# HOW MULTI-FRAME MODEL FITTING AND DIFFERENTIAL MEASUREMENTS CAN IMPROVE LIDAR-BASED VEHICLE TRACKING ACCURACY

By

Steven J. Chao

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering – Master of Science

2015

# ABSTRACT

HOW MULTI-FRAME MODEL FITTING AND DIFFERENTIAL MEASUREMENTS
CAN IMPROVE LIDAR-BASED VEHICLE TRACKING ACCURACY

By

Steven J. Chao

Over the past few decades, much work has been done in the field of laser based (Lidar) vehicle tracking. The most common approach is to fit a simple rectangular model to a point cloud of vehicle data, and then process the measurements using an Extended Kalman Filter. In this study we explore the use of techniques which could improve tracking performance in situations with sparse data, system noise or clutter, difficult vehicle orientations, or poorly modeled vehicle shapes.

We specifically consider: Multi-Frame Measurements and Differential Measurements. Multiple-Frame Model Fitting can improve tracking accuracy through batch processing of multiple frames rather than the usual single-frame processing. Differential Measurement tracking avoids estimating a vehicle's current pose and instead estimates change in pose between subsequent frames by comparing point clouds, and has the advantage that it does not require a prior vehicle shape model. Since our goal is to focus specifically on these types of vehicle measurement, we ignore other important tasks such as background removal, object detection and classification, vehicle occlusion, and real-time speed considerations. We ultimately show that each of these measurement techniques have different, complementary strengths, which can be combined to improve vehicle tracking performance in difficult situations.

# ACKNOWLEDGEMENTS

I would like to thank: my graduate research advisor Dr. Daniel Morris for encouraging and mentoring me within the field of Vehicle Tracking, Dr. Xiaobo Tan and Dr. Hassan Khalil for serving on my defense committee and providing their experienced feedback, my fellow researchers at the Michigan State University Vision Laboratory for their collaboration and help, and my friends and family for their continued support, especially my parents, Dave and Annette.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# KEY TO ABBREVIATIONS

| | |
|---|---|
| EKF: | Extended Kalman Filter |
| GPS: | Global Positioning System |
| ICP: | Iterative Closest Point |
| MB: | Model-Based |
| MBT: | Model-Based Tracking |
| MF: | Multi-Frame |
| MFMB: | Multiple-Frame Model-Based |
| SF: | Single-Frame |
| SFMB: | Single-Frame Model-Based |
| VASM: | Variable-axis Ackerman Steering Model |

# INTRODUCTION

Over the past few decades, there has been much research done in the field of Autonomous Vehicles. Many large automotive corporations, smaller research groups, and universities have devoted significant time and money into the advancement of this field, and self-driving cars are soon to be reality. While this technology faces many huge hurdles, it will likewise provide numerous benefits, many of which are immediately apparent, but also some which could never have been expected. For example, autonomous vehicles will be able to prevent accidents caused by drunk or distracted drivers, recover countless hours squandered in daily commutes, improve traffic flow and efficiency, and save lives by removing human error. On a broader scale, autonomous driving will redefine how we as a society commute, likely creating autonomous taxis, and provide other benefits currently unimagined.

While autonomous vehicles could provide many benefits to society, creating this technology has many challenges. These include detecting and classifying environmental objects, predicting where they will be in the future, understanding road signs and traffic laws, and making numerous decisions independently and in real time.

In order to detect, classify, and track objects, sensors are needed to measure the environment. There has been significant work done researching sensors such as sonar, radar, and GPS, but two of the most commonly used types are cameras and Lidar (laser) scanners. For the task of classifying what an unknown object may be, the most effective sensors are cameras. Computer Vision is a large field of research that explores techniques for processing camera

images, and classifying objects within them. However, camera-based-methods have a very difficult time determining the precise location of an object, and from that extracting the velocity of the target. A common solution is to use a combination of cameras and laser scanners within object tracking; image processing is used to initially classify objects, and the laser scanner is then used to track the known objects' position over time.

For this study, we focus specifically on the issue of using laser scanners to track and predict vehicle movement over time. We ignore other important fields such as background removal, object detection and classification, non-vehicle object tracking, and real-time computational constraints, in order to study this topic in detail.

State of the art research in Lidar-based vehicle measurement focuses heavily on improving the Extended Kalman Filter or Particle Filter framework, which tracks vehicle movement over time, based on individual vehicle measurements. A large part of this involves modeling the uncertainty of possible vehicle poses and selecting the most likely one. This research compliments the ideas presented within of this paper, which are to fundamentally improve the vehicle measurements, which are then fed into these trackers.

The laser scanner we use is a Velodyne HDL-32E Lidar sensor. This Lidar operates by using 32 lasers arranged vertically at small, constant, angular intervals, mounted on a rotating platform. The laser array spins at a variable rate of around ten Hertz, and returns over 700,000 depth points per second, effectively creating a dense environmental point cloud map over a 360 degree field of view. Examples of Lidar point cloud data are shown in the images below. In the following chapters, we will describe and evaluate two different measurement techniques to estimate the position of a vehicle, based on the available point cloud data.

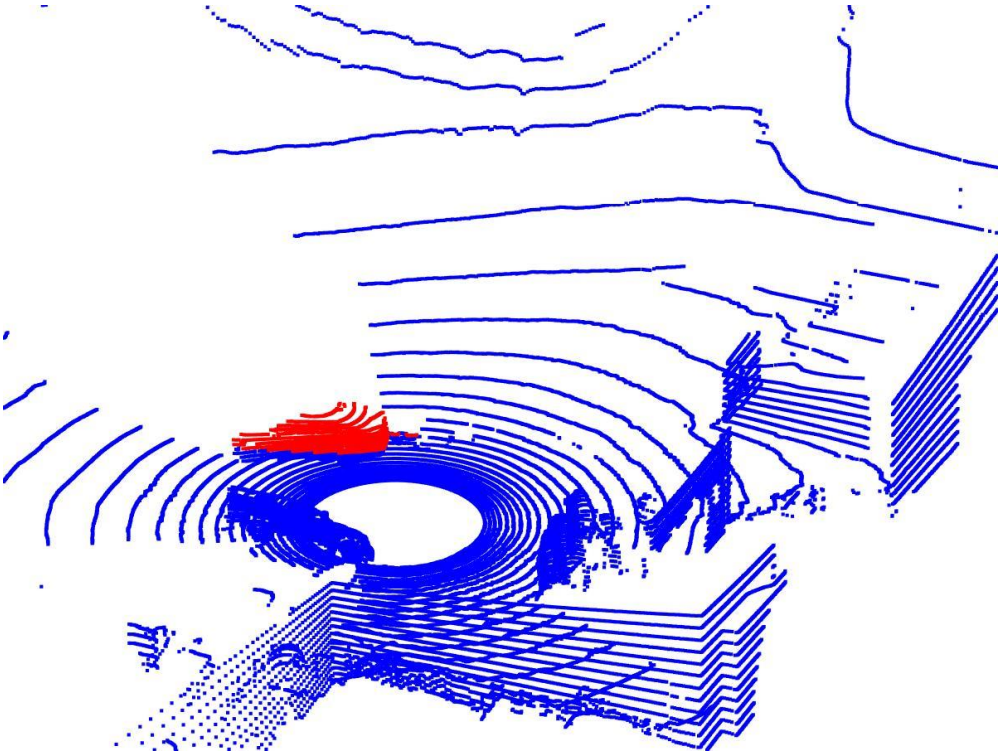

**Figure 1**: *Image*: The surrounding environment, as seen through a Lidar scanner, including ground, a building's walls, and a target vehicle (in red)

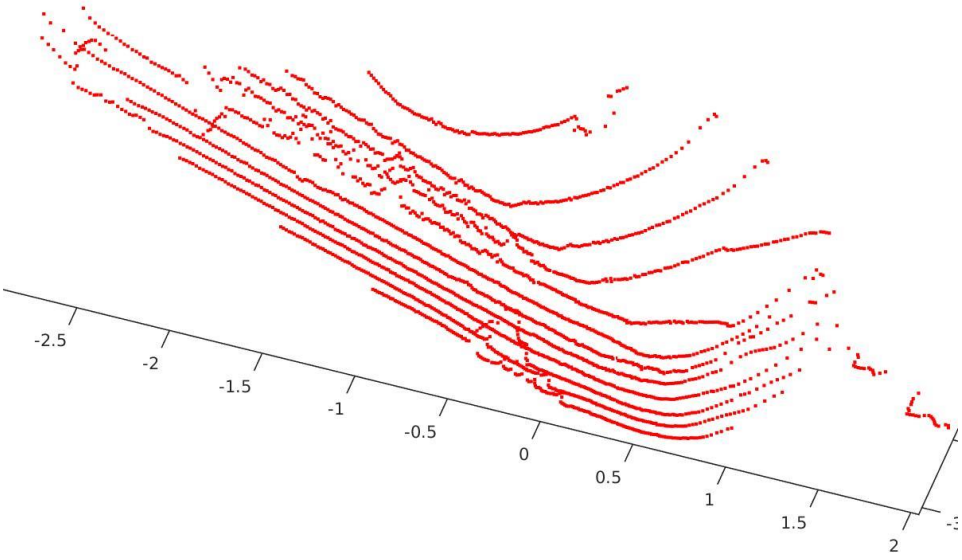


**Figure 2**: *Image*: The point cloud of a vehicle returned by a Lidar scanner

# CHAPTER 1

# VEHICLE TRACKING FRAMEWORK

## Background

Over the past few decades, multiple different approaches to vehicle tracking have been developed, most with a similar overall architecture. For example, past research ( (Zhao & Thorpe, 1999), (Streller, Furstenberg, & Dietmayer, 2002), (Wang C. , 2004), (Morris, Colonna, & Haley, 2006), (Wender & Dietmayer, 2008), (Petrovskaya & Thrun, 2009), (Morris, Hoffman, & Haley, 2009)) has generally resulted in a three stage tracker approach: data segmentation, data association, and Bayesian filter update. During the data segmentation stage, sensor data is divided into meaningful pieces, which are often represented as line features or point clusters. During the data association stage, these pieces are assigned to currently tracked vehicle models. Finally a recursive Bayesian filter, generally an extended Kalman filter, is utilized to fit the targets to the data.

The overall vehicle tracking framework is as follows. At every time step, the extended Kalman filter (EKF) predicts the expected vehicle location, based on some kinematic vehicle model of the previous state. The actual location of the vehicle is also measured, by optimizing the point cloud data to fit a rectangular vehicle model, and fed into the filter. By using the measurement and prediction step's associated covariance matrices, the EKF combines the two estimates, weighted according to their uncertainties. The desired operation is for the tracker's

estimated vehicle position to initially follow the Model-Based measurements, but eventually more closely follow the kinematic model predictions as the tracker improves in certainty. This makes the tracker less susceptible to small measurement noises while also improving the accuracy of the vehicle track over time. The components needed to build this tracker therefore include a model-fitting measurement component, the extended Kalman filter, and a kinematic model to predict vehicle motion.

## Vehicle Location Measurement

The first component of vehicle tracking is Model-Based vehicle location measurement. Once vehicles have been detected and segmented, their 2D position and orientation need to be estimated using the visible point cloud data. This is more difficult than it sounds, because with a Lidar sensor it is only possible to see two edges of a vehicle at a given instant, and because there exist many diverse vehicles of very different shapes. This challenge is frequently solved by optimizing a rectangular model to the point cloud cluster ( (Zhao & Thorpe, 1999); (Wender & Dietmayer, 2008)). Most vehicles are effectively represented by a rectangular model with the dynamic variables width and length (W, L). Matching a general vehicle model to the point cloud data is a very powerful tool, because it is able to reliably estimate the position and orientation of the entire vehicle using only the visible portion of vehicle point cloud data.

Generally, optimizing this dynamic rectangular model is a very accurate measurement technique, even for vehicles of diverse shapes. This is extremely important, because such a descriptive yet simple model allows us to quickly and robustly estimate a vehicle's position at

any given moment. Since Lidar-based vehicle tracking involves processing hundreds of thousands of data points per second, utilizing this fast, consistent, and accurate rectangular model to estimate the vehicle's location is a necessity.

## Extended Kalman Filter Framework

Next, the location measurements found by fitting the rectangular vehicle model are fed into an extended Kalman filter (EKF). The EKF uses kinematic predictive equations to estimate the vehicle's current location using the previous state vector. Using the equations shown in Appendix A, the EKF estimates a weighted combination of the measurement and model prediction steps to get the final tracker solution.

In Figure 3, we can see all three of these EKF components, compared over time. We recall that the overall model tracker is a combination of both the measurement and prediction inputs, weighted based on the covariance of each. It can be seen that initially the tracker more closely follows the Model-Based measurement input, but as time passes and the tracker is refined, it begins to follow the kinematic model's predictions. This results in a smoother trajectory, thanks to the kinematic model predictions, that is still very accurate due to the individual measurements component. Notice that the large measurement noise spike occurring at about 3.5s has very little effect on the overall EKF estimate, which is one of the major benefits of this filter.

