Innovation Cell

Obstacle tracking

Ruchika Chavhan

July 21, 2019

Preface

I am sorry if it's a little too detailed. Shillong is a really boring place so I had nothing to do! And I hate writing stuff in LaTeX. Video results are already on slack. Code has been pushed on git.

Contents

1 Obstacle tracking using 3D Lidar

1.1 The SLAM Problem

SLAM stands for Simultaneous Localization and Mapping.

A robot in a unknown arena, is exploring the area as well as following a desired trajectory. The task is to estimate the map of it's surroundings and path of the robot given the robot's control signals and observation of features.

To estimate a map of the surrounding, the bot needs the location with respect to initial location. For a good pose estimate, a map of the surroundings need to be built.

The SLAM can be lanmark based or map based. The structure of a landmark based SLAM is shown in figure 1.1.

In real world, both the map and the exact location of the bot is unknown. It can only measure them with available sensors but at the price of some error in observations of the sensor.

An example is provided in the figure below:

In the figure 1.2, the yellow stars represent true position of landmarks. The blue circles around the true landmarks represents the uncertainity with which the landmark is measured. This uncertainity arises due to the error in desired position of the bot.

1.1.1 Notations in SLAM

SLAM is a completely probability based approach. To avoid confusion in notation, we set a fixed notation through out

$x_{1:t}$ represnts the position of the bot from time 1 sec to t sec. $u_{1:t}$ represnts the control signal applied to the bot from time 1 sec to t sec. $z_{1:t}$ represnts the measurements made by the bot from time 1 sec to t sec. m represents the map of the surrounding.

From 1.3 and 1.4, the flow of data is understood. Using control signals, the bot is able to change it's poistion. Using it's position, it is able to estimate it's surroundings.

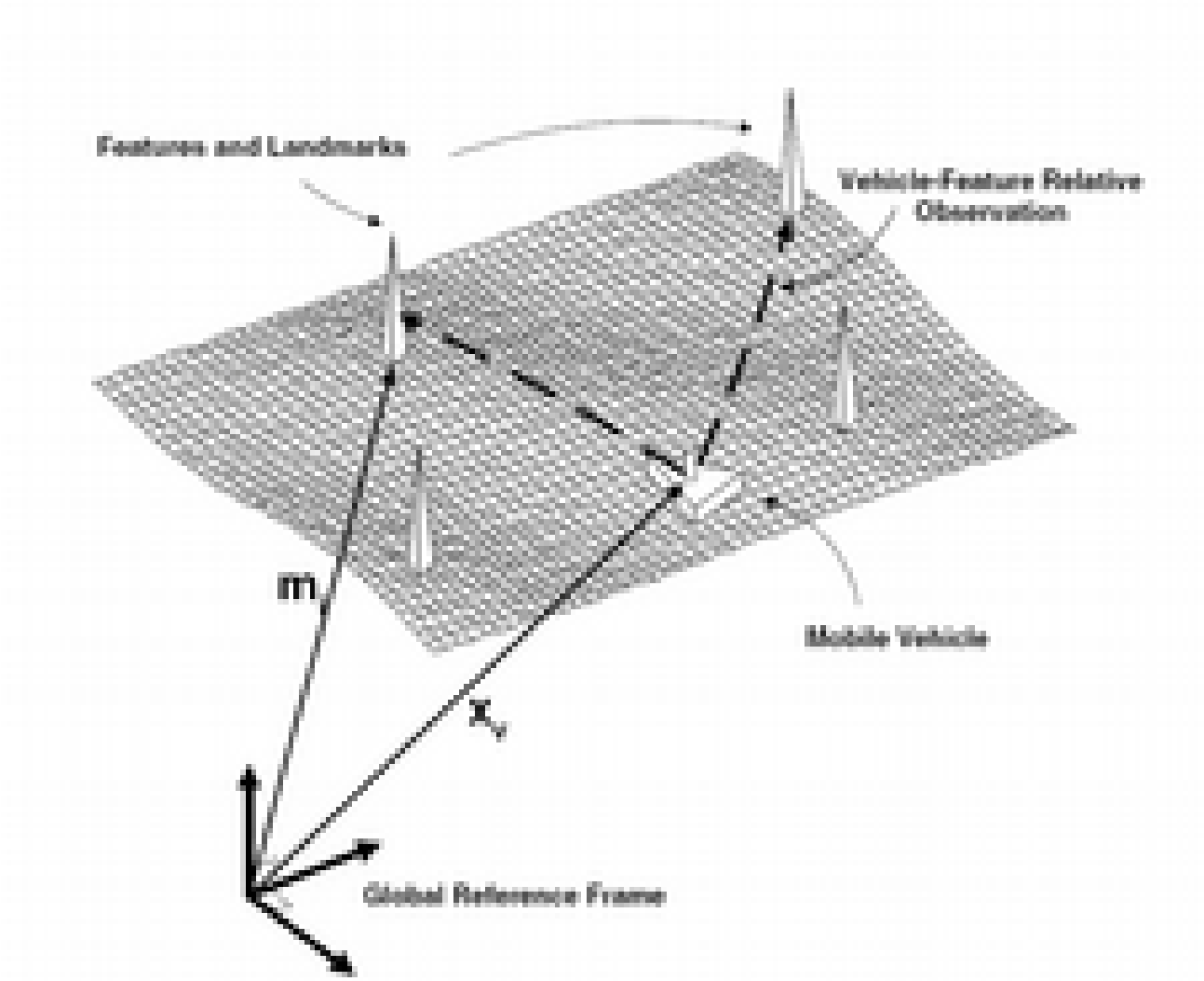


Figure 1.1: Structure of SLAM

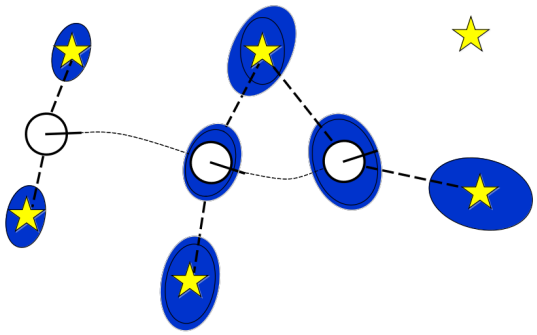


Figure 1.2: Accumulation of errors in both mapping and pose estimation

1.1.2 Full SLAM

The portions shaded in gray imply that the data is being carried forward to the next node. As suggested in figure 1.3, x_t uses data from all x_is i.e.$x_{0:t-1}$. This process can become slow with increase in data points and time.

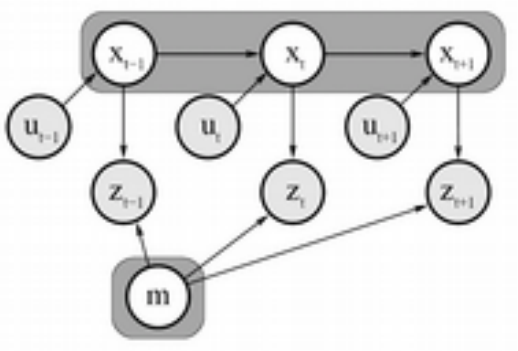


Figure 1.3: Flow of Full Slam

1.1.3 Online SLAM

As suggested in figure 1.4, only the previous data point is used.This is a much faster process and hold valid in most real life cases under Markov Assumption.

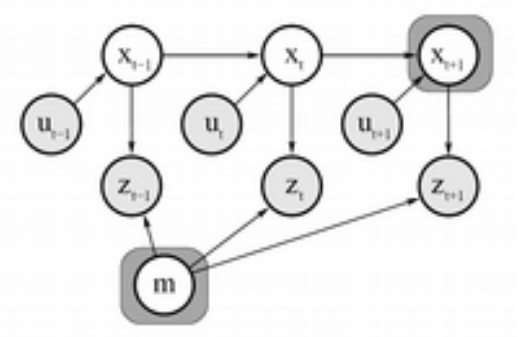


Figure 1.4: Flow of Online Slam

1.2 Bayesian Filters

Our goal is to estimate the state x given the observations z and controls u. We need to estimate $P(x|z,u)$.

Let us define a quantity $P(x_t|z_{1:t},u_{1:t})$ and let us call this belief ($bel(x_t)$). Assuming that the measurements are independant of one another and considering belief to represent a posterior probability, we have

$$bel(x_t) = \eta P(z_t|z_{1:t-1},u_{1:t},x_t)P(x_t|u_{1:t},z_{1:t-1})$$

where η is a normalisation constant. The above statement is obtained by applying Bayes' Rule.

Using the Markov Assumption,

$$bel(x_t) = \eta P(z_t|x_t)P(x_t|u_{1:t},z_{1:t-1})$$

Using the Law of Total Probability,

$$bel(x_t) = \eta P(z_t|x_t)\int P(x_t|x_{t-1},u_{1:t},z_{1:t-1})P(x_{t-1}|u_{1:t},z_{1:t-1})dx_{t-1}$$

Using the Markov Assumption twice in both the terms of the integrand,

$$bel(x_t) = \eta P(z_t|x_t)\int P(x_t|x_{t-1},u_t)P(x_{t-1}|u_{1:t-1},z_{1:t-1})dx_{t-1}$$

$$bel(x_t) = \eta P(z_t|x_t)\int P(x_t|x_{t-1},u_t)bel(x_{t-1})dx_{t-1}$$

We can observe that $bel(x_t)$ is recursively dependent on previous beliefs. Intuitively, the probability of estimating a state is dependent on previous states and measurements and the accumulation of errors in subsequent time steps.

The term $P(x_t|x_{t-1},u_t)$ is called the motion model of the filter. It describes how the motion of the bot evolves on providing control signals.

The term $P(z_t|x_t)$ is called the sensor model or observation model. It describes how measurements made by the bot change due to change in pose.

Any Bayesian Filter can be viwed as a two-step process:

STEP 1: Prediction step

$$bel(x_{t-1}) = \int P(x_t|x_{t-1}, u_t)bel(x_{t-1})dx_{t-1}$$

STEP 2: Correction step

$$bel(x_t) = \eta P(z_t|x_t)bel(x_(t-1))$$

As seen from the above two steps, the Prediction step involves the motion model and the Correction step involves the observation model

1.3 Kalman Filter

The Kalman Filter is regarded as the optimal solution to many tracking and data prediction tasks. It uses a linear motion model and assumes a gaussian noise with zero mean.

Therefore

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

$$z_t = C_t x_t + \delta_t$$

A_t is Matrix that describes how the state evolves from previous state without controls or noise.

B_t is Matrix that describes how the state evolves from $t-1$ to t when a particular control signal is applied.

C_t is Matrix that describes how to map the state to an observation.

ϵ_t is the error in measurement of state. It is a random variable assumed to be sampled from a gaussian distribution with zero mean and covariance R_t

δ_t is the error in estimating the map. It is a random variable assumed to be sampled from a gaussian distribution with zero mean and covariance Q_t

1.3.1 Linear Motion Model

Under the assumption of Gaussian noise,

$$P(x_t|u_t, x_{t-1}) =$$
$$det(2\pi R_t)^{\frac{-1}{2}} exp(-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t))$$

R_t describes the noise of the motion.

1.3.2 Linear Measurement Model

Under the assumption of Gaussian noise,

$$P(z_t|x_t) = det(2\pi Q_t)^{\frac{-1}{2}} exp(-\frac{1}{2}(z_t - C_t x_t)^T Q_{-1}(z_t - C_t x_t))$$

Q_t describes the noise of the measurement.

