. Although many of the proposed image-based tracking systems worked quite well in predicting the trajectories of vehicles [11, 12] or people [13], they were found to be hard to implement in practice. There is not sufficient resolution from image-based approaches for applications requiring accurate distance information, for instance the collision avoidance system and ego-motion estimation system [14]. Additionally, Lindstrom and Eklundh summarized other key limitations of camera-based tracking systems including: cameras have a field-of-view that is too narrow; the visual appearance of a feature is not robust to position or lighting changes; and the visual processing gives little explicit distance information . In order to address these limitations, radar and other distance sensors began to draw researchers'interest.

2.2 The late 1990s: Introduction of laser rangefinders

Introduced several decades ago (Bachman, 1979), laser based radar systems, also called laser rangefinders or LIDAR, combines the advantages of many traditional radar techniques. This sensor obtains the relative distance between the sensor and nearby objects by sending out the laser beam towards the target and calculating the time difference between sending and receiving. A key advantage of this approach over vision systems is that laser rangefinders are not sensitive to target illumination.

As an electro-optical sensor, LIDAR usually operates on a short wavelength which leads a narrow beam width. As a result, typical LIDAR systems have relatively fine spatial resolution. Because of these merits, many applications based on LIDAR measurement techniques have been developed since the 1990s, discussed below.

2.2.1 Historical development of laser-based tracking system

At the beginning, researchers tended to use the accurate distance information from the laser rangefinder to solve the robot motion estimation problem [16], and possibilities of applying this technique to tracking systems were considered [17]. Meanwhile, Meier and Ade's work and Sobottaka and Bunke's research proved the feasibility of tracking objects by distance information, although the former used coarse 3D information from the infrared sensor and the latter used range cameras [18, 19]. Earlier than that, in 1995, researchers from Robotics Institute of CMU began to use range sensors to perform obstacle detection [20]. Following that, laser intensity-based tracking systems were intensively studied by A. Hancock (1999) in his doctoral thesis [21].

Since the late 1990s, research on tracking objects based on laser range data has been rapidly gaining popularity. As shown in previous Figure 1.1 and discussed in Chapter 1, the problem of detecting and tracking multi-object generally consists of three parts: Detection, Filtering and Data Association [5]. Detection addresses the problem of extracting the foreground from the raw data. Filtering is applied to improve the estimate of the target state by combining the existing model and observation. Data association deals with matching the observations from the sensor to existing tracked objects. This "matching" process for moving objects can be extremely hard for crowded scenes. The following section will introduce the history of laser-based tracking systems based on the different algorithms they use.

2.2.2 Moving objects detection for laser-based tracking system

Because the segments obtained by detection part are used in the data association algorithm step for matching the existing tracks, the performance of detection is very significant to the whole tracking system. To improve this part, many algorithms for background subtraction, segmentation and classification have been discussed.

2.2.2.1 Background subtraction

For LIDAR sensors operating in cluttered environments, the raw data contains multiple sources of information including the noise, background, and of course the foreground moving objects. As seen in Figure 2.1, the foreground moving objects are marked by red rectangles and number the other dots represent the stationary background. Most tracking systems apply a "background filter" first before processing the data, and the goal of a background filter is to separate the foreground and background, and to remove noise.

Background subtraction techniques have been widely used for detecting moving objects from static cameras [23]. One approach, presented by Fod et,al., considers the different features provided by LIDAR within a background model based on range information [24]. In order to simultaneously detect moving targets and maintain the position of stationary objects, an occupancy grid map is employed to detect moving objects [25]. The grid-based map technique has been widely used in Simultaneous Localization and Mapping (SLAM) for representing complex outdoor environments (Figure 2.1) [7, 35, 59]. Due to the few features extracted from LIDAR data, by building motion-map and stationary-map separately, Wang showed this approach is more robust than a feature-based approach (Figure 2.1) [7].

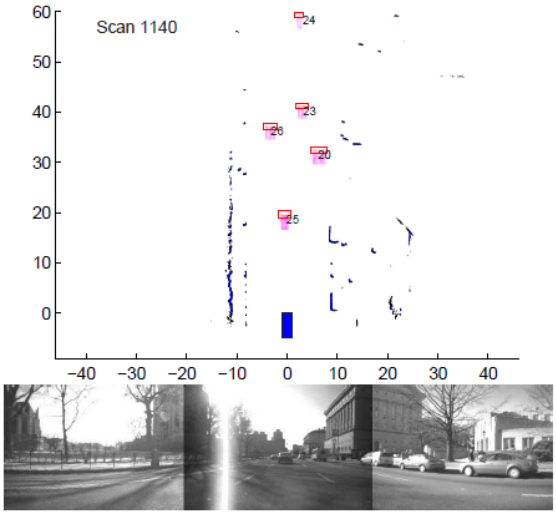


Figure 2.1. Moving objects detection by grid map (from [7])

2.2.2.2 Segmentation

The data segmentation process seeks to divide the collected data points into distinct segments such that points associated with the same object are grouped together. As a first criterion, a common method is to perform segmentation based on a distance threshold. Examples of this approach can be found in work done by Wang et al., Maclanchlan and Mertz and Vu et al. [28, 35, 66].

As the next step, consecutive points on the same object are sought. This is challenging because of the limited angular resolution of the LIDAR sensor in this study. Specifically, the distance between two consecutive scan points on the same surface can change dramatically depending on the varying position and angle of the surface relative to the sensor. For example, the side of a truck or vehicle perpendicular to the laser beam will have consecutive points that are closely spaced. Conversely, a truck that is nearly parallel to the sensor's scan direction may only have a few points impacting the side of the vehicle, and these points may be very widely spaced (Figure 2.2).

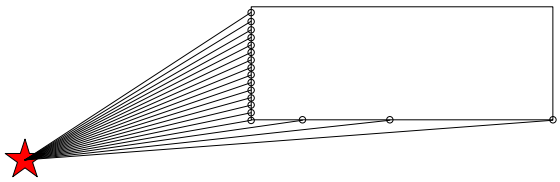


Figure 2.2. Example of angular resolution limition

A possible solution to this problem has been suggested by Sparbert et al. and Mendes et al. who used an adaptive distance threshold to perform segmentation [29, 30]. Their methods are based on the distance between data point to the LIDAR. In their method any two consecutive points r_k and r_{k+1} will be regarded as belonging to the same object if the distance between them $r_{k,k+1}$ fulfill the Equation 2.1:

$$r_{k,k+1} \leq C_0 + r_{min} \cdot \frac{tan\beta\sqrt{2 \cdot (1 - cos\phi)}}{cos(\frac{\phi}{2}) - sin(\frac{\phi}{2})} \tag{2.1}$$

where $r_{min} = min\{r_k, r_{k+1}\}$, ϕ is the angular resolution of the LIADR and β represents the maximum angle of the surface to x axis when the object could be distinguished as a unique one.

Because the algorithm mentioned above is based on the consecutive beams, it is very sensitive to the noise point. Therefore, in this thesis, a density-based spatial clustering algorithm is applied for data segmentation. This clustering algorithm is a well-known data mining method named "Density-Based Spatial Clustering of Applications with Noise" (Figure 2.3) [31]. DBSCAN is a clustering method based on the notion of "density reachability". A point, q, is defined as density-reachable from a point p, if the cardinality of set Q is greater than a given number N and for every element q_i in Q Equation 2.2 is satisfied.

$$d(q_i, p) < \varepsilon, q_i \in Q \tag{2.2}$$

where $d(q_i, p)$ is the distance between q_i and p and ε is a given threshold.

To avoid asymmetry of density-reachable points, density-connected is introduced. Here, two points p and q are density-connected if there is a point O such that both p and q are density-reachable from O. Given the distance eps and minimum number of points to form density-reachable N, the cluster could be defined by two constrains:

- All points within the cluster are density-connected;
- If a point is density-connected to any point of the cluster, it belongs to this cluster;

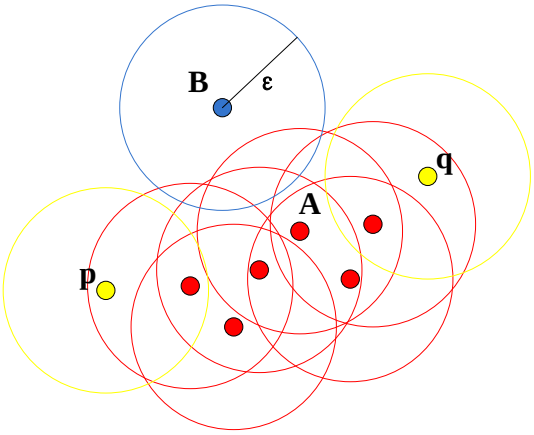


Figure 2.3. Examples for DBSCAN. ε is the radius of the circle and N is 2. Points p and q are density-connected and all the red and yellow points form a cluster. Point B is noise.

DBSCAN is useful because it can deal with arbitrary shaped clusters without knowing the number of the clusters, which is very important for the LIDAR data in this thesis. Because the cluster with numbers of elements less than N is considered as noise, this de-noise characteristic is also very useful (Figure 4.6). Since most physical objects, including vehicles and people, are well defined by lines when viewed in profile, the minimum number of points required to form a cluster is set to 2. The distance is empirically chosen as 120cm.

2.2.2.3 Occlusion

Occlusion also presents a problem in segmentation, where the shadow of one object may partially block the view of a second object, causing segmentation to try to classify the single second object as several different objects moving in unison (Figure 2.4). To avoid dividing one vehicle or object into several segments led by partially occluded and black surfaces, Petrovskaya and Thrun, Aycard and Ogawa et al. implemented a geometric vehicle model that requires continuity between disjoint point segments [47, 52, 81]. Other researchers have solved this problem by performing image processing approaches before clustering step. In these approaches, the LIDAR image is processed as a 2D birds-eye-view image, and afterwards image-processing methods are applied. For example, Zhao and Thorpe used Hough transformation to extract the lines [22], and Burke applied a median filter following principal component analysis [40].

2.2.2.4 Classification

A traffic scene can be quite complicated because it contains different mixtures of vehicles, pedestrian, pets and backgrounds. Usually, the motion dynamics are significantly different among the

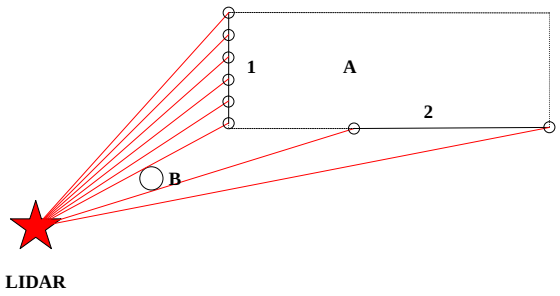


Figure 2.4. Example of partial occlusion. Vehicle A is occluded by pedestrian B, so it is divided into two parts: 1 and 2 in the LIDAR data.

classes of objects, but not always. For example, the velocity of a pedestrian normally is 5 mph and the velocity of a vehicle is normally much higher than this. But there are speeds, particularly near construction zones, intersections, and stop signs, where the speeds of all objects may be similar.

More suitable classification methods have been investigated in surveillance and traffic flow analysis literature. Specifically, object classification in laser-based multi-object tracking system has been studied extensively [29, 30, 34, 36]. Mendes, et al. uses a voting scheme presented by [37]. By considering all the hypotheses over time, an object is assigned with a class until the confidence level reaches a reasonable value [30]. Although the features used are not discussed in detail, the results showed that voting classification approach can assign the right class after several frames (Figure 2.5).

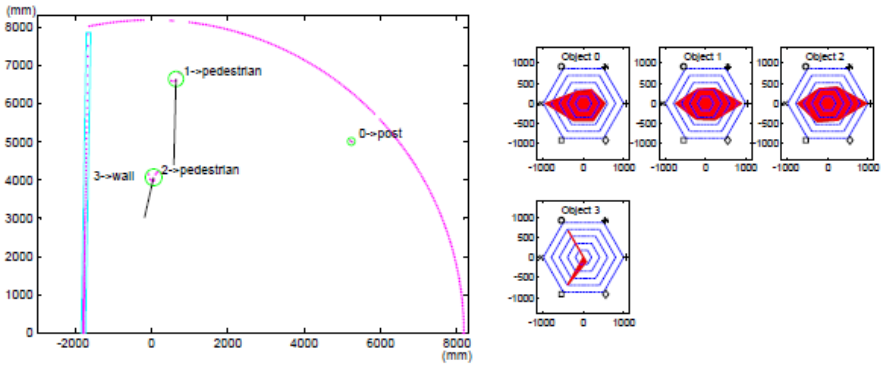


Figure 2.5. Voting scheme based classification, where:+ → pedestrian; ∗ → car; ○ → truck; × → post; □ → wall; ◇ → bush [30]

Also based on the voting scheme, Nashashibi applied the following rules to vote [36]:

- A vehicle has minimum width of 1 meter;
- A L-shaped object is considered a vehicle;
- A partially occluded segment is considered as a vehicle;

- Motorcycle is a small non-occluded object with width less than 1 meter;
- Certain number of regularly aligned and non occluded obstacles, in other words, comes out repetitively, are considered to be environment elements;

Figure 2.6 presents this rule-based classification method, and the results show that this approach is fairly successful.

Figure 2.6. Rule based classification [36]

2.2.3 Filtering methods for laser-based tracking system

In the multi-object tracking literature, filtering is necessary to smooth the trajectory and to predict the vehicles pose state when the observation cannot be obtained directly. To perform this filtering, many Bayesian based filters, for instance the Kalman Filter, the Extended Kalman Filter, the particle filter and the IMM algorithm, are widely used.

2.2.3.1 Kalman Filter

The Kalman Filter is an optimal recursive filter designed to estimate the state of a dynamic system from multiple sequential measurements [82]. The Kalman Filter is generally applied to linear system models that have Gaussian noise both in the state propagation (dynamics) as well as the measurement model. The state is estimated by the following general equations recursively. The Kalman Filter assumes a dynamic system model of the form:

$$x_k = Ax_{k-1} + Bu_k + w_k \tag{2.3}$$

$$y_k = Cx_k + v_k \tag{2.4}$$

where x_k denotes a state of system at discrete time k, A a state matrix, u_k an input, y_k a measurement, $w_k \sim N(0, Q)$ and $v_k \sim N(0, R)$ are the zero mean Gaussian white noise with covariance Q and R. The states are recursively updated as:

$$\hat{x}'_k = A\hat{x}_{k-1} + Bu_k \tag{2.5}$$

$$\hat{P}'_k = AP_{k-1}A^T + Q \tag{2.6}$$

$$K_k = \hat{P}'_k C^T [C\hat{P}'_k C^T + R]^{-1} \tag{2.7}$$

$$\hat{x}_k = \hat{x}'_k + K_k[y_k - Cx_k{}'] \tag{2.8}$$

$$P_k = [I - K_k C]P_k{}' \tag{2.9}$$

where K_k is the Kalman Gain and P_k is the covariance of the a-posteriori estimate error.

Given a role as a classic filtering method in signal processing, the Kalman Filter has been widely applied in the laser-based tracking systems [22, 24, 26, 27, 28, 33, 34, 35, 38, 39, 40, 41]. Among them, the initial and most widely cited research is briefly presented below.

In 1998, Zhao and Thorpe introduced a real-time car tracking system that is based on a laser range-finder [22]. Based on the framework of Interactive Multiple Model (IMM) (Section 2.2.5.3), with three significant motion models and extended Kalman Filter, this system was evaluated using real range data from the laser scanner mounted on Navlab5 traveling on Highway. Results showed it could provide accurate motion estimates, motion classification and reliable maneuver detection. In a different approach, Wang, et al. tried to resolve the simultaneous localization and mapping (SLAM) problem as well as the detection and tracking moving objects (DATMO) problem at the same time [27]. The advantage of this approach is that the map from SLAM can help to detect moving objects, and in turn, the map will be more reliable after removing the moving objects detected and tracked by DATMO. See Figure 2.7 and Figure 2.8 for example. After combining several algorithms (IMM, extended Kalman Filter) demonstrated by Zhao and Thorpe, Wang, et al. improved the SLAM with DATMO system and tested it using 100 miles real range datarecorder from a moving vehicle [28]. The results showed the effectiveness of this approach (Figure 2.9). A Kalman Filter based on a linear model has also been applied for people tracking [24]. By searching the small neighborhood region of every "blob" (object) and adding small errors in velocities, this approach could track several moving people in a small room (Figure 2.10). This approach of combining IMM and EKF was also discussed by Keampchen, et al. who included a "Stop and Go" model in their IMM approach [42].

2.2.3.2 Particle Filter

Instead of Kalman Filter, several multi-object tracking systems apply a particle filter or similar Monte Carlo algorithms because they can deal with nonlinear and non-Gaussian models [43]. Particle filter estimates the posterior over unobservable state from sensor measurements [44]. Because the location of pedestrians cannot be easily characterized by a general dynamic

Figure 2.7. SLAM without DATMO [27]

Figure 2.8. SLAM with DATMO [27]

model, a particle filter is first used for pedestrian tracking [45]. Their results suggest that a sample-based particle filter performs better than a Kalman filter (Figure 2.11) in this particular scene. If the environment is fixed and the map is previously obtained, by breaking the high dimensional particles into two sets of lower dimensional particles and one conditionally depended upon the other, the particle filter showed the ability to deal with simultaneous robot localization and people-tracking problem [6] (Figure 2.12). Frank, et al. applied sequential Monte Carlo approach on multi-object tracking and tested it off-line by real laser range data [46]. Although the particle filter outperforms Kalman Filter in situations dealing with unpredictable motion tracking problem, it has not been extensively applied in laser-based tracking system until recent

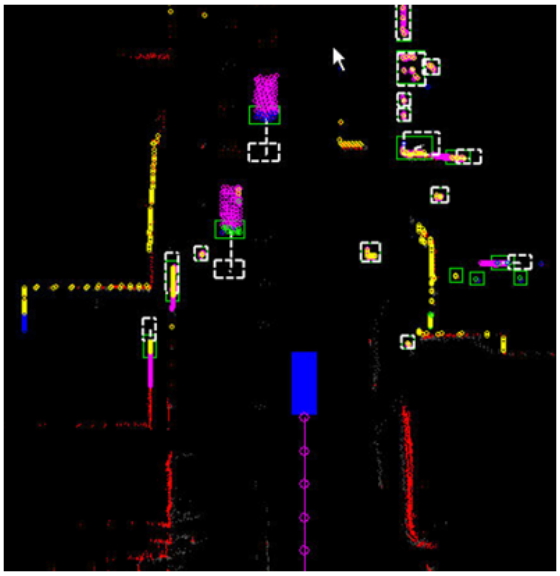


Figure 2.9. SLAM and DATMO with advanced algorithm [28]

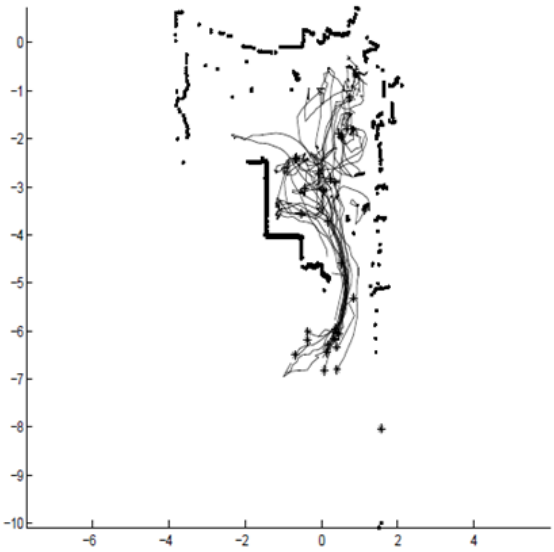


Figure 2.10. Trajectory of people obtained by tracking with Kalman Filter [24]

years because of the high computational complexity [28]. Recent example include Petrovskaya and Thrun 's work where they applied a Rao-Blackwellized particle filter (RBPF) for their robot, Junior, who won the second place in the 2007 DARPA Urban Grand Challenge [47]. RBPF was first introduced by Doucet, et al. in 2000 [48] and has been applied in DATMO problems using both simulated data and real data [49, 50, 51].