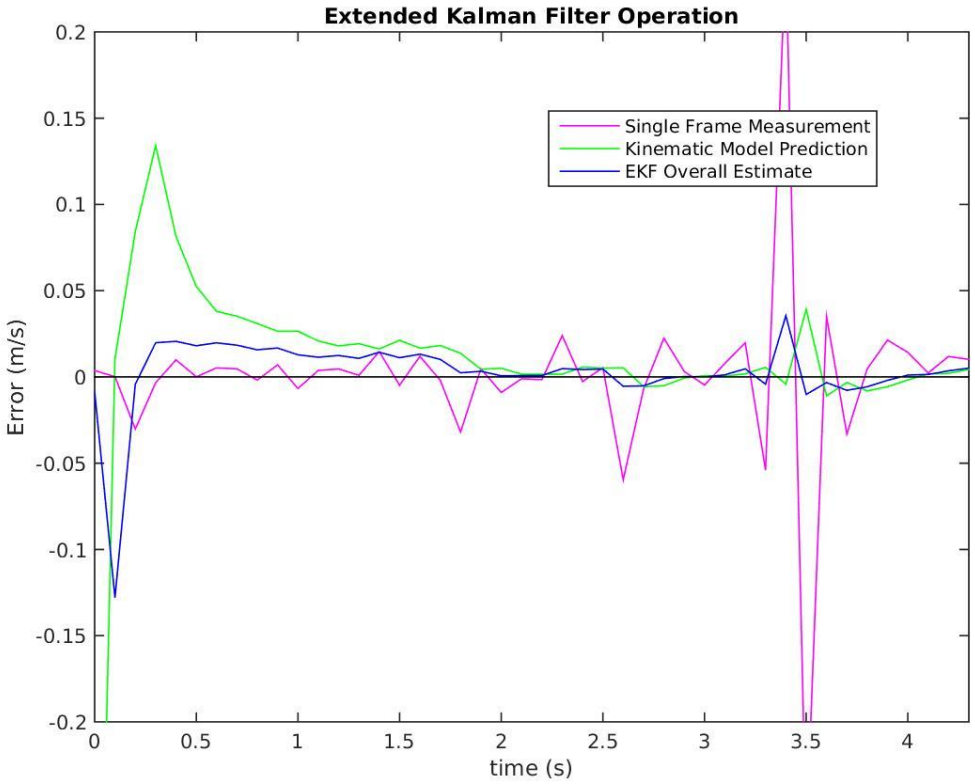


**Figure 3**: *Plot*: A comparison of the performance of the three components of the EKF over time

## Single-Frame Model-Based Measurement Limitations

This is the general flow of a vehicle tracking algorithm. Individual measurements of a vehicle's location are collected and fed into an EKF for processing. While this method does track vehicles quite well in most cases, there are certain disadvantages to using a generic, rectangular model. Specifically, grossly incorrect measurements can occur when the vehicle is matched to the vehicle point cloud incorrectly, such as when attempting to measure vehicles of non-rectangular shapes, when the vehicle data is sparse, or in the presence of clutter, to name just a

few examples. The EKF filter is capable of smoothing out small measurement errors and effectively ignoring them, but large measurement errors are a different story. In many cases, even a single, grossly incorrect measurement is enough to cause the tracker to completely lose the true vehicle trajectory.

While these incorrect measurements do not happen very often, they are enough of a problem to warrant significant further investigation. If an autonomous vehicle completely lost the track of even a single oncoming vehicle, it could cause a very serious accident. The current, widely used approach to deal with this issue is to couple the EKF with a Particle Filter. Particle Filters sample the probability space of the vehicle state vector, rather than just modeling the most likely region as an EKF does. Since they are capable of modeling multi-modal distributions, individual incorrect measurements can eventually be filtered out, after they have been shown to diverge drastically with the true vehicle point cloud data and additional measurement inputs. This has proven to be quite effective in filtering out individual grossly inaccurate measurements. However, challenges using particle filters include modeling higher dimensional spaces, as well as spaces whose probability is spread out widely, since in both cases many samples are needed.

Significantly less research has been done approaching this problem from the other end, and attempting to improve the actual vehicle measurement process. In this study we explored two additional methods with the potential to compensate for the gross measurement errors occasionally returned using Single-Frame Model-Based measurement techniques. First we investigated how optimizing the measurements over multiple frames of data could be used to improve performance. Second, we studied how Model-Free Differential Measurements could be used in combination with Model-Based methods to improve measurement performance.

# CHAPTER 2

# SINGLE-FRAME VEHICLE MEASUREMENT

In the first chapter, we discussed the general framework for vehicle tracking. The bulk of our work in this study was focused on improving the measurement portion of this process, by comparing typical Single-Frame Model-Based (SFMB) measurements with our Multi-Frame Model-Based (MFMB) and Differential measurements.

We began by building a simple SFMB measurement technique. We first had to develop a rectangular vehicle model to fit the vehicle points to, as done by (Morris, Colonna, & Haley, 2006). In this study, we used a modified form of this technique that relies on 1D filters to improve computation speed. The next step was to build a measurement filter to estimate the vehicle's position without any prior knowledge about the vehicle's current pose, using this rectangular vehicle model. Once a rough estimate of the vehicle's location was known, a more precise model optimization technique was needed to refine the pose measurement. This gave us the finalized SFMB measurement, which could be fed into the EKF, or compared directly with the measurements of the two other techniques.

## Vehicle Location Estimation

Building the location estimation component was a challenging part of SFMB measurement process, due to the lack of any prior knowledge about the vehicle's position. In order to quickly and accurately estimate the vehicle's position, we took advantage of a simple fact: when the vehicle's points are projected to a line perpendicular to an edge of the vehicle, we expect a unique distribution of points. There should be a small uniform distribution of points over the width of the vehicle, except for a dense cluster where the edge of the vehicle projects to, as shown in Figure 4. Points projected to a line that isn't perpendicular to a vehicle edge would instead return a more uniform distribution, as shown in Figure 5.

To utilize this knowledge, we created a cost function model that accurately represented this expected perpendicular point distribution. Next, we rotated a series of 45 lines over the vehicle point cloud in a 180 degree span (4° resolution), and projected the vehicle points to each line. Finally, we convolved the cost function model with each of these vehicle point projections. This allowed us to determine which line projections best fit our expected perpendicular distribution, giving us both the vehicle's orientation and also the location of the vehicle's edges. This is similar to the 2D model fitting described in (Morris, Colonna, & Haley, 2006), but this novel 1D variation is extremely useful due to its quicker speed and reliable measurement accuracy.

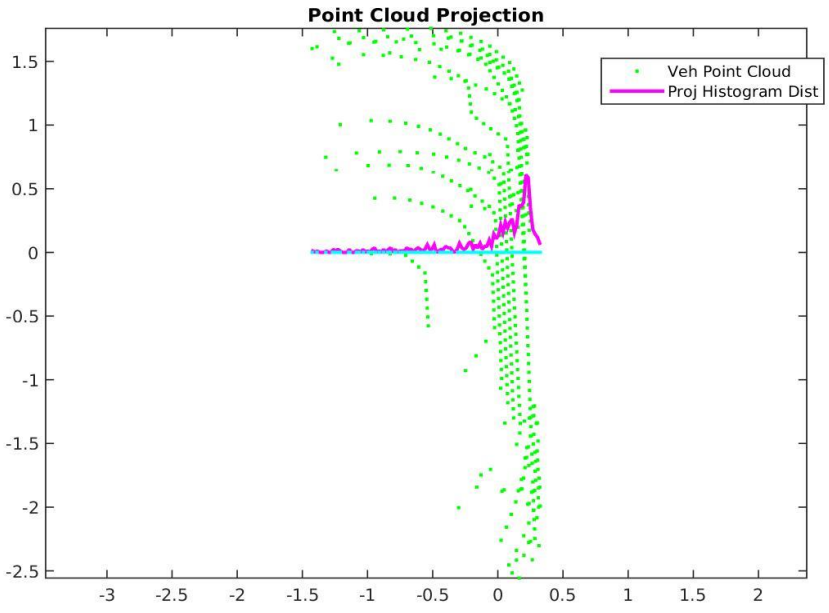


**Figure 4**: *Image*: The distribution of the points (shown by the pink histogram) projected to this line, which is perpendicular to an edge of the vehicle

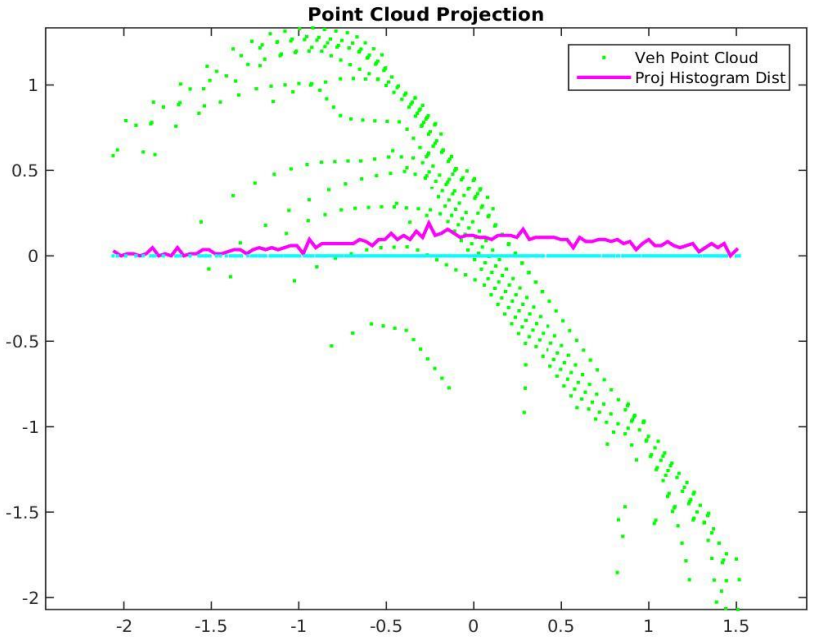


**Figure 5**: *Image*: The distribution of the points (shown by the pink histogram) projected to this line, which is *not* perpendicular to an edge of the vehicle

## Cost Function Optimization

The cost function kernel designed to model perpendicular distributions is shown in Figure 6. It can be seen that there is a small, uniform, negative cost over the entire width of the vehicle, which is where we expect most of the vehicle's non-edge points to project to. Near the expected vehicle edges is a large, parabolic, negative cost. This does two things: it estimates the location of the vehicle's edges, and it also picks which line is most perpendicular to the vehicle, in order to estimate the vehicle's orientation. Finally, outside the edges of the vehicle is a large, positive cost, since we don't expect any vehicle points outside this. In the current cost function implementation, the vehicle length and width are user-specified parameters, but in the future it could be made into a dynamic variable.

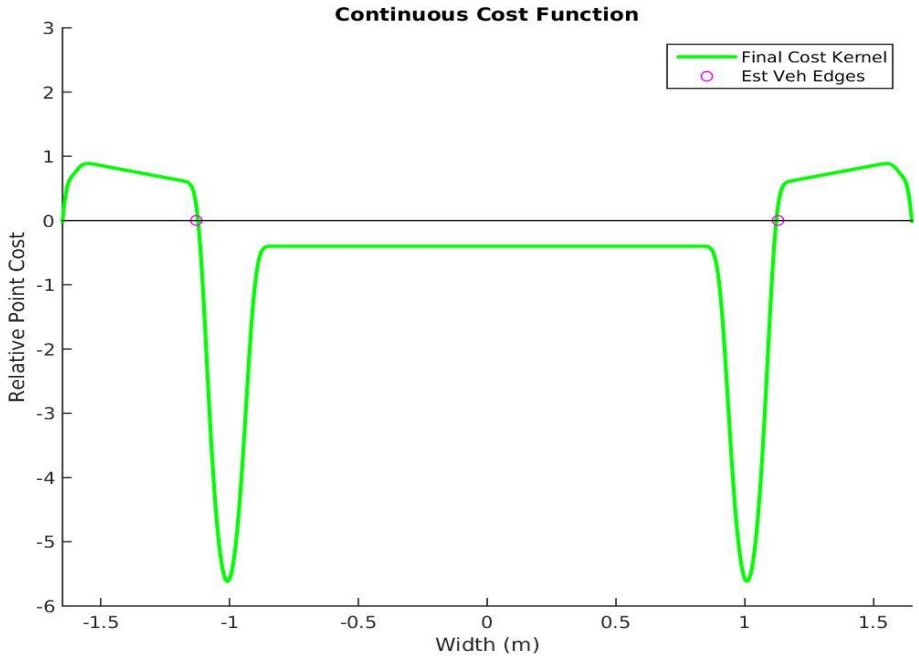


**Figure 6**: *Diagram*: The cost function kernel, designed to model the expected distribution of the perpendicular point cloud projections

For each projected line (Equation 1), the points are binned to reduce computation time and discretely convolved (Equation 2) with a likewise discrete version of this cost function kernel over intervals (***n***, ***m***). This is done twice, convolving with a kernel of both parameters (***w***, ***L***). Theta ($\boldsymbol{\theta}$) represents the relative direction of each projection. By picking the two perpendicular projections with the minimum combined cost, we are able to both fit the vehicle edges to the point cloud data, and also determine the direction of vehicle. The final equation is given in Equation 3. This is a robust estimation technique, capable of consistently and quickly returning an approximate estimate of vehicle pose with no prior location knowledge. An image of the final vehicle location estimation is shown below in Figure 7.

Projection of Vehicle Points (***V***):

$$\boldsymbol{V_p} \;=\; \cos(\boldsymbol{\theta}) \cdot |\boldsymbol{V}| \qquad \textit{Eq. 1}$$

Discrete Convolution:

$$(f * g)[\boldsymbol{n}] \;\stackrel{\text{def}}{=}\; \sum_{\boldsymbol{m}=-\infty}^{\infty} f[\boldsymbol{m}]\, g[\boldsymbol{n}-\boldsymbol{m}] \qquad \textit{Eq. 2}$$

Overall SFMB Cost Function:

$$[\boldsymbol{\theta}, \boldsymbol{n}, \boldsymbol{m}] = \min_{\boldsymbol{\theta} \,\epsilon\, [0°, 180°)} \Big( \min_{\boldsymbol{n}} \left(kernel(\boldsymbol{w}) * \boldsymbol{V_p}\right)[\boldsymbol{n}] + \min_{\boldsymbol{m}} \left(kernel(\boldsymbol{L}) * \boldsymbol{V_p}\right)[\boldsymbol{m}] \Big) \qquad \textit{Eq. 3}$$

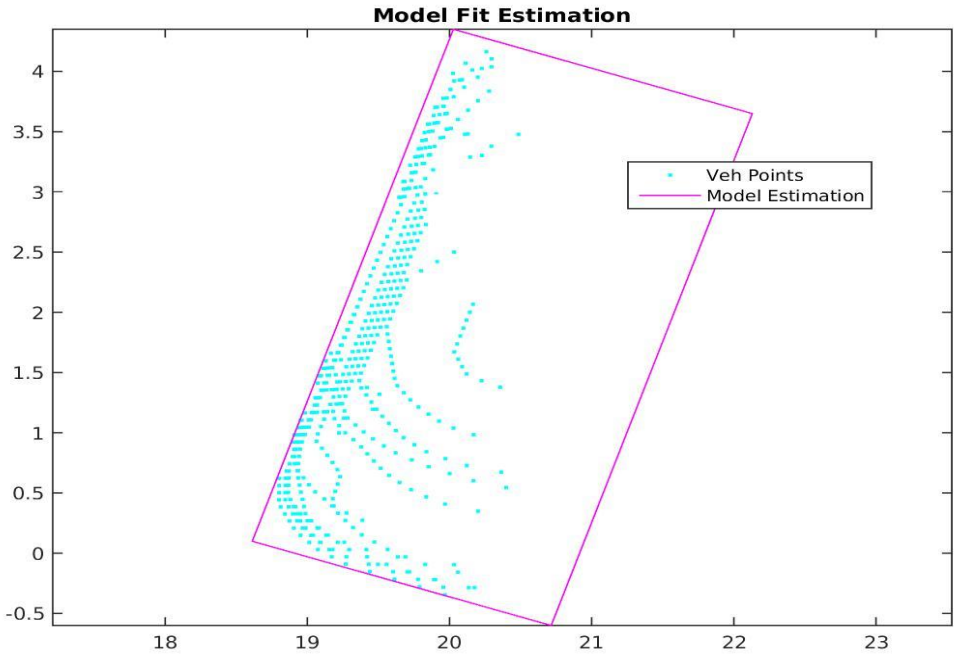


**Figure 7**: *Image*: An example of the cost function estimating a rectangular model to vehicle data

## Vehicle Location Refinement

Once we have an initial estimate of the vehicle's pose, we can refine the estimate using nonlinear optimization. For this we use MATLAB's nonlinear minimization functions (see Appendix B). Matlab's optimization of a continuous cost function (Equation 15) is able to return much more precise measurements than the estimation (4° resolution) returned by our discrete technique. This more powerful local optimization technique is run second because it needs an initial pose estimate to avoid local minima and correctly optimize the measurement.