1.3.3 State space derivation

Assume that we want to know the value of a variable within a process of the form

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

and the observations model: $z_t = C_t x_t + \delta_t$

For the minimisation of the MSE to yield the optimal lter it must be possible to correctly model the system errors using Gaussian distributions. The covariances of the two noise models are assumed stationary over time and are given by

$$R = E[\epsilon_t \epsilon_t^T]$$

$$Q = E[\delta_t \delta_t^T]$$

Since our goal is to estimate x_t. The difference between predicted state $\hat{x}_t$ and x_t is called the error and is denoted by e_t. Our objective is to reduce the square of this difference. Therefore the loss function is:

$$lossfunction = E[e^2] = E[\hat{x} - x]$$

This is equivalent to $E[e_t(e_t)^T] = P_t$ where P_t is the error covariance matrix. The above equation can also be written as $E[e_t e_t^T] = P_t = E[(x - \hat{x})(x - \hat{x})^T]$

Assuming the prior estimate of $\hat{x}_t$ is called $\hat{x'}$, and was gained by knowledge of the system. It posible to write an update equation for the new estimate,combing the old estimate with measurement data thus

$x_t = \hat{x'} + K_t(z_t - C\hat{x'})$

where K_t is the Kalman gain. The term $C\hat{x'}$ comes from the measurement model which is linear. The term $(z_t - C\hat{x'})$ is called the innovation or measurement residual.

$$i_t = z - C\hat{x'}$$

Substitution of the measurement model in the above equation gives:

$$x_t = \hat{x'} + K_t(Cx_t + \delta_t - C\hat{x'})$$

Noticing that $e_t = (\hat{x} - x$, the error covariance P becomes:

$$P_t = E[[(I - K_tC)(x_t - \hat{x'}) - K_t\delta_t][(I - K_tC)(x_t - \hat{x'}) - K_t\delta_t]^T]$$

The above equation can re-written as:

$$P_t = (I - K_tC)E[(\hat{x_\prime} - x)(\hat{x_\prime} - x)^T](I - K_tC) + K_tE[\delta_t\delta_t^T]K_t^T$$

Notice that $E[(\hat{x_\prime} - x)(\hat{x_\prime} - x)^T]$ is the prior estimate of P_t and $E[\delta_t\delta_t^T] = Q_t$, where R_t is the covariance matrix for the measurement model.

Therefore,

$$P_t = (I - K_tC)P_t'(I - K_tC) + K_tQ_tK_t^T$$

The diagonal of the covariance matrix contains the mean squared errors as shown:

$$P_{kk} = \begin{bmatrix} E[e_{t-1}e_{t-1}^T] & E[e_te_{t-1}^T] & E[e_{t+1}e_{t-1}^T] \\ E[e_{t-1}e_t^T] & E[e_te_t^T] & E[e_{t+1}e_t^T] \\ E[e_{t-1}e_{t+1}^T] & E[e_te_{t+1}^T] & E[e_{t+1}e_{t+1}^T] \end{bmatrix}$$

The sum of the diagonal elements of a matrix is the trace of a matrix. In the case of the error covariance matrix the trace is the sum of the mean squared errors. Therefore, the mean squared error may be minimise by minimising the trace of P_t which in turn will minimise the trace of P_{kk}.

The trace of P_t is first differentiated with respect to K_t and the result set to zero in order to find the conditions of this minimum.

Rewriting the above equation:

$$P_k = P_k^{'} - K_t C P_k^{'} - P_k^{'} K_t^T C^T + K_t (C P_k^{'} C^T + Q_t) K_t^T$$

The trace of a matrix is equal to the trace of its transpose, therefore it may written be as:

$$T[P_t] = T[P_k^{'}] - T[K_t C P_k^{'}] - T[P_k^{'} K_t C] + T[K_t (C P_k^{'} C^t + Q_t) K_t^T]$$
$$T[P_t] = T[P_k^{'}] - 2T[K_t C P_k^{'}] + T[K_t (C P_k^{'} C^T + Q_t) K_t^T]$$

where T[.] is the trace of a matrix.

Differentiating with respect to K_t gives:

$$\frac{dT[P_t]}{dK_t} = -2(C P_k^{'})^T + 2K_t (C P_k^{'} C^T + Q_t)$$

Equating the right hand side to zero:

$$(C P_k^{'})^T = K_t (C P_k^{'} C^T + Q_t)$$
$$K_t = P_k^{'} C^T (C P_k^{'} C^T + Q_t)^{-1}$$

Thus, we have derived the expression for Kalman Gain.

We define $C P_k^{'} C^T + Q_t = S$, the associated measurement prediction covariance.

Substituting the value of Kalman Gain in the expression of P_t:

$$P_t = P_t^{'} - P_t^{'} C^T (C P_t^{'} C^T + Q_t)^{-1} C P_t^{'}$$
$$P_t = (1 - K_t C) P_t^{'}$$

The flow chart of Kalman filter is shown in 1.5

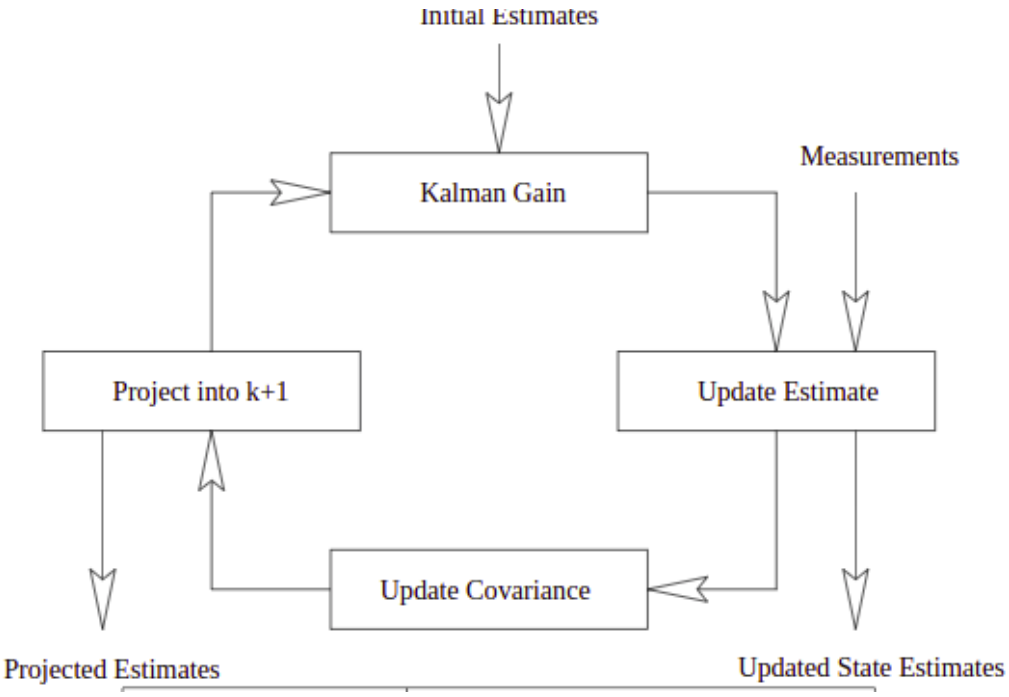


Figure 1.5: Flowchart of Kalman Filter

Update steps:

$$\text{Kalman Gain: } K_t = P_k^{'} C^T (C P_k^{'} C^T + Q_t)^{-1}$$

$$\text{Update state estimate: } x_t = \hat{x^{'}} + K_t(z_t - C\hat{x^{'}})$$

$$\text{Update measurement estimate: } z_t = C_t \hat{x}_t + \delta_t$$

$$\text{Update Covariance: } P_t = (1 - K_t C) P_t^{'}$$

$$\text{Project into } k+1\text{: } \hat{x_{'}} = A\hat{x} + B\hat{z} + \epsilon_t$$

Majority of the below theory has been refered from https://repository.tudelft.nl/islandora/object/uuid:f536b829-42ae-41d5-968d-13bbaa4ec736?collection=education. It is a master thesis by A.S. Abdul Rehman.

1.4 Extended Kalman Filter

Most realistic problems in robotics involve non-linear dynamics. We deal with Gaussian distributions while deriving the equations of Kalman Gain. The new estimated probabilities remain Gaussian because of the linear transformations. If the linearity is broken, the estimated probabilities will not be Gaussian thus breaking the initial assumption. Therefore, in Extended Kalman Filter we use local linearlization.

1.4.1 EKF Linearization

We use the first order taylor expansion of the non-linear function. Let $g(x_t, u_t)$ be a non linear function describing the motion model of the system.

The first order taylor expansion of $g(x_t, u_t)$ can be written as:

$$g(u_t, x_t) = g(u_t, \hat{x}) + \frac{dg(u_t, x_t)}{dx_{t-1}}(x_{t-1} - \hat{x_{t-1}})$$

where $\frac{dg(u_t, x_t)}{dx_{t-1}}$ is called the Jacobian Matrix. Let us denote it by G_t.

Let $h(x_t)$ be the non linear function describing the measurement model of the system.

The first order taylor expansion of $h(x_t)$ can be written as:

$$h(x_t) = h(\hat{x}) + \frac{dh(x_t)}{dx_{t-1}}(x_t - \hat{x_{t-1}})$$

where $\frac{dh(x_t)}{dx_t}$ is called the Jacobian Matrix. Let us denote it by H_t.

1.4.2 Linear Motion Model

$$P(x_t|u_t, x_{t-1}) = det(2\pi R_t)^{\frac{1}{2}} exp(-\frac{1}{2}(x_t - g(u_t, x_t))R_t^{-1}(x_t - g(u_t, x_t))^T)$$
$$P(x_t|u_t, x_{t-1}) = det(2\pi R_t)^{\frac{1}{2}} exp(-\frac{1}{2}(x_t - g(u_t, \hat{x}) - G_t(x_{t-1} - \hat{x_{t-1}})R_t^{-1}(g(u_t, \hat{x}) - G_t(x_{t-1} - \hat{x_{t-1}}))^T)$$

1.4.3 Linear Measurement Model

$$P(z_t|x_t) = det(2\pi Q_t)^{\frac{-1}{2}} exp(-\frac{1}{2}(z_t - h(x_t))Q_t^{-1}(z_t - h(x_t))^T$$

$$P(z_t|x_t) =$$

$$det(2\pi Q_t)^{\frac{-1}{2}} exp(-\frac{1}{2}(z_t - h(\hat{x}) + H_t(x_t - \hat{x_{t-1}})Q_t^{-1}(z_t - h(\hat{x}) + H_t(x_t - \hat{x_{t-1}})^T$$

1.4.4 Update steps in EKF

The update flowachart is same as 1.5, but only some matrices have been changed in the steps:

Kalman Gain: $K_t = P_k^{'} H^T (H P_k^{'} H^T + Q_t)^{-1}$

Update state estimate: $x_t = \hat{x^{'}} + K_t(z_t - H\hat{x^{'}})$

Update measurement estimate: $z_t = H_t \hat{x}_t + \delta_t$

Update Covariance: $P_t = (1 - K_t H) P_t^{'}$

Project into $k+1$: $\hat{x_{'}} = g(\hat{x}, \hat{z}) + \epsilon_t$

1.5 Unscented Kalman Filter

The UKF on the other hand, avoids computationally expensive linearization altogether and opt to use an approximation based on so-called sigma points from a Gaussian distribution. Instead of linearizing around the mean and using the Jacobian as the system function, the UKF propagates the chosen sigma points through the original non-linear function. The Gaussian then can be recovered from the newly transformed points. Note that the resulting probability density function is only an approximation of Gaussian distribution.

1.5.1 Sigma-Point Sampling

We form a set of sigma points and denote them with χ

We take into consideration $2L+1$ points, L being the number of points chosen on one side of the Gaussian.

$$\chi^0_{t-1|t-1} = x^*_{k-1|k-1}$$

$$\chi^i_{t-1|t-1} = x^*_{k-1|k-1} + (\sqrt{(L+\lambda)P^*_{k-1|k-1}})_i \text{ for } i = 0, 1, 2....L$$

$$\chi^i_{t-1|t-1} = x^*_{k-1|k-1} - (\sqrt{(L+\lambda)P^*_{k-1|k-1}})_{i-L} \text{ for } i = L+1, L+2.....2L+1$$

where $(\sqrt{(L+\lambda)P^*_{k-1|k-1}})_i$ is the square root of the i_{th} column of the square root of the matrix $((L+\lambda)P^*_{k-1|k-1})$.