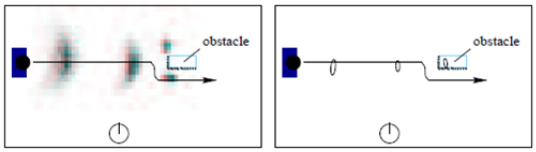


Figure 2.11. Comparison of Kalman Filter (right) and Particle Filter (left). The arrow indicates the trajectory of the object. The Kalman Filter predicts the object into the obstacle. [45]

2.2.4 Data association approaches for laser-based tracking system

Data association seeks to match data points to a specific object, and it is a significant step for the DATMO problem resolution. Because of the limited features provided by the distance sensor, accurate data association is very difficult, especially for crowded scenes. As the most intuitive approach to assign the nearest segment to the object, a Greedy Nearest Neighbor (GNN) filter was widely applied in the early era of tracking [5]. This method is simple to implement and computationally simple, and as a result many tracking systems still use it as the main data association approach [35, 38, 52]. Because the GNN is a simple implementation with acceptable error, this thesis also applies the GNN filter to do the data association.

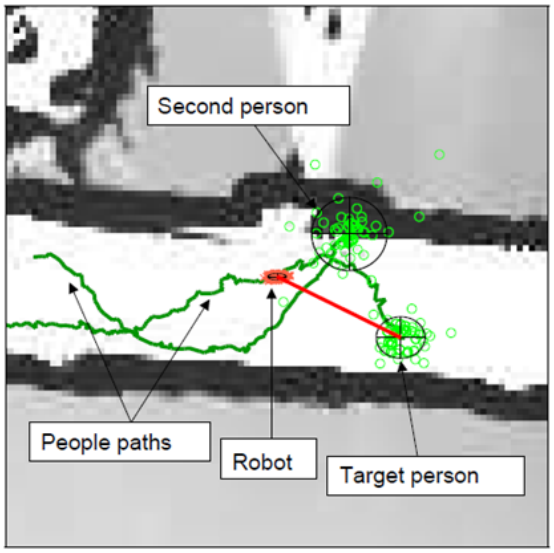


Figure 2.12. Simultaneous robot localization and people-tracking [6]

For the completeness of the literature review, two more advanced well-known Bayesian approaches are introduced here: 1) multiple hypothesis tracking (MHT) algorithm and 2) joint probabilistic data association (JPDA). Though they were developed several decades ago, their application has grown extensively in the last ten years as the result of increasing computational power in readily-available processors [53, 54]. Recently, a batch of multi-object tracking systems have achieved notable success by applying Markov chain Monte Carlo data association (MCMCDA) [55, 56, 57]. Unlike other data-association methods that seek to maintain or test all possible data assignments, MCMCDA uses Markov chain Monte Carlo sampling [58]. Vu, et al, applied this data association for laser-based tracking systems and obtained positive results [59] (Figure 2.13).

Figure 2.13. Tracking results for laser-based tracking systems using MCMCDA [59]

2.2.4.1 Greedy Nearest Neighbor (GNN) Filter

The simplest and probably the most widely applied data association approach is the Greedy Nearest Neighbor Filter.The NNF only takes into account the prediction of the existing track from the last frame and the new observation obtained from the sensor. For each new data set, every segment is assigned to the nearest previous track after predicting motions for previous tracks from previous measurements to the current measurement. Prassler, et al. introduced a people tracking system for a crowded scene (railway station during rush hour) [61]. Considering that a person's location cannot be easily characterized by a general dynamic model, the algorithm had to rely almost entirely on the greedy nearest neighbor filter to track people for consecutive range images. The result indicate that this system could track 5 to 30 moving objects in real-time (Figure 2.14).

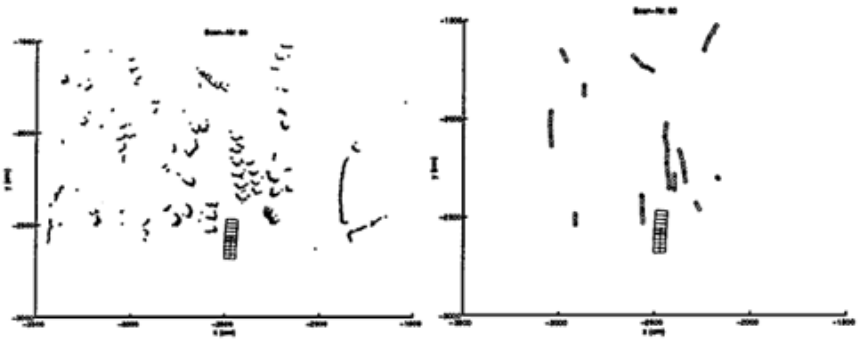


Figure 2.14. People tracking results in a railway station [61]

Most laser-based vehicle tracking systems usually apply the Mahalanobis distance as the proximity metric for GNN. The Mahalanobis Distance was introduced by P.C.Mahalanobis in 1936.This measurement is used instead of Euclidean distance because it considers the geometric information implicit in: that they primarily move forward and not side-to-side.

For a point x and a group of points Y, the Mahalanobis distance between them is defined as

$$D_M(x,Y) = \sqrt{(x-\mu_Y)^T S_Y^{-1}(x-\mu_Y)} \tag{2.10}$$

where x is a n dimensional vector $(x_1, x_2, ..., x_n)^T$ and Y is a set of n dimensional vectors, $(Y_1, Y_2, ..., Y_n)^T, Y_n = (y_{n,1}, y_{n,2}, ..., y_{n,m})$. $\mu_Y \in n \times 1, S \in n \times n$ are the mean value vector and covariance matrix of Y. The value $D_M(x,Y)$ is called the Mahalanobis distance between x and Y.

For the 2-dimensional data, since the covariance matrix represents the axis of the ellipse covering the distribution of the data, it can represent the direction and size of the entire data cluster. For points with the same mean value of the Euclidean distances to every point in the data cluster, the ones near the long axis direction have a small Mahalanobis distance versus the ones near the short axis direction. This property is very useful for finding the corresponding feature points of a vehicle since the vehicle always tends to move towards the long axis direction.

The difference between the two metrics is illustrated by the following example.

In Figure 2.15, the circle points are a measurement of a vehicle in the last frame, and the star points represent two probable locations of the vehicle in the current frame. If the Euclidean distance is used to measure the distance, the Euclidean distance between the probably positions and the data set are $D_e(A) = 252.5$ and $D_e(B) = 361.4$. However, as a vehicle, point B should have a higher probability of being the next cluster position than point A since it´ s nearly impossible for a vehicle to suddenly move laterally. If the Mahalanobis distance is applied, then the distances become $D_m(A) = 77.9$ and $D_m(B) = 6.6$, which illustrates that the association along the long axis of a data cluster tends to generate trajectories consistent with expected vehicle motion. Similarly, if the trajectory of a cluster is known, the Mahalanobis distance metric can use weightings aligned with the expected trajectory. Thus, the Mahalanobis distance better associates new measurements to a vehicle projected along its probable path, not to vehicles that happen to be close to the new measurement but whose paths are not nearby the measurement [38].

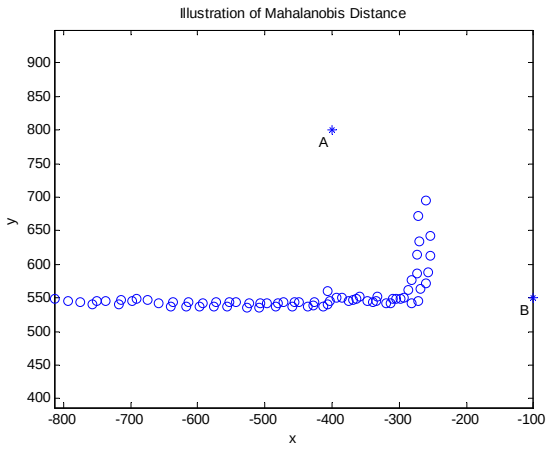


Figure 2.15. Example of Mahalanobis Metric

In this thesis, for easiness implementation with acceptable error, GNN filter is applied as data association approach. The advanced data association algorithms are reviewed following.

2.2.4.2 Joint probabilistic data association (JPDA)

To eliminate association ambiguity in complex scenes, especially for multi-objective tracking, JPDA is a data association algorithm that takes into account every possible association. It computes the Bayesian estimate of the correspondence between segments detected by sensor and possible existing tracks and forms a hypothesis matrix including all possible associations. The assignments with highest probability are picked out.

As an example of JPDA, Schulz applied sample-based JPDA in laser-based tracking system at first and showed its effectiveness for the multiple people tracking problem (Figure 2.16) [45, 62]. By modifying JPDA to separate highly-correlated target-path combinations from poorly-

correlated combinations, Frank, et al. proposed two extended JPDA approaches and tested them off-line (Figure 2.17) [46]. The robot in DARPA 2007 challenge, Junior, mentioned earlier in Section 2.2.2.2, also used this JPDA approach.

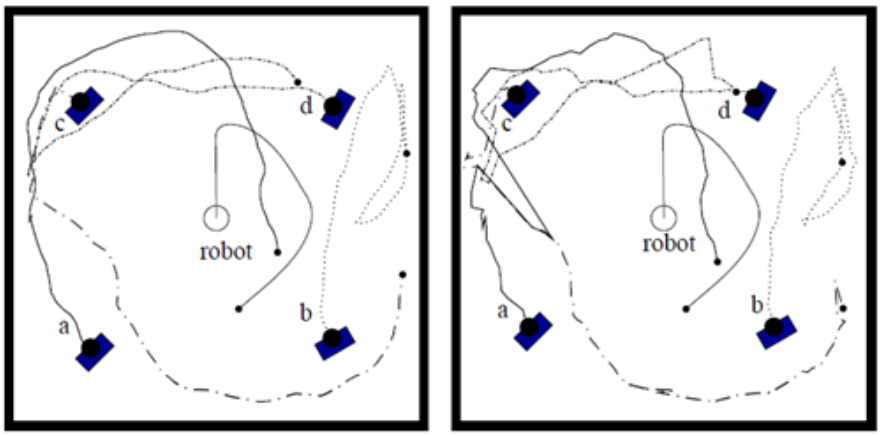


Figure 2.16. Trajectory of tracking four people by sample-based JPDA: the left scene is the true, the right is the estimated trajectory [62]

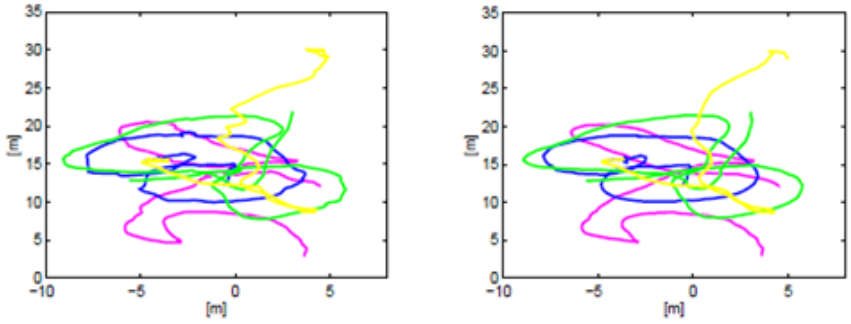


Figure 2.17. Results of racking four people: the left one is based on correlated samples and the right one is based on independent samples [46]

2.2.4.3 Multiple Hypothesis Tracking (MHT)

Instead of considering just the information of one frame, Multiple Hypothesis Tracking (MHT) is a multi-scan deferred decision logic tracking algorithm that keeps multiple alternative data association hypotheses whenever the ambiguity situation occurs. Rather than combining these hypotheses in JPDA method, all the hypotheses are propagated and the one with highest posteriors is returned as solution. Since all potential tracks are maintained and updated, this method is very useful when the target motion model is unpredictable. Although the number of hypothesis may grow exponentially over time, MHT is still applied in many multi-object tracking systems, supplementary with hypothesis deleting mechanism [26, 28, 63]. Wang, et al. also implemented this method on Navlab for real-time road tests (see the previous Figure 2.9).

2.2.5 Approaches for occlusion handling

When the targets are visible or partially occluded, nearly all the systems mentioned previously are able to track them and obtain reasonable trajectories. However, when the environment is crowded, for instance in the rush hour traffic and transport hub stations, the targets may occlude each other, especially when some of them move close to the sensor. The temporary occlusion of the objects may lead to mismatch when they return to view. To maintain estimates of the tracks during occlusion, it is critical to have an estimate of the motion model of the objects during occlusion [64].

In the early era of the laser-based tracking systems, some systems failed to keep tracking when the target was temporary occluded [61, 65]. Figure 2.18 shows a situation where the track is lost when the target is occluded.

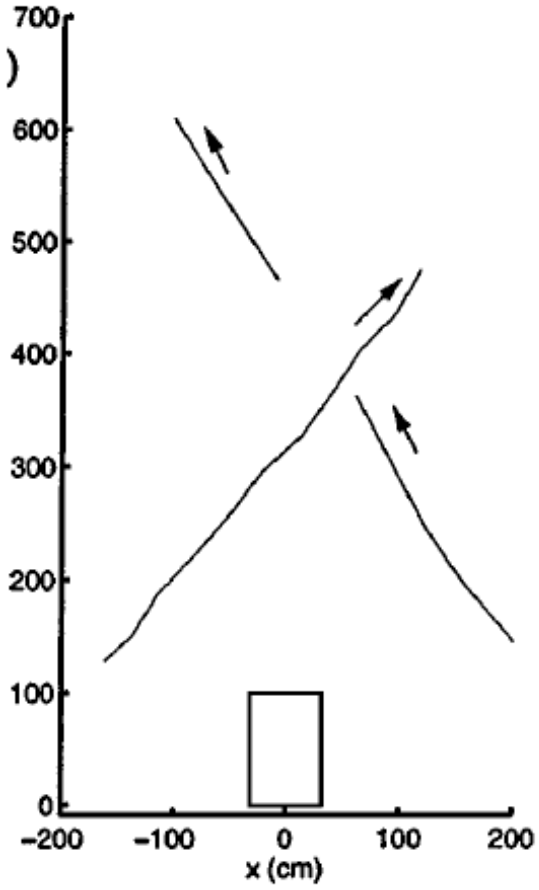


Figure 2.18. Occlusion of an object may cause temporary loss-of-tracking of the occluded moving object [61]

To some extent, this occlusion problem can be resolved using some of the previously-mentioned filtering methods necessary in laser-based tracking systems. In literature, both single motion model and multiple models have been applied. As an intuitive approach and possibly the most

widely used method, one can predict motion during occlusion using the prediction steps of a Kalman Filter along with the same motion model used in the "prediction/correction" steps of the Kalman filtering process. This conditional updating process allows one to predict an object's position during occlusion [24, 26, 33, 38, 40]. Among this prior work, different motion models are seen to demonstrate different performances. For convenience and easiness of implementation, a constant velocity model is nearly the most common model for vehicle tracking. Since the driver tends to keep the same speed for a short time, this model can readily solve the occlusion problem. Unfortunately, the real traffic always includes different motion types for vehicles, for example accelerated motion, turning and stopping. Not to mention the wondering people, whose motion is nearly impossible to predict [45]. For this reason, the Interactive Multiple Model (IMM) method has been applied broadly, which runs several Kalman Filters with different models parallel and merge the outputs to predict the positions [22, 28, 42, 52]. On the other hand, unlike the explicit motion model of Kalman Filter, the probability based model in a particle filter can deal with complex dynamic motion [43].

2.2.5.1 Explicit model with Kalman Filter

As reviewed in filtering section, Kalman Filter is widely applied in multi-object tracking literature. Therefore, many researchers used the same motion model to propagate the states during occlusion. Fod, et al. used a linear model to estimate the motion recursively [24]. After data association, the new state of an object is the linear combination of the predicted state vectors from the matched object of last frame. The weighting is calculated by minimizing the sum-of-squares error matric. The result in previous Figure 2.10 shows it is an effective approach for multiple target tracking.

As the less complicated model, the constant velocity model is extensively used to describe the state of the targets with acceptable error. Streller, et al. used position, velocity, orientation and rotational speed to describe the target state[26], while Burke and Fayad with Cherfaoui only considered the position and velocity [38, 40]. The motion model used by Streller, et al. is presented as Equation 2.11:

$$X_{k+1,n} = \begin{bmatrix} 1 & T & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} X_{k,n} \tag{2.11}$$

$$X_{k,n} = (x_{k,n}, v_{x_k,n}, y_{k,n}, v_{y_k,n}, \varphi_{k,n}, \omega_{k,n})^T$$

where $X_{k,n}$ represents the state of object n at discrete time k, $(x_{k,n}, y_{k,n})$ the position, $v_{x_k,n}, v_{y_k,n}$ the velocity, $\varphi_{k,n}$ the orientation of the object $\omega_{k,n}$ the rotational speed and T the sampling time.

This model could predict the state of the objects with acceptable error, and deal with some

occlusion situations. In Figure 2.19, object 10 is temporary occluded by object 4 and reacquired by the tracking system later.