This summarizes the process of our Single-Frame Model-Based measurement technique, the performance of which will be described in Chapter 6, Results.

# CHAPTER 3

# MULTIPLE-FRAME VEHICLE MEASUREMENT

## Objectives of Multi-Frame Measurement

One potential solution to correct severe Model-Based Tracking errors is optimizing the best fit trajectory over multiple frames. The idea behind this is that gross measurement errors often only occur sporadically, typically caused by insufficient data in a single frame of time. If a measurement technique optimizes the best fit over multiple frames, the fitting process should be more robust and less susceptible to bad measurements in individual frames. Additionally, a nonlinear optimization over multiple frames may be able to give more precise pose measurements than just a single frame measurement.

In order to optimally match rectangular models to multiple frames of previous data, we needed to define certain constraints. For our implementation, we assumed a constant speed and angular velocity between each previous set of data. This is actually a reasonable assumption over short periods of time for true vehicle movement. In the future, different constraints could be used, such as constant acceleration assumptions, or even higher order state variable optimizations, such as dynamic velocity and acceleration. However, for this work, we optimize the measurements using a simple constant velocities constraint, in order to determine the potential of further exploration into Multi-Frame Model-Based (MFMB) measurement techniques. We now need to discuss how predictive kinematic models can be

used to estimate future vehicle movement by using the current state vector variables.

## Predictive Kinematic Models

A kinematic model is a method to predict the future state of a vehicle (which represents its future trajectory) based on the current state variables. These models are needed for two reasons: first, they are used by the EKF to predict future vehicle motion, and secondly are used here by our MFMB measurement technique, in order to constrain the vehicle measurement optimizations. We model the vehicle's 2D position using three state variables, the center-point of the vehicle (x, y) and the orientation of the vehicle (θ). Next, the constant velocity assumptions are represented by the two constant state variables, speed (s), and angular velocity ($\dot{\theta}$). The next component is to define how these five variables (x, y, θ, s, $\dot{\theta}$) predict their next state within each kinematic model.

In the most basic model, the Center Point Steering Model, it is assumed that the vehicle rotates about the center point of the vehicle. In this model, the next state is simply predicted by multiplying the estimated speed and angular velocity by the change in time, and adding them to the current location of the vehicle. This is a simple kinematic model and generally real vehicles do not behave in this manner. A more realistic model is called the Ackerman Steering Model, which assumes the vehicle actually rotates around a fixed back axle, which is defined by a user estimated parameter ($L_a$) that represents the distance from the vehicle center point to the rear axle. In this case, the state vector will be identical (x, y, θ, s, $\dot{\theta}$), but the predictive equations will be adjusted account for the new point of rotation ($L_a$). This model is significantly

more realistic than the Center-Point Steering model. Finally, we implemented a Variable-axis Ackerman Steering Model (VASM), which is useful in situations where the location of the vehicle's back axle is unknown. In this kinematic model, the predictive equations are identical to the Ackerman Model, except that the location of the back axle is now a continuously refined variable within the state vector ($x, y, \theta, s, \dot{\theta}, L_a$) (Morris, Haley, Zachar, & McLean, 2008). With each of these models, which are depicted in Figure 8, as the complexity of the equations increase, so too does the overall predictive power of the model; in our studies we therefore use the best type of model for each situation.

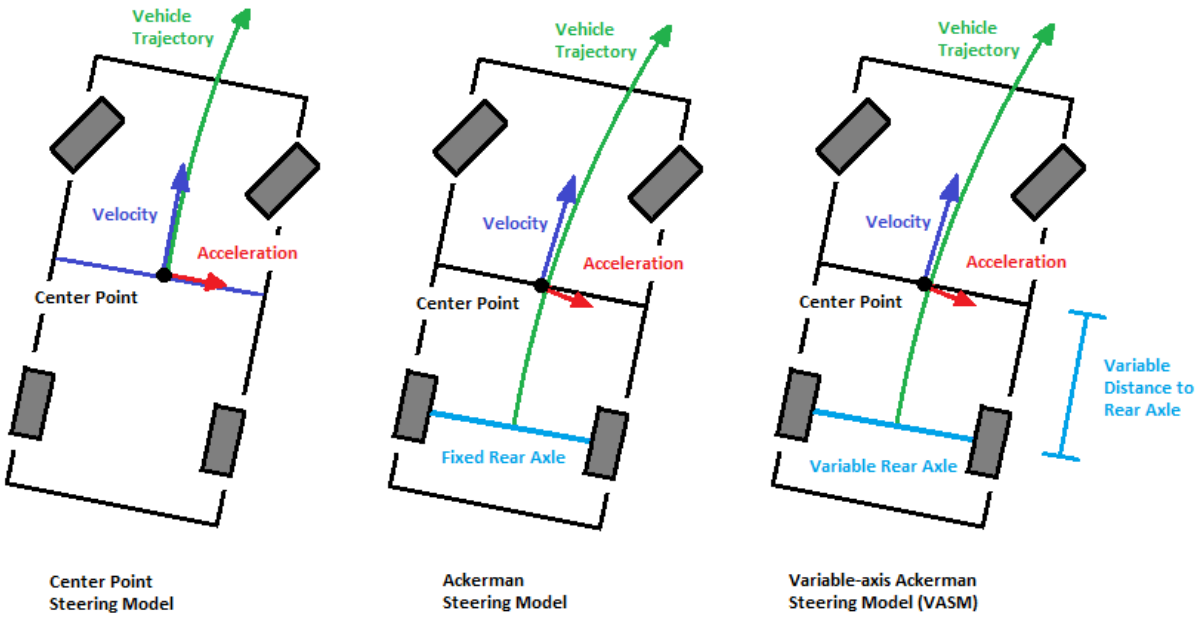


**Figure 8**: *Diagram*: Depiction of how the kinematic models predict vehicle trajectory

## Multi-Frame Model-Based Measurement Technique

In order to build this measurement technique, we incorporated many of the measurement techniques from the single-frame tracking algorithm. First, a variable number of previous frames are chosen to optimize the measurement model fits over. Next, the known poses of the two previous vehicle measurements are utilized to estimate the initial values of the state vector $(x, y, \theta, s, \dot{\theta})$. Then by maintaining our constant speed and angular velocity assumptions, we predict where the other vehicle models will be, by using the equations from the kinematic vehicle models. We utilize the same rectangular vehicle model as discussed in the Single-Frame measurement chapter. Finally, we use a nonlinear MATLAB minimization function (see Appendix B) to best optimize the fit of each model cost function to its respective point cloud, while maintaining constant velocities between consecutive frames of data. This returns the optimized current pose, as well as the estimated speed and angular velocity between each of the frames in the optimization. An example of this optimization is shown below in Figure 9. This final MFMB measurement can be fed into an EKF for overall vehicle tracking. The performance of this measurement technique will be discussed thoroughly in Chapter 6, Results.

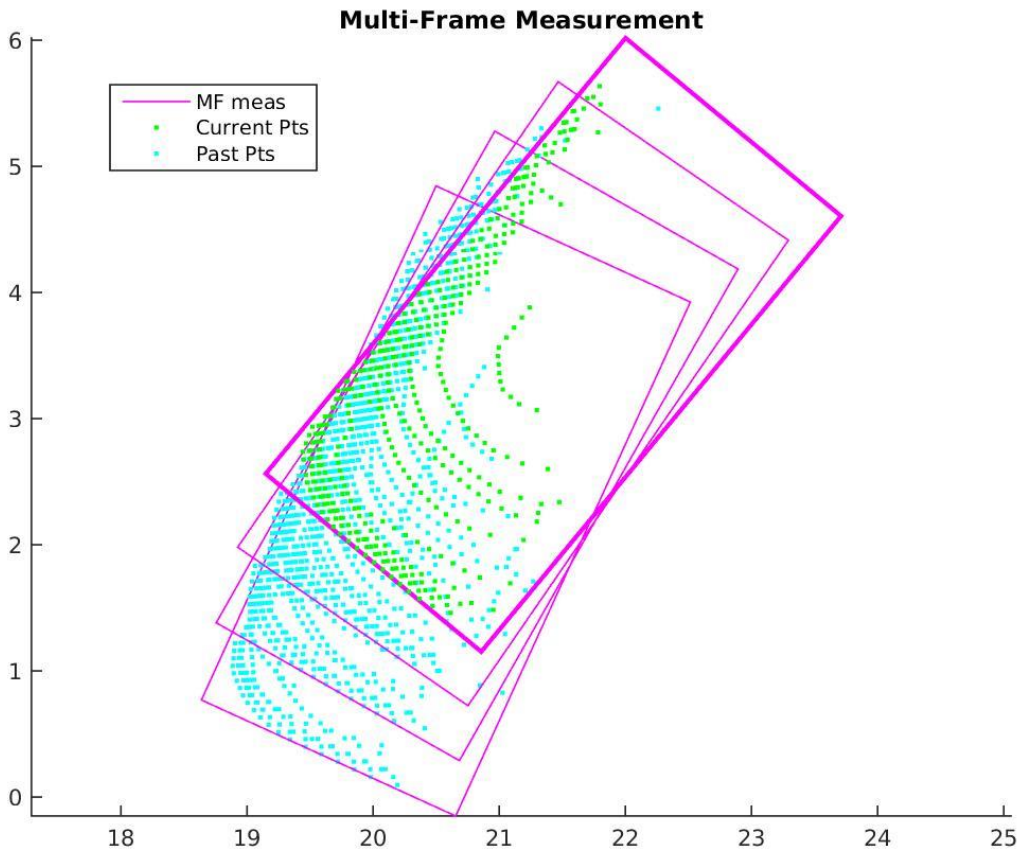


**Figure 9**: *Image*: An example of how MFMB measurement optimizes over multiple frames of previous vehicle data

# CHAPTER 4

# DIFFERENTIAL MEASUREMENT

In order to compensate for the weaknesses of the Model-Based approaches, there has been significant research done on Model-Free tracking. This is often accomplished by using a registration algorithm such as Iterative Closest Point (ICP), which is able to measure differences in frames without relying on a generic model. This differential measurement works by optimizing the transformation that best describes the vehicle's movement between two different frames in time. However, this method too has downsides. First and foremost, it loses the predictive power of a kinematic vehicle model, which is essential for our application of vehicle tracking and future trajectory prediction. Additionally, ICP loses the tracking accuracy possible from a Model-Based method, in the situations where this method performs well. Therefore, in our investigations we considered how ICP can be used in conjuncture with existing Model-Based methods, to prevent gross measurement errors, while still retaining the strength of Model-Based measurements.

## Iterative Closest Point Algorithm

The Iterative Closest Point (ICP) algorithm is one of the most commonly used registration methods. Significant research has been done studying this algorithm, on both improving its performance (Besl & McKay, 1992) and also using it in many model free tracking applications ( (Wang, Posner, & Newman, 2015); (Pooja & Govindu, 2010)). For a concise overview of the variations of ICP and its many applications, see (Rusinkiewicz & Levoy, 2001).

ICP attempts to measure the transformation that best describes the movement of an object over time, by optimizing the transformation between two similar point clouds. For our application, ICP uses two consecutive frames of vehicle points and estimates the optimal transformation between them. ICP generally provides a quite a good estimate of the vehicle's movement, without needing any assumed vehicle model. For this reason ICP is commonly used for tracking applications where a model is less descriptive, such as the tracking of pedestrians or other non-uniformly shaped objects.

There are many minor variations of the ICP technique, but the general algorithm is the same. Throughout the process, one point cloud, the reference, is kept fixed while the other, the source, is transformed to best match the reference. The algorithm iteratively updates the transformation to minimize the distance between the source and the reference. The essential steps of the algorithm are as follows, and are depicted in Figure 10. First, each point in the reference cloud is matched with the closest point in the source cloud. Second, some sort of mean-squared cost function is used to estimate the rotation and translation that will best align the two point clouds. Third, the source points are transformed using the above transformation. Finally, this process is iterated until a stopping criteria is met (the reference and source cloud

points are re-associated, and the algorithm is repeated). There are many minor variations to this algorithm to improve its speed and performance, but this is the essential process. An example of using ICP to estimate a vehicle's transformation over time is shown in Figure 11.