1.5.2 Prediction step

The sigma points are transformed to a different set of points using a non-linear function f.

$$\chi^{*,i}_{t|t-1} = f(\chi^i_{t-1|t-1})$$

The transformed sigma points are used to calculated the predicted state and covariance matrix.

$$x_{t|\hat{t-1}} = \sum W^i_s \chi^{*,i}_{t|t-1}$$
$$P_{t|\hat{t-1}} = \sum W^i_c (x_{t-\hat{1}|t-1} - \chi^{*i}_{t|t-1})(x_{t-\hat{1}|t-1} - \chi^{*i}_{t|t-1})^T + Q_{t|t-1}$$

where the weights for state and covariance matrix are calculated as follows:

$$W^0_s = \frac{\lambda}{L+\lambda}$$
$$W^0_c = \frac{\lambda}{L+\lambda}(1-\alpha^2+\beta)$$
$$W^i_c = W^i_s = \frac{1}{2(L+\lambda)}$$
$$\lambda = \alpha^2(L+\kappa) - L$$

where α and κ control the spread of the sigma points. β is related to the distribution of x.

1.5.3 Update step

From the transformed sigma points, we then sample $2L+1$ sigma points to produce a prediction of the measurement.

$$\chi^0_{t|t-1} = x_{k|\hat{k-1}}$$
$$\chi^i_{t|t-1} = x_{k|\hat{k-1}} + (\sqrt{(L+\lambda)P^*_{k-1|k-1}})_i \text{ for } i = 0, 1, 2....L$$
$$\chi^i_{t|t-1} = x_{k|\hat{k-1}} - (\sqrt{(L+\lambda)P^*_{k-1|k-1}})_{i-L} \text{ for } i = L+1, L+2.....2L+1$$

Propagating these sigma points through the non-linear observation function h.

$$z_t^i = h(\chi_{t|t-1}^i)$$

Calculating the predicted measurement and predicted measurement covariance:

$$\hat{z}_t = \sum W_s^i z_{t|t-1}^i$$
$$\hat{P}_t = \sum W_c^i (\hat{z}_t - z_t^i)(\hat{z}_t - z_t^i)^T + R_{t|t-1}$$

The state-measurement cross-covariance matrix becomes

$$P_{x_t z_t} = \sum W_c^i (\hat{z}_t - z_t^i)(x_{t-\hat{1}|t-1} - \chi_{t|t-1}^{*i})$$

The expression of Kalman gain is:

$$K_t = P_{x_t z_t} P_{z_t z_t}^{-1}$$

1.6 Interacting Multiple Models

1.6.1 Introduction

The state estimation problem relies on the object motion model to predict and update the object states. However, in an urban situation tracked objects do not necessarily move in well-defined pattern. Even if a perfect motion model representing the object trajectories is available, there is no guarantee that the object will follow a specific motion model all the time.

IMM filter aims to generate better estimates for anobject with ambiguous dynamic behaviour. This approach used by IMM is as follows: by using multiple models representing different potential target manoeuvres to run state estimation filters and subsequently rank the most probable estimation results. Typically, the output of IMM is either a probability-weighted combination of the individual filters or the output of the filter with the highest probability. IMM consists of j filters running in parallel and each filter use different motion model to represent state evolution and measurement. It means for a single track, j state filter(s) is to be maintained.

First, the predicted state vectors and covariance matrices are generated by each individual filter estimates with respect to the predicted model probabilities. Next, each of the estimates is compared to current measurement to update the

model-match probability. During the update step, the computed probabilities of the dynamical target model are added to the updated state vector and error covariance matrix information produced by the corresponding filter.

The mode probabilities are then used in the so-called merge (or combination) and mixing step. During the merge stage, the state vectors and covariance matrices of each motion model are merged into a single state vector and covariance matrix. Finally, in the mixing process, new filtered state estimates, error covariance matrices and corresponding model probabilities are computed for each model using weighted state estimates. The derivation is as follows, we assume that the system with motion model uncertaintymis evolving as Jumping Markov Linear Systems (JLMS), the stochastic state space system dynamics and measurement equation representing each model j are then defined as:

$$x_{k+1} = f_j(x_t, u_t) + w_{j,t}$$
$$z_k = h_j(x_t, u_t) + v_{j,t}$$

where $j = 1, 2...r$ is part of model set $M = M_j^r = 1$

In the Bayesian framework, we can infer the posterior pdf. of IMM as $P(x_t, M_t|z_t)$ that is given all measurements z up until time k, with discrete state variable x_t , and M_t for the motion mode we can infer the estimate of joint pdf. We can further decompose using conditional probability lemma and further rewrite it to becomes

$$P(x_t, M_t|z_k) = \sum P(x_t, M_{j,k}|z_t)P(M_{j,k}|z_t)$$

$P(M_{j,k}|z_t)$ is also denoted by $u_{j,t}$ is the posterior mode probability that the object motion matches the dynamics of motion model ran by filter j at time t. In recursive form, the pdf becomes:

$$P(x_t, M_t|z_k) = \sum P(x_t, M_{j,k}|z_t)u_{j,t}$$

1.6.2 Mixing

IMM incorporates weighted average of each j_{th} model filter state $\hat{x_{i,k1}}$ to determine a single combined estimates state $\hat{x^*_{i,k1}}$ and its corresponding variance $P^*_{j,k1}$. This is done by summing up each model's joint posterior density, and the process is called mixing. Therefore, $\hat{x^*_{i,k1}}$ and $P^*_{j,k1}$ are given as:

$$\hat{x^*_{i,k1}} = \sum_{(i|j),k1} \hat{x_{i,k1}}$$
$$P^*_{j,k1} = \sum_{(i|j),k1} \hat{x_{i,,k1}}[P_{i,k1} + (\hat{x_{j,k1}} - \hat{x^*_{j,k1}})(\hat{x_{j,k1}} - \hat{x^*_{j,k1}})^T$$

1.6.3 Prediction

The prediction is made based on the mixed initial states $\hat{x^*_{j,k1}}$ and $P^*_{j,k1}$ by the individual j_{th} Kalman filter employing its own motion model.

The resulting model-specific predicted states are to be denoted as $x_{j,k}$ and $P_{j,k}$. In addition, we also have model-specific predicted measurement $z_{j,k}$ and innovation covariance $S_{j,k}$.

1.6.4 Updates

The resulting posterior states and innovation covariances are denoted by $\hat{x_{j,k}}$ and $\hat{P_{j,k}}$. The mode probability reflects the degree of fitness of current measurement to active model j, it is updated by:

$$u_{j,t} = \frac{\lambda_{ij}}{\sum \lambda_{ij} u_{i,k}}$$

where $\lambda_{j,k}$ is the Gaussian likelihood of the measurement given by

$$\lambda_{j,k} = det(2\pi S_{j,k})^{\frac{-1}{2}} exp(-\frac{1}{2}(z_k - \hat{z_k}) S^{-1}_{j,k} (z_k - \hat{z_k})^T)$$

1.6.5 Combination

The filter output of each j filter is then recombined as a single states x_k and P_k by:

$$\hat{x_k} = \sum u_{j,k} \hat{x_{j,k}}$$
$$\hat{P_k} = \sum Pj, k + (x_{j,k} - x)(x_{j,k} - x)^T$$

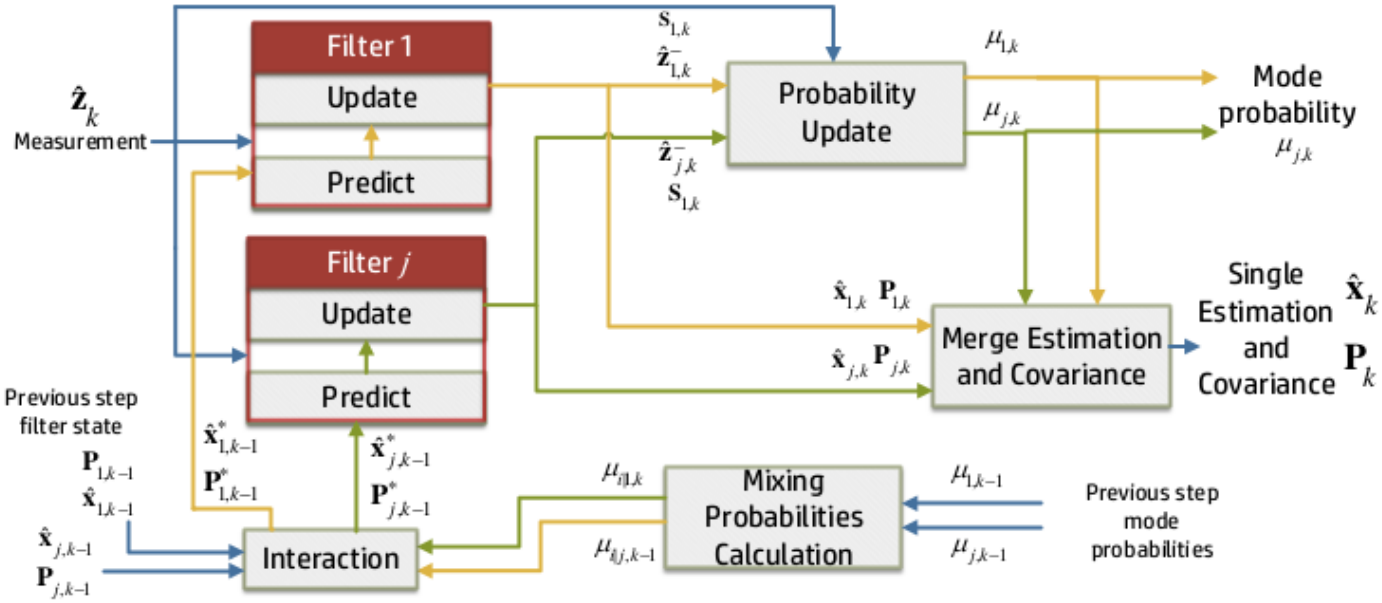


Figure 1.6: Schematic of IMM

1.7 Probabilistic Data Association Filter

PDAF and its derivative all use the incoming measurement inside a validation gate to approximate the probability distribution function of the tracked object after each update. This is done by assuming the measurement follows a Gaussian probability distribution.

1.7.1 Assumptions

1. Only one target of interest is present, whose state x is assumed to evolve in time according to the equation $x_{k-1} = f_{k-1}(x_{k-1}) + v_{k-1}$ with the true measurement z given by: $z_k = h_k(x_k) + w_k$. where v_{k-1} and w_k are zero-mean white Gaussian noise sequences with covariances matrices of Q_{k-1} and R_k .
2. The tracks are initialized.
3. The past information through time $k-1$ about the target is summarized approximately by a sufficient statistic in the form of the Gaussian posterior $P[x_{k-1}|z_{k-1}] = N[x_{k-1}; x_{k-\hat{1}|k-1}, P_{k-1|k-1}]$
4. If the target was detected and the corresponding measurement fell into the validation region, then, only maximum of one of the validated measurements can be target originated.
5. All non-object originated measurements are assumed to be originated from a clutter that is uniformly distributed in space and Poisson distributed in time.

The PDAF algorithm is divided into 4 main stages, Prediction, Measurement Validation, Data Association and State Estimation.

1.7.2 Prediction

The PDAF predict the state and covariance matrices one step ahead of time just like KF, given by

$$x_{k|\hat{k-1}} = f_{k-1}(x_{k-\hat{1}|k-1})\ z_{k|\hat{k-1}} = h_k(x_{k|\hat{k-1}})$$
$$P_{k|k-1} = f_{k-1}P_{k-1|k-1}f_{k-1} + Qk - 1$$

where $P_{k|k-1}$ is covariance matrix and the innovation covariance matrix corresponding to correct measurement given as $x_k = h_k P_{k|k-1} h_k^{'} + R_k$

1.7.3 Measurement Validation

In essence, detections that exceed a predefined distance threshold to the closest track are excluded. Those detections are considered as non-associable, (i.e. it is very unlikely that those detections represent objects which are already being tracked). Instead, those detections are considered to be potential objects that are not yet tracked and thus are used to create new object hypotheses.