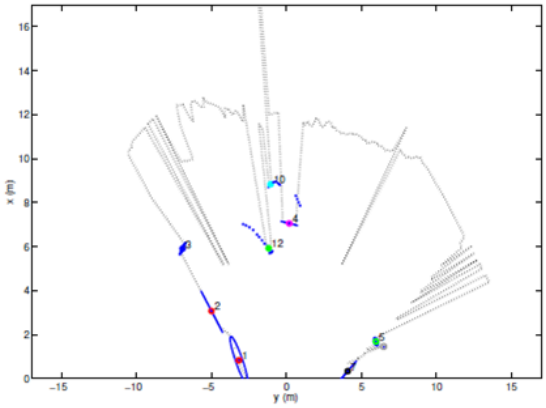


Figure 2.19. Tracking by constant velocity model. Object 10 is tracked through occlusion, where blue dots represent the trajectories of the moving objects, blue lines stationary obstacle.[40]

Without any explanation, Maclachlan applied a single dynamic model with linear acceleration and constant turn rate [66]. Although the occlusion situations were not discussed in detail, this system has been tested in a prototype collision warning system on two transit buses during 7000 km of regular passenger service.

2.2.5.2 Probability model with particle filter

Besides those explicit models, multi-object tracking systems with particle filter tend to apply Brownian motion model [6]. Since this model even does not attempt to model the dynamics but assume the target could move to any direction with uniform probability, it could be applied for targets without a consistent motion behavior, for instance people. It works fairly well for short duration of occlusion (Figure 2.12). However, the prediction could become equally spread out over a broad area as the occlusion time increases, because all of the motion is represented as dispersion. Considering that humans do not move randomly, some more sophisticated motion models have been proposed [67, 68, 69, 70]. Assuming that people tend to follow an "efficient path" rather than a random one, Bruce and Gordon added common destinations in the environment and then used a path planner to predict the probably routes between the current location of human to those destinations [70]. Figure 2.20 shows pedestrian passing an obstacle. The Brownian model (left) tends to spread out after a while, but the plan-based model (right) provides the more concentrated distribution and gives the right prediction (green dots).The result showed a better performance than Brownian model.

2.2.5.3 Interacting Multiple Models (IMM)

The IMM algorithm and its variants [71, 72] have been successfully implemented in many tracking applications for overcoming the motion modeling problem by using more than one model [74,

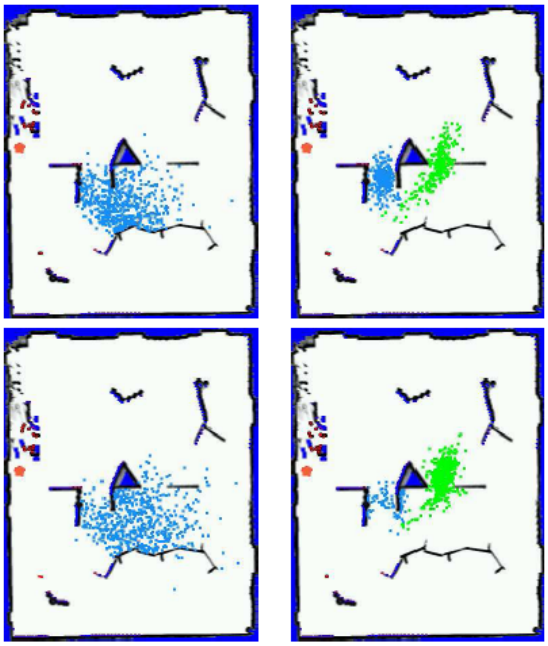


Figure 2.20. Tracking people during occlusion using Brownian model (left) and plan-based model (right) [70].

73, 1]. The principle is to run several elemental filters with different possible motion models in parallel. The final estimation of the state is obtained by merging the outputs of all filters based on the distribution probability over the set of motion models. Zhao and Thorp applied the IMM with three models: constant velocity, constant acceleration and turning model [56]. This approach was able to predict the position of the object accurately enough so that it could be re-acquired later (Figure 2.21). In some situations, however, the object may stop in its trajectory. Considering this situation, one can add "stationary motion" model to IMM algorithm. This approach was discussed as well [42, 76]. Although it could smooth the tracks of the target (Figure 2.22), Coraluppi and Carthel claimed that this method tends to degrade the performance for targets that may take the same path without temporary stop behavior [76]. Wang improved this approach by adding moving-stop hypothesis tracking [7].

In Wang's work, the probability distribution of different motion models is represented by a Transition Probability Matrix (TPM) In practice, these critical parameters have to be defined. At the beginning of data capture, these parameters are predefined without relating to the real data [77, 78], and so unrealistic results may be obtained. To overcome this, several on-line adaptive TPM estimation methods have been proposed [79, 80, 81] and these techniques show improvements of this algorithm (Figure 2.16).

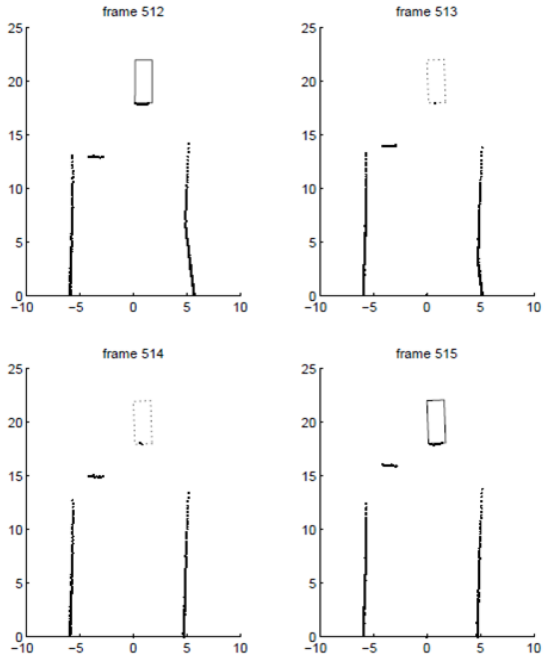


Figure 2.21. Car tracking over missing data by IMM approach [22]

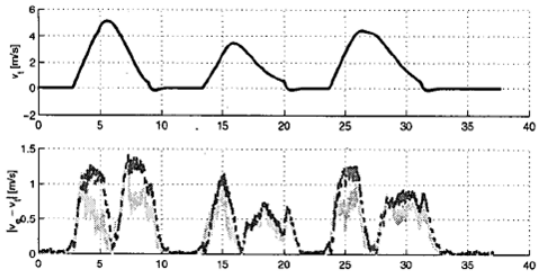


Figure 2.22. Velocity of the temporary stop target. Solid line represents the IMM with stationary model and dash line the CV model. [42]

2.2.5.4 Dynamic Model

As shown above, a single explicit model is widely used since it is easy to implement. However, since the dynamic of the objects cannot be represented by any specific model, for instance a constant velocity model or a constant acceleration model, it may lose the track when the target is occluded for a relatively long period. Although Brownian model is robust to all kinds of motion, the random characteristic led to a wide spread in the prediction. The interacting multiple models (IMM) algorithm combines the merits of both kinds of models above. This method runs a set of candidate models in parallel and merges the outputs of all filters to obtain a relatively accurate estimation. The drawback of this approach is that the choice of critical parameters used by the IMM algorithm is still an open question and the temporary stationary motion is hard to address by IMM. Based on above descriptions, a general time series dynamic model is applied in this

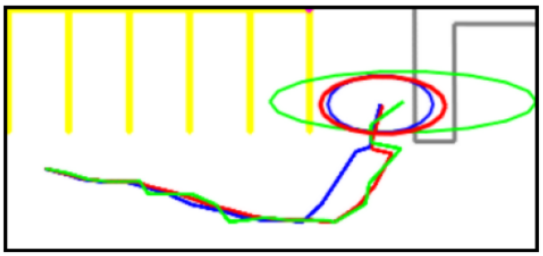
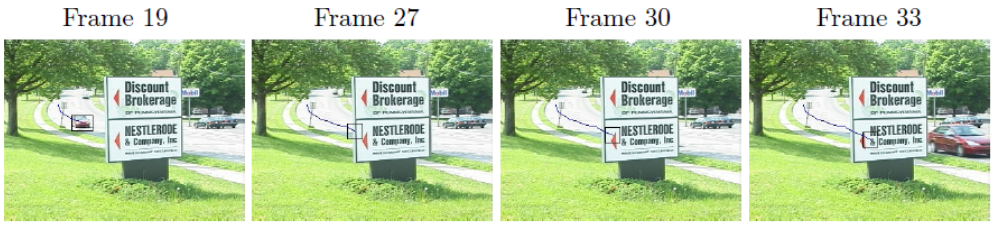


Figure 2.23. Trajectory of people tracked by IMM with adaptive TPM. Green line represents the ground truth, red line the trajectory with adaptive TPM and blue line the trajectory without TPM. [81]

thesis to predict the state of targets during occlusion.

Dynamic modeling has been theoretically studied and applied in dynamic computer vision and image system [83, 84]. Because it provides more dynamics than any predefined simple model, e.g. constant velocity model or constant acceleration model, Lim applied Caratheodory-Fejer interpolation to identify the dynamics of the model [83]. Figure 2.24 shows that dynamic model overcomes the constant velocity model.



(a) Kalman filter with constant velocity dynamics.

(b) Kalman filter with CF-identified dynamics.

Figure 2.24. Example of dynamic modeling for robust visual tracking [83]

Chapter 3

System Setup

3.1 LIDAR system

To test and evaluate the tracking system, a laser measurement system is constructed with the instruments described in Table 3.1. The whole system is shown in Figure 3.1.

Table 3.1. List of instruments

Instruments	Major Properties	
SICK LMS 291 LIDAR	Range	up to 80m
	Angular Resolution	0.25°/0.5°/1°
	Frequency	$18Hz/37.5Hz/75Hz$
	Measurement Resolution	10mm
	Scanning Angle	180°
DeviceMaster D139M	4-port device server	
Csi/SPECO Regulated Power Supply	115VAC input, 24DC 2Amp output	
410W Power Inverter	Invert battery energy into AC power	
Battery	Battery Warehouse High Performance Marine Deep Cycle Battery	
Panasonic Toughbook	Save the LIDAR data	

In the experiments, the LIDAR is working on 0.5° angular resolution and $37.5Hz$ frequency and supplied by 24VDC. The power of the whole system is provided by two deep-cycle lead-acid batteries. The batteries have enough capacity that the LIDAR system can work more than 6 hours without any external power supply. This DC voltage is converted to AC power using a 410W Power Inverter. The Panasonic Toughbook uses the AC power directly and the LIDAR and device server uses the DC power converted by Csi/SPECO Regulated Power Supply. The LIDAR measurements are transmitted via a high-speed RS422 serial cable on the device server and sent to computer via a TCP/IP connection.

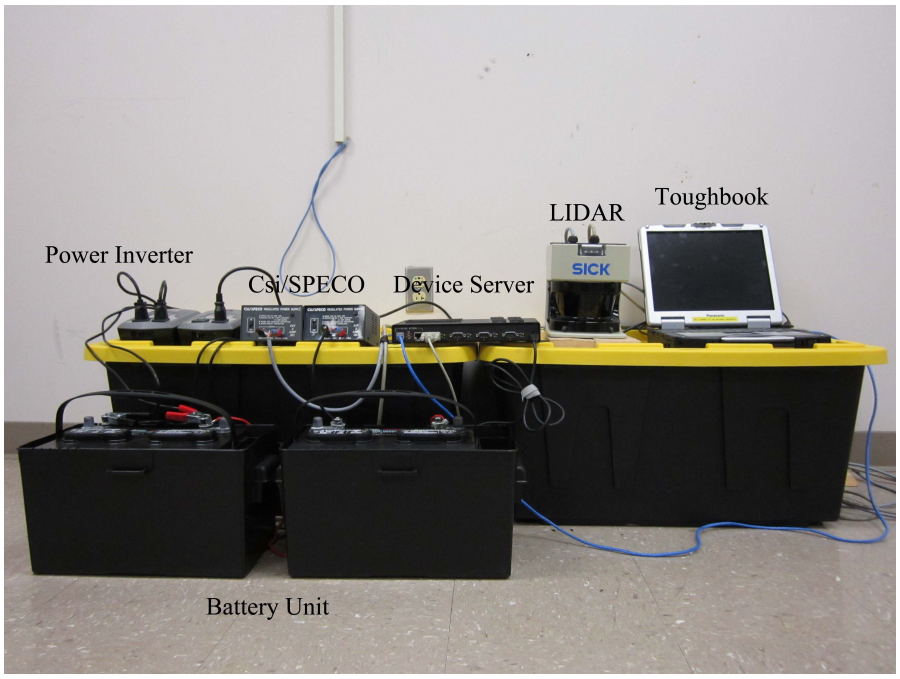


Figure 3.1. LIDAR system

3.2 Experimental Protocol

The data is collected at two locations in Penn State University Park. To cover most urban traffic scenes, a position on Park Avenue and the intersection between Curtin Road and Bigler Road are chosen.

3.2.1 Park Avenue

As a road with 35 mph speed limit, Park Avenue can represent most normal urban roads. The experiment was taken in February 9th between 11am to 1pm. Since the vehicles tended to drive with a constant speed and only a few pedestrians were present around this location, this data set could be used as a relatively simple test to evaluate the effectiveness of the system. Figure 3.2 shows the location on Google Map and Figure 3.3 is the picture of the location.

3.2.2 Intersection between Curtin Road and Bigler Road

Vehicles in real urban traffic scenes may have a complex dynamic motion model besides constant velocity, for instance, the turning and move-stop-move model. This intersection was chosen for data collection because it has a stop sign on every direction and many pedestrians involved in this area. The data was collected during peak time (February 9th, 5pm-6pm). This data set is used to test the robustness of the system to deal with complex dynamic motion models and tracking pedestrians and vehicles at the same time. The location is shown in Figure 3.4 and

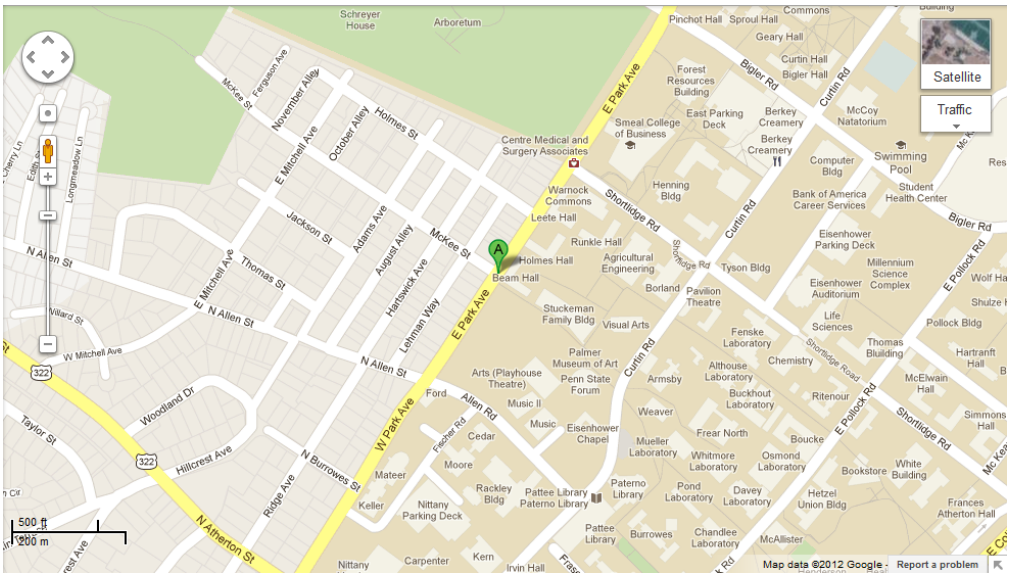


Figure 3.2. The experiment location on Park Avenue

Figure 3.3. Picture of the experiment location on Park Avenue

Figure 3.5 shows a picture of this location.

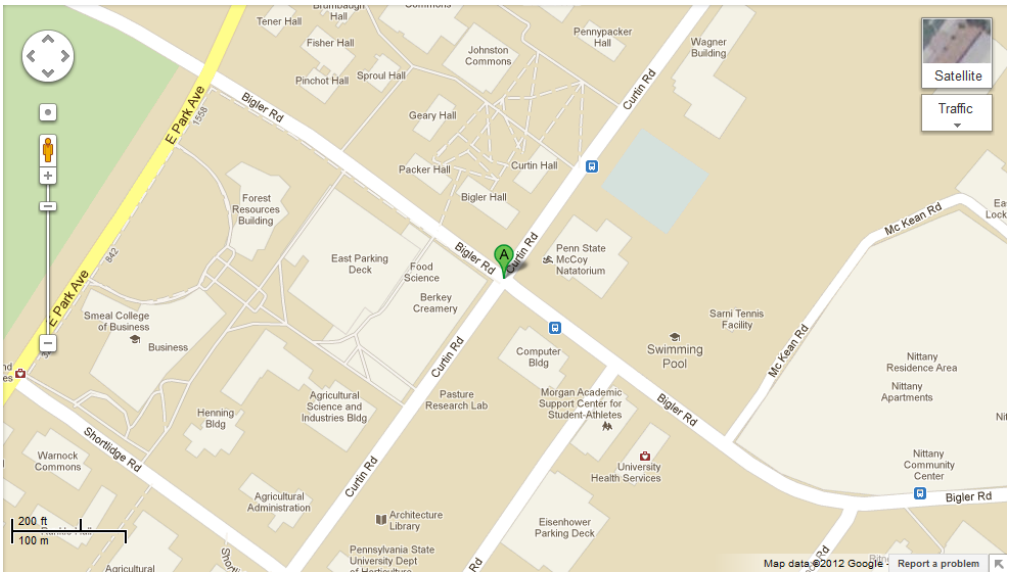


Figure 3.4. The experiment location of the intersection

Figure 3.5. Picture of the experiment location of the intersection

Chapter 4

Methodology

This chapter presents the methodology used for feature tracking in this thesis. A detailed overview of each step is described in the sections that follow, but an overview of the entire process, shown in the flow chart of Figure 4.1, is helpful to explain how each step relates to each other.

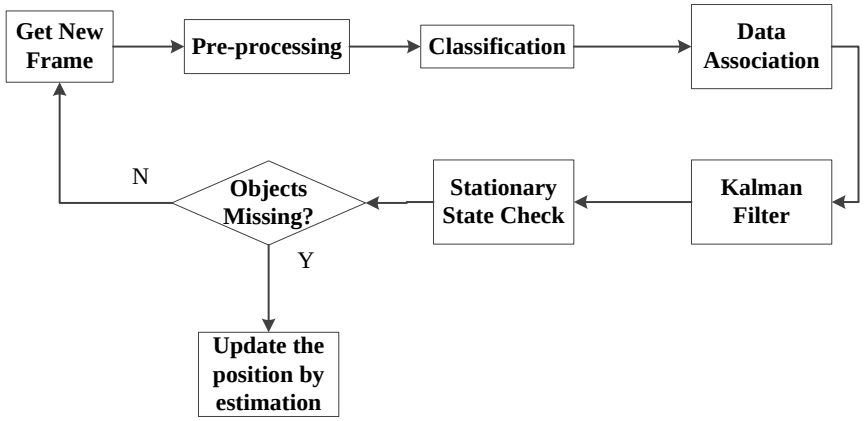


Figure 4.1. Flow Chart of entire process

The process begins when a frame of data is collected from a LIDAR system, with a "frame" representing one LIDAR sweep. Each frame of the LIDAR data is pre-processed to remove noise and to extract foreground information from background information. Next, the data is separated into different segments and each segment is classified and labeled as either belonging to a vehicle or a pedestrian. During the data association step, the system tries to match an object to the nearest segment with the same label, which constrains vehicles classified in the current frame to only match vehicle segments seen in previous frames.