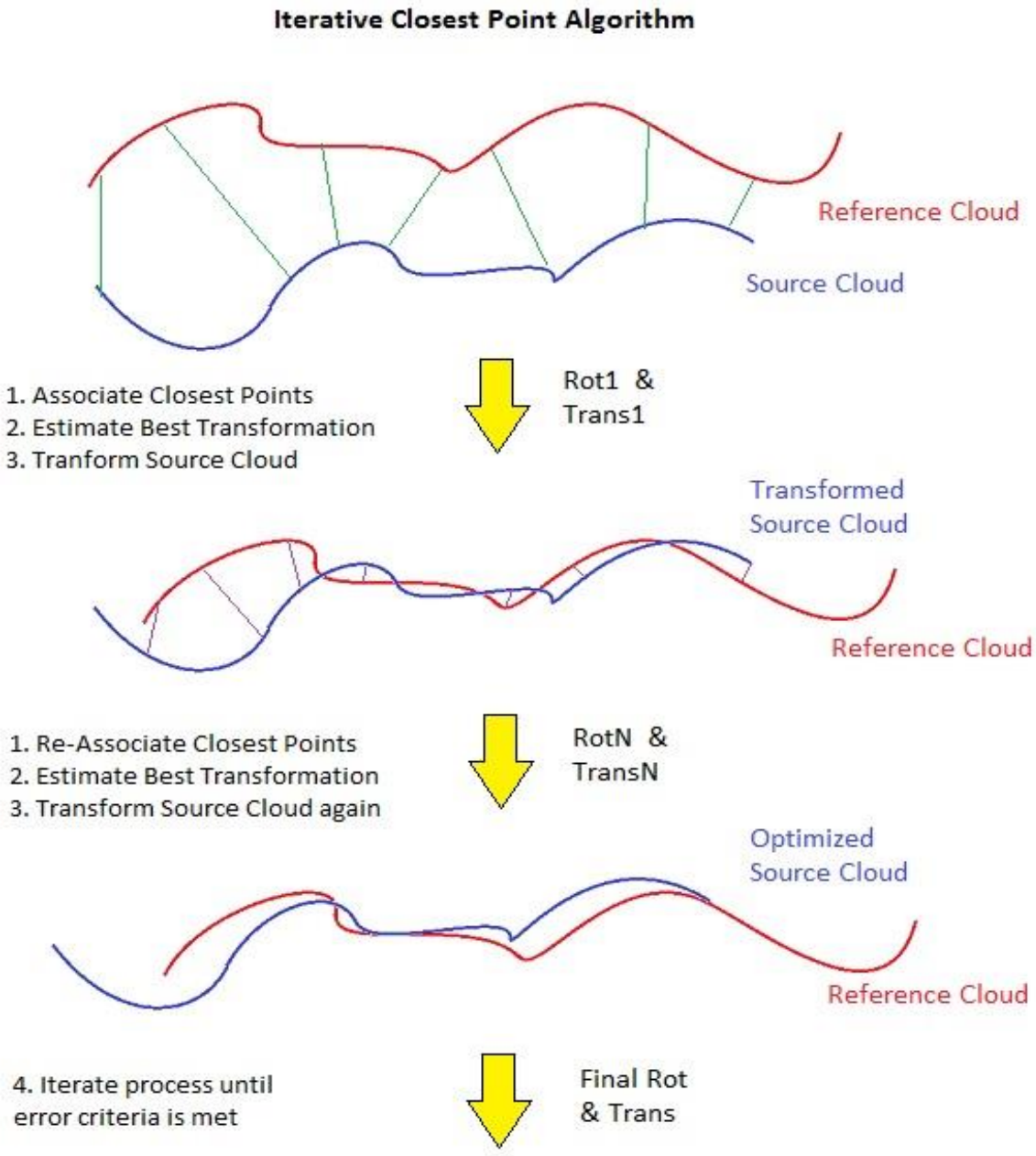


**Figure 10**: *Diagram*: A visual depiction of the Iterative Closest Point Algorithm

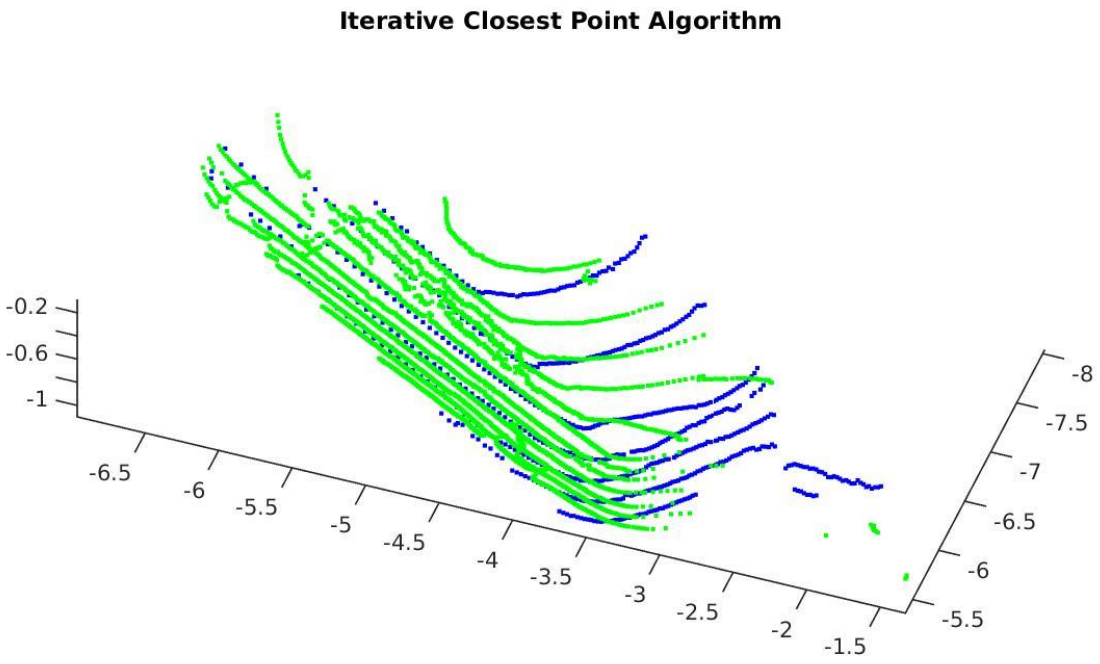


**Figure 11**: *Image*: An example of how ICP can be used to estimate vehicle movement

## Limitations of Local Minimization

The ICP algorithm is an extremely powerful tool; it is capable of quickly returning a decent estimate of the transformation between two similar frames without any dependence on a given model. That being said however, it also has severe limitations. This is mainly because when it minimizes the cost function to find the best fit, it is not actually finding the global best

fit, but instead the best local minima. This is due to the point association step; if extreme differences exist between the two frames initially, the algorithm will match incorrect point pairs, and then the procedure will likewise optimize to a bad overall match. This causes two major issues for our vehicle tracking application. First, it is generally only effective when used over frames with similar times, when the vehicle hasn't drastically changed its pose. Secondly, it performs very poorly when a vehicle turns, and a different edge of the vehicle becomes visible. This is because the algorithm will try to minimize the distance between the two opposite sides of the vehicle, greedily assuming they are the same side. These are two significant issues that we will need to address in order for ICP to be useful to us.

## Three Degrees of Freedom Constraint

To improve the performance of our ICP algorithm in the application of vehicle tracking, we constrained the freedom of vehicle motion. Instead of allowing a six degree of freedom transformation through 3D space, we restricted vehicle motion to a single plane parallel to the ground. We made this assumption because for normal vehicle operation, a vehicle is only capable of moving with its wheels touching the ground. By limiting the possible vehicle transformations over time to three degrees of freedom ($x, y, \theta$), we get rid of many cases where the ICP returns a meaningless and incorrect estimated transformation. Below is an example of the improvement returned by constraining vehicle motion to three degrees of freedom.

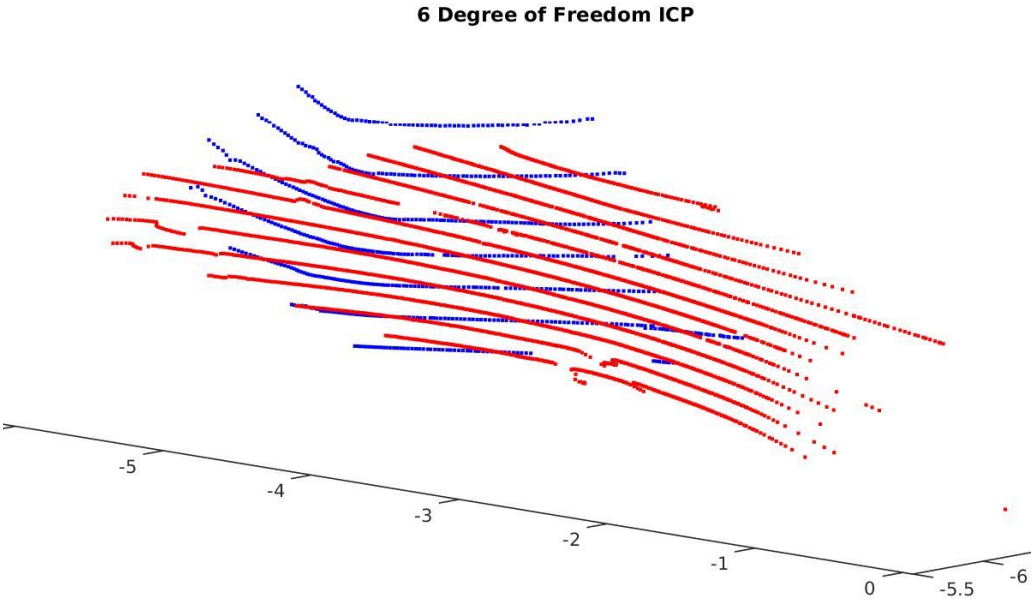


**Figure 12**: *Image*: Operation of the vehicle tracking ICP with six degrees of freedom

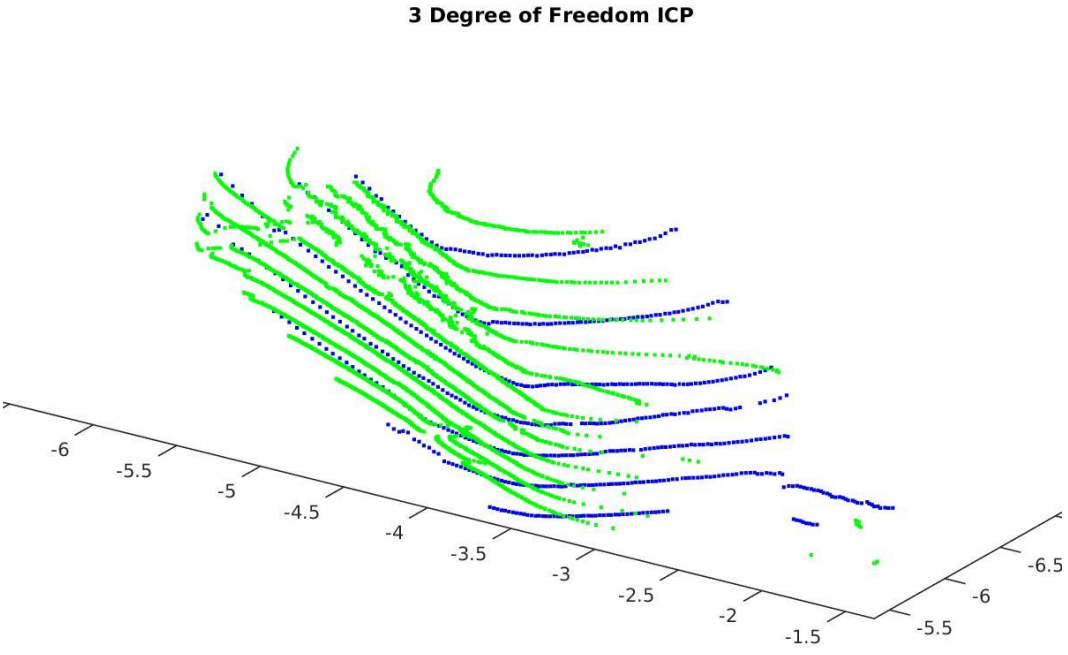


**Figure 13**: *Image*: Operation of the vehicle tracking ICP with three degrees of freedom

## Differential Measurement Technique

The ICP algorithm essentially measures the difference between two vehicle clouds over time. Differential measurement therefore works rather differently than conventional Model-Based measurement. ICP doesn't require an assumed vehicle model; this can be a strength when the vehicle is not rectangular, but it isn't as accurate in cases when the model does fit well. Secondly, ICP estimates the linear and angular velocity of a vehicle over a span of time, but it is incapable of estimating the actual pose of a vehicle at a single instant. This actually makes integrating it within a typical vehicle tracking framework slightly more complicated. In order to compensate for this difference, when we use differential measurements with our EKF filter, it is necessary to utilize a Model-Based measurement technique for the first frame of data, to get the initial first order variable measurements $(x, y, \theta)$. Once we have these initial measurements, we can use the measurements ICP does return $(s, \dot{\theta})$ to extrapolate the precise pose measurements found by ICP over the rest of the vehicle trajectory.

In order to compare ICP differential measurements $(s, \dot{\theta})$ with Model-Based position measurements $(x, y, \theta)$, we extract the delta position measurements from two frames of the Model-Based position measurements, which can then be directly compared to the differential measurements. This performance will be summarized in Chapter 6, Results. Since both of these algorithms work in different ways and have distinct strengths, we consider both methods in the hope to use them in a complementary fashion to improve overall vehicle tracking accuracy and precision.

# CHAPTER 5

# MODEL SIMULATION

In order to get meaningful results to compare the effectiveness of different methods, it is crucial we collect vehicle data with a known pose. To initially test the trackers, we used physical vehicle data collected with an HDL-32E Lidar. However, a large challenge to existing vehicle tracking research is the difficulty in getting effective, quantitative results to draw meaningful conclusions from. This is because for accurate tracking method comparisons, we need point cloud vehicle data with a known ground-truth vehicle pose, which is a decidedly complicated task during real world data collections. One study (Held, Levinson, & Thrun, 2013) cleverly simulated known vehicle movement by tracking stationary vehicles using a known, moving sensor trajectory. This allowed them to collect physical data with precisely known ground-truth trajectories. However, since our Lidar scanner is not yet mounted to a vehicle, we decided that the best way to achieve this was to build a program to simulate the point clouds from a vehicle moving in a user-specified path. This provided us with the precise known trajectory of the vehicle, and also allowed us to simulate diverse vehicles.