A common approach of measurement gating is to assume the measurements are distributed according to a Gaussian, so the hyper-ellipsoid gate is used as gating region and the Mahalanobis distance of the obtained measurement state vector is computed and compared against that of predicted state vector[115]. A threshold based on inverse value χ_2 square distribution is often used, this threshold is called gate level. The measurement validation is done by the gating area given by elliptical region: $(V)(k,\gamma) = z : [zz_{k|\hat{k-1}}]^{'} x_k^{-1} [zz_{k|\hat{k-1}}] < \gamma$

where γ is the value of gate threshold equal to inv-$\chi_2(P_G)$, $P_(G)$ is the probability that the gate contains the true measurement if detected, and S_k is the covariance of the innovation corresponding to the true measurement. The volume of the validation region for q-dimensional measurement is expressed as

Finally, the set of validated measurements given as:

$$V_k = \frac{\pi^{\frac{q}{2}}}{\Gamma(\frac{q}{2}+1)}\sqrt{|\gamma \mathbf{S}_k|}$$

Figure 1.7: V is the elliptical region

1.7.4 Data Association

The non-parametric PDAF is used, we assume a diffuse prior clutter model clutter density, which is suitable for heterogeneous clutter environment. The association probabilities β of measurement i at time k is given as

$$\beta_{i,k}(x) = \begin{cases} \dfrac{e^{\frac{1}{2}(\mathbf{z}_{m,k}-\hat{\mathbf{z}}_{m,k|k-1})^T \mathbf{S}^{-1}(\mathbf{z}_{m,k}-\hat{\mathbf{z}}_{m,k|k-1})}}{(\frac{2\pi}{\gamma})^{\frac{q}{2}}\frac{m_k(1-P_D P_G)}{V_k P_D} + \sum_{j=1}^{m_k} e^{\frac{1}{2}(\mathbf{z}_{j,k}-\hat{\mathbf{z}}_{j,k|k-1})^T \mathbf{S}^{-1}(\mathbf{z}_{j,k}-\hat{\mathbf{z}}_{j,k|k-1})}}, i = 1, \\ \dfrac{(\frac{2\pi}{\gamma})^{\frac{q}{2}}\frac{m_k(1-P_D P_G)}{V_k P_D}}{(\frac{2\pi}{\gamma})^{\frac{q}{2}}\frac{m_k(1-P_D P_G)}{V_k P_D} + \sum_{j=1}^{m_k} e^{\frac{1}{2}(\mathbf{z}_{j,k}-\hat{\mathbf{z}}_{j,k|k-1})^T \mathbf{S}^{-1}(\mathbf{z}_{j,k}-\hat{\mathbf{z}}_{j,k|k-1})}}, i = 0 \end{cases}$$

Figure 1.8: Data association probabilities

1.7.5 State Estimation

The state update equation is given by

$$\hat{x_{k|k}} = x_{k|\hat{k-1}} + K_k v_k$$

with combined innovation $\hat{v_k}$ and filter gain K_k given as

$$v_k = \sum \beta_{j,k}(z_k - \hat{z_{k-1|k}})$$
$$K_k = P_{k|k-1} h_k^{'} x_k^{-1}$$

and associated covariance of updated state given by

$$Pk|k = \beta_{0,k} P_{k|k-1} + [1 - \beta_{0,k}](P_{k|k-1} - K_k S_k K_k^T) + K_k(\sum \beta_{j,k} v_{j,k} v_{j,k} - v_k v_k) K_k^T$$

the probability weighting $\beta_{0,k}$ of correct measurement is accounted in during update process. Note that P_k will get updated by probabilities weighting $(1\beta_{0,k})$ which correspond to the probabilities that correct measurement is received.

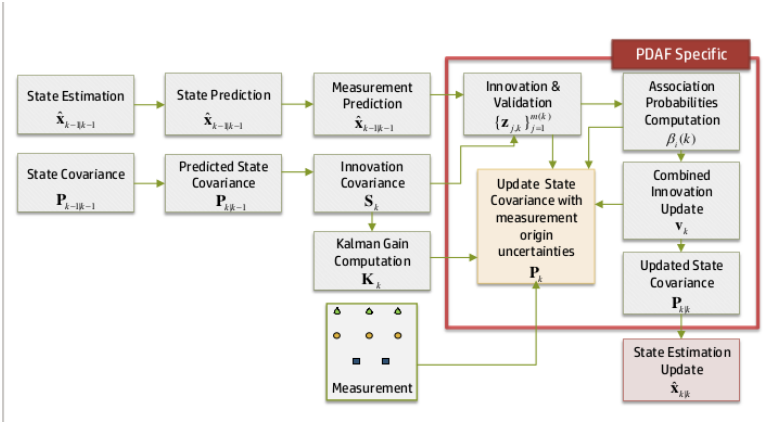


Figure 1.9: Cycle of JPDA

1.8 Ground Removal

The procedure is as follows: first, the raw LIDAR scans is divided into a polar grid with m_{bins} x $n_{channels}$ - cells with minimum radius r_{min} being the radius closest to ego vehicle and r_{max} being the farthest radius also from ego vehicle. The minimum radius is the radius in which the reflection of the ego vehicle is no longer can be seen while the maximum radius is determined by the effective range of the sensor. Mapping of each point cloud data $p_i = [x_i, yi, z_i, I_i]$ to each channel and bin is thus given as:

$$channel(p_i) = \frac{atan2(y_i, x_i)}{2\pi}$$
$$bin(p_i) = \frac{\sqrt{x_i^2 + y_i^2} - r_{min}}{2\pi}$$

Note the height information of the point coming from z_i is stored as the mapped cell's attribute. The lowest z_i point, hereby called prototype point is to be used as 'local' ground to determine the lowest possible point for all cells. Additionally, the absolute (local) height in the cell can be enumerated by computing the difference between lowest z_i and the highest z_i within the same cell.

Next, we define the interval $[T_{hmin} T_{hmax}]$ of possible ground height h_i for threshold-based classification. Since the sensor's mounting height above ground

level h sensor is known a priori, we can extrapolate that the closest point to the sensor must be situated above ground point.

Thus h_{sensor} is then used as the T_{hmax} for a point to be considered as ground. The cell information is filled from the bins which are closest to ego-vehicle to the farthest, if the prototype point of a cell lies inside the interval $[T_{hmin}T_{hmax}]$ then it is set as the h_i of that cell. If the prototype point is higher than T_{hmax} , then the ground level is set to be equal to that of h_{sensor}.

After all cells have been enumerated, a slope (simple Euclidean gradient) m between cells thus can be calculated to inspect if there is sudden height increase between cells. In addition, absolute height difference is also computed as a secondary check to identify possible non-ground object residing in the neighbouring cell.

Consistency Check and Median Filtering are employed to further flatten the ground plane and yield better ground estimate. Consistency Check is done by iterating non-ground cells which are flanked by non-ambiguous ground cells and then comparing the cells' height consistency with the neighbouring cells. The cell height is compared against a predefined absolute height T_{flat} and its height difference with adjacent cells is compared to threshold T consistency . A value below thresholds indicates the cells should belong to the ground and thus they are to be re-classified accordingly.

Median filtering on the other hand deals with missing ground plane information (common due to occlusion), as the name implies, the height value of the missing cell is to be replaced with the median value of neighbouring cells. The polar grid is again iterated to identify ground cells which have missing information but is surrounded by ground cells, the tunable parameter of the filter is kernel (window) size s_{kernel} which indicates the number of neighbouring cells involved.

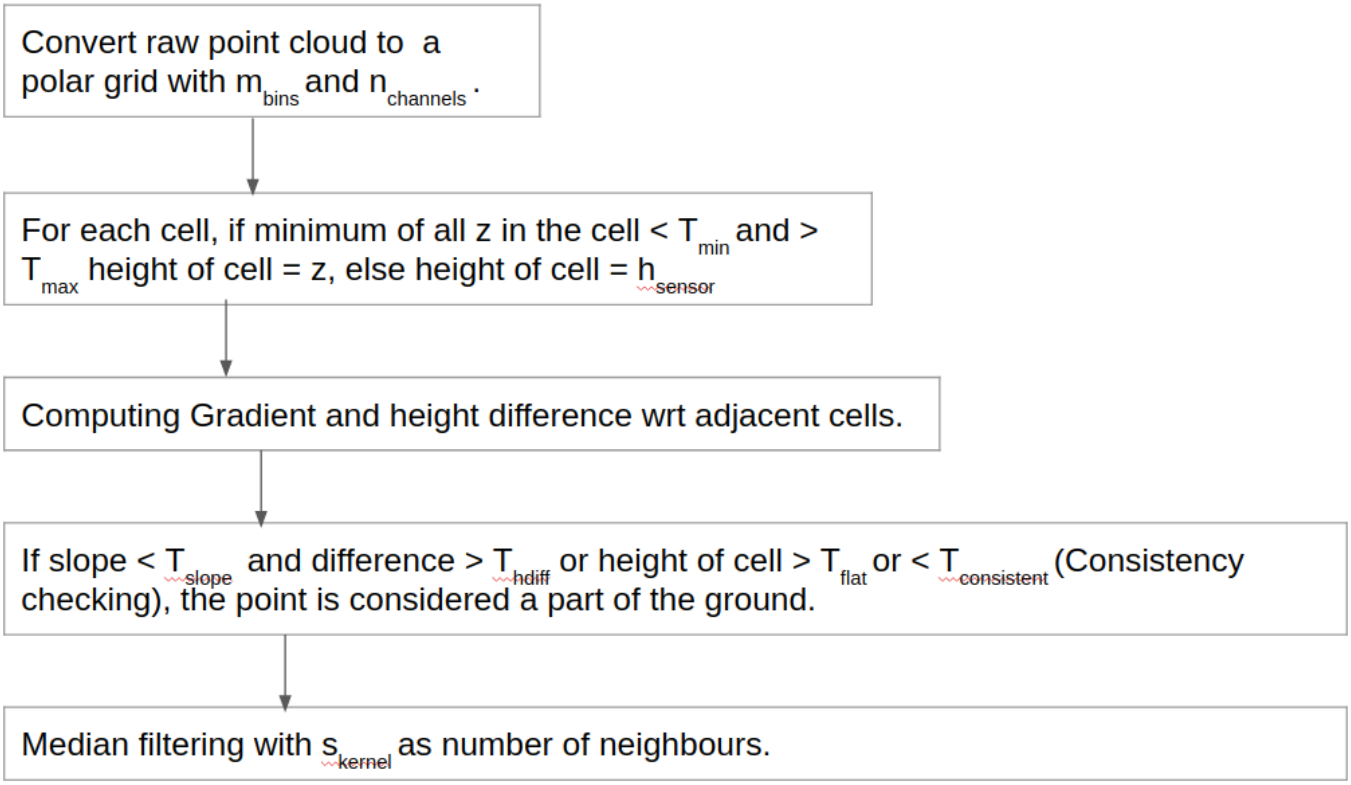
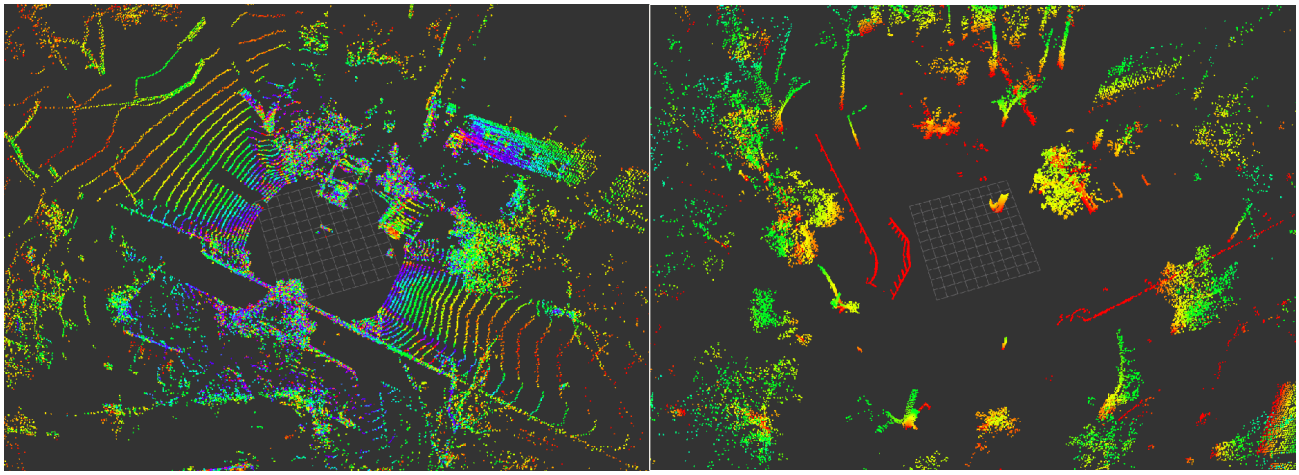
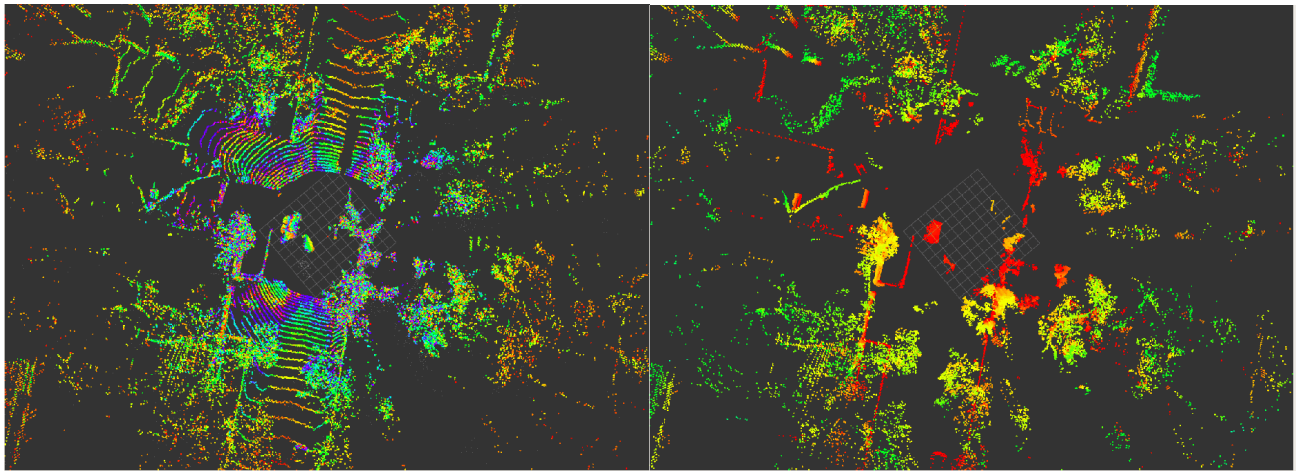