If a new segment cannot be associated to a previously-existing object, it is stored for a short period. Such stored segments are marked as missing and deleted if no prior or subsequent matches

are found for several frames. If a segment is not found to match a new object, and the object persists for a specific time interval, it is marked as a new object and added to the object list for future tracking. Once the association is completed, each object's position is predicted and correlated with incoming data using Kalman filters. Objects remaining within a small boundary for a long time interval are marked as stationary. If any object is marked as missing during the association step, its new position will be estimated by the dynamic model. Finally, the system obtains a new data frame and repeats the processing loop again.

4.1 Pre-processing

As discussed in Chapter 2, background subtraction is very useful to separate the foreground and background, and to remove noise. In autonomous vehicle literature, to simultaneously detect moving targets and maintain the position of stationary objects, an occupancy grid map [25] is employed to detect moving objects. A grid-based map technique has been widely used in Simultaneous Localization and Mapping (SLAM) for representing complex outdoor environments [7, 35]. In this thesis, an intuitive approach is applied to build the grid map since the LIDAR is fixed relative to the map, e.g. the LIDAR is only used for moving objects detection.

At the beginning, all the points with maximum distance from the LIDAR, e.g. those at the limits of the LIDAR range, are regarded as open space and deleted (Figure 4.2). The pseudo-code is:

onlimitdataDel(D)
for each beam j of data set D
$L(j)$ = the length of the ith beam
    if $L(j) == max(D)$
        $L(j) = NAN$
return D

To form the occupancy map, the field of the LIDAR view is separated into $40cm \times 40cm$ grids, which is empirically selected. At any time frame, if a grid is occupied by the segments detected by sensor, its corresponding value will be increased by 1 and if no segment detected in this gird, the value will be decreased by 1. After a reasonable time interval, the grids representing stationary obstacles will have a relatively high value and any segments in these grids will be regarded as stationary objects. The stationary map is formed by all the grids with high enough value. In the experiment without temporary static objects, the map tends to be stable after 80 to 100 frames. The pseudo-code implementation of this algorithm is given below:

mapUpdate($oldMap, D$)
$addMap$ = same size to $oldMap$ with all grids are 1

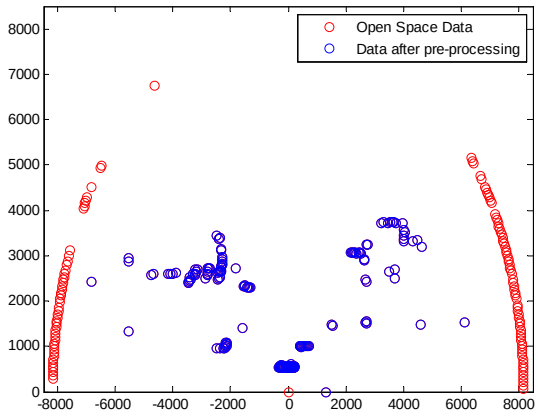


Figure 4.2. Example of pre-processing

for each beam j with valid value in data set D
    the value of corresponding grid of $addMap$ plus 1
for the value of each grid V_m of the $addMap$
    if $V_m >= 1$
        $V_m = 1$
$newMap = oldMap + addMap$
return $newMap$

Figure 4.3 shows the different influence of the different thresholds to the area of stationary map. The blue line with threshold value 10 shows that background objects are "oscillating" in and out of background classification, even after 300 frames. That is because it may take more than 10 frames (around 0.3s) for a large vehicle driving through a $40cm \times 40cm$ grid and this grid will be marked as background temporarily.

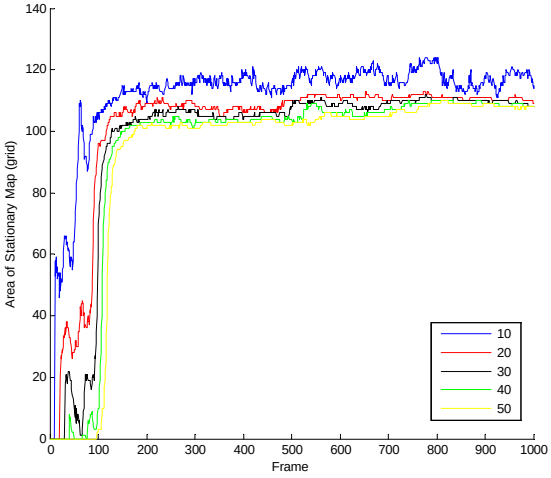


Figure 4.3. Stationary map area verses threshold value of Park Ave. data set

Table 4.1. Statistic information of the stationary map

Threshold	10	20	30	40	50
Mean	116.9165	111.8367	110.4451	109.2618	108.0798
Standard Deviation	4.1089	2.3833	1.9982	1.9031	2.2343

Table 4.1 presents the mean values and the standard deviation values of the area of the stationary map after 200 frames based on different thresholds. Since the area of the stationary map has the smallest standard deviation with threshold 40, this threshold is used in this work.

Although this algorithm cannot distinguish the temporarily stopped objects from the background, their states are still being tracked. One probable improvement of this approach is taking into account the space occluded by other objects. If only the grids on open space minus their values, the map may be more stable.

Figure 4.4 shows the stationary map of the data collected on Park Ave. As discussed above, the threshold is 40, which means any grid with value greater than 40 will be counted as background. The background map demonstrates that the grid-map based background detection algorithm is fairly good and is reliable for different kinds of objects, for instance the trees around (230, 160) and the bushes around (120, 140).

4.2 Segmentation and classification

Once the noise is removed from the raw data, the LIDAR scan is separated into different segments and classified by the similar rule-based classification method applied by Nashashibi [36]. Considering that motorcycles are less common than pedestrians in an urban traffic scene and the environment elements are extracted by grid map, the segments are classified either as vehicles or pedestrians. The details are discussed in Section 4.2.3. After classifying, every segment will be marked by feature points for the convenience of data association. Figure 4.5 presents the flow chart of the classification process. For convenience of implementation and to have an intuitive interpretation, the data will be translated from polar coordinate to Cartesian coordinate after preprocessing. Unless noted, all the approaches discussed are based on a Cartesian coordinate system.

4.2.1 Segmentation

As one of the most critical steps, data segmentation algorithm attracts many researchers'interests (Section 2.2.2.2). In this thesis, for easiness of implementation with acceptable error, DBSCAN is applied for data segmentation. To implement this approach in Matlab, a code written by Daszykowski is used [32]. Figure 4.6 shows the example of DBSCAN and compares it with the adaptive distance gap algorithm presented by [30].

Since the LIDAR calculates the distance by sending out a laser beam and computing the time difference between sending and receiving, the black surface could absorb most of the laser and reflect such small energy that the sensor thinks this beam does not hit any obstacles. Figure 4.7

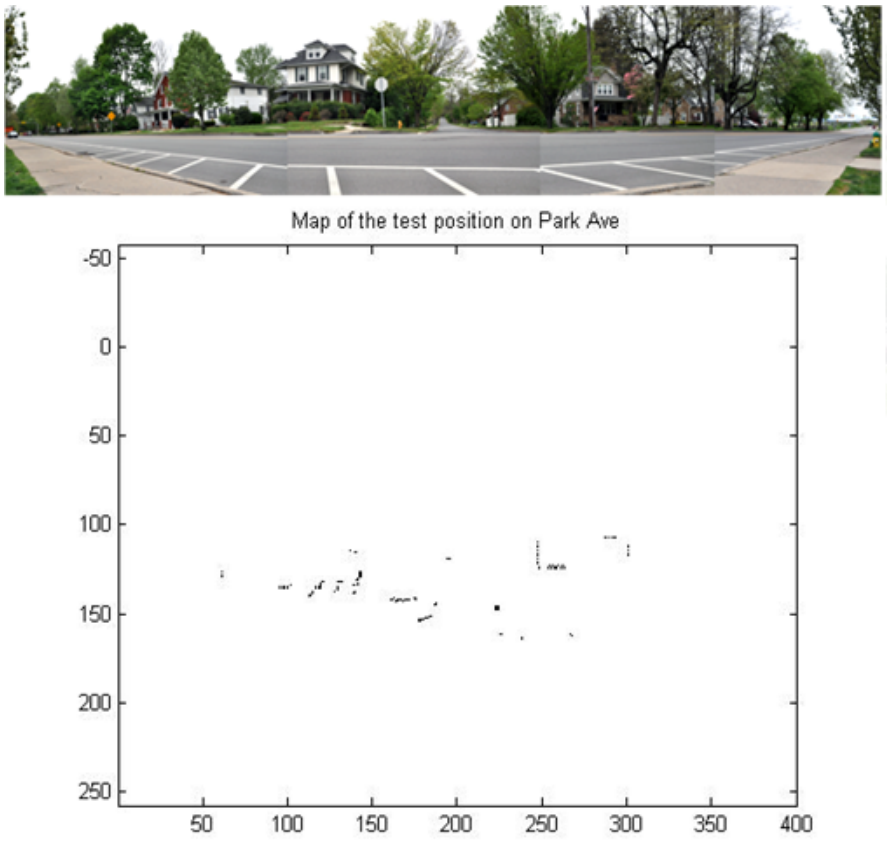


Figure 4.4. Grid-map of one experiment location

shows the segment results of consecutive 10 frames. Because of the inaccurate measurement, both the positions and the shapes are not reliable.

As discussed in Chapter 3, the object detected by the sensor might be partially occluded in the middle and the data segmentation approach may mark it as two separated segments, when the gap caused by occlusion is too large. One example is showed in Figure 4.8, where red dots represent the sensor measurements, blue rectangular the segment results and the frame is marked by number. In five consecutive frames, one object is occluded by a small obstacle around (800,250) with the LIDAR on (0,0). The data points are segmented into two objects in frame 1, 2 and 3. Then, in frame 4 and 5, they are successfully segmented as a uniform object.

4.2.2 Partially occluded object detection

The segment detected by the LIDAR sensor can be partially occluded by other objects (Figure 4.9). Since the segment boundaries of real objects are often spurious, some features provided by them are vague and should be ignored [66]. As stated by Fayad and Nashashibi [38, 36], in this thesis, the partial occlusion can be detected by the distance information and classified as

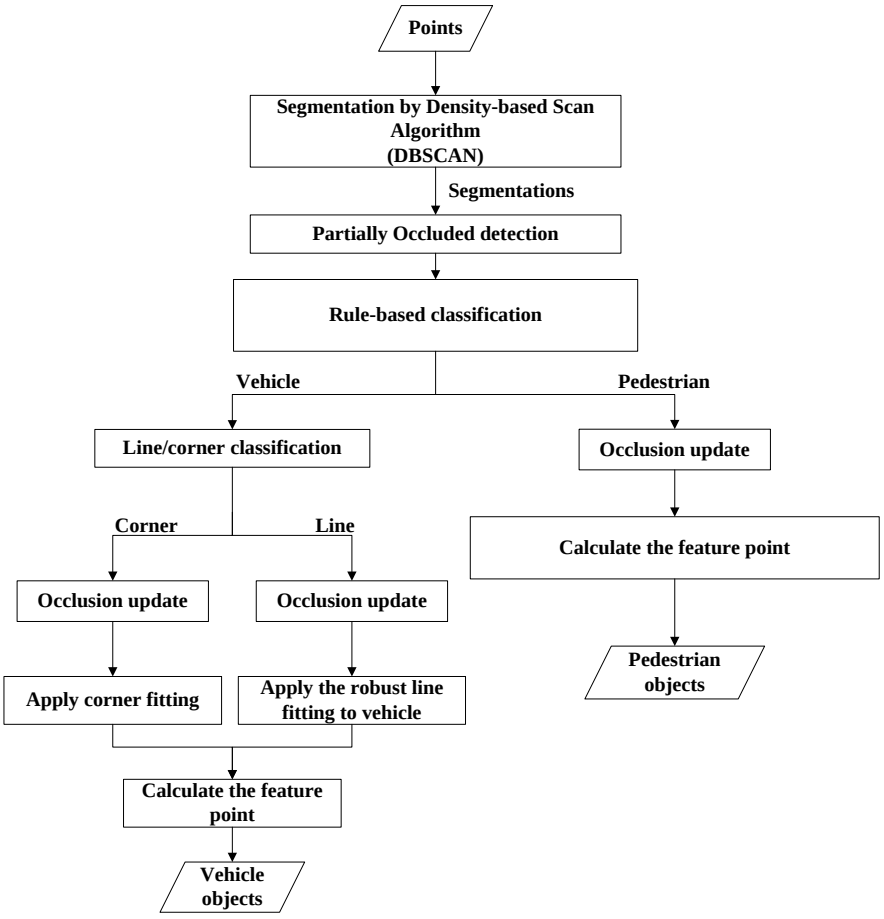


Figure 4.5. Flow chart of the classification step

follows :

- Occluded one endpoint;
- Occluded both endpoints;
- Occluded middle part;

If the gap in the middle part occlusion is less than the threshold of DBSCAN, it could be resolved by DBSCAN. Even the vehicle is divided into two parts, the system could still keep tracking the target by feature point association discussed in Section 4.3.

In this thesis, for any segment detected by beams l_a to l_b $(a < b)$, the endpoint at beam l_a is marked as occluded if there is an $i \in [a-5, a)$ such that $D_{l_i} < D_{l_a}$, where D_l is the distance value of beam l. The endpoint at beam l_b is marked as occluded if there is an $i \in (b, b+5]$ such that $D_{l_i} < D_{l_b}$, where D_l is the distance value of beam l. The constant 5 is selected by trial to avoid the sensor noise between two objects. The pseudo-code of the occlusion detection is:

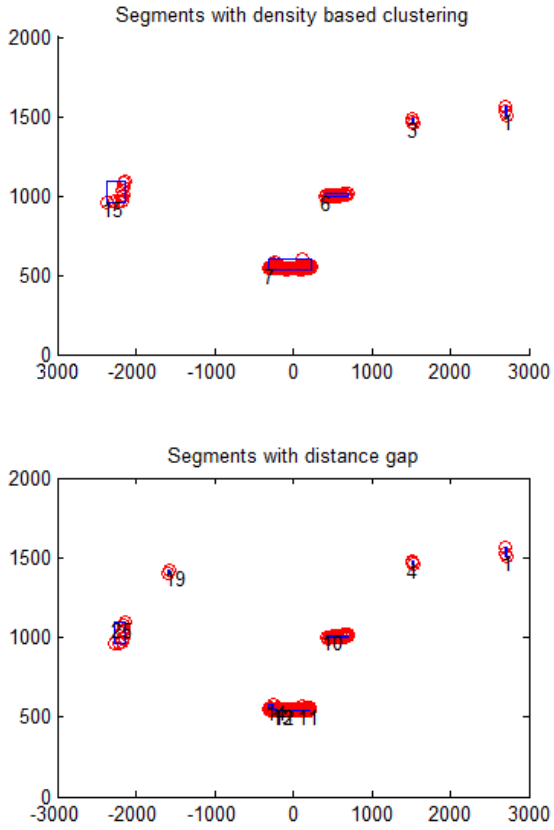


Figure 4.6. Comparison of different segment methods, DBSCAN (up) groups the segments 11 to 14 obtained by distance threshold segmentation (down) and eliminate the noise segment 19.

occlusionDetec(S, D)
for each segment i in S
    for each beam j in the 5 neighbor beams outside the endpoints of i
        if the distance of j is less the distance of corresponding endpoint
            $D(j) = 1$
return D

Because of the angular resolution limitation, in some particular situations the object detected by LIDAR may have a significant error, like the horizontal side of vehicle A and vertical side of vehicle D in Figure 4.9. In this thesis, when the distance between the beam of the endpoint and its consecutive one on the same surface is larger than 50cm, this endpoint will be treated as occluded.

4.2.3 Object classification and feature extraction

Similar to the method mentioned by Nashashibi [36], a rules-based classification is performed to distinguish pedestrian and vehicles:

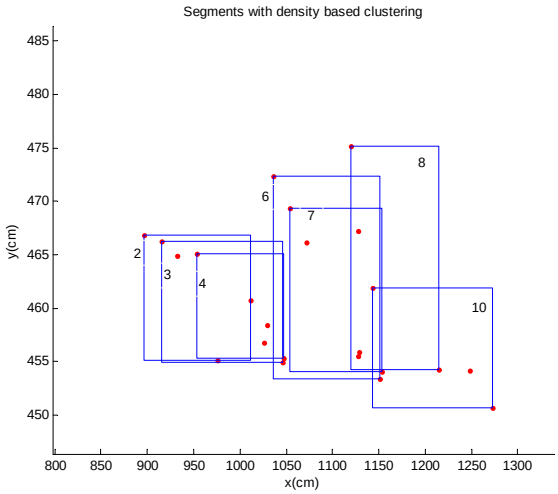


Figure 4.7. Example of the segmentation results of a dark object

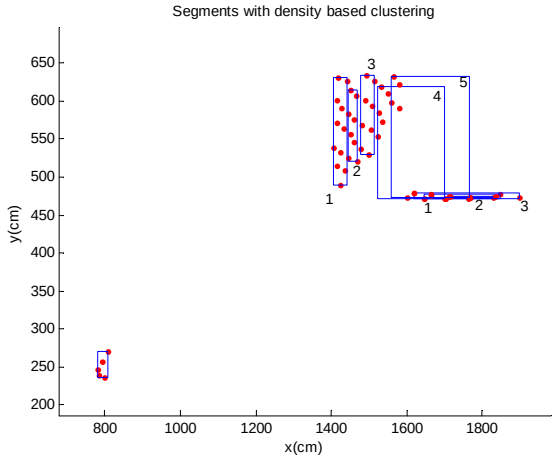


Figure 4.8. Example of middle occluded object

- Segments with width less than 80cm are pedestrian;
- Segments with width larger than 80cm are vehicle candidates and will be fitted to line shape or L shape;
- L-shaped segments with both sides less than 80cm and no occlusion detection are vehicle.