In order to build a vehicle point cloud simulator, we began with a vehicle mesh. This is useful because there are large online warehouses full of diverse, free vehicle meshes, allowing the simulation of many different types of vehicles. Next, we simulated a HDL-64E Velodyne Lidar by gathering the locations, angles, and frequency of lasers from data-sheets. Then, by

placing the vehicle mesh at a specified location relative to the Lidar sensor, an accurate point cloud was created by intersecting the known lasers directions with the mesh's triangular facets, and filtering out any redundant intersections. Finally, we also built in an additional feature to simulate Gaussian depth-noise, to further test the effectiveness of the measurement methods.

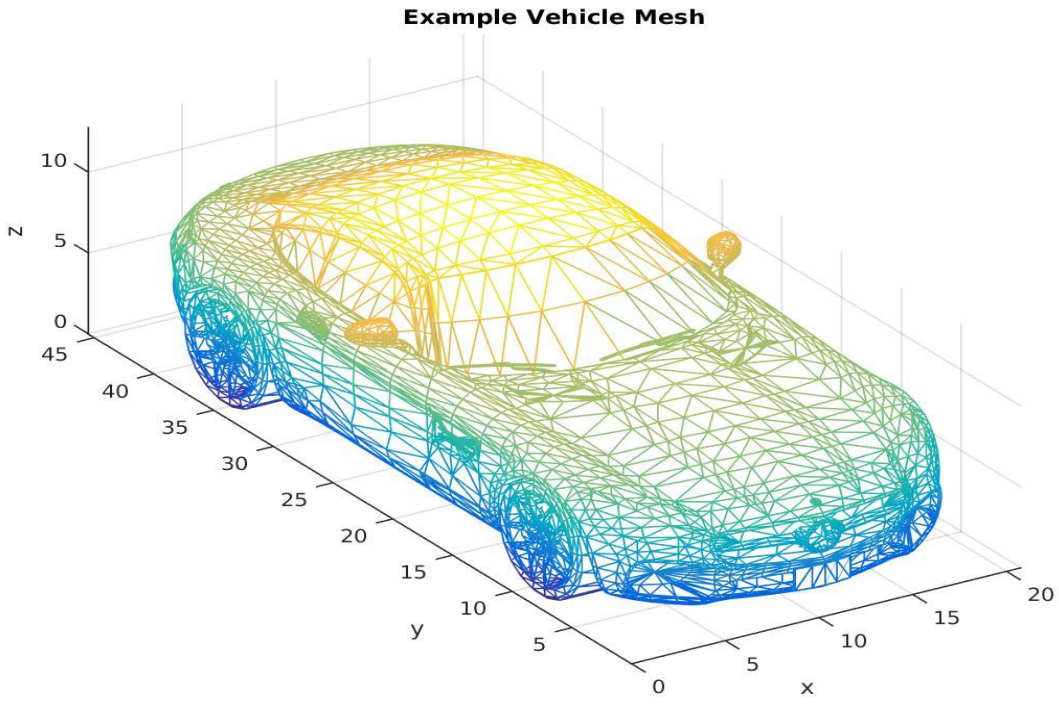


**Figure 14**: *Image*: An example of a vehicle mesh available online

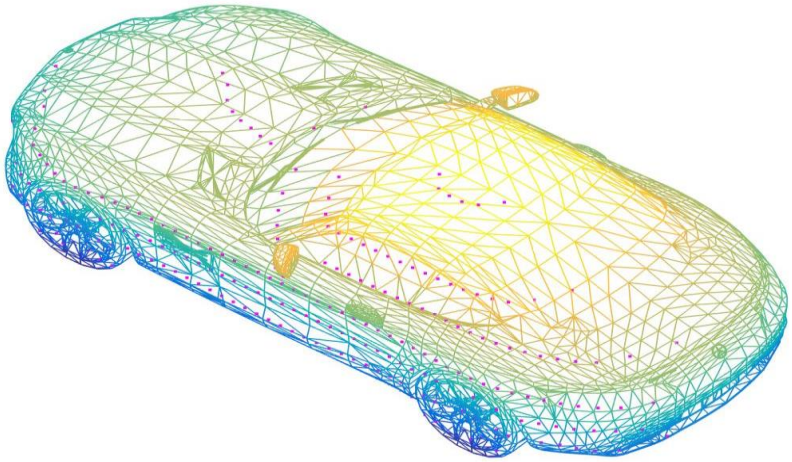

**Figure 15**: *Image*: A vehicle mesh with the Lidar points from the simulation algorithm

The overall operation of the point cloud simulator involves feeding in any vehicle mesh and desired trajectory, based on one of our three kinematic models, and returns the exact point clouds and known pose at every point in time. This is a powerful simulation algorithm; it allows us to generate any available vehicle mesh along any path we desire, uses various types of Lidar sensors, and returns a precise series of point cloud measurements with the corresponding known pose of the vehicle.

Using this simulation algorithm allowed us to collect diverse datasets. This included data at far distances, data with different amounts of noise, and test many different vehicle shapes. It also returned a precise ground-truth pose in order to calculate the absolute error between the different measurement methods. This point cloud simulator was essential for obtaining measurement data with ground truth poses needed to evaluate our methods.

# CHAPTER 6

# RESULTS

In order to summarize the relative strengths of the different tracking methods, they were tested under diverse situations. The major findings and differences between each type of tracker are summarized below. Because model-fitting performance is based on many factors including the type and shape of the vehicle, the sparsity of data, Lidar vision conditions, environmental clutter, and model accuracy, it is difficult to try to extract an overall quantitative grade of each method. Instead we will focus on individual, diverse situations and compare how the trackers perform in each one, in order to draw general conclusions about the overall effectiveness and strengths of each.

## Gross Measurement Error Avoidance

Our first goal for investigating these two additional measurement techniques was to better avoid gross measurement errors. Gross errors are cases where the fitting process matches different parts of the vehicle data to the vehicle model, causing major a measurement error. The reason we chose to study ICP and Multi-Frame measurements was because they are both inherently more robust than Single-Frame Model-Based measurements. Below we show three cases under which the standard SFMB measurement technique fails, including cases with

vehicle clutter, sparse data points, and situations where pickup trucks or semi-truck cabs cause unusual point distributions. It can be seen in each of the three figures below that both tracking methods ICP and MFMBT avoid these situational gross errors significantly better than SFMBT, confirming our hypothesis that these methods would be more robust.

We notice that both ICP and MFMB are more robust, but in different ways. MFMB measurements work better in situations where the model is generally a good representation of the vehicle, but there are individual frames where the model fails, often due to noise, sparse data, or the vehicle's orientation. This can be seen in the example with a single frame of exhaust clutter above (Figure 16), and additionally in the case with very sparse data (Figure 17). ICP on the other hand works better (relatively) when the true shape of the vehicle is misrepresented by a rectangular 2D model, which can be seen in the example above of tracking a semi-truck cab (Figure 18). In this case, since the model doesn't accurately match the vehicle's underlying shape, MFMB also returns an incorrect measurement.

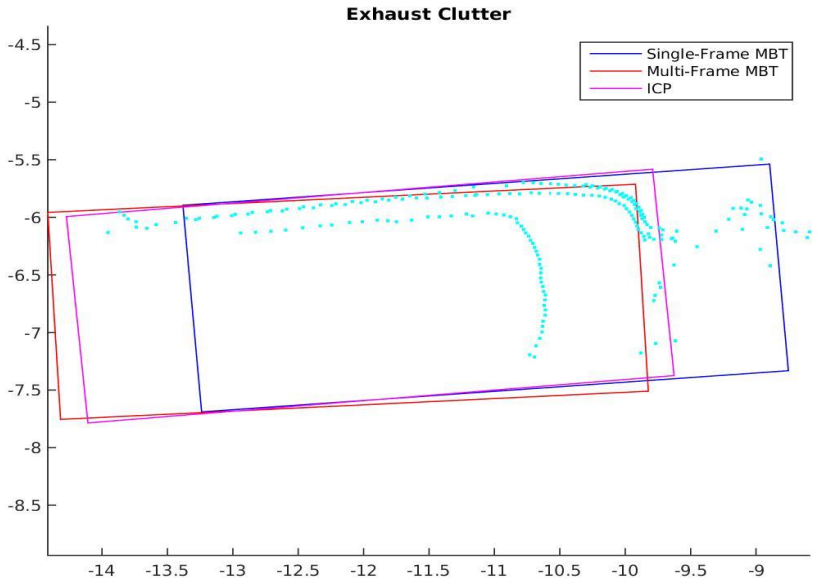


**Figure 16**: *Image*: Clutter from the vehicle's exhaust causes a gross measurement error

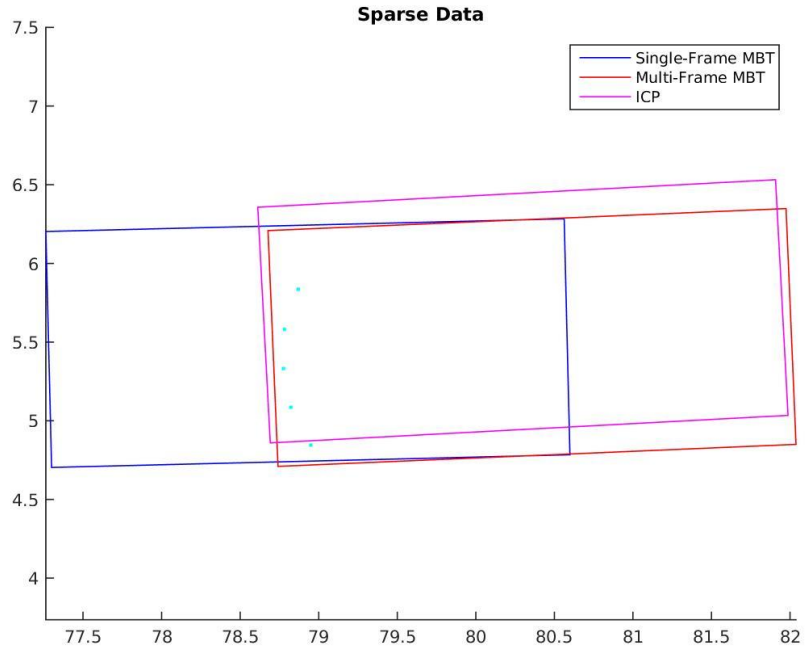


**Figure 17**: *Image*: Sparse data from a distant vehicle causes Single-Frame measurement errors

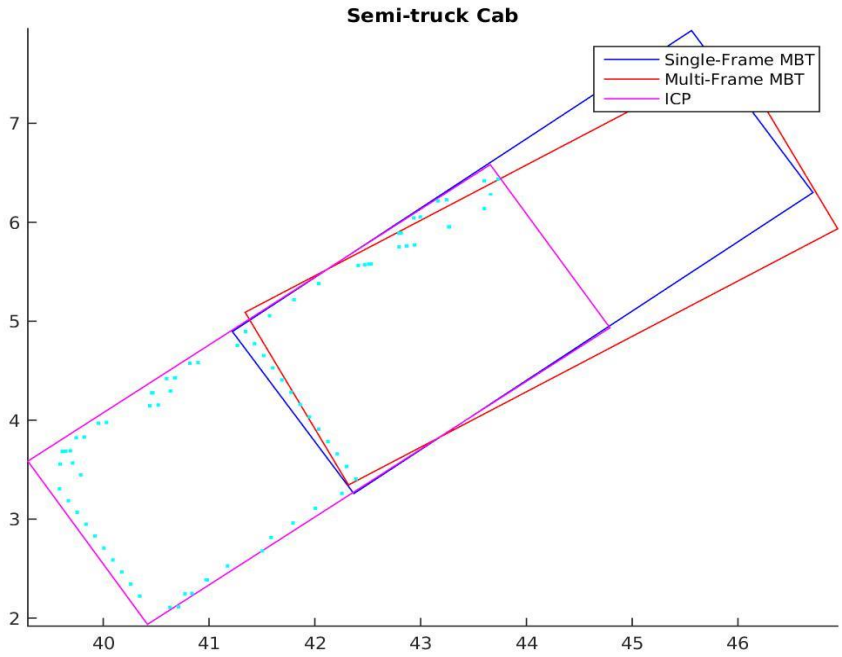


**Figure 18**: *Image*: A semi-truck cab's nonrectangular 3D shape causes gross measurement errors

We have shown that both of these measurement techniques are capable of addressing this first objective: improving our ability to avoid gross measurement errors. On the other hand, both methods are generally more expensive in terms of computation time. Therefore, a simple combined solution to avoid gross measurement errors would be to run a tracker that generally feeds SFMB measurements into the EKF, until we get a bad measurement (which we could determine from cases where the measurement and model prediction estimations are very different), and in this case use one of these two more robust measurement techniques. This would offer the fastest overall performance, while also addressing the issue of gross measurement errors. However, a general solution of how best to utilize all measurement techniques together is completely dependent on the desired application of the tracker.

Therefore, we will now attempt to characterize the strengths of both of these two measurement methods, so that they can be combined in a manner the user desires.

## Performance Metrics

In order to analyze the effectiveness of our measurement techniques, we will compare both ICP and MFMB individually to the existing SFMB measurement method. We do this because we need to use slightly different metrics for the ICP and MFMB comparisons. The reason for this is that ICP inherently returns only differential measurements, whereas because both SF and MF are model-based, they return location measurements. Additionally, our reference data of the known vehicle trajectory is also in terms of location measurements. Therefore, we will individually compare SFMB and MFMB performances in terms of location errors, and secondly we will compare SFMB and ICP in terms of pairwise differential errors.

In order to compare the effectiveness of both measurement techniques, we first need to define the metrics we will be using to judge their performances. For the location measurements we will plot the errors between the measured location, and the known true location. We will plot this in local coordinates; that is we will measure the error between the measured and true vehicle locations in terms of the direction the vehicle is facing at the given instant. Once we have all local errors, we will combine the x and y errors in a scatter plot to compare the performance and accuracy of both methods. Comparing the performance of the differential measurements will be done in the same manner, except instead of calculating the location error, we will be calculating the differential position errors.

## Measurement Bias

This introduces an important topic, measurement bias. In a general error measurement situation, we would expect there to be some inherent difference between the center-point of our rectangular model and the center-point of the true vehicle location. This is called the measurement bias. Measurement bias is created because we are using a generically shaped rectangular model to represent unique vehicles with unknown true shapes. An example of measurement bias is shown below (Figure 19).