Figure 1.10: Ground Removal Algorithm



(a) Original Point Cloud (b) Point cloud with elevated points

Figure 1.11: Results of ground removal



(a) Original Point Cloud (b) Point cloud with elevated points

Figure 1.12: Results of ground removal

1.9 Object Clustering

Connected Component Clustering is utilised in order to distinguish each possible object in the elevated points. The connected component clustering is originally designed to find the connected region of 2D binary images. However, it is also applicable to LIDAR point cloud since in urban situation traffic object does not stack vertically, and thus the top view of the LIDAR measurement can be treated as a 2D binary image.

The XY plane of the elevated points is discretized into grids with m x n cells. The grid is assigned with 2 initial states, empty (0), occupied (-1) and assigned. Subsequently, a single cell in x, y location is picked as a central cell, and the clusterID counter is incremented by one. Then all adjacent neighbouring cells (i.e. x - 1, y + 1, x, y + 1, x + 1, y + 1 x - 1, y, x + 1, y, x - 1, y - 1, x, y - 1, x + 1, y + 1) are checked for occupancy status and labelled with current cluster ID. This procedure is repeated for every and each x, y in the m x n grid, until all non-empty cluster has been assigned an ID. The clustered are displayed in different colors as in 1.16.

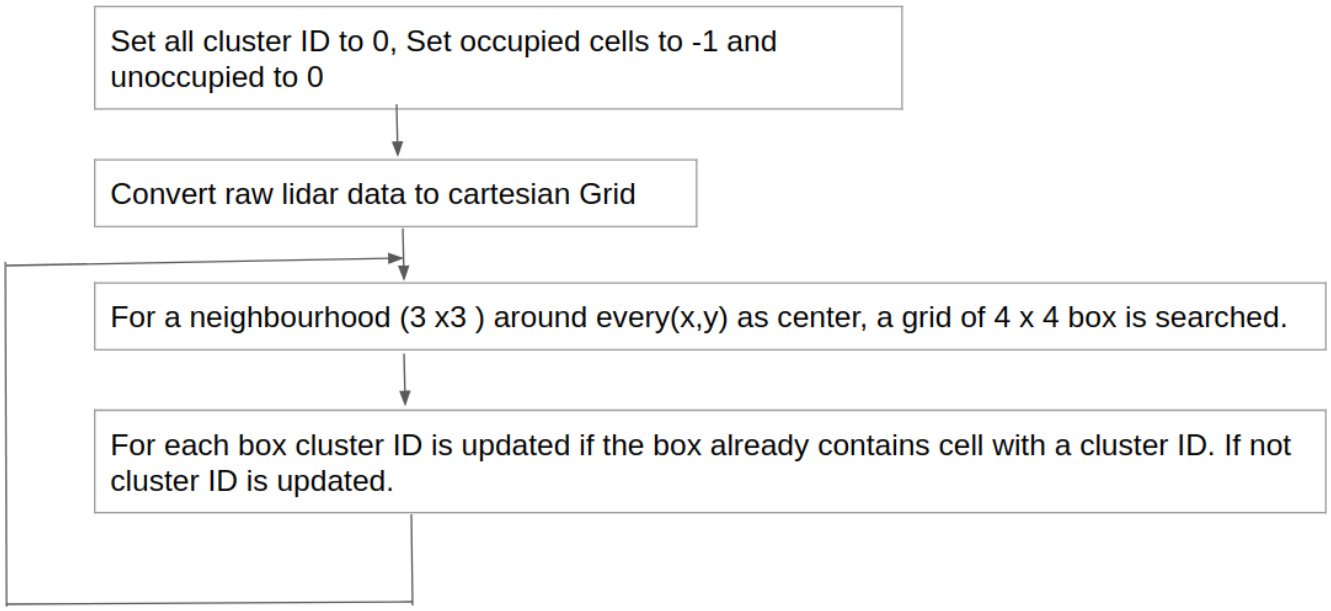
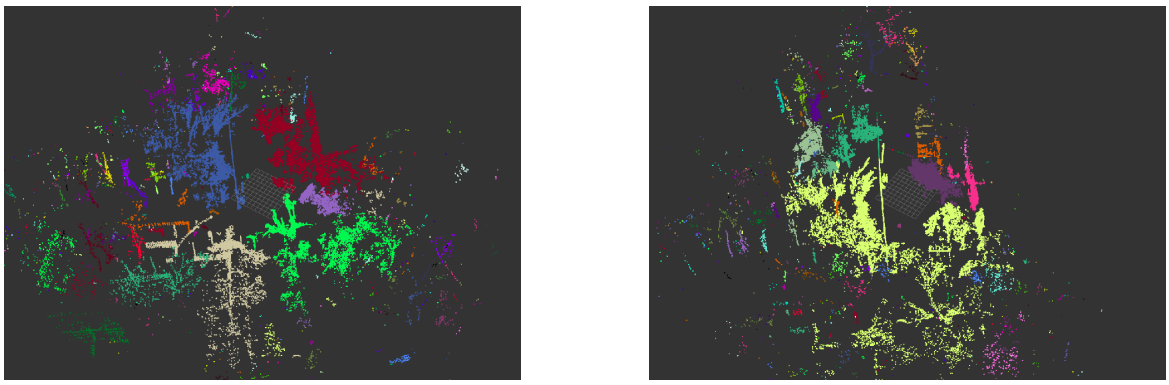


Figure 1.13: Component Clustering Algorithm



(a) (b)

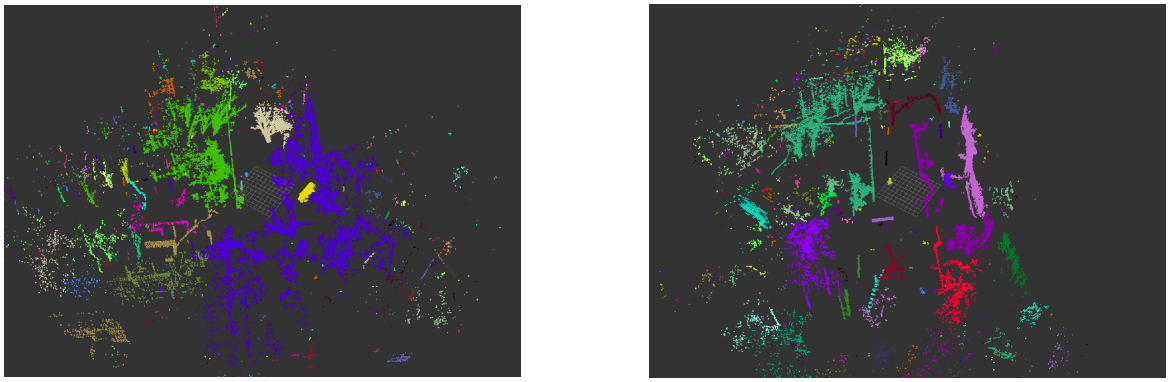
Figure 1.14: Results of Component Clustering

1.10 Bounding Box Fitting

A Minimum Area Rectangle (MAR) is applied to each clustered object which results in a 2D box, and when combined with height information retained in the clustering process, becomes a 3D bounding box.

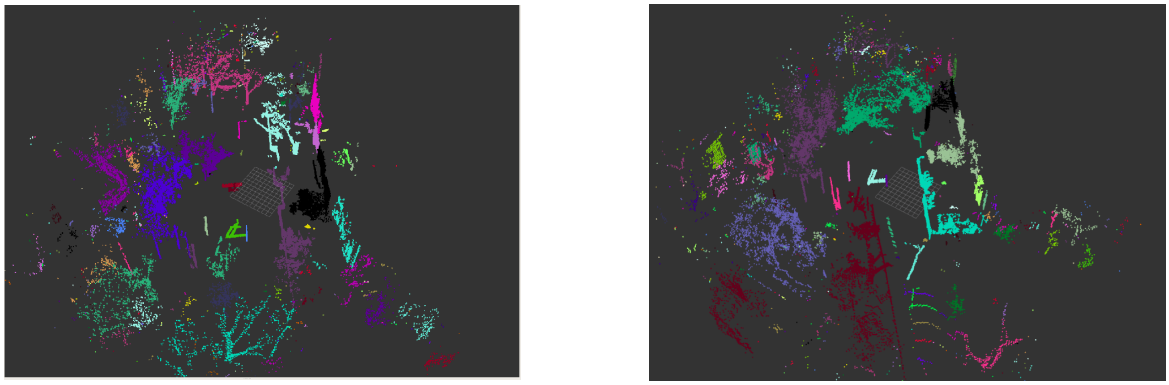
The L-shape fitting procedure is done as follows: first, we select two farthest outlier point x_1 and x_2 which lies on the opposite sides of the object facing the LIDAR sensor. A line L_d is then drawn between the two points, then an orthogonal line L_o is projected from L_d toward the available points.

The projected L_o with maximum distance d max and angle close to 90 degrees



(a) (b)

Figure 1.15: Results of Component Clustering



(a) (b)

Figure 1.16: Results of Component Clustering

is then obtained using iteration end-point fit algorithm. The points connected to the orthogonal line then becomes the corner point x_3 . Closing a line between x_1, x_2 and x_3 would form an L-shaped polyline. The orientation of the bounding box is then determined by the longest line, as most traffic object (e.g. car, cyclist) is heading in parallel with its longest dimensional side.

1.11 Rule Based Filter

Some of the bounding boxes obtained from the above step, can be objects of non-interest. To track vehicles, other objects like walls, bushes, pedestrains, bikers etc, must be neglected. Noticing that cars and other four wheelers have significantly different dimensional profile when compared to other objects. Therefore, some thresholds have been decided to discard such objects.