The pseudo-code of this step is:

classification(S)
for each segment i in S
    if the diagonal of the bounding box of i is less than 80cm
        mark i as Pedestrian
    else

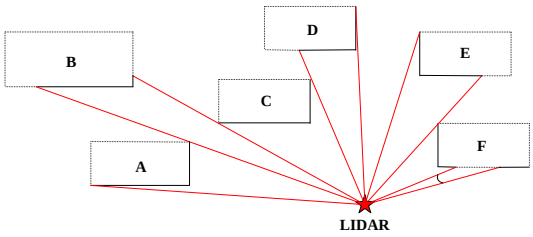


Figure 4.9. Illustration of occlusion. A, C is totally visible, E has one endpoint occluded, B,D have two endpoints occluded and F is occluded in the middle part. The vertical side of D and the horizontal side of A is marked as occluded due to the angular resolution limitation.

mark i as potential vehicle
LlDistinguish(i)*
if the segment is a line
    robustLinefit(i)*
else
    cornerFit(i)*
    if both sides less than 80cm and no occlusion marked
        mark i as Pedestrian
return S
(* these parts are presented in the following section)

4.2.4 Line and "L-shape" classification

For each vehicle candidate, the corner point is found by searching the distance between the points and the line formed by the two endpoints. The farthest point will be regarded as the corner point. After that, the segment will be separated into two parts and the weighted line fitting (discussed in Section 4.2.6) will be applied to each part. If the angle between the two lines is less than 45 degree, this segment will be marked as line. Otherwise, it will be marked as "L-shape". When the number of the points of either part is less than 3, the segment will be marked as line since the information of this part is too vague to determine a feature. Figure 4.10 presents an illustration of this step and it is implemented in following pseudo-code:

LlDistinguish(i)
for each point n in segment i
    $d(n)$ = distance between n to the line formed by two endpoints
mark the point with greatest d as potential corner point C
divided i into two sides A, B based on C
if $sizeof(A) <= 3$ or $sizeof(B) <= 3$
    mark i as a line

return
else
$lineA$ = weightedlineFit(A)*
$lineB$ = weightedlineFit(B)*
a = angle between $lineA$ and $lineB$
if $a < 45$
mark i as a line
else
mark i as a L shape
(* this part is presented in following section)

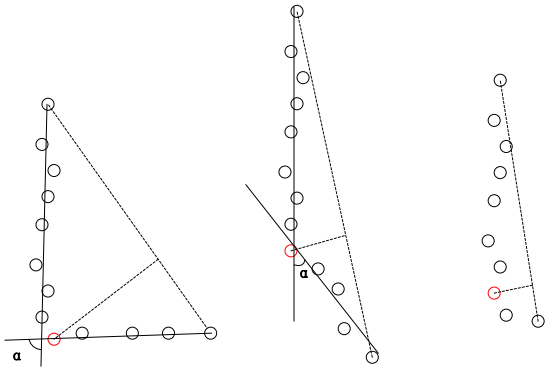


Figure 4.10. Illustration of line/corner classification. Left one represents the L shape, middle one the line shape and right one the line shape

4.2.5 Robust line fitting

Because vehicles are built differently, a fixed-height horizontal scan by a LIDAR may capture different features depending on the vehicle. For example, a scan that captures the straight side panels and fender lines for a normal car might capture the tires and fender wells of a large truck. As a result, the segment may not be a straight line [27]. One solution, suggested by MacLachlan, is to apply a robust line-fitting algorithm to resolve this problem [33]. This approach is also used in this thesis and the details are described below:

At the beginning, the segment is fitted to a line by weighted line fitting and then 20% worst-fitted points will be deleted. Following that step, weighted line fitting is applied again to get a more robust result. As we can see from Figure 4.11, after deleting some bad points, the robust line fitting algorithm can provide more reliable results without being disturbed by sensor noise. The pseudo-code of this algorithm is:

robustlineFit(i)
$line1$ = weightedlineFit(i)*

for each point n in segment i

$d(n)$ = distance between the point n and the $line1$

descending sort d

set the points corresponding to the top 20% d to NAN

$line2$ = weightedlineFit(i)

return $line2$

(* this part is presented later)

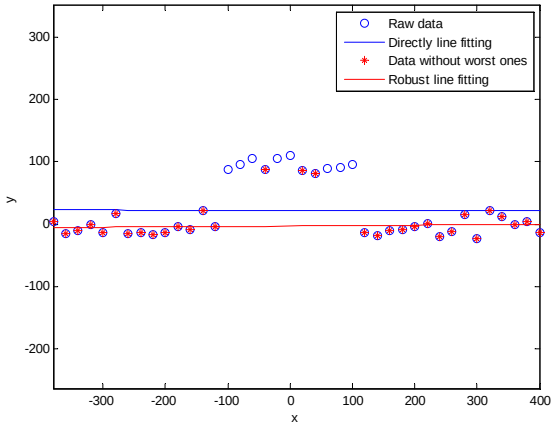


Figure 4.11. Example of Robust Line Fitting

4.2.6 Weighted line fitting

Assume a set of points could be fitted into a line as Equation 4.1:

$$y = kx + b \tag{4.1}$$

After rewriting it into a matrix form and substitute x, y with n points in the data set, a more general equation could be obtained:

$$Y = X\beta \tag{4.2}$$

where

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}, \beta = \begin{bmatrix} k \\ b \end{bmatrix} \tag{4.3}$$

Applying weighted least squares estimation, the parameters of the line could be estimated by

$$\beta = (X^T W X)^{-1} X^T W Y \tag{4.4}$$

where W is a diagonal matrix with the weighted factor of every point. Considering near points are denser than far ones, the weight factor is the square of the Euclidean distance between the point to the LIDAR:

$$W_i = x_i^2 + y_i^2 \tag{4.5}$$

where W_i is the weighted factor of point i at position (x_i, y_i). This algorithm is implemented in the following pseudo-code:

weightlineFit(i)
for each point n in segment i
    $w(n) = x(n) * x(n) + y(n) * y(n)$
    $w = w/max(w)$
calculate slope k and intercept b based on Equation 4.4
return k, b

4.2.7 Corner fitting

Since the geometric model of vehicles can be regarded as rectangular, any L shape should be a corner with right angle. For robust feature extraction, as suggested by RA.Maclachlan [66], a corner fitting is applied here. The first step in corner fitting is the same as that of line and "L-shape" classification: dividing the cluster into two sides based on the roughly corner point. Then the longer one will be used as the base side since it is more robust than the shorter one. The base side will be fitted into a line first. Next, the perpendicular line is obtained by least-squares linear regression by enforcing a slope requirement on the fit. The corner point will be refined by the intersection point of lines fitting the two sides. Both sides are fitted by weighted line fitting. This algorithm is implemented in the following pseudo-code:

cornerFit(i)
for each point n in segment i
    $d(n)$ = distance between n to the line formed by two endpoints
mark the point with greatest d as potential corner point C
divided i into two sides: A and B based on C
if the length of the diagonal of $A's$ bounding box > the length of the diagonal of $B's$ bounding box
    $[k1, b1]$ = robustlineFit(A)
    $k2 = -1/k1$
    $x = x$ values of all the points in B
    $y = y$ values of all the points in B
    $b2 = (sum(y) - k2 * sum(x))/sizeof(x)$
else

$[k1, b1] = \text{robustlineFit}(B)$
$k2 = -1/k1$
$x = x$ values of all the points in A
$y = y$ values of all the points in A
$b2 = (sum(y) - k2 * sum(x))/sizeof(x)$
$CP =$ point of intersection between $[k1, b1]$ and $[k2, b2]$

As the example in Figure 4.12, the corner point calculated by the algorithm is more accurate than picking any other specific point from the data since the actual corner of real vehicles tends to be rounded.

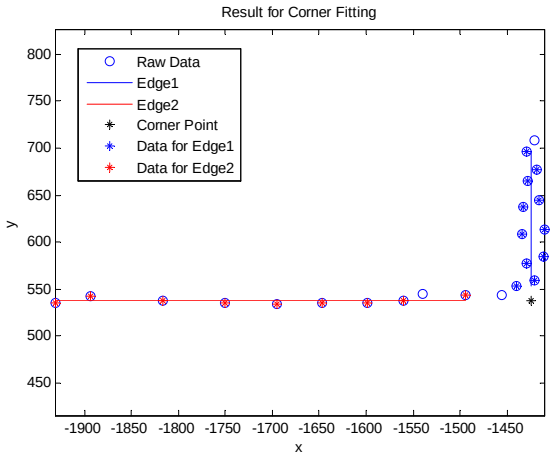


Figure 4.12. Example of corner fitting

4.2.8 Feature points calculation

The feature points are used to represent a vehicle depend on the shape and occlusion situation of the segment of points representing the vehicle. For a vehicle-classified object, the object may be in the shape of a line, an "L", or be uncertain. For a line segment, the two endpoints can serve as feature points. For this line representation, any occluded endpoints are deleted. For an "L-shaped" object, the corner point will also be included as a feature. If no feature point is detected, the mean point of the whole segment will be counted as the feature point. For pedestrian-classified objects, the feature point is represented by the mean point of the segment. Figure 4.13 illustrates some examples of the feature points calculation.

4.3 Data association

The data association algorithm used in this thesis is called the Greedy Nearest Neighbor (GNN) algorithm. The experiment results from others (as well as this thesis) show that it is good enough with acceptable errors.

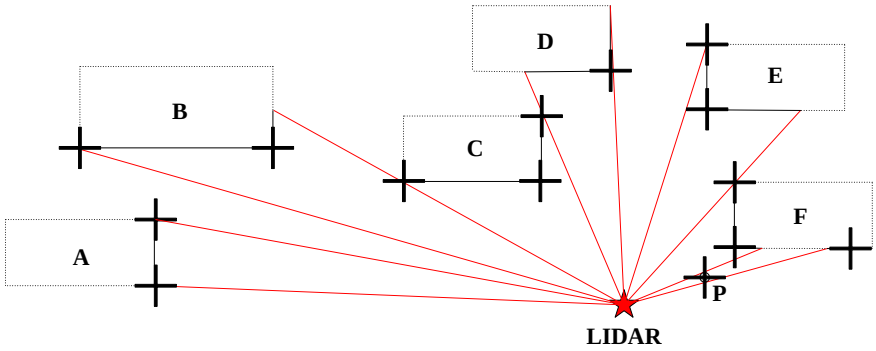


Figure 4.13. Examples of feature points calculation. The black crossings represent the feature points.

To explain the GNN algorithm, we first consider the physical characteristics of a moving vehicle. Specifically, the Mahalanobis distance is used to measure the distance instead of Euclidean distance because it takes into account the distribution of the data (the details will be discussed in following section). When matching the object to its nearest cluster, the feature points are used to reduce the computational complexity and decrease the influence of the noise and bias. According to the matching result, every object obtains a new speed. If the new speed is under a threshold determined by the type of the object, the match is considered successful and the object gets a new position. Otherwise, it is stored for a short period. Such stored objects are marked as missing and deleted if no prior or subsequent matches are found for several frames. The detailed logical structure is showed in Figure 4.14 and the pseudo-code of this part is showed as following:

dataAssociation($objectList, S$)
T = time interval between two consecutive frames
for each object o in $objectList$
    if S is empty
    mark o as "missing"
    $missTime + 1$
    if $missTime > thresholdDelet$
        delet o from $objectList$
    for each segment i in S
        $D(i)$ = Mahalanobis distance between o and i
    match o and segment p with minimum D based on corresponding feature point
    $speed$ = (distance between corresponding feature points)$/T$
    if $speed > threshold$
        mark o as "missing"
        $missTime + 1$
        if $missTime > thresholdDelet$
            delet o from $objectList$

elseif the state of the o is "normal"
update the object by Kalman Filter
delet p from S
elseif the state of the o is "adding"
$addTime + 1$
delet p from S
if $addTime > thresholdAdd$
set the state of o to "normal"
elseif the state of o is "missing"
set the state of o to "normal"
delet p from S
if S is non-empty
add all the left segments in $objectList$
return $objectList$

4.3.1 GNN data association

Every existing object in the last frame is compared to all the new classified clusters to find the nearest match. The algorithm gives priority to the oldest objects first, which helps to eliminate the accidental miss matching caused by noise or wrong classification of dark objects (Figure 4.15).

Shown in Figure 4.15 are two consecutive frames of the same object (red circles). The black crossings are the four corner feature points of the vehicle. Since the error or accuracy of the sensor, there is gap between two edges of the right angle. In the first frame (shown at left), the gap is notable such that the DBSCAN regarded them as different clusters and association step marks the horizontal edge as an "adding" object (blue circles). In the new frame (shown at right), the object is tracked successfully again because the older objects have priority to match to the segments. The artificial "added" object is deleted since it does not last for enough time

4.3.2 Feature point association

To form a feature point association, the new position and speed of the object are calculated by determining the feature correspondence between measured data and previously associated object. In this study, there are two kinds of feature points considered: feature points representing the mean position of the segment, and corner figure points.

As discussed in section 4.2.8, the feature points of the segment are computed during the classification step. The object will keep the feature points of prior data after successful matching, and the object will add new feature points when the segment is associated with more information from new data. Eventually, every object should have four corner feature points unless the segments it matched cannot provide enough information. One example of where there is insufficient

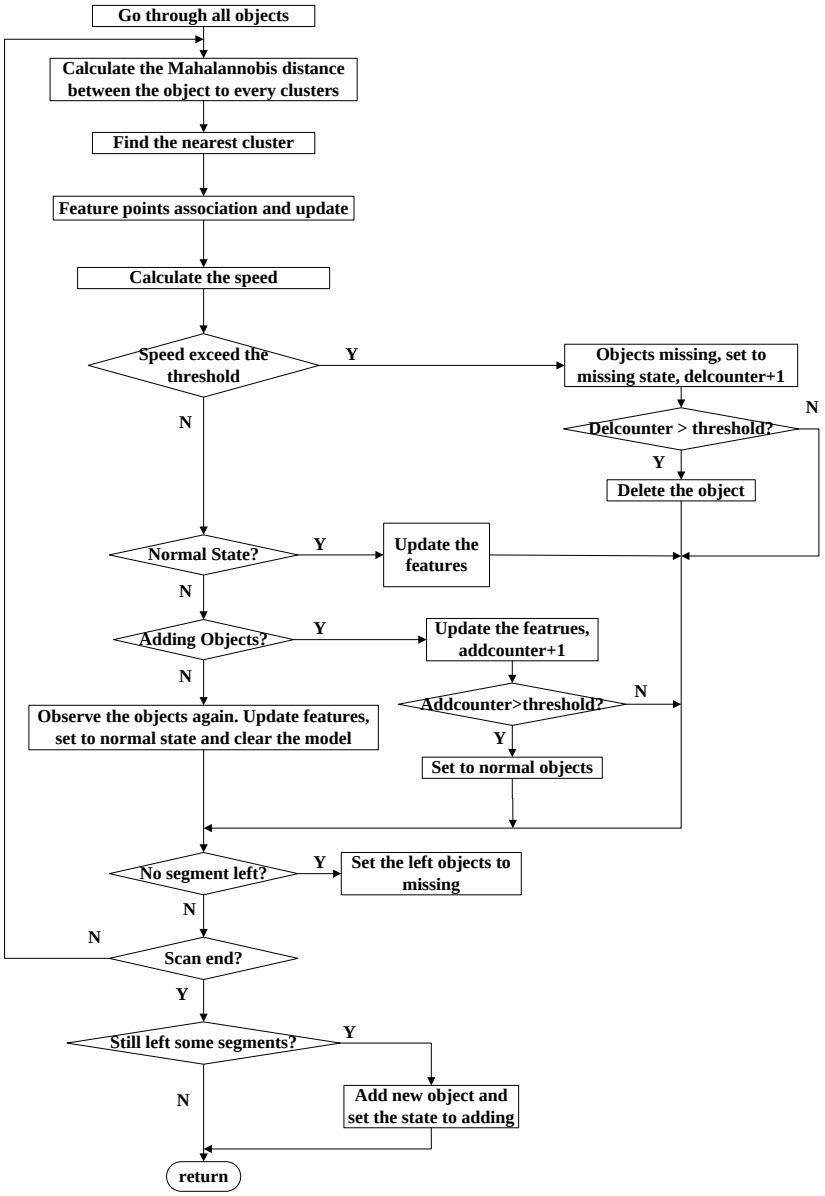


Figure 4.14. Flow char of data association step.

information would be when the vehicle is scanned from the frontal direction such that the back of the vehicle is never visible.

Figure 4.16 is an example of data association of 20 consecutive frames. The black dots are background extracted by grid map approach, the red dots the segments obtained by sensor and the black crossing the feature points of the vehicle object. The number is calculated automatically by the tracking system, which represents the order of the object (including the noise object with

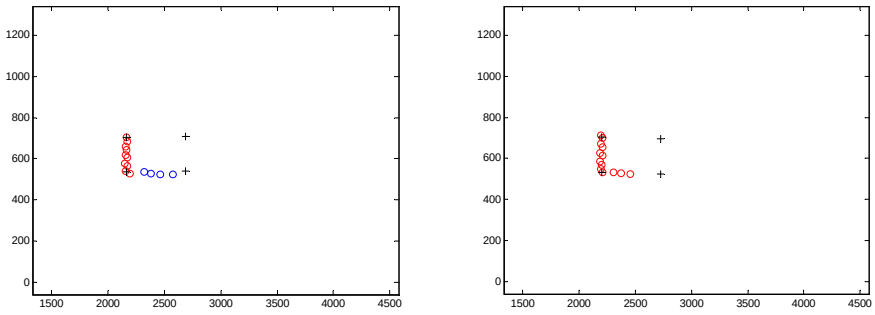


Figure 4.15. Example of GNN data association

short survival time and the obstacle on background). The true number of the moving objects can be computed by counting the normal marked objects on foreground.

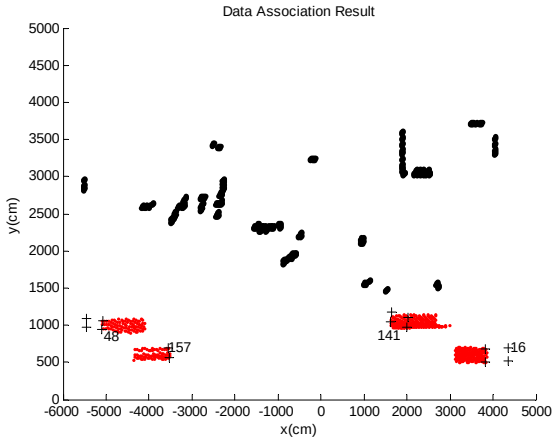


Figure 4.16. Example of data association

Figure 4.17 is another example of data association. Black star represents the feature point of the pedestrian object. It indicates the algorithm can deal with the vehicle object and pedestrian object at the same time.

The feature statistics such as the mean and corner positions will obviously change as the object moves through the scanning field, and sometimes these changes are quite abrupt. To make the model of the vehicle stable, the length of the edge is computed by averaging all the corresponding history valid edge values. In other words, the feature "remembers" the extent of previous vehicle scans and uses this information to improve the expected mean and corner positions of the vehicle. Figure 4.18 shows the lengths of two sides of the vehicle 15 in previous Figure 4.17 versus the time.