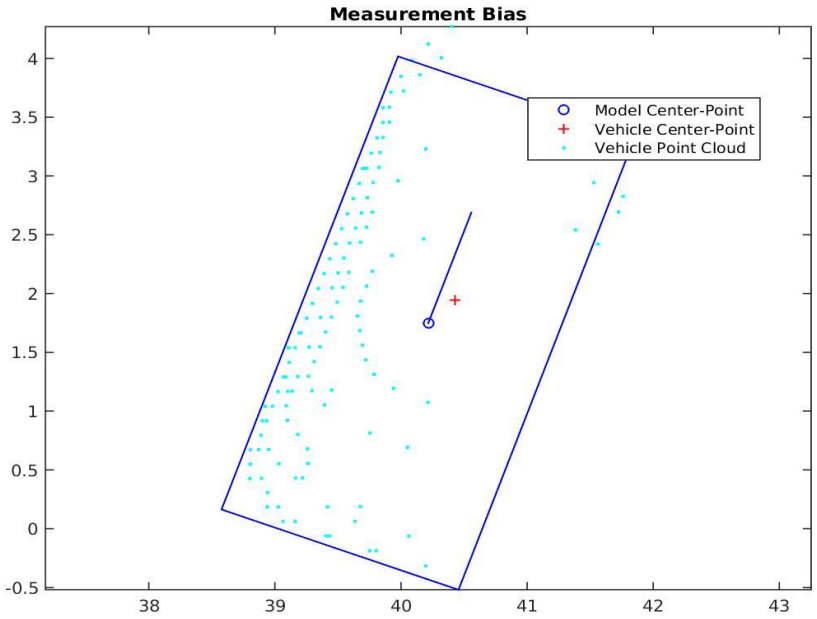


**Figure 19**: *Image*: An example of the measurement bias between the model and vehicle Center-Points

When we measure the error between a given measurement and the true vehicle location, there can be two sources for error: error due to the failures of the measurement

technique or error due to the inherent measurement bias. This is why we represent the measurement errors with a scatter plot. In the case where the measurement technique performs consistently and accurately, but there exists a constant measurement bias, we would expect the errors to be tightly clustered, but not necessarily around the origin (which represents zero error). Therefore, it is misrepresentative to use accuracy alone to describe the performance of a measurement technique. Precision is a significantly more useful metric to compare the performance of different techniques, because we want our techniques to provide consistent measurements over similar datasets. For this reason, for each measurement method, we will create a scatter plot of the localized error, and compare the variance of the different measurement approaches, in order to determine which method performs best. We also count the number of gross measurement errors each technique makes, but don't include these errors within the standard deviation calculations, since they correspond to fitting incorrect portions of the vehicle.

## Multiple-Frame Model-Based Measurement Analysis

First we will compare the two types of Model-Based fitting, both the single-frame implementation and our multi-frame version. As previously mentioned, since both measurement approaches are model based, we are able to compare the localized position errors. Since our MFMB implementation assumes constant speed and angular velocity, the vehicle trajectories we measured met those constraints. Additionally, we used the same kinematic model (Center Point Steering) in both the MFMB measurement technique and the

trajectory of the vehicle point cloud simulation, in order to remove any possible error which could be caused by that inconsistency.

In Figure 21, we see initial scatter plot comparisons of the SFMB and MFMB measurement techniques, compared over multiple data collections. For each collection the vehicle and its trajectory were exactly the same; in each case, the vehicle drove in an equally sized circle, but different parameters were changed in order simulate various environmental conditions. The below scatter plot summarizes cases with simple data, noisy data, sparse data, and data with missing points.

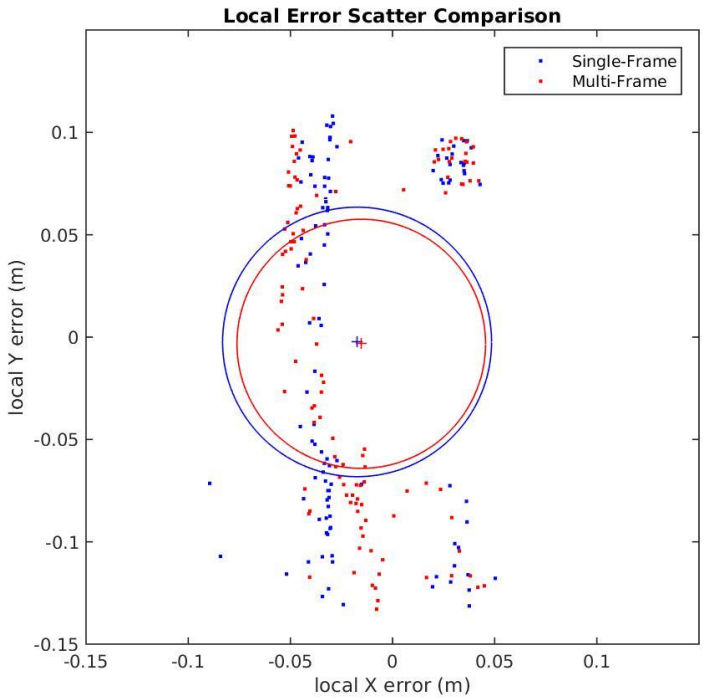


**Figure 20**: *Scatter Plot*: Comparison of the Single and Multiple Frame Model-Based Measurement Techniques

| Measurement Method | Standard Deviation | Number of Gross Errors |
|---|---|---|
| Single-Frame MB | 0.0658 | 2 |
| Multi-Frame MB | 0.0608 | 0 |

**Table 1**: Corresponding SFMB and MFMB measurement data

Inspecting the scatter plot above and comparing the standard deviations of each technique, we can see that the Multi-Frame measurements were marginally more precise over a series of various data collections, although they had the significant advantage of not making any gross errors. We also noticed that the overall point distribution of each technique was oddly shaped, without a well-defined center. This was an unexpected result suggesting an unmodeled source of error. In order to explain it we consider the effect of the measurement bias.

### *Varying Measurement Bias*

We have already discussed the effect of constant measurement bias on the accuracy of the measurement techniques: a constant measurement bias would shift the center of the measurement errors by a constant amount, while leaving the precision of the measurements unaffected. However, since we are measuring a circular vehicle trajectory, the measurement bias we see isn't a constant value. In the example below, the assumed vehicle model is smaller than the true size of the vehicle. In this case, the model will be matched over the visible vehicle points, which will cause the model center-point to be closer to the visible corner than the vehicle's true center-point, which creates measurement bias (Figure 21). However, as soon as a different corner of the vehicle is visible, the measurement bias shifts in that direction (Figure 22). This is because the center of the model is not also the center of the vehicle. This could be fixed by adjusting the width of the 1D filters of the vehicle model for each unique vehicle.

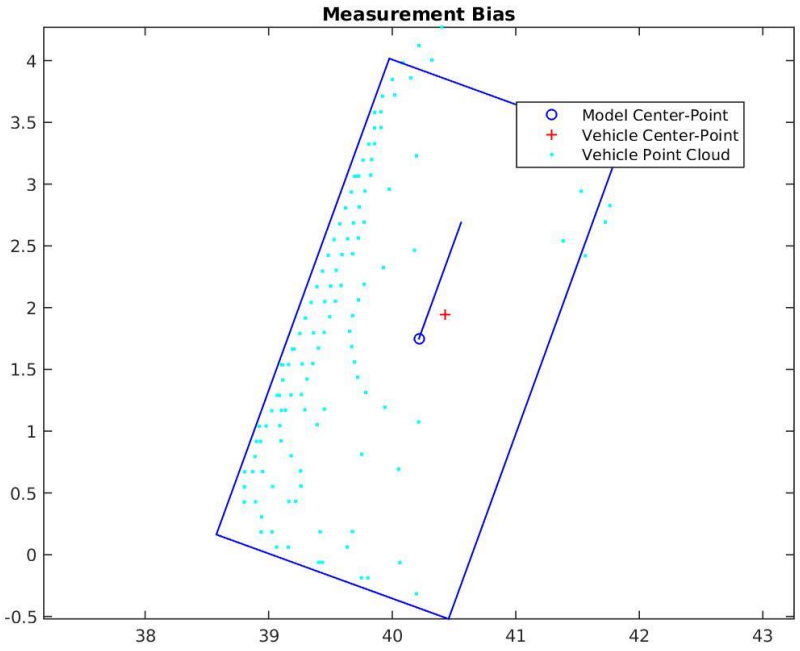


**Figure 21**: *Image*: An example of the measurement bias between the model and vehicle centers, as seen from a certain vantage point

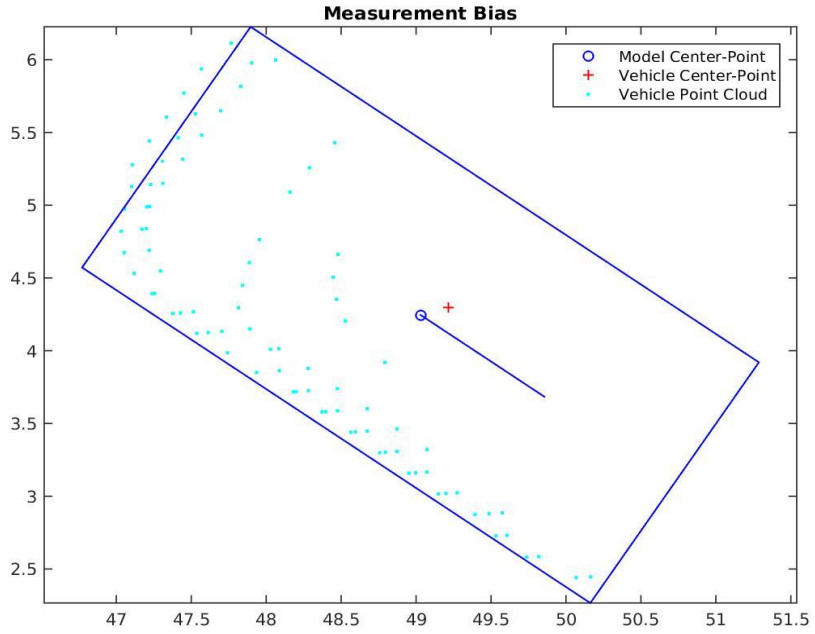


**Figure 22**: *Image*: A second example of the measurement bias between the model and vehicle centers, as seen from a different vantage point

In order to account for this, we divided the above scatter plot measurement errors based on which corner of the vehicle is visible. Since the vehicle trajectory is circular, this effectively means dividing the circular measurements based on which quadrant the vehicle is in, since each quadrant contains a different visible corner. When we evaluate the same earlier results, separated based on which corner of the vehicle is visible, we get more meaningful results (Figures 23 and 24).

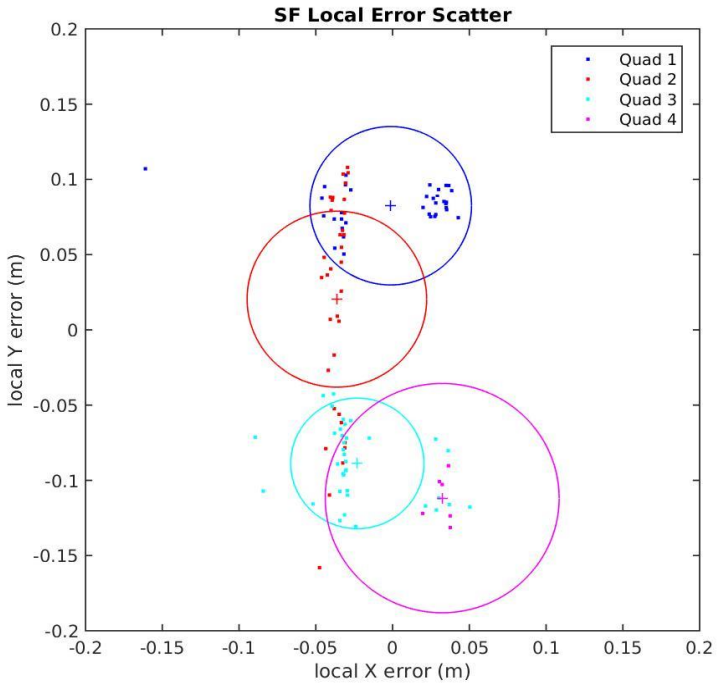


**Figure 23**: *Scatter Plot*: A scatter plot of the local error measurements for the Single-Frame approach, separated based on the quadrant of the circle the vehicle is in

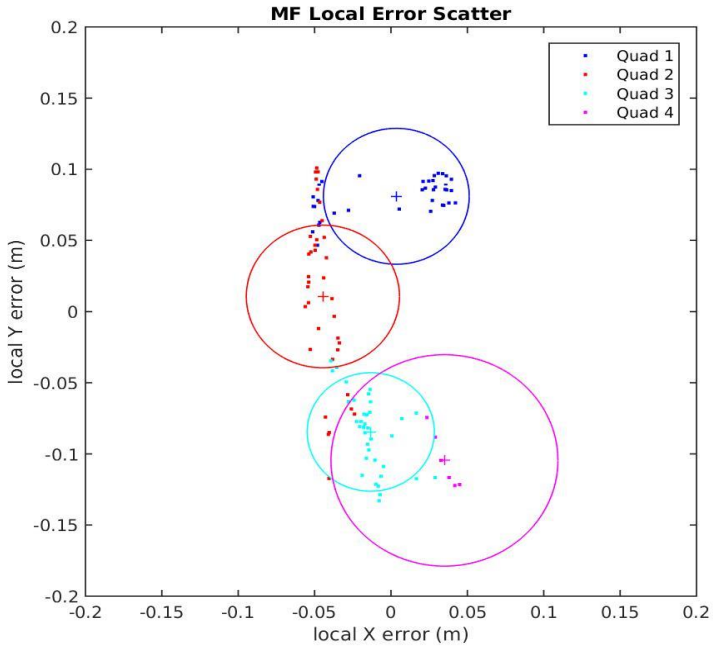


**Figure 24**: *Scatter Plot*: A scatter plot of the local error measurements for the Multi-Frame approach, separated based on the quadrant of the circle the vehicle is in

| Measurement Method | Average Standard Deviation | Total Number of Gross Errors |
|---|---|---|
| Single-Frame MB | 0.0577 | 2 |
| Multi-Frame MB | 0.0534 | 0 |

**Table 2**: The measurement data from the two scatter plots in Figures 24 and 25, with averaged standard deviation and the total gross measurement error count

In the above scatter plots, we first notice that the errors within each quadrant are much more precise, with each quadrant containing a distinct center-point. This confirms our earlier hypothesis that the measurement bias does change based on the corner of the vehicle that is visible. When we compare the results of the exact same data, separated based on the visible corner of the vehicle, we see that both the performances of the SFMB and MFMB improve, with the Multi-Frame optimization still performing better both in terms of measurement precision and gross error avoidance.