Some thresholds used are:

1. $T_{height_(min|max)}$ min and max height of the object.
2. $T_{width_(min|max)}$ min and max height of the object.
3. $T_{length_(min|max)}$ min and max length of the object.
4. $T_{area_(min|max)}$ min and max top-view area of the object.
5. $T_{ratio_(min|max)}$ min and max ratio of length and width.
6. $T_{ratiocheck_l(min)}$ min length of object for ratio check.
6. $T_{pt_per_m3_(min)}$ min point count per bounding box volume.

```
float tHeightMin = -2.00;
float tHeightMax = 3.00;

float tWidthMin = 0.25;
float tWidthMax = 4.5;

float tLenMin = 0.1;
float tLenMax = 20.0;

float tAreaMax = 4.0;

float tRatioMin = 0.5;
float tRatioMax = 4.5;

float minLenRatio = 0.5;
float tPtPerM3 = 8;
```

Figure 1.17: Parameters

1.12 IMM-UKF-JPDA

IMM-UKF-JPDA stands for Interacting Multiple Model- Unscented Kalman Filter-Joint Probability Data Association. The IMM-UKF-JPDA consists of four steps: Interaction, Prediction-and-Measurement Validation Step, Data Association-and-Model-Specific Filtering Step, and Mode Probability Update-and-Combination Step.

Now we will describe in detail the state vector and measurement vector we use.

It is reasonable to assume that road objects does not have vertical motion (in Z-axis) and thus we are left with measurement in the position vector of z in x and y-axes:

$$z = \begin{bmatrix} x_{pos} \\ z_{pos} \end{bmatrix}$$

The tracking target dynamic motion model can be described mathematically by a discrete time, stochastic state-space model in the form of:

$$x_{k+1} = f_k(x_k, u_k) + w_k \; z_k = H(x_k, u_k) + v_k$$

with state vector x_k, measurement vector z_k system function F , and measurement function H at each time step k. The system is disturbed with zero-mean, white, Gaussian noise sequences w_k and v_k which are mutually independent with covariance matrices Q_k (motion noise) and R_k (measurement noise).

In most traditional models, the state vector includes the position, the velocity, and the acceleration in both dimensions, here we also incorporate the velocity magnitude and heading angle

The state vectors x_k thus becomes:

$$x_k = \begin{bmatrix} x_{posk} \\ z_{posk} \\ k \\ v_k \\ \dot{\psi}_k \end{bmatrix}$$

with x_{posk} is the position in the x-axis, z_{posk} is the position in the y-axis, ψ_k is the yaw (heading), v_k is the magnitude of velocity and $\dot{\psi}_k$ is the yaw rate, all computed at time instance k.

1.12.1 Interaction

This step utilises the IMM probability mixing of different models. In our case, we have used three different models namely the Constant Velocity, constant Turn rate and Random Motion. Motion of target objects is assumed to be superposition of these three motion models.

Constant Velocity Constant Velocity (CV) Model represents motion behaviour of an object with nearly constant velocity and the object is assumed to have zero turning rate and thus is heading to the same bearing on all time steps. The system function is given as:

$$x_k = \begin{bmatrix} x_{posk} + v_k T sin(\psi_k) \\ z_{posk} + v_k T cos(\psi_k) \\ 0 \\ v_k \\ \dot{\psi}_k \end{bmatrix}$$

CONSTANT TURN RATE The assumptions are velocity is (nearly) constant and the turn rate is also constant. These assumptions are appropriate for both rectilinear motion with a uniform acceleration and a coordinated turn

$$x_k = \begin{bmatrix} x_{posk} + \frac{v_k}{\dot{\psi}_k}(-sin(\psi_k) + sin(T\psi_k + \psi_k) \\ z_{posk} + \frac{v_k}{\dot{\psi}_k}(cos(\psi_k) - cos(T\psi_k + \psi_k) \\ T\psi_k + \psi_k \\ v_k \\ \dot{\psi}_k \end{bmatrix}$$

RANDOM MOTION The Random Motion model uses the state vector values at the time instant in ehich they are measured.

$$x_k = \begin{bmatrix} x_{posk} \\ z_{posk} \\ k \\ v_k \\ \dot{\psi}_k \end{bmatrix}$$

This step utilises the IMM probability mixing, The conditional mode probabilities $u_{(i|j),k-1}$ correspond to probabilities that mode i has been established in the previous cycle given that mode j is active in current time step is given by:

$$u_{(i|j),k-1} = \frac{p_{ij} u_{ik}}{\bar{u_{j,k}}}$$

where $\bar{u_{j,k}} = (\bar{u_{1,k}}, \bar{u_{2,k}} ... \bar{u_{r,k}})$ is the a priori mode probabilities at the current time step, which in turn is given by prediction of the mode probabilities of the last time step p_{ij} , which denotes the probability that a mode transition occurs from model i to model j:

$$\bar{u_{j,k}} = \sum p_{ij} u_{ik}$$

Then, assuming we have j individual filters, combined initial state(s) $x_{j,\hat{k-1}}$ and covariance $P_{j,k-1}$ are mixed from each filter j to form single initial state $\hat{x^*_{i,k1}}$ and covariance $P^*_{j,k1}$. They are given by calculating:

$$\hat{x^*_{i,k1}} = \sum {}_{(i|j),k1} \hat{x_{i,,k1}}$$
$$P^*_{j,k1} = \sum {}_{(i|j),k1} \hat{x_{i,,k1}} [P_{i,k1} + (\hat{x_{j,k1}} - \hat{x^*_{j,k1}})(\hat{x_{j,k1}} - \hat{x^*_{j,k1}})^T$$

1.12.2 Prediction and Measurement Validation Step

This step involves the implementation of all streps of the UKF.

$$\chi^0_{t-1|t-1} = x^*_{k-1|k-1}$$
$$\chi^i_{t-1|t-1} = x^*_{k-1|k-1} + (\sqrt{(L+\lambda)P^*_{k-1|k-1}})_i \text{ for } i = 0, 1, 2....L$$
$$\chi^i_{t-1|t-1} = x^*_{k-1|k-1} - (\sqrt{(L+\lambda)P^*_{k-1|k-1}})_{i-L} \text{ for } i = L+1, L+2.....2L+1$$

where $(\sqrt{(L+\lambda)P^*_{k-1|k-1}})_i$ is the square root of the i_{th} column of the square root of the matrix $((L+\lambda)P^*_{k-1|k-1})$.

$$\chi^{*,i}_{t|t-1} = f(\chi^i_{t-1|t-1})$$

The transformed sigma points are used to calculated the predicted state and covariance matrix.

$$\hat{x_{t|t-1}} = \sum W^i_s \chi^{*,i}_{t|t-1}$$
$$\hat{P_{t|t-1}} = \sum W^i_c (\hat{x_{t-1|t-1}} - \chi^{*i}_{t|t-1})(\hat{x_{t-1|t-1}} - \chi^{*i}_{t|t-1})^T + Q_{t|t-1}$$

where the weights for state and covariance matrix are calculated as follows:

$$W^0_s = \frac{\lambda}{L+\lambda}$$
$$W^0_c = \frac{\lambda}{L+\lambda}(1 - \alpha^2 + \beta)$$
$$W^i_c = W^i_s = \frac{1}{2(L+\lambda)}$$
$$\lambda = \alpha^2(L+\kappa) - L$$

where α and κ control the spread of the sigma points. β is related to the distribution of x.

From the transformed sigma points, we then sample $2L+1$ sigma points to produce a prediction of the measurement.

$$\chi^0_{t|t-1} = x_{k|\hat{k}-1}$$
$$\chi^i_{t|t-1} = x_{k|\hat{k}-1} + (\sqrt{(L+\lambda)P^*_{k-1|k-1}})_i \text{ for } i = 0, 1, 2....L$$
$$\chi^i_{t|t-1} = x_{k|\hat{k}-1} - (\sqrt{(L+\lambda)P^*_{k-1|k-1}})_{i-L} \text{ for } i = L+1, L+2.....2L+1$$

Propagating these sigma points through the non-linear observation function h.

$$z^i_t = h(\chi^i_{t|t-1})$$

Calculating the predicted measurement and predicted measurement covariance:

$$\hat{z}_t = \sum W^i_s z^i_{t|t-1}$$
$$\hat{P}_t = \sum W^i_c (\hat{z}_t - z^i_t)(\hat{z}_t - z^i_t)^T + R_{t|t-1}$$

The state-measurement cross-covariance matrix becomes

$$P_{x_t z_t} = \sum W^i_c (\hat{z}_t - z^i_t)(x_{t-\hat{1}|t-1} - \chi^{*i}_{t|t-1})$$

The expression of Kalman gain is:

$$K_t = P_{x_t z_t} P^{-1}_{z_t z_t}$$

1.12.3 Data Association and Model-Specific Filtering Step

This step is implemented directly from the PDA filter.

$$x_{\hat{k}|k} = x_{k|\hat{k}-1} + K_k v_k$$

$$v_k = \sum \beta_{j,k}(z_k - z_{\hat{k-1}|k})$$
$$K_k = P_{k|k-1} h'_k x^{-1}_k$$

and associated covariance of updated state given by

$$Pk|k = \beta_{0,k} P_{k|k-1} + [1 - \beta_{0,k}](P_{k|k-1} - K_k S_k K^T_k) + K_k(\sum \beta_{j,k} v_{j,k} v_{j,k} - v_k v_k) K^T_k$$

1.12.4 Model Probability Update and Combination Step

The mode probabilities are updated based on the likelihood the measurement fit to a model, given as

Finally, each filter updated states are to be combined again into single final state and covariance estimate:

$$\hat{x_k} = \sum u_{j,k} \hat{x_{j,k}}$$
$$\hat{P_k} = \sum Pj, k + (x_{j,k} - x)(x_{j,k} - x)^T$$

1.12.5 Schematic of IMM-UKF-JPDA

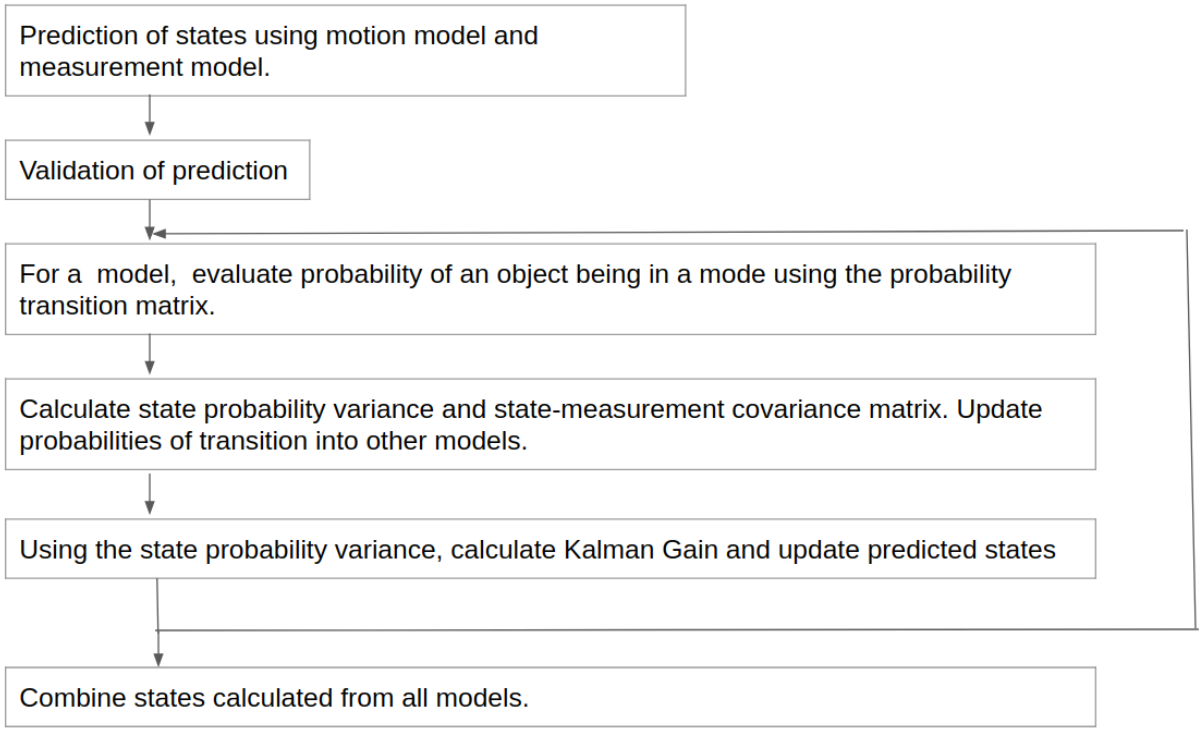


Figure 1.18: IMM-UKF-JPDA

1.13 Bounding box Tracking

The IMM-UKF-JPDA filter provides estimated poses for point-based object, in order to incorporate dimensional and heading information (i.e. that of bounding box) a logic-based algorithm is implemented which essentially has three major functionalities:

(1) Associate detected box with a track
(2) to retain detector best-known output and if later a better detection output is found, update the retained bounding box parameter
(3) perform geometrical correction to compensate occlusion and ego-car perspective change.