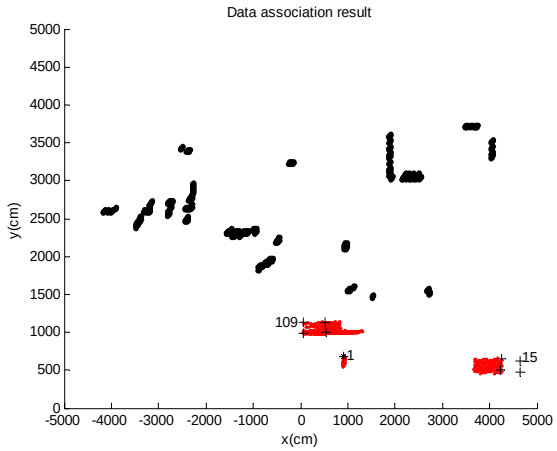


Figure 4.17. Example of data association

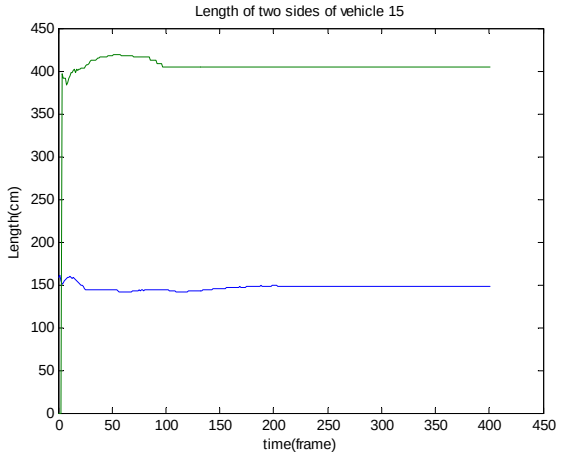


Figure 4.18. Lengths of the sides

4.4 Model motion

Similar to Lim used in visual tracking system [83], a general time series dynamic model is applied in this work to predict the state of targets during occlusion.

4.4.1 Dynamic modeling for motion model

In this thesis, the motions considered include at least constant velocity, constant acceleration, and geometric turning. To avoid over-fitting of these trajectories, both second-order and third-order difference equation models were applied. A constant parameter is involved by setting the input to 1 for obtaining a relatively accurate initial value.

To obtain a dynamic model representation, an l-th order time series dynamic model can be

written as Equation 4.6,

$$y_k = \sum_{n=1}^{l} a_n y_{k-n} + bu \tag{4.6}$$

where y_k is the position of the first feature of object in k-th frame, u is the input and always set to 1. The model with constant velocity v is a particular case; i.e.:

$$y_k = y_{k-1} + v, k = 2, 3, ... \tag{4.7}$$

The model with constant acceleration a is also a particular case, i.e.:

$$y_k = 2y_{k-1} - y_{k-2} + a, k = 3, 4, ... \tag{4.8}$$

Since the x position and y position are modeled separately, the turning motion could be modeled by differentiating the velocities in the x and y directions.

4.4.2 Implementation

Similar to an optimization problem, the dynamic model coefficients can be learned in a manner that best fits the observed data. The goal of this process is to minimize the objective function below, which is the sum of the least-square errors. This thesis uses System Identification toolbox in Matlab to build the dynamic model.

$$\sum_{k} (y_k - \sum_{n=1}^{i} a_n y_{k-n} + b)^2 \tag{4.9}$$

After the dynamic model is built, the states of the objects will be estimated by Kalman Filter using following state space models:

Third order:

$$\begin{bmatrix} x_k \\ x_{k-1} \\ x_{k-2} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ x_{k-2} \\ x_{k-3} \end{bmatrix} \tag{4.10}$$

$$Z = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k-1} \\ x_{k-2} \end{bmatrix} \tag{4.11}$$

Second order

$$\begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ x_{k-2} \end{bmatrix} \tag{4.12}$$

$$Z = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix} \tag{4.13}$$

Once the object gets 20 valid observations, the model is updated by system identification

using observed data. Because motion dynamics of the object cannot be described by only one model, the system will update the motion model every 20 frames. To reduce the error of the model estimates during the temporary occlusion, the model is updated based on 50 history observations before predicting any future trajectories. This work applies the "arx" function in Matlab to perform the system identification and the psudo-code of the kalman filtering step is:

updateKF(z, KF)
$xp = KF.A * KF.X' + KF.Bu$
$pp = KF.A * KF.P * KF.A' + KF.Q$
$k = pp * KF.H' * (KF.H * pp * KF.H' + KF.R)^{-}1$
$newKF.X = (xp + k * (z' - KF.H * xp))'$
$newKF.P = (I3 - k * KF.H) * pp$
where I3 is a 3X3 identity matrix.

4.4.3 Experimental results

To test the dynamic model and select its order, many experiments are performed based on simulated data. In these simulated data, all the objects involved are represented by $40 \times 40cm^2$ squares. Because the object's width is less than 60cm, it will be marked as pedestrian class and the mean feature points are used to build the model. For the vehicle class object, only one of the corner feature points will be used to build the model. Therefore, the results based on square-shape objects can evaluate the dynamic modeling of both pedestrian and vehicle objects.

4.4.3.1 Tracking of different dynamic motions

Figure 4.19 to Figure 4.25 show the results of data filtering from simple scenarios to hard scenarios. Seven scenarios are created:

1. Constant velocity motion
2. Constant acceleration motion
3. Turning motion
4. Forward-backward motion
5. Moving-stationary-moving motion
6. Lane change motion
7. Lane change (hard) motion

Every scenario is tested by both 3^{rd} order model (blue line) and 2^{nd} order model (green line). The red line represents the true motion. For every scenario, the trajectory (left) and the estimated velocity (right) are shown.

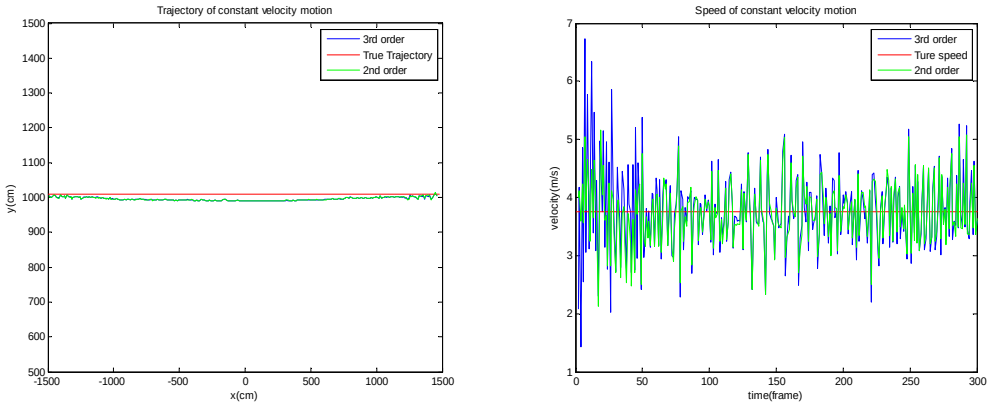


Figure 4.19. The constant velocity motion

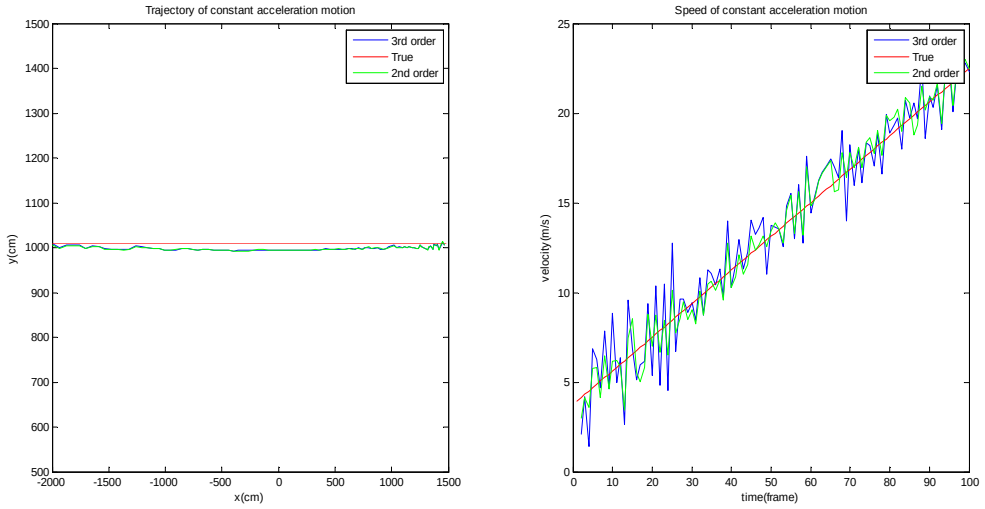


Figure 4.20. The constant acceleration motion

The tracking trajectories of all the scenarios can follow the true ones fairly well. The reason of the slight bias between the true trajectory and the tracking trajectory is because the true trajectory is produced by the path of the square's center of mass and the tracking trajectory is formed by the path of the feature points.

For relatively simple scenarios 1, 2, 3, 4 and 5, the velocity approximated by the 2^{nd} order model has better performance. However, for relatively complex scenarios 6 and 7, the speed plots show that the 3^{rd} order model is more robust than the 2^{nd} order model.

4.4.3.2 Prediction during occlusion

Several occlusion scenarios are created to evaluate the prediction performance of the identified dynamic model during occlusion. Same to the experiments above, 3^{rd} order model (blue line) and 2^{nd} order model (green line) are tested and compared. The black dots represent the obstacle

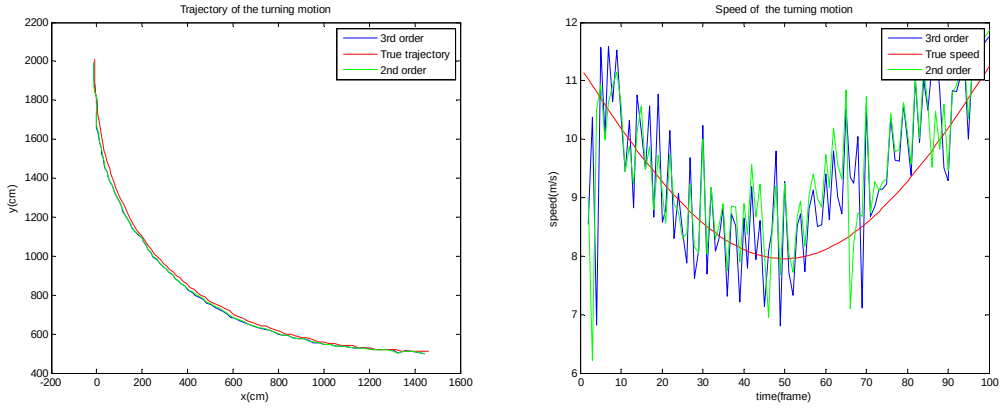


Figure 4.21. The turning motion

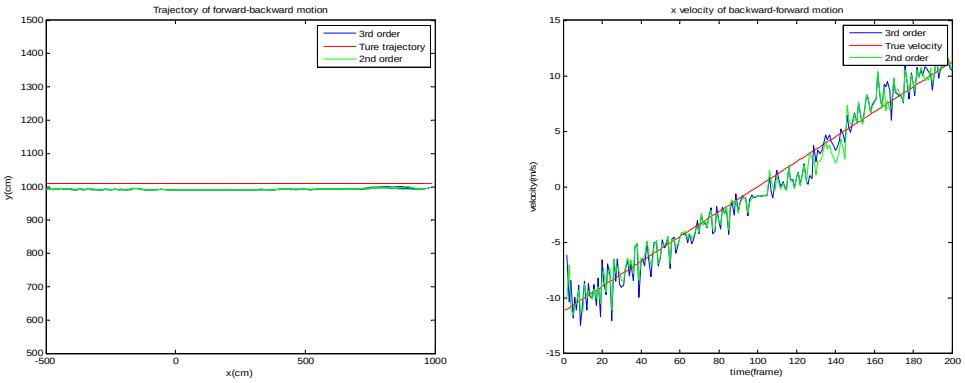


Figure 4.22. The forward-backward motion

and the star points are the positions predicted by the identified model.

According to the simulated experiment results, when the model is identified by enough training data and the occlusion time is relatively short, both 3^{rd} order model and 2^{nd} order model work well. However, when the training data set is small (Figure 4.29, Figure 4.30) or the object is occluded by a relatively long period of time (Figure 4.31, Figure 4.32), the 2^{nd} order model will lose the track. Therefore, for the better performance, the 3^{rd} order model is applied in this thesis.

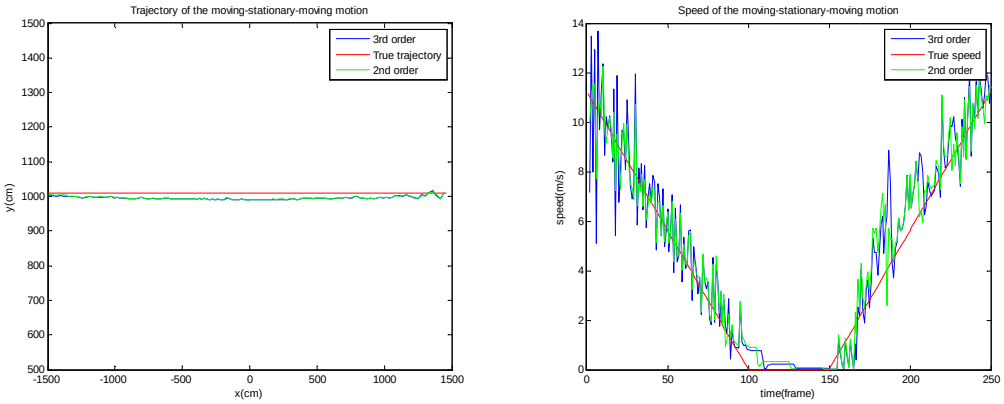


Figure 4.23. The moving-stationary-moving motion

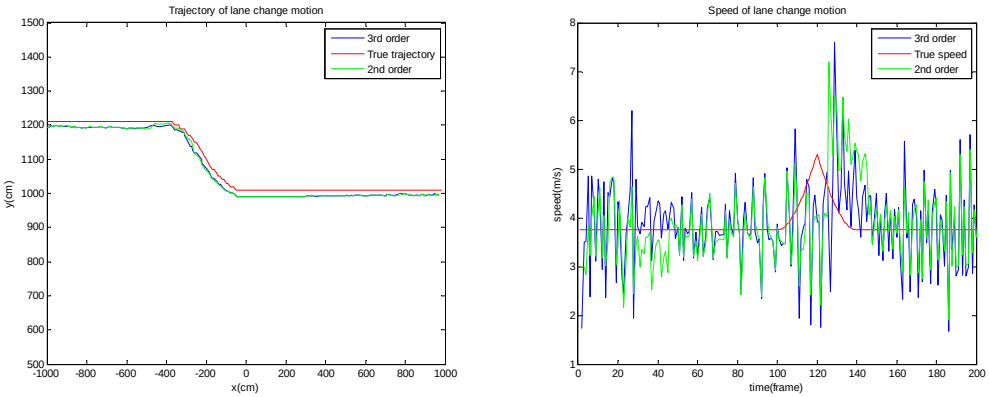


Figure 4.24. The lane change motion

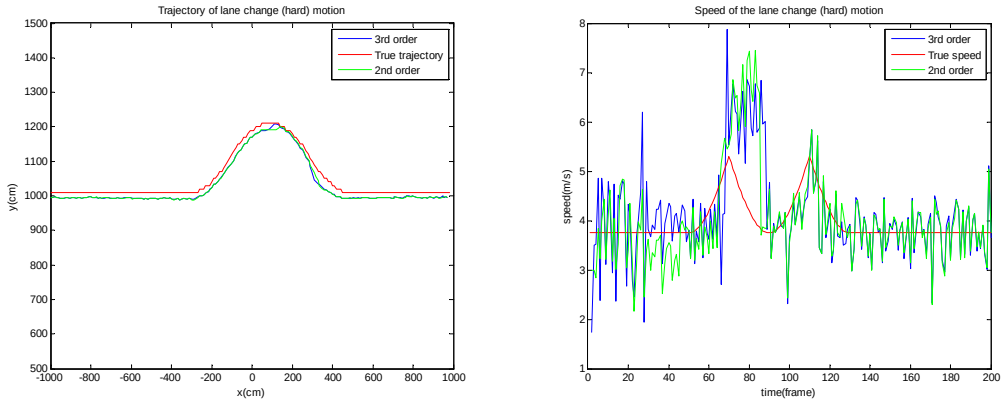


Figure 4.25. The lane change (hard) motion

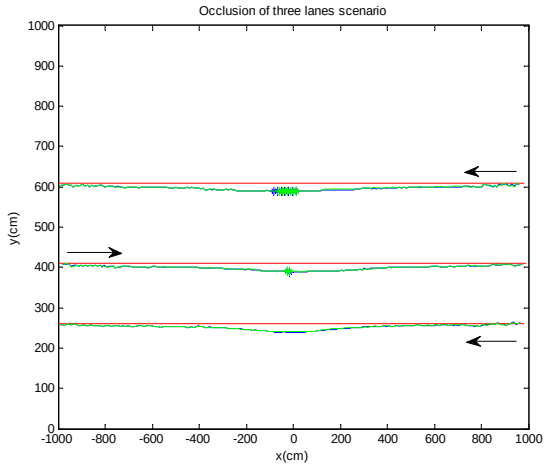


Figure 4.26. Three objects pass by each other. The arrows represent the directions of the movements.

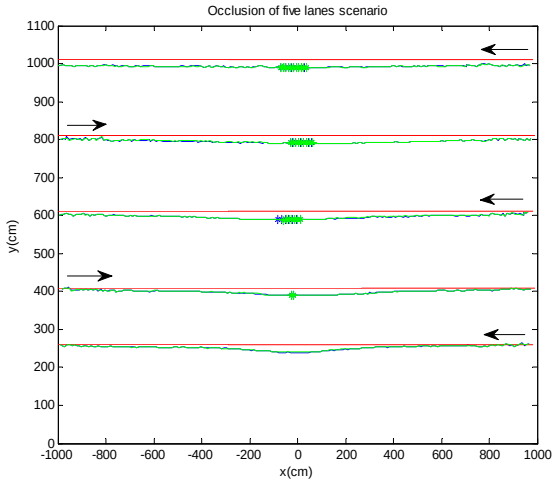


Figure 4.27. Five objects pass by each other. The arrows represent the directions of the movements.