### *Removal of Measurement Bias*

In order to remove the issue of the varying measurement bias, we conducted another series of data collections all with the same vehicle corner visible, to minimize the measurement bias. Additionally, we refined the rectangular model to fit the true vehicle size as best we could, in order to further minimize the measurement bias. Next we compile a series of diverse measurements, including cases with sparse points, noisy data, and clutter, all over equivalent vehicle trajectories. The local position errors are compared within Figure 25 and Table 3. In these results, it can be seen that the local error scatter plot has a more typical distribution with a strong center point. Overall, it can clearly be seen that MFMB performs better than SFMB in terms of local position error, orientation error, and gross error avoidance.

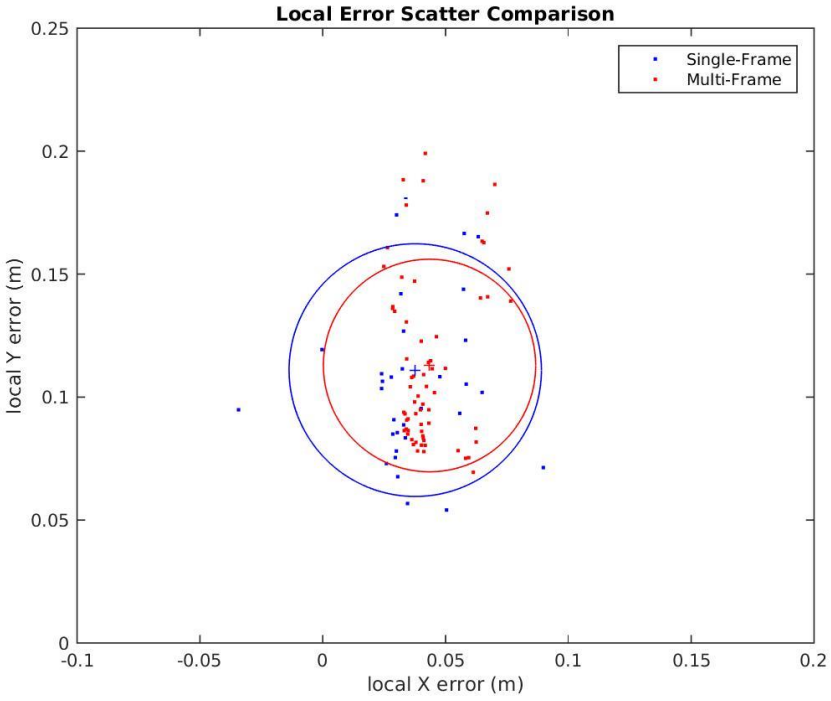


**Figure 25**: *Scatter Plot*: Comparison of the Single and Multiple Frame Model-Based Measurement Techniques in a case with minimal measurement bias

| Measurement Method | Standard Deviation | Number of Gross Errors |
|---|---|---|
| Single-Frame MB | 0.0514 | 2 |
| Multi-Frame MB | 0.0432 | 0 |

**Table 3**: Corresponding SFMB and MFMB measurement data with minimal bias

### *Multi-Frame Performance Spectrum*

The previous results show that once bias is removed, Multi-Frame optimization reliably performs better than its Single-Frame counterpart. We then wanted to further explore how the number of frames we optimize over affects the overall MFMB measurement performance. In order to do this we collected a series of data sets over linear vehicle trajectories, under various noise and distance conditions. Next MFMB measurements were collected while varying the number of optimized frames, and the results are summarized in the table below.

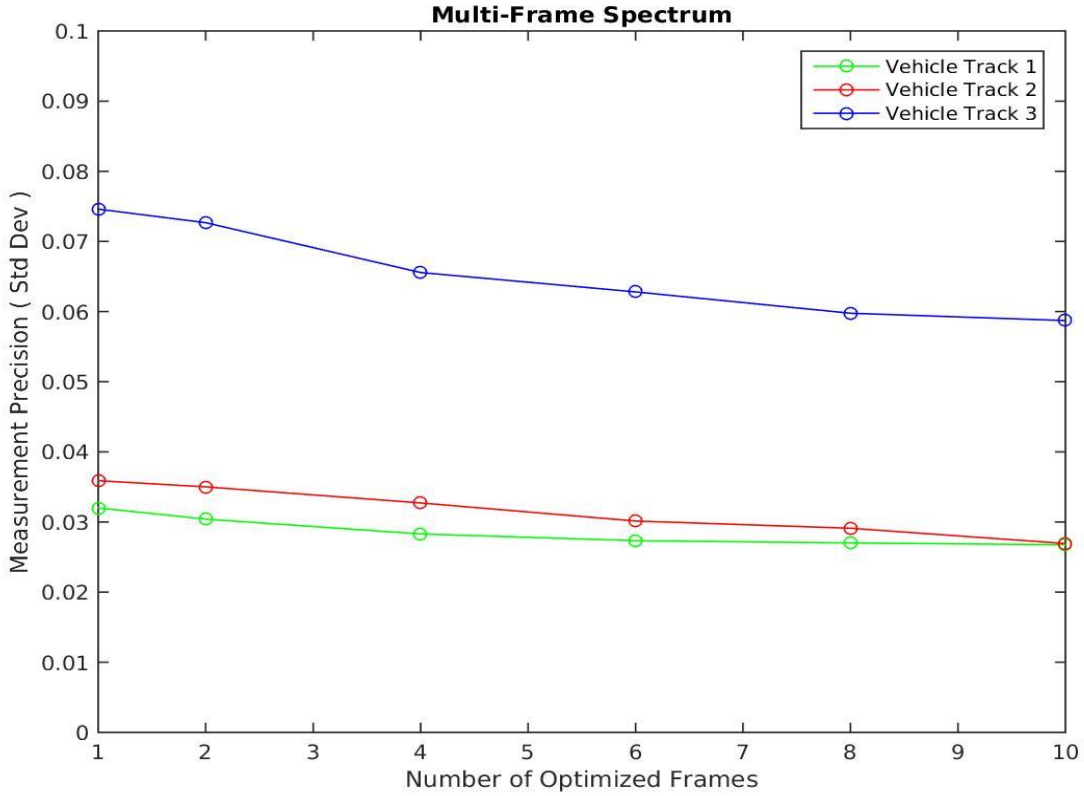


**Figure 26:** *Plot*: How optimizing Multi-Frame model fitting over different numbers of previous data frames affects measurement precision

The above chart shows three distinct data sets. For each one, the X-axis contains the number of frames the MFMB technique optimized over, and the Y-axis shows each measurement's corresponding precision. Note that the data optimized over 1 single frame corresponds to the SFMB technique. It can be seen that in each of the cases, increasing the number of optimized frames increases the precision of the measurements, shown by the decreasing standard deviation, but with diminishing effect. However, a drawback to increasing the number of optimized frames is that it significantly increases the time taken to solve the optimization.

Overall MFMB performs consistently with what we would expect. Increasing the number of frames likewise increases the number of vehicle points optimized over, which therefore results in more consistent and robust measurements. An example of the single and multiple frame operation is below, to showcase why and how MFMB works better (Figure 28). However, there is a trade-off between diminishing measurement improvement and increasing computation time. Therefore future applications will need to decide on the number of optimized frames to achieve the desired balance between those two factors.

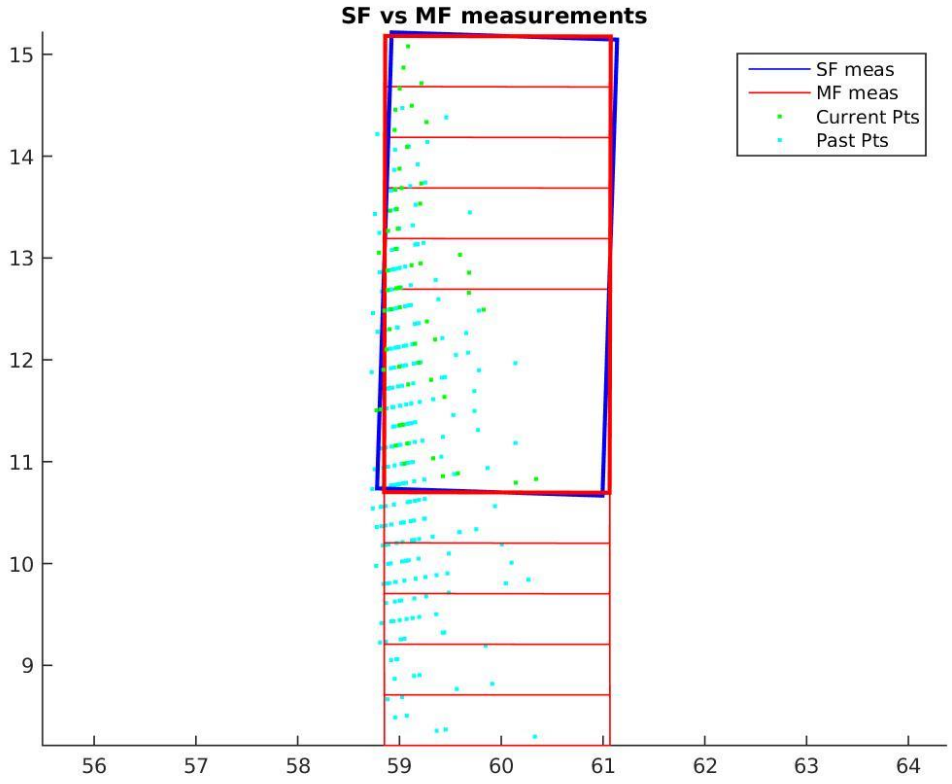


**Figure 27:** *Image*: An example of Single and Multiple Frame Model-Based optimizations over a set of noisy data

## Differential Measurement Analysis

Now we will compare the overall performance of ICP versus traditional Model-Based tracking. It is difficult to quantify how well Model-Based tracking performs, because it is inherently dependent on how well a specific vehicle can be represented by the generic rectangular model; its performance is completely situational. On the other hand, since ICP is model-independent, it performs significantly more consistently for vehicles with diverse shapes. Another major difference between ICP and Model-Based fitting is that ICP is a local minimization whereas SFMBT is global.

In cases where a rectangular model accurately describes the true shape of a vehicle, SFMB is able to return significantly more accurate measurements. As we did in the previous section, we compiled a series of diverse measurement cases and compared the effectiveness of the two measurement methods, SFMB and ICP. However, recall that since ICP is a differential measurement, in this case the scatter plot is showing results of the pairwise differential errors. However, other than that slight distinction, the two comparison approaches are very similar.

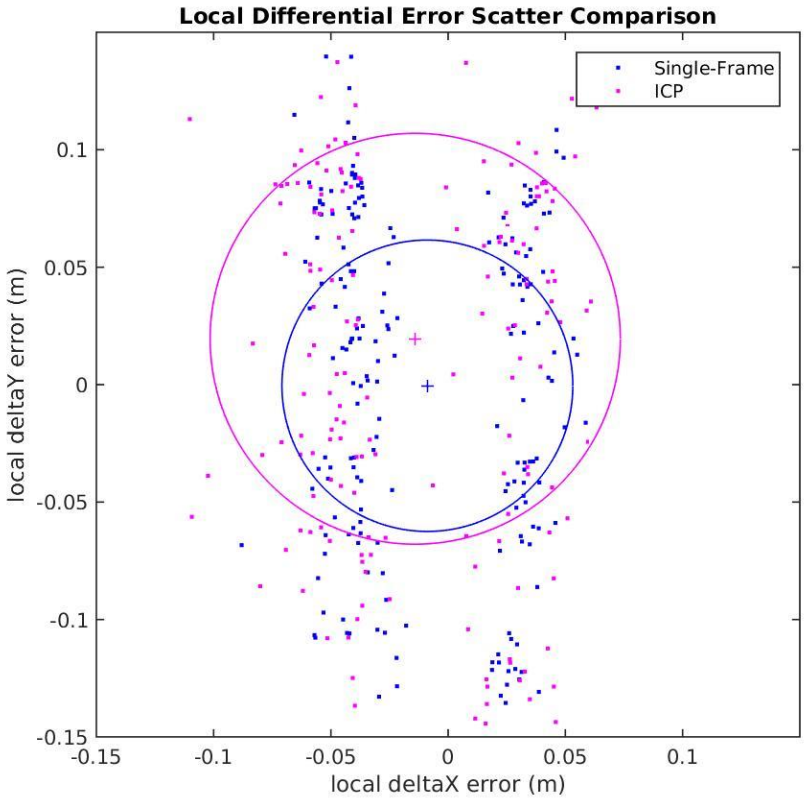


**Figure 28**: *Scatter Plot*: Comparison of the Single-Frame Model-Based and ICP Measurement Techniques

| Measurement Method | Standard Deviation | Number of Gross Errors |
|---|---|---|
| Single-Frame MB | 0.0621 | 5 |
| Iterative Closest Point | 0.0875 | 12 |

**Table 4**: Corresponding SFMB and ICP measurement data

In the above results it can be seen that SFMB performs significantly more accurately that ICP, in cases where the model fits well. However, that doesn't necessarily mean SFMB is better overall. As the true vehicle shape is represented less well using a rectangular model, ICP

performs better and better, relatively to SFMB, and at some point will even outperform Model-Based measurements completely. Additionally, even in cases where the rectangular model is generally a good description of the vehicle's shape, in some instances ICP will still perform better than SFMB due to the inherent limitations of using a generic model. For example in the below figure (Figure 30), SFMB returns a drastically worse measurement error than ICP, due to the vantage point at which the vehicle is seen.