1.14 Tracking states and maturity

First, to distinguish the track status, each and every track is assigned a unique finite 'state' to reflect its validity and maturity status of the track. The state numbers also act as a scoring threshold. A track that has successful measurement association will be assigned state = 1 (Initializing) at time frame k, and if at time frame k + 1 another successful association occurs, the state number is incremented by 1 until the state number has reached number of 3. After that, in the next time frame the track state will be upgraded to Tracking (state number 5).

1. 0: Invalid

2. 1-3: Initializing

3. 5: Track with associated measurement that passes gating for more than n time frame (here n = 3)

4. 7-10: Track with lost measurement may return to Tracking state if measurement is found again in the future

1.14.1 Bounding box association

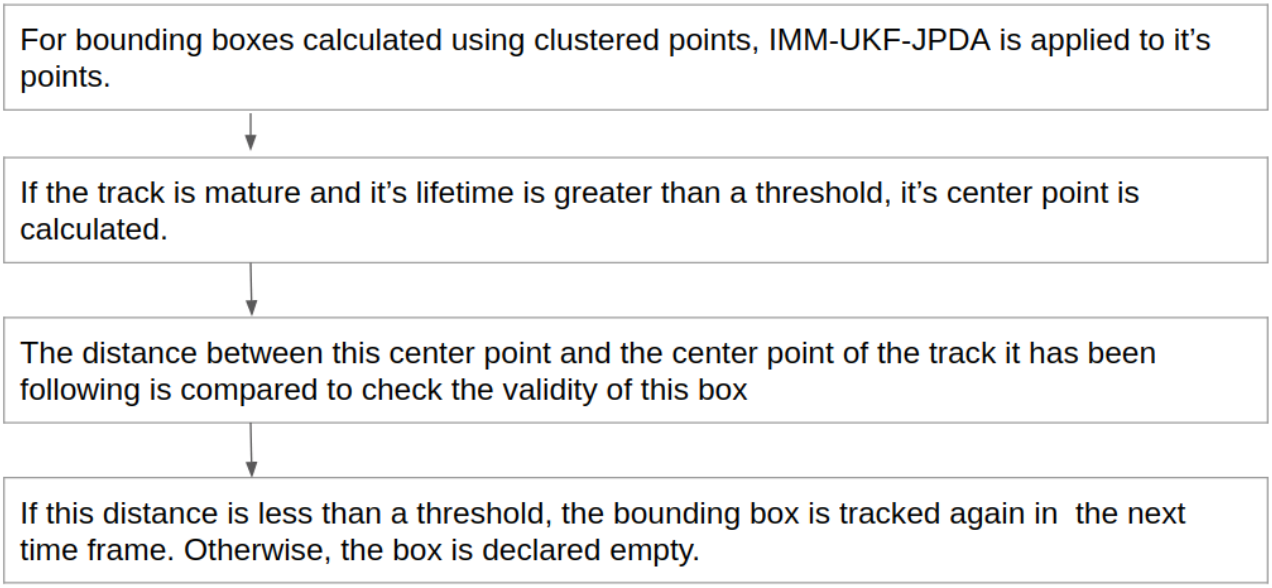


Figure 1.19: Bounding Box association algorithm

1.14.2 Bounding box update

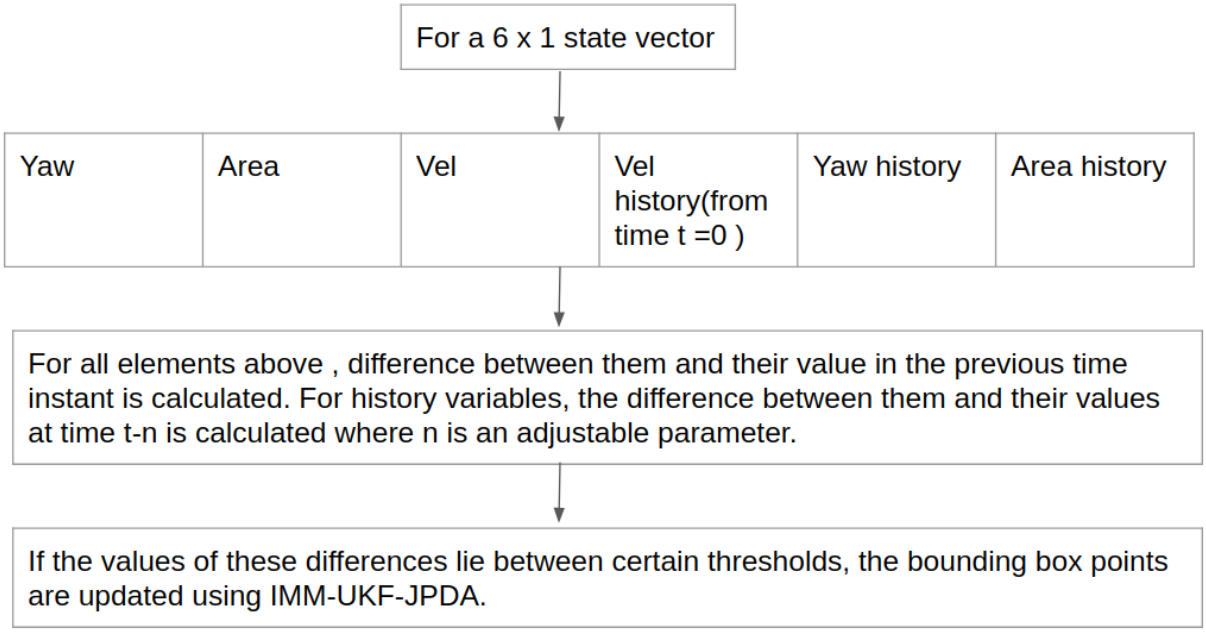


Figure 1.20: Bounding Box update algorithm

1.15 Results on real life Data

This section contains results of the above on real life data captured by Ouster OS1 64-channel Lidar.

The green boxes are the bounding boxes fit into the clustered cloud. The green arrows projecting around the center is the angle of yaw with respect to the ego-vehicle. The length of the arrow represents the velocity of the object.

As one can notice from the images, the algorithm is successful in neglecting false positive data points and only focusses on vehicles, pedestrains and trees.

The paramenters have been tuned to values so that pedestrians are also detected. This is done because self driving cars need to be extra careful while tracking people. Tress have been included to take care of some scenarios in Indian Road Conditions.

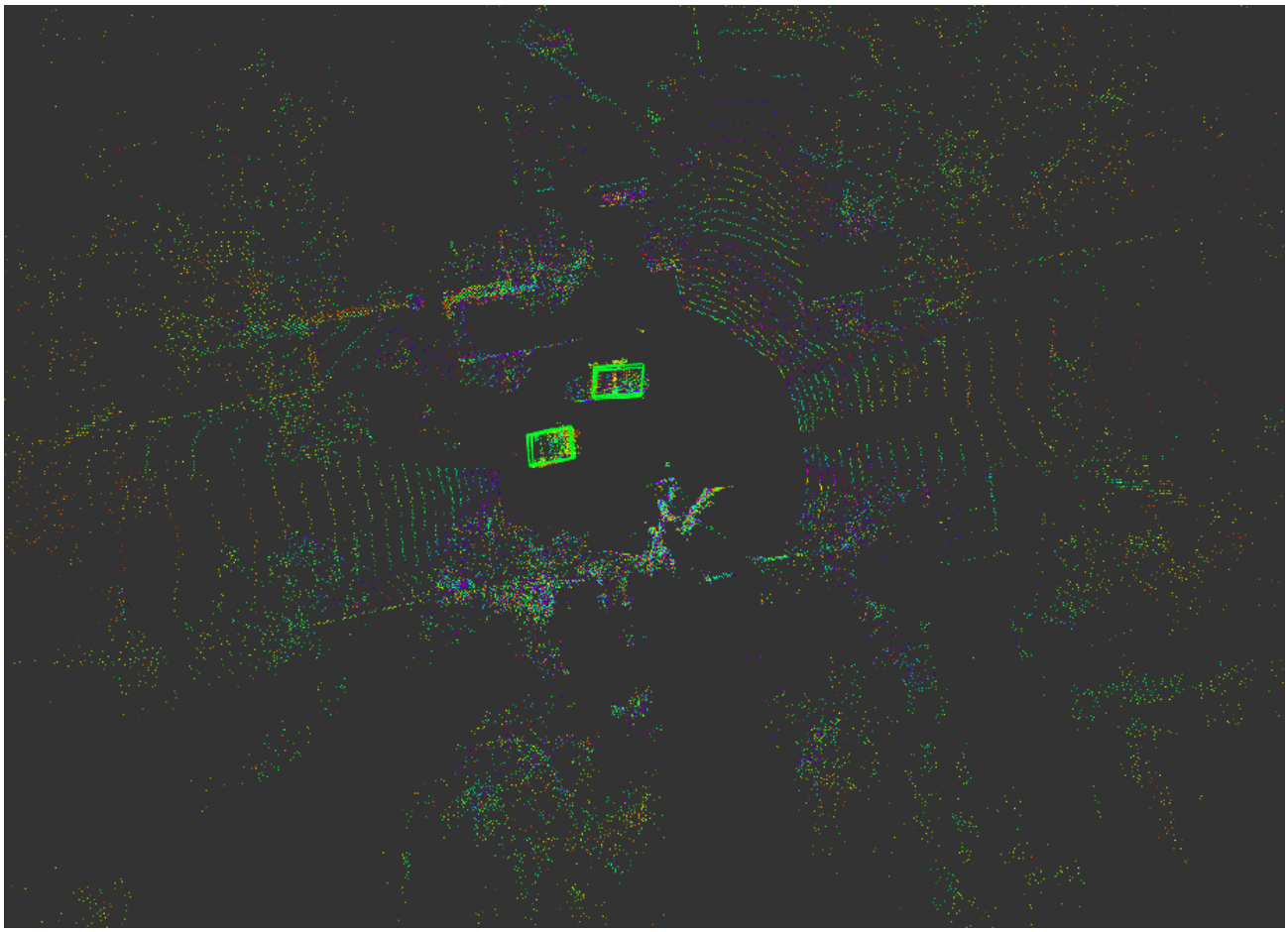


Figure 1.21: Bounding Boxes fit into cars

Two cars are detected in the above figure.