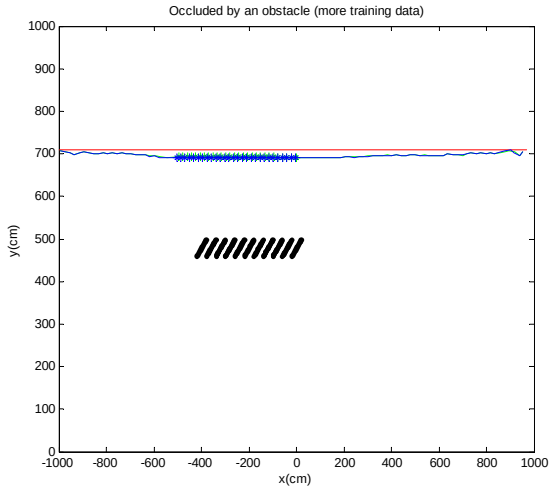


Figure 4.28. Object is occluded by an obstacle. The object moves from the right to the left. The width of the obstacle is 4m and size of the training data set is 49.

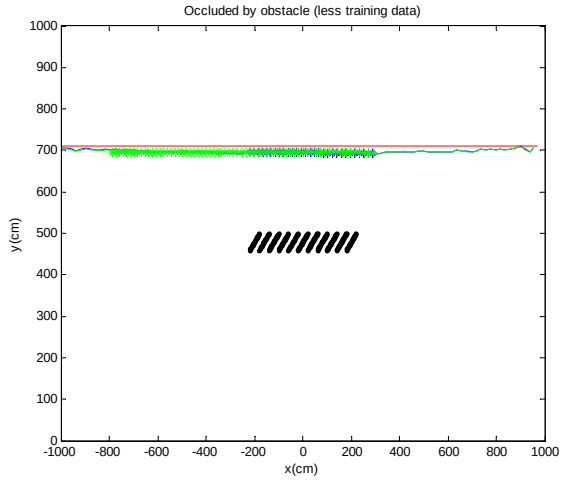


Figure 4.29. Object is occluded by an obstacle (less training data). The object moves from the right to the left. The width of the obstacle is 4m and size of the training data set is 34.

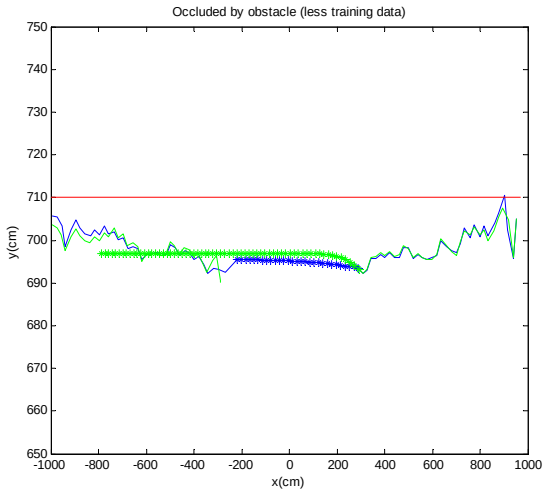


Figure 4.30. Zoom in on Figure 4.29

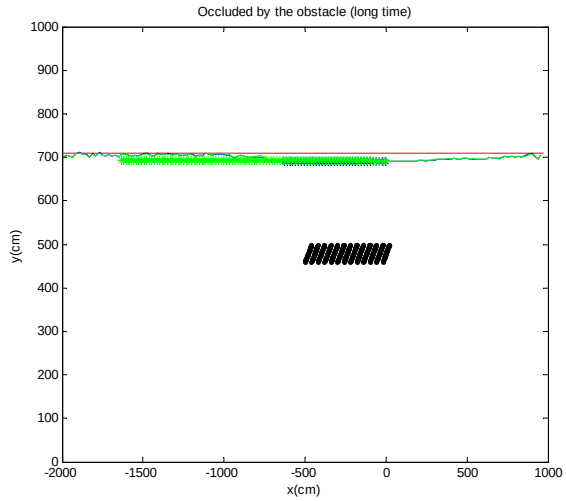


Figure 4.31. Object is occluded by an obstacle (long occluded time). The object moves from the right to the left. The width of the obstacle is 5m and size of the training data set is 49.

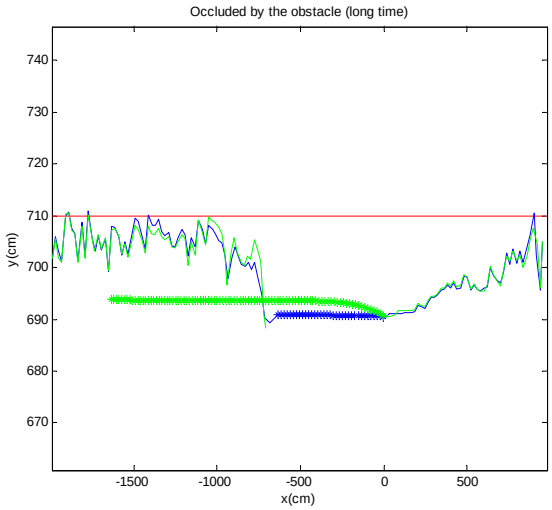


Figure 4.32. Zoom in on Figure 4.31

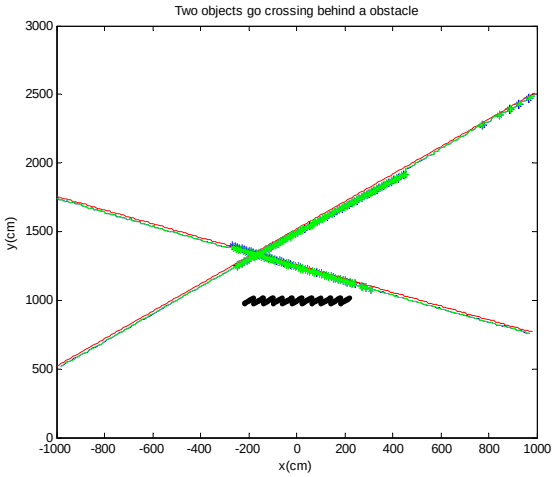


Figure 4.33. Two pedestrians go crossing after an obstacle. Both of them move from the bottom to the top.

Chapter 5

Experimental Results

The experimental data was collected with SICK LIDAR in two different locations in Penn State - University Park (see Chapter 3). This chapter presents both successes and failures in this process. The reasons of the failures are explained.

5.1 Experimental results

Since the previous chapter explained and showed specific examples of the many steps and algorithms, only the primary results of significant parts are shown in this section. Two main steps, background detection, data association and the performance of the whole tracking system are presented below.

Figure 5.1 is a typical snapshot during tracking, where the "*" represents the feature point of pedestrian class object, the "+" the feature point of vehicle object, blue point the LIDAR, red circles the tracked moving objects and yellow circles the missing objects. All the uncolored feature points represent stationary objects, including background obstacles and temporary stationary objects. It demonstrates that the system could track multiple objects simultaneously, including three visible vehicles and one occluded vehicle. Although marked as stationary, the system keeps updating the states of these uncolored objects in case that the object is temporarily stationary.

5.1.1 Result one

Figure 5.2 shows a time sequence of tracking results on Park Ave. location, where the black dots represent the background, black "+" the feature points of vehicle objects in current frame, black "*" feature points of pedestrian objects in current frame, red lines the trajectories of the vehicle objects and blue line the trajectory of the pedestrian object.

During more than 300 frames showed in Figure 5.2, the system successfully tracks four vehicle objects and one pedestrian object. In Figure 5.2(d), the vehicle object in the upper lane is occluded by the vehicle object closer to the sensor and reacquired in Figure 5.2(e) based on the

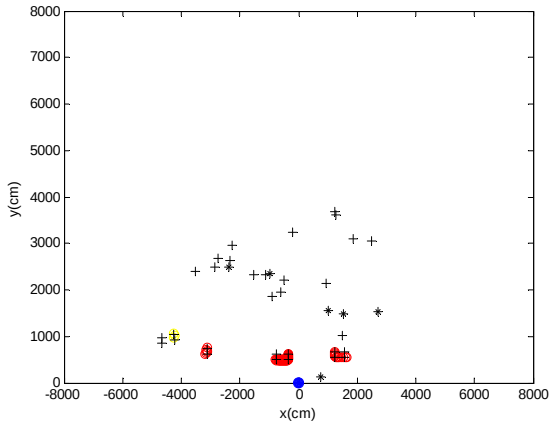


Figure 5.1. Snapshot of the tracking processing

positions predicted by dynamic model during occlusion. The pedestrian object in Figure 5.2(e) also survives from being occluded by the vehicle object near the sensor.

5.1.2 Result two

The data collected near an intersection is used to test the robustness of the system. Considering the vehicle is always occluded by very close pedestrians in the intersection, the sensor is places behind three small obstacles. Therefore, tracking multiple targets in this environment is much more difficult than the experiment on Park Ave. In addition, the stop sign and the crossing also make the dynamics of the object more complex. A time sequence of tracking results are showed in Figure 5.3, where the black dots represent the background, black "+" the feature points of vehicle objects in current frame, black "*" feature points of pedestrian objects in current frame, red lines the trajectories of the vehicle objects and blue line the trajectory of the pedestrian object.

Figure 5.3 presents several tracking results during 1000 consecutive frames. The system successfully tracks three vehicle objects and one pedestrian object. Besides them, the tracks of one vehicle object and one pedestrian object are lost during occlusion, which will be discussed later. This figure indicates the ability of tracking complex motion of the identified dynamic model. One moving-stationary-turning motion object and one moving-stationary-moving object are tracked under the influence of closet obstacles. Although the system loses the track of the pedestrian in Figure 5.3(b), Figure 5.3(c) and Figure 5.3(d), the pedestrian in Figure 5.3(d), Figure 5.3(e) and Figure 5.3(f) survives from being greatly occluded by a closed vehicle object.

5.2 Failure analysis

As showed in Chapter 4, when the moving object is occluded for a long period of time or the training data set is relatively small, the predictions propagated by the identified model may be

not accurate enough to be reacquired by the system. In this section, two examples of the failed tracking are presented.

5.2.1 Failed pedestrian tracking result

Figure 5.4 and Figure 5.5 present a result of a failed pedestrian tracking. In Figure 5.4, the lines represent the observed trajectories, the stars the predicted trajectories, blue color the former part and the green color the latter part. Figure 5.5 shows the speeds of the former part (blue) and latter part (red). This pedestrian object is occluded for nearly 100 frames. Figure 5.4 indicates that the motion is failed modeling as a deceleration motion instead of a nearly constant velocity one. It may be caused by the motion of the pedestrian which is relatively unstable such that recent 50 observations do represent a deceleration motion. For this particular case, using more training data may resolve the problem. However, more training data can make the system fragile to the real motion change.

5.2.2 Failed vehicle tracking result

Figure 5.6 and Figure 5.7 show the failed vehicle object tracking results. Same as the results above, in Figure 5.6, the lines represent the observed trajectories, the stars the predicted trajectories, blue color the former part and the green color the latter part. Figure 5.7 shows the speeds of the former part (blue) and latter part (red). Although Figure 5.7 indicates the estimated speed is very close to the reacquired speed, from Figure 5.6, we can see the predicted trajectory tends to be parallel to the reacquired trajectory, which is hard to be associated because of the Mahalanobis metric.