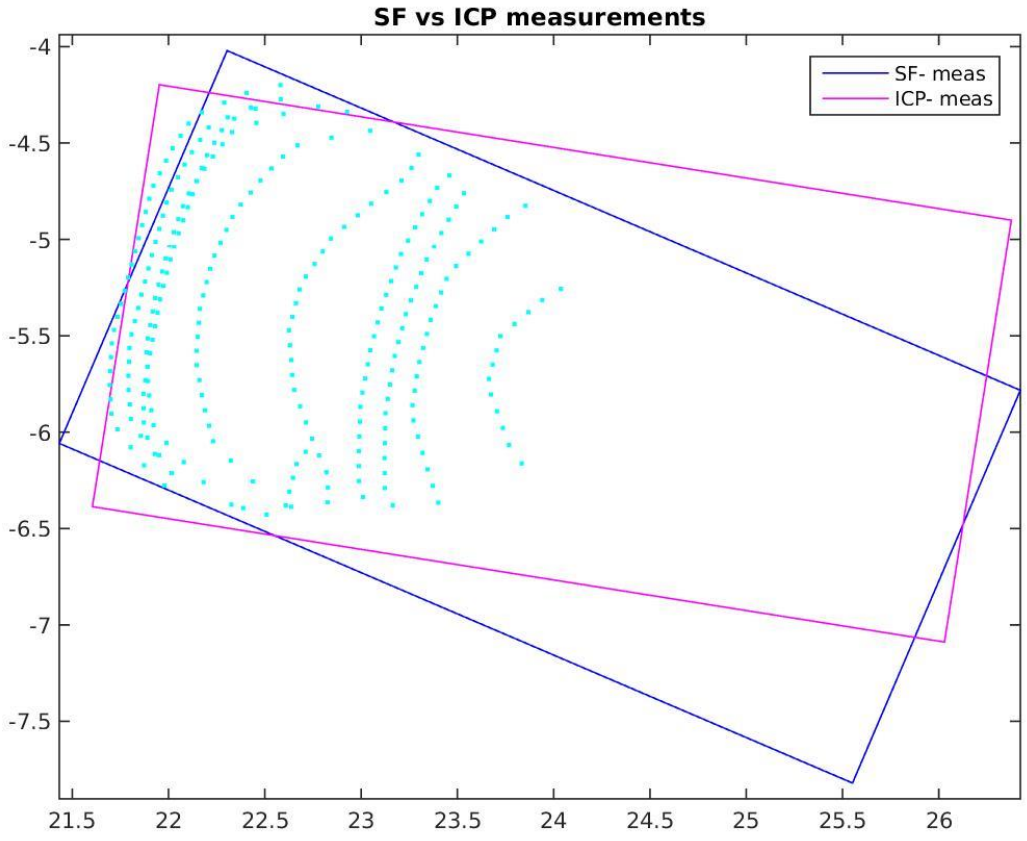


**Figure 29**: *Image*: Example of ICP outperforming SFMB with only one visible edge

We can draw three conclusions from this. First, in every situation either Iterative Closest Point or Model-Based Tracking will provide more accurate results, based on how well the rectangular model fits the true vehicle shape. Creating a combined measurement technique that will return either ICP or MBT measurements based on the shape of each individual vehicle would be a possible idea for future work. Secondly, the fact that ICP is only a local minimization is quite limiting. It can only be applied in situations where the two vehicle locations are relatively close, often being used in consecutive time steps. Model-Based tracking however benefits from being able to find the global best measurement of a vehicle at any instant of time; this robustness will be extremely useful in dealing with other difficult vehicle tracking issues such as occlusion. Finally, since ICP is only capable of providing relative, differential measurements, it has no way of estimating a vehicle's current location at a single instant in time; therefore implementing an ICP only measurement technique would be extremely difficult.

# CHAPTER 7

# CONCLUSION

In this study we explored how Multi-Frame Model-Based and Differential measurement techniques can be used to improve vehicle measurement accuracy. Increasing the accuracy and robustness of vehicle pose measurements is an important goal within autonomous vehicle research. The techniques of batch processing multiple frames and the Iterative Closest Point algorithm have been studied and used in different fields of research, but haven't been significantly explored in combination with model-based vehicle tracking.

In difficult situations Single-Frame Model-Based (SFMB) fitting occasionally matches incorrect parts of the rectangular model to the true point cloud data. These gross measurement errors are a major problem for vehicle tracking, and in some cases can be strong enough to completely derail the vehicle tracker. However, by increasing vehicle measurement robustness, we can minimize the occurrence of these gross errors. Increasing vehicle measurement accuracy is also important, because the extracted higher order parameters of velocity and acceleration are then likewise more accurate. Throughout this study, we have found ways that both accuracy and robustness can be improved by leveraging Multi-Frame and Differential measurements.

Multi-Frame Model-Based (MFMB) fitting provides a consistent way to increase both vehicle measurement accuracy and robustness, in cases where the rectangular model represents the true vehicle shape well. In the Results section we saw that increasing the

number of optimized frames results in more accurate measurements, and also reduces the number of gross measurement errors. MFMB is a very powerful tool for dealing with cases involving distant, noisy, or sparse data, or situations with individual frames of occlusion or inaccuracy.

Differential or model-free measurements are much more useful in cases where the true vehicle shape is not represented well by a rectangular model. In these situations both single and multi-frame model-based methods are ineffective and the best way to estimate vehicle movement is by using differential measurements.

By integrating multi-frame optimizations and differential measurements with conventional rectangular model based vehicle measurement, we provide additional tools to deal with difficult vehicle tracking situations. Conventional SFMB measurements are very useful: they often provide accurate measurements quickly, with low computational cost. However, the poor performance of SFMB in challenging situations can be improved using the two techniques we've introduced in this study. Multi-Frame optimization uses additional computation power to improve measurement accuracy and robustness, and model-free ICP measurement provides a way to measure non-rectangular vehicle movement.

# CHAPTER 8

# FUTURE WORK

In the course of our investigations on differential and multi-frame measurement techniques, we discovered many areas for future work. How each idea could benefit the measurement research we conducted will be explained in the following section, as well a brief description of implementation ideas.

## Multi-Frame Optimization of Higher Order Variables

One major limitation of traditional single-frame measurements fed into an Extended Kalman Filter is that the filter optimizes a linear cost function. For a single-frame measurement, the only input is a simple position and orientation estimate. By tracking that over time, the best fit optimization only has the strength to optimize the vehicle's location and velocities. However, this simplification loses a good deal of potential information, as true vehicle movement frequently includes acceleration and even changes in acceleration. While this cannot be measured using only single-frame measurements, these could be estimated with the additional frames used by the Multi-Frame Model-Based measurement technique. Similarly, it would also be possible to refine the vehicle length and width estimations, in order to minimize the effects of bias on MFMB.

## Iterative Closest Point Unique Model Accumulation

In general, ICP performs worse than Model-Based measurement methods, in situations where the vehicle fits the rectangular model relatively well. However, that doesn't mean it doesn't have potential; aside from being crucial in situations where Model-Based fitting fails, one huge possible field for future research is a unique vehicle model accumulation. The weakness with Model-Based fitting is that it assumes a generic model, which limits the effectiveness of the kinematic tracker for individual vehicles. The main idea behind unique vehicle model accumulation is that by using ICP, we could combine points from multiple frames in time to estimate a unique vehicle model, which could be built up as a re-sampled 3D point cloud. This would have numerous benefits, including very specific knowledge of the vehicle's shape to avoid collision, it would prevent gross measurement errors caused by rectangular vehicle model assumptions, and would provide more accurate vehicle location measurement

However as powerful as this technique would be, it comes with considerable challenges. First and foremost, it would be difficult to prevent ICP transformation errors from each frame from accumulating and creating an extremely incorrect combined model. In order to deal with this issue, significant work would need to be done to create a robust method to remove or improve poor matches. Secondly, clever algorithms must be designed to resample and condense redundant points, since our program's memory is finite. A possible solution could involve iteratively refining a vehicle mesh using each frame's point cloud data, starting from a basic rectangular box model. A tool like this would be a powerful addition to the field of vehicle tracking.

## Situationally Utilized Measurement Models

Creating a combined measurement approach that will return either ICP or MBT measurements based on the shape of every unique vehicle would provide the most accurate measurements. This idea would be to combine both Model-Based and model-free tracking and utilize the technique that will best measure the vehicle position. Every vehicle would be classified based on how well it is represented by a rectangular vehicle model. However once that is complete, by continuing to use that method to track the vehicle's movement, a situationally unique technique would return the best possible measurements using the ICP or MB techniques.

# APPENDICES

# Appendix A: Extended Kalman Filter Equations

The extended Kalman filter (EKF) is the nonlinear version of the Kalman filter, which linearizes about an estimate of the current mean and covariance. These equations are widely known and there are many small variations to this process. The EKF works in the following general manner. The state transition and observation models do not need to be linear functions, and instead are represented as:

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \qquad \textit{Eq. 4}$$

$$z_k = h(x_k) + v_k \qquad \textit{Eq. 5}$$

Where $w_{k-1}$ and $v_k$ are the process and observation noises, which are both assumed to be zero mean value multivariate Gaussian noises with covariance $Q_k$ and $R_k$ respectively. $u_k$ is the control vector. At each time step, the Jacobian is evaluated with current predicted states. These matrices can be used in the Kalman filter equations. This process essentially linearizes the non-linear function around the current estimate.

The EKF equations are as follows:

**Predict**

Predicted state estimate $\boldsymbol{x}_{k|k-1} = f\left(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_{k-1}\right)$ *Eq. 6*

Predicted covariance estimate $\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_{k-1}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}_{k-1}^{\mathrm{T}} + \boldsymbol{Q}_{k-1}$ *Eq. 7*

**Update**

| | | |
|---|---|---|
| Innovation or measurement residual | $\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k\|k-1})$ | *Eq. 8* |
| Innovation (or residual) covariance | $\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_{k\|k-1} \boldsymbol{H}_k^{\mathrm{T}} + \boldsymbol{R}_k$ | *Eq. 9* |
| Near-optimal Kalman gain | $\boldsymbol{K}_k = \boldsymbol{P}_{k\|k-1} \boldsymbol{H}_k^{\mathrm{T}} \boldsymbol{S}_k^{-1}$ | *Eq. 10* |
| Updated covariance estimate | $\boldsymbol{P}_{k\|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k\|k-1}$ | *Eq. 11* |
| Updated state estimate | $\hat{\boldsymbol{x}}_{k\|k} = \hat{\boldsymbol{x}}_{k-1\|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k$ | *Eq. 12* |

Where the state transition and observation matrices are defined to be the Jacobian Matrices:

$$\boldsymbol{F}_{k-1} = \left.\frac{\partial f}{\partial x}\right|_{\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_{k-1}} \qquad \textit{Eq. 13}$$

$$\boldsymbol{H}_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{\boldsymbol{x}}_{k|k-1}} \qquad \textit{Eq. 14}$$

# Appendix B: Nonlinear Function Minimization

An integral portion to the tracking algorithms we built was the nonlinear function minimization tool in MATLAB's Optimization Toolbox. We used this function to optimize the model fit to point cloud data for both single and multi-frame trackers. It works by accepting a user-defined cost function (*func*), certain variables used to minimize this cost (***x***), and their initial values ($\boldsymbol{x_0}$). By using the gradient, which is either automatically derived or manually provided, it accurately returns the optimized variables ($\boldsymbol{x_f}$) to minimize the cost function. Since this is a local minimization, we need relatively accurate initial values. This tool minimizes the function equation:

$$\boldsymbol{C} = \min_{\boldsymbol{x}:\, \boldsymbol{x0} \rightarrow \boldsymbol{xf}} func(\boldsymbol{x}) \qquad \textit{Eq. 15}$$

For our purposes, we define the cost function as being the vehicle model cost function from chapter one, multiplied by the true point cloud data. The optimizable parameters of the cost function are the pose of the vehicle's rectangular model, which includes its position and orientation. The initial estimation of the vehicle's position and orientation are used to initialize the cost function. By tuning the estimated pose variables to minimize the overall cost function, the nonlinear minimization function is able to return the best estimation of the vehicle's true pose, based on the provided model.

# BIBLIOGRAPHY

# BIBLIOGRAPHY


Besl, P., & McKay, N. (1992). A method for registration of 3-d shapes. *PAM1*, 14(2):239-256.

Dellaert, F., & Thorpe, C. (1998). Robust car tracking using kalman filtering and bayesian templates. *Proceedings of SPIE*, vol 3207, p72.

Held, D., Levinson, J., & Thrun, S. (2013). Precision tracking with sparse 3D and dense color 2D data. *Robotics and Automation*, 1138-1145.

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 35-45.

Morris, D., Colonna, B., & Haley, P. (2006). Ladar-based mover detection from moving vehicles. *Proceedings of the 25th Army Science Conference*.

Morris, D., Haley, P., Zachar, W., & McLean, S. (2008). LADAR-Based Vehicle Tracking and Trajectory Estimation for Urban Driving. *Association for Unmanned Vehicle Systems International (AUVSI)*, (pp. 1-15). San Diego.

Morris, D., Hoffman, R., & Haley, P. (2009). A view-dependent adaptive matched filter for ladar-based vehicle tracking. *Proceedings of 14th IASTED Int. Conf. on Robotics and Applications*.

Petrovskaya, A., & Thrun, S. (2008). Model based vehicle tracking for autonomous driving in urban environments. *RSS, Zurich, Switzerland*.

Petrovskaya, A., & Thrun, S. (2009). Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26:123-139.

Pooja, A., & Govindu, V. (2010). A multi-view extension of the ICP algorithm. *Proc. 7th Indian conference on Computer Vision, Graphics and Image Processing*, 235-242.

Rusinkiewicz, S., & Levoy, M. (2001). Efficient variants of the ICP algorithm. *3DIM01*, 145-152.

Streller, D., Furstenberg, K., & Dietmayer, K. (2002). Vehicle and object models for robust tracking in traffic scenes using laser range images. *The IEEE 5th International Conference on Intelligent Transport Systems*, 118-123.

Wang, C. (2004). *Simultaneous localization, mapping and moving object tracking.* Pittsburgh, PA: Robotics Institute, Carnegie Mellon University.

Wang, D., Posner, I., & Newman, P. (2015). Model-free detection and tracking of dynamic objects with 2D lidar. *The International Journal of Robotics Research (IJRR)*.

Wender, S., & Dietmayer, K. (2008). 3D vehicle detection using a laser scanner and a video camera. *Intelligent Transport Systems (IET)*, 2(2):105-112.

Zhao, L., & Thorpe, C. (1999). Qualitative and quantitative car tracking from a range image sequence. *1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 496-501.