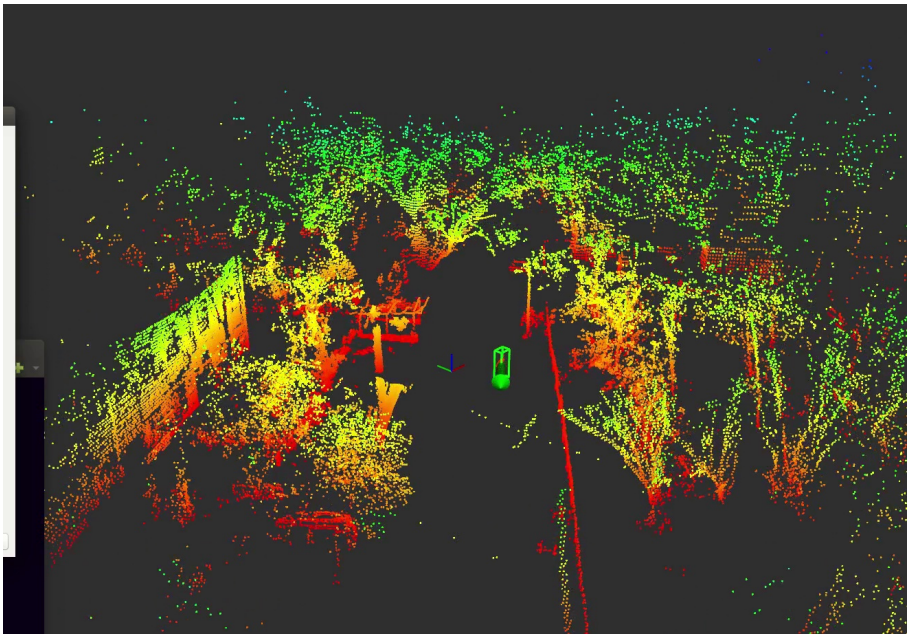


Figure 1.22: Results on 3D Lidar Data

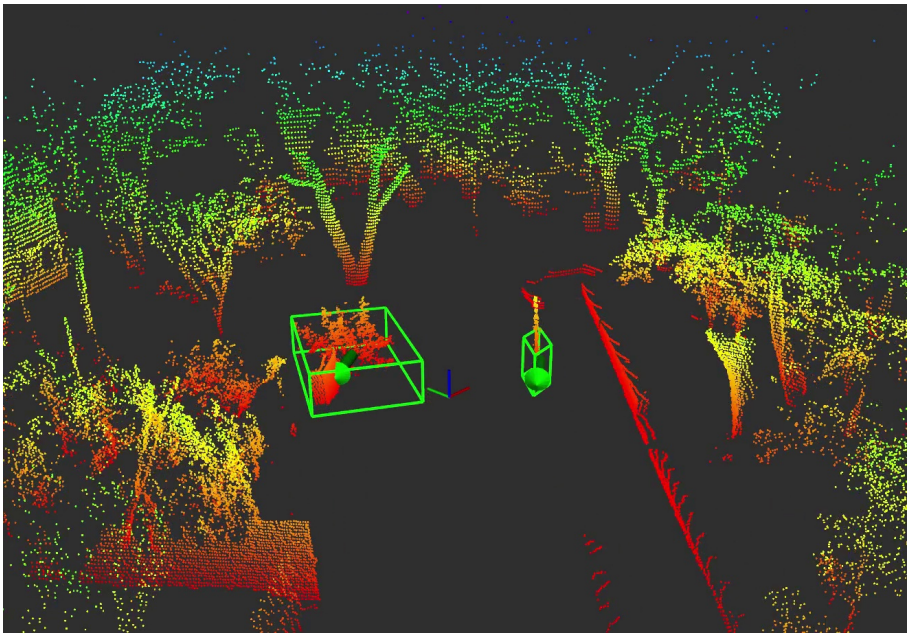


Figure 1.23: Results on 3D Lidar Data

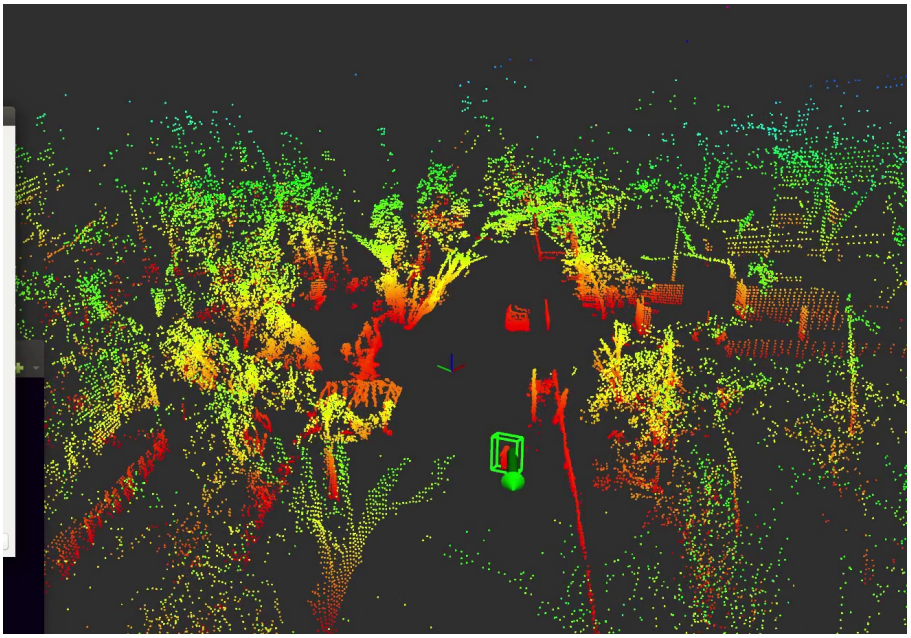


Figure 1.24: Results on 3D Lidar Data

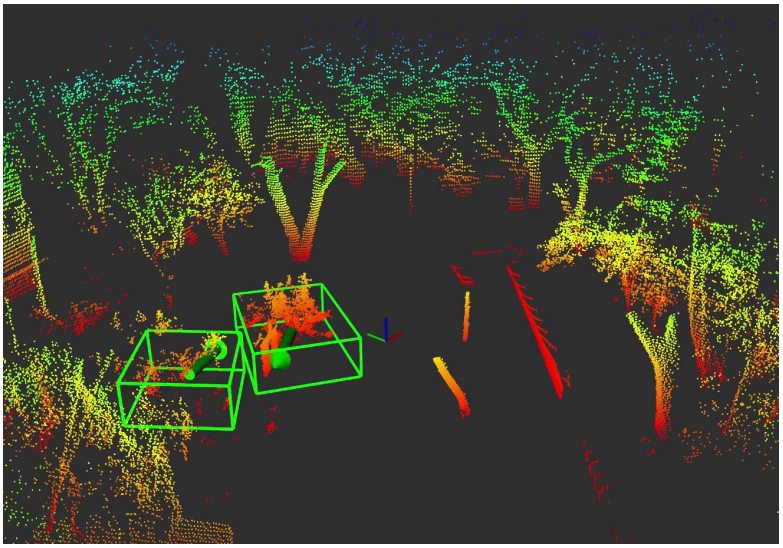


Figure 1.25: Results on 3D Lidar Data

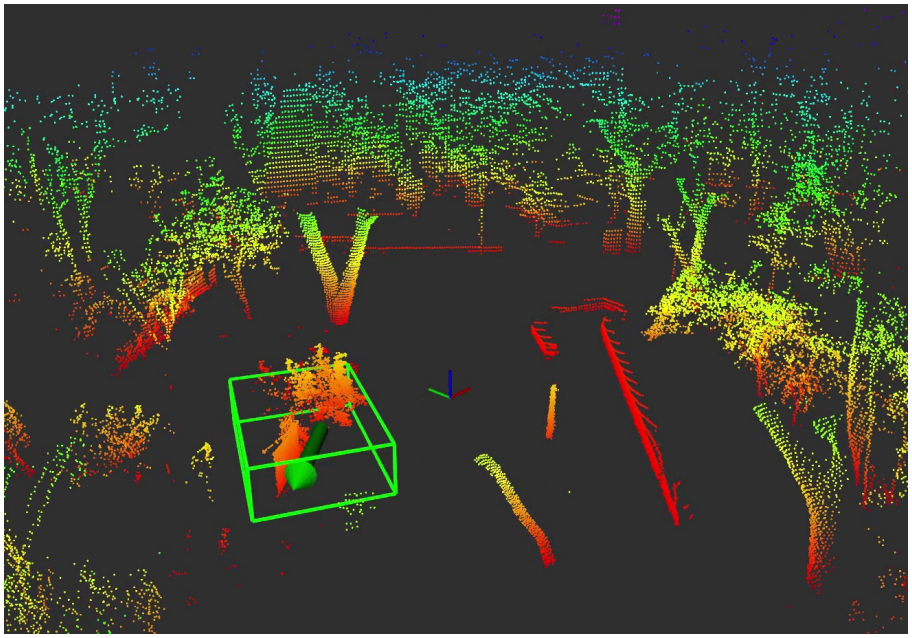


Figure 1.26: Results on 3D Lidar Data

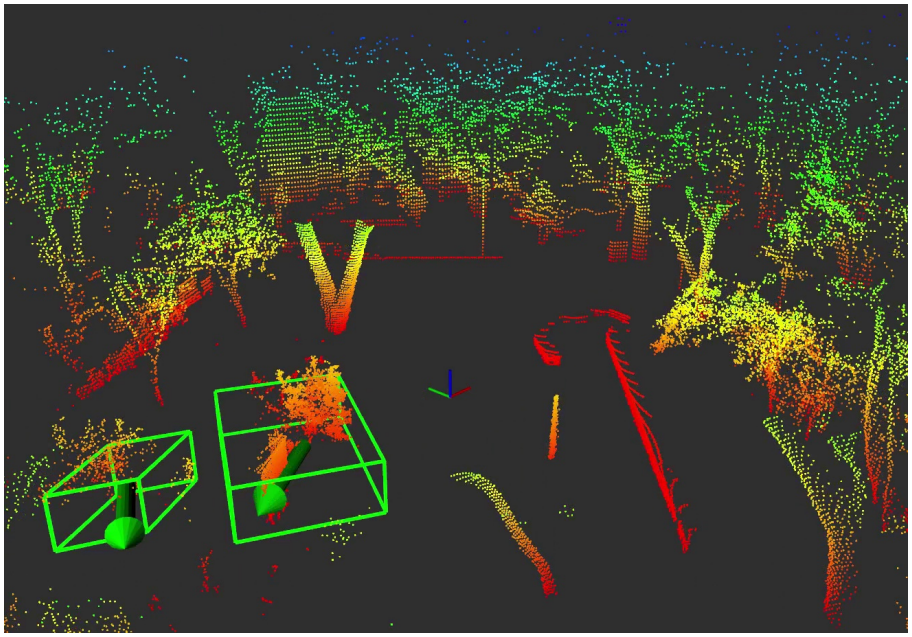


Figure 1.27: Results on 3D Lidar Data

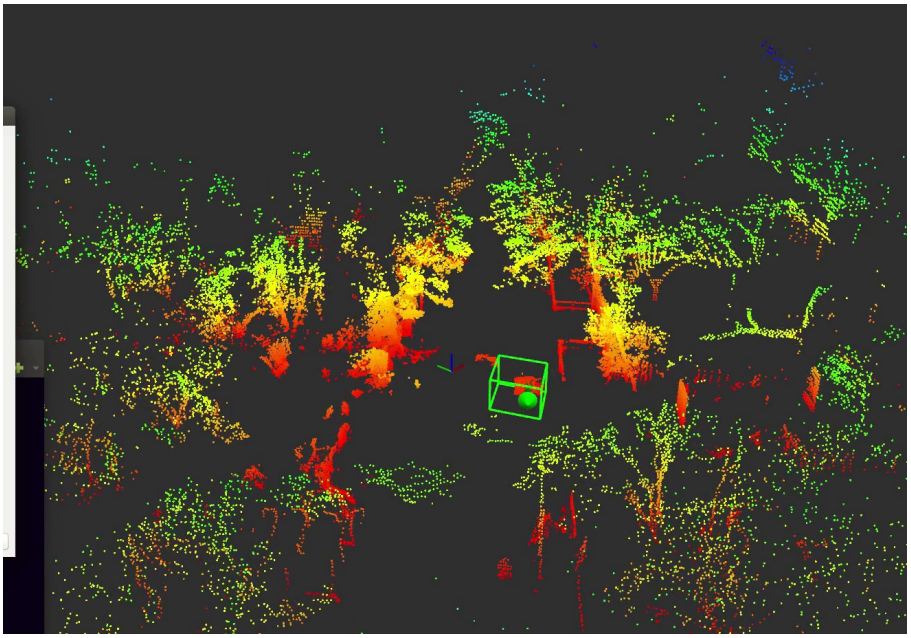


Figure 1.28: Results on 3D Lidar Data