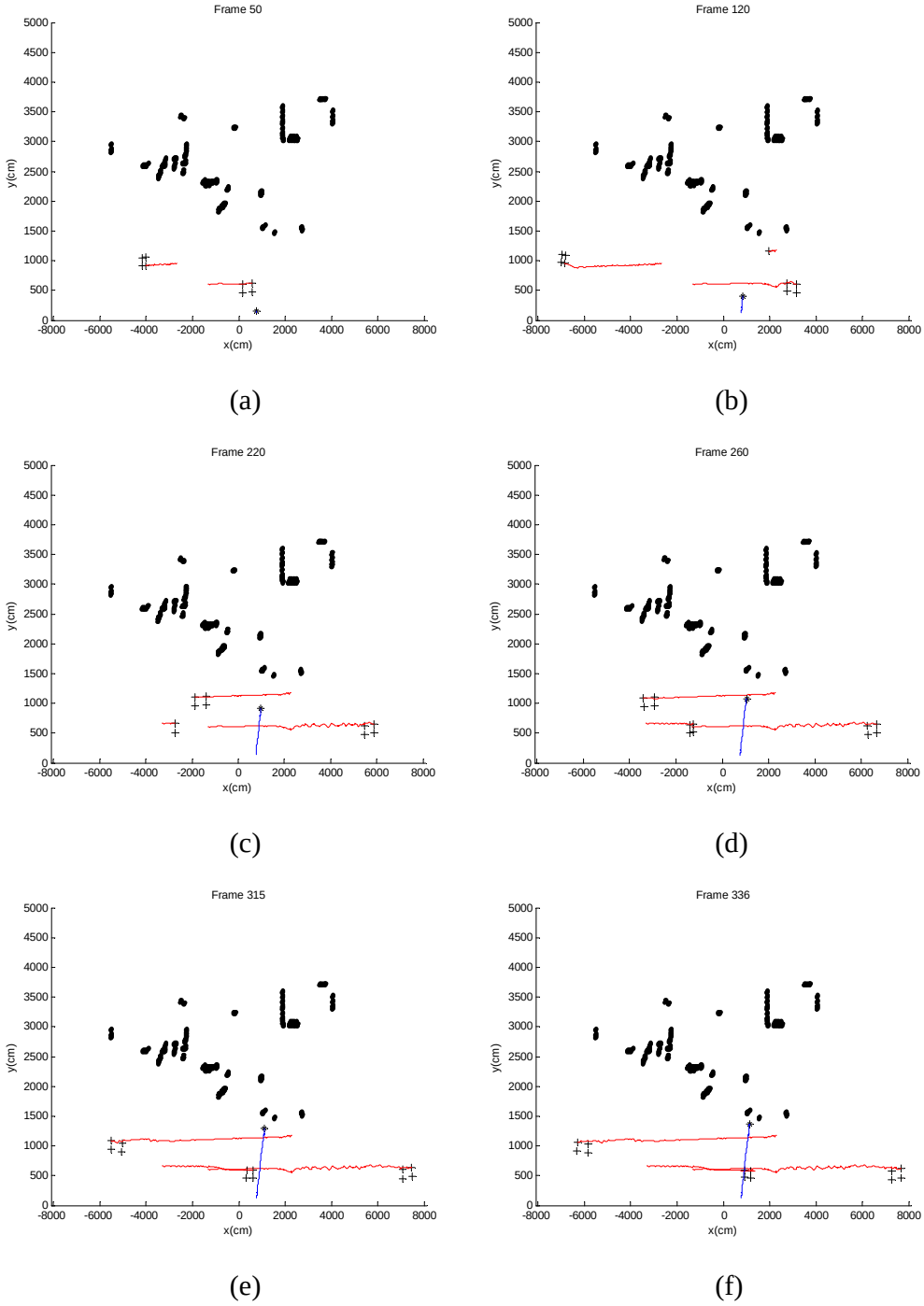


Figure 5.2. Example of multi-object tracking result

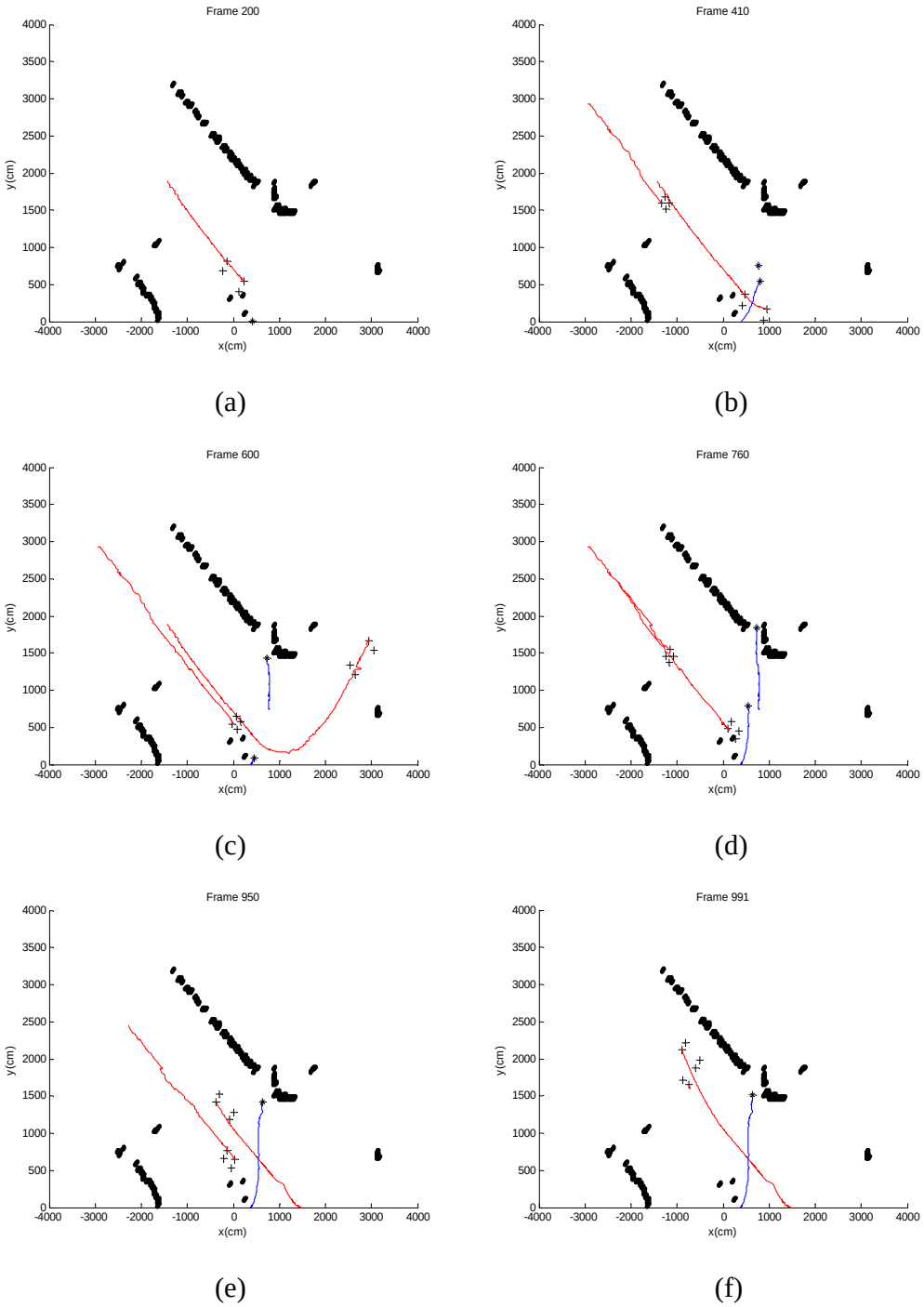


Figure 5.3. Example of multi-object tracking result

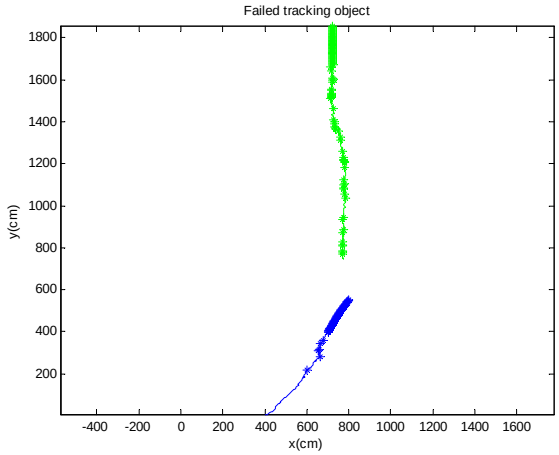


Figure 5.4. Trajectory of a failed tracking pedestrian

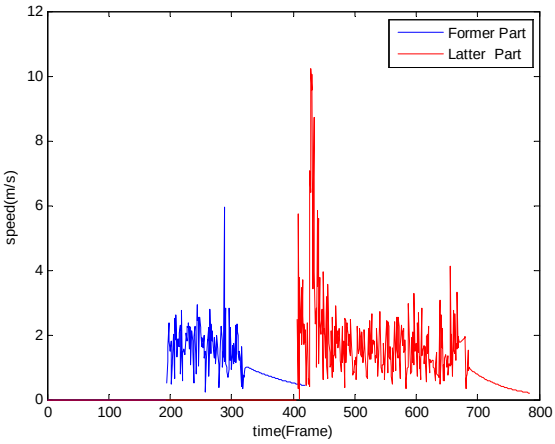


Figure 5.5. Speed of a failed tracking pedestrian

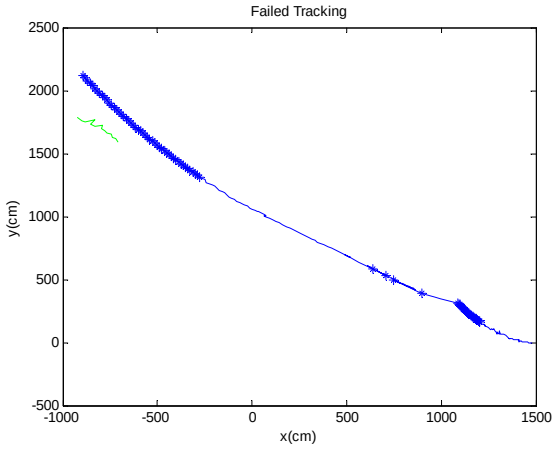


Figure 5.6. Speed of a failed tracking vehicle

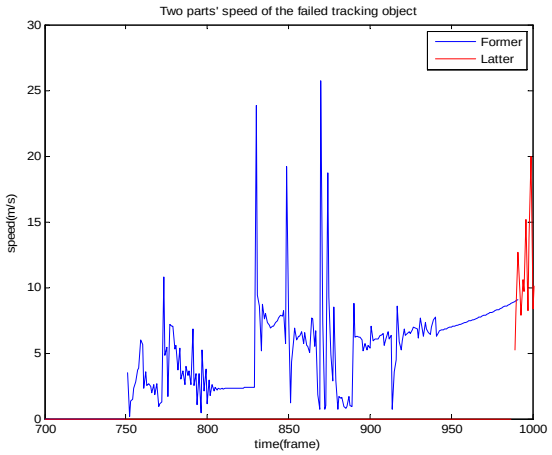


Figure 5.7. Speed of a failed tracking vehicle

Chapter 6

Conclusion

In the past decades, detection and tracking multiple objects has been studied extensively and a large number of applications have been involved. Because LIDAR provides accurate distance information, many researchers introduced it in multi-object tracking systems in recent ten years and many techniques have been implemented in them. In multi-object tracking systems, motion model is one of the crucial factors because it will affect the performance of the data filtering and the prediction when valid observations are missing.

In this thesis, we have shown that besides applying one specific model or combining several possible model candidates, the dynamic model identified by recent history observations can address this problem fairly well. As evaluated offline by simulated data and the real data collected in urban scenes, the dynamic model based multi-object tracking system could track and predict objects with simple motion dynamic very accurately, e.g. constant velocity, and predict the states of the objects with complex motion dynamics with acceptable error. As illustrated by the results, third order dynamic model is good enough for most situations and the performance is much better than the specific constant velocity model.

Future work includes incorporating other advanced data association algorithm, e.g. JPDA or MHT to deal with very crowded scenes, introducing geometric model to resolve the partial occluded problem and applying a technique to update the dynamic model based on the error of the old model.

Bibliography

[1] S. Blackman and R. Popoli, "Design and Analysis of Modern Tracking Systems," Artech House, 1999

[2] L. Brown, "A Radar History of World War II: Technical and Military Imperatives," Institute of Physics, Philadelphia, 1999

[3] J. W. Taylor and G. Bronins, "Design of a New Airport Surveillance Radar (ASR-9)," *IEEE Proc*, 73, 1985, pp. 284–189

[4] D. Atlas, "Radar in meteorology," American Meteorological Society, Boston, MA, USA, 1990

[5] Y. Bar-Shalom and T.E. Fortman, "Tracking and Data Association," Academic Press, 1988

[6] M. Montemerlo, S. Thrun, and W. Whittaker, "Conditional particle filters for simultaneous mobile robot localization and people-tracking," *IEEE International Conference on Robotics and Automation*, Vol. 1, 2002, pp. 695–701

[7] C. C. Wang, "Simultaneous localization, mapping and moving object tracking," PhD thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Apr. 2004

[8] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robotics and Autonomous System*, Vol. 32, 2000, pp. 1–16

[9] D. A. Pomerleau, "Neural network perception for mobile robot guidance," PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1992

[10] F. Dellaert, D. Pomerleau, and C. Thorpe, "Model-based car tracking integrated with a road-follower," *IEEE International Conference on Robotics and Automation*, Vol. 3, 1998, pp. 1889-1894

[11] D. Koller, K. Daniilidis, and H. H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *International Journal of Computer Vision*, Vol. 10, No. 3, 1993, pp. 257-281

[12] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell, "Towards robust automatic traffic scene analysis in real-time," *12th IAPR International Conference on Pattern Recognition*, Vol. 1, 1994, pp. 126–131

[13] L. Davis, V. Philomin, and R. Duraiswami, "Tracking humans from a moving platform," *15th International Conference on Pattern Recognition*, Vol. 4, 2000, pp. 171–178

[14] M. Lindstrom and J. O. Eklundh, "Detecting and tracking moving objects from a mobile platform using a laser range scanner," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawaii, USA, 2001

[15] C.G. Bachman, "Laser radar systems and techniques," Norwood, MA: Artech House, 1979

[16] P. Jensfelt and H. I. Christensen, "Laser based position acquisition and tracking in an indoor environment," *International Symposium on Robotics and Automation*, vol. 1, 1998

[17] J. Gonzalez and R. Gutierrez, "Mobile robot motion estimation from a range scan sequence," *IEEE International Conference on Robotics and Automation*, Albuquerque, New Mexico, Apr. 1997

[18] E. B. Meier and F. Ade, "Object detection and tracking in range image sequences by separation of image features," *IEEE International Conference on Intelligent Vehicles*, 1998

[19] K. Sobottka and H. Bunke, "Vision-based driver assistance using range imagery," *IEEE International Conference on Intelligent Vehicles*, 1998

[20] D. Langer and C. Thorpe, "Range sensor based outdoor vehicle navigation, collision avoidance and parallel parking," *Autonomous Robots*, Vol. 2, 1995, pp. 147–161

[21] J. A. Hancock, "Laser intensity-based obstacle detection and tracking," PhD thesis, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1999

[22] L. Zhao and C. Thorpe, "Qualitative and 1uantitative car tracking from a range image sequence," *CVPR*, Santa Barbara, CA, June 1998, pp. 496–501

[23] M. Piccardi, "Background subtraction techniques: A review," *IEEE International Conference on Systems, Man and Cybernetics*, 2004, pp. 3099–3104

[24] A. Fod, A. Howard, and M. J. Mataric, "Laser-based people tracking," *IEEE International Conference on Robotics and Automation*, Washington DC, May, 2002, pp. 3024–3029

[25] A. Elfes, "Occupancy grids : a probabilistic framework for robot percpetion and navigation," PhD thesis, Carnegie Mellon University, 1989

[26] D. Streller, K. Furstenberg, and K. Dietmayer, "Vehicle and object models for robust tracking in traffic scenes using laser range images," *IEEE 5th International Conference on Intelligent Transportation System*, Singapore, sept. 2002

[27] C. C. Wang and C. Thorpe, "Simultaneous localization and mapping with detection and tracking of moving objects," *IEEE international Conference on Robotics and Automation*, 2002

[28] C. C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas," *IEEE International Conference on Robotics and Automation*, Vol. 1, Sept. 2003, pp. 842–849

[29] J. Sparbert, K. Dietmayer, and D. Streller, "Lane detection and street type classification using laser range images," *IEEE Intelligent Transportation System Conference*, Oakland, CA, USA, Aug. 2001

[30] A. Mendes, L. C. Bento, and U. Nunes, "Multi-target detection and tracking with a laser-scanner," *IEEE Intelligent Vehicles Symposium*, Parma, Italy, June 2004

[31] M. Ester, H. Kriegel, J. Sander, X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," proc. *2nd Int. Conf. on Knowledge Discovery and Data Mining*, Portland, OR, USA, 1996, pp. 226

[32] M. Daszykowski, B. Walczak, D. L. Massart, "Looking for Natural Patterns in Data. Part 1: Density Based Approach," *Chemom. Intell. Lab. Syst.* 56, 2001, pp. 83–92

[33] R. Maclachlan, "Tracking moving objects from a moving vehicle using a laser scanner," Robot. Inst., Canegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-07, Jun. 2005

[34] C. Premebida and U. Nunes, "A multi-target tracking and GMM-classifier for intelligent vehicles," *IEEE Intelligent Transportation Systems Conference*, Toronto, Canada, Sept. 2006

[35] T. D. Vu, O. Aycard, and N. Appenrodt, "Online localization and mapping with moving object tracking in dynamic outdoor environments," *IEEE Intelligent Vehicles Symposium*, Istanbul, Turkey, June 2007

[36] F. Nashashibi and A. Bargeton, "Laser-based vehicles tracking and classification using occlusion reasoning and confidence estimation," *IEEE Intelligent Vehicles Symposium*, Eindhoven, The Netherlands, June 2008

[37] T. Deselaers, D. Keysers, R. Paredes, E. Vidal, and H. Ney, "Local representations for multi-object recognition," *DAGM 2003, Pattern Recognition, 25th DAGM Symp*, pp. 305312, September 2003

[38] F. Fayad and V. Cherfaoui, "Tracking objects using a laser scanner in driving situation based on modeling target shape," *IEEE Intelligent Vehicle Symposium*, Istanbul, Turkey, June 2007

[39] M. Carms, P. E. Rybski, C. Baker, and C. Urmson, "Obstacle detection and tracking for the urban challenge," *IEEE Transportations on Intelligent Transportation systems*, Vol. 10, No. 3, Sept. 2009

[40] M. Burke, "Laser-based target tracking using principal component descriptors," *21st Annual Symposium of the Pattern Recognition Association of South Africa*, Stellenbosch, South Africa, Nov. 2010, pp. 45–50

[41] M. Hashimoto, T. Konda, Z. Bai and K. Takahashi, "Laser-based tracking of randomly moving people in crowded environments," *IEEE International Conference on Automation and Logistics*, Hong Kong and Macau, Aug. 2010

[42] N. Kaempchen, K. Weiss, M. Schaefer, and K. C.J. Dietmayer, "IMM object tracking for high dynamic driving maneuvers," *IEEE Intelligent Vehicles Symposium*, Parma, Italy, June 2004

[43] S. Thrun, "Particle Filters in Robotics," *The Eighteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, 2002

[44] A. Doucet, J.F.G. de Freitas, and N.J. Gordon, editors, "Sequential Monte Carlo Methods in Practice," Springer, 2001

[45] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers "Tracking multiple moving targets with a mobile robot using particle filters and statistical data association," *IEEE International Conference on Robotics and Automation*, Seoul, Korea, 2001, pp. 1665–1670

[46] O. Frank, J. Nieto, J. Guivant, and S. Scheding, "Multiple target tracking using sequential monte carlo methods and statistical data association," *IEEE International Conference on Intelligent Robots and Systems*, Vol. 3, 2003, pp. 2718–2723

[47] A. Petrovskaya and S. Thrun, "Model based vehicle tracking in urban environments," *IEEE International Conference on Robotics and Automation, Workshop on Safe Navigation*, Vol. 1, 2009, pp. 1–8

[48] A. Doucet, Nd. Freitas, K. Murphy, S. Russell, "Raoblackwellised filtering for dynamic bayesian networks," In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, pp 176183, 2000

[49] S. Sarkka, A. Vehtari, and J. Lampinen, "Rao-Blackwellized particle filter for multiple target tracking," *Information Fusion*, Vol. 8, 2007, pp. 2–15

[50] J. Cui, H. Zha, H. Zhao and R. Shibasaki, "Robust tracking of multiple people in crowds using laser range scanners," *18th International Conference on Pattern Recognition*, Vol. 4, 2006, pp. 857–860

[51] J. Cui, H. Zha, H. Zhao, and R. Shibasaki, "Laser-based detection and tracking of multiple people in crowds," *Computer Vision and Image Understanding*, Vol. 106, No. 2-3, 2007, pp. 300–312

[52] T. Ogawa, H. Sakai, Y. Suzuki, K. Takagi, K. Morikawa, "Pedestrian detection and tracking using in-vehicle lidar for automotive application," *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011

[53] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 19, No. 1, 2004, pp. 5–18

[54] Y. B. Shalom, T. Kirubarajan, and X. Lin, " Probabilistic data association techniques for target tracking with applications to sonar, radar and EO sensors," *IEEE Aerospace and Electronic System Magazine*, Vol. 20, No. 8, Aug. 2005, pp. 37–56

[55] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for general multiple-target tracking problems," *IEEE Conference on Decision and Control*, Atlantis, Paradise Island, Bahamas, Dec. 2004

[56] T. Zhao, R. Nevatia, and B. Wu, "Segmentation and tracking of multiple humans in crowded environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008

[57] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for multi-target tracking," *IEEE Transactions on Automatic Control*, Vol. 54, No. 3, Mar. 2009

[58] R. Karlsson and F. Gustafsson, "Monte carlo data association for multiple target tracking," *Target Tracking: Algorithms and Applications*, Vol. 1, Oct. 2001

[59] T. D. Vu and O. Aycard, "Laser-based detection and tracking moving objects using data-driven markov chain monte carlo," *IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009

[60] E. Prassler, J. Scholz, and A. Elfes, "Tracking people in a railway station during rush-hour," *Computer Vision Systems*, 1999

[61] E. Prassler, J. Scholz, and A. Elfes, "Tracking multiple moving objects for real-time robot navigation," *Autonomous Robots*, Vol. 8, 2000, pp. 105–116

[62] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "People tracking with a mobile robot using sample-based joint probabilistic data association filters," *The International Journal of Robotics Research*, Vol. 22, No. 2, 2003, pp.99–116

[63] C. C. Wang, C. Thorpe, M. Hebert, S. Thrun, and H. D. Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Jounral of Robotics Research*, Vol. 26, No. 9, Sept. 2007, pp.889–916

[64] R. Rosales and S. Sclaroff, "Improved tracking of multiple humans with trajectory prediction and occlusion modeling," *IEEE CVPR Workshop on the Interpretation of Visual Motion*, 1998

[65] B. Kluge, C. Kohler, and E. Prassler, "Fast and robust tracking of multiple moving objects with a laser range finder," *IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001, pp. 21–26

[66] R. A. Maclachlan, "Tracking of moving objects from a moving vehicle using a scanning laser rangefinder," *IEEE Intelligent Transportation Systems Conference*, Toronto, Canada, Sept. 2006

[67] B. G. F. M.Berna, B.Lisien and S.Thrun, "A learning algorithm for localizing people based on wireless signal strength that uses labeled and unlabeled data," in *International Joint Conference on Artificial Intelligence*, 2003

[68] J. H. H. K. L. Liao, D. Fox and D. Schulz, "Voronoi tracking: Location estimation using sparse and noisy sensor data," in *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 2003

[69] M. Bennewitz, W. Burgard, and S. Thrun, "Learning motion patterns of persons for mobile service robots," in *IEEE Conference on Robotics and Automation*, 2002

[70] A. Bruce and G. Gordon, "Better motion prediction for people-tracking,", *International Conference on Robotics and Automation (ICRA)*, New Orleans, USA, Apr. 2004

[71] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: a survey," *IEEE Trans. Aerosp. Electron. Syst.* Vol. 34, 1998, pp.103–123

[72] X. Rong Li and Vesselin P.Jilkov, "A survey of maneuvering target tracking-part v : Multiple-model methods," *IEEE Transactions on Aerospace and Electronic Systems*, 2003

[73] M. Busch and S. Blackman, "Evaluation of imm filtering for an air defence system application," In *Proceedings SPIE Signal Data Process, Small targets*, volume SPIE 2561, pages 435447, 1995

[74] S. Blackman, "Multiple Target Tracking with Radar Applications," Artech House : Dedham, 1986

[75] S. Coraluppi, M. Luettgen, and C. Carthel, "A hybrid-state estimation algorithm for multi-sensor target tracking," *International Conference on Information Fusion*, Paris, France, 2000, pp. 18–23

[76] S. Coraluppi and C. Carthel, "Multiple-hypothesis IMM (MH-IMM) filter for moving and stationary targets," *International Conference on Information Fusion*, Montreal, QC, Canada, 2001, pp. 18–23

[77] W. D. Blair and G. A. Watson, "IMM algorithm and aperiodic data," In *Proc. SPIE Vol. 1697, p. 83-91, Acquisition, Tracking, and Pointing VI, Michael K. Masten ; Larry A. Stockum ; Eds.*, pages 8391, November 1992.

[78] O.Drummond M.Miller and A. Perrella, "Multiple-Model Filters for Boost-to-Coast Transition of Theater Ballistic Missiles," In *Proceedings SPIE Signal Data Process. Small targets*, volume SPIE 2561, pages 355376, 1998

[79] Zoubin Ghahramani and Geoffrey E. Hinton, "Variational learning for switching state-space models," *Neural Computation*, 12(4) :831864, 2000

[80] J. Burlet, O. Aycard, A. Spalanzani, and C. Laugier, "Adaptive interactive multiple models applied on pedestrian tracking in car parks," In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, China, 2006.

[81] O. Aycard, "Contribution to perception for intelligent vehicles," PhD thesis, Universite de Grenoble, France, 2010

[82] R.E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, 35, Mar 1960

[83] Hwasup Lim, "Dynamic motion and appearance modeling for robust visual tracking, " PhD thesis, The Pennsylvania State University, 2007

[84] M. Sznaier and O. Camps, "The role of dynamics in computer vision and image processing," In *IEEE Conference on Decision and Control*, pages 39233934. IEEE, Dec. 2008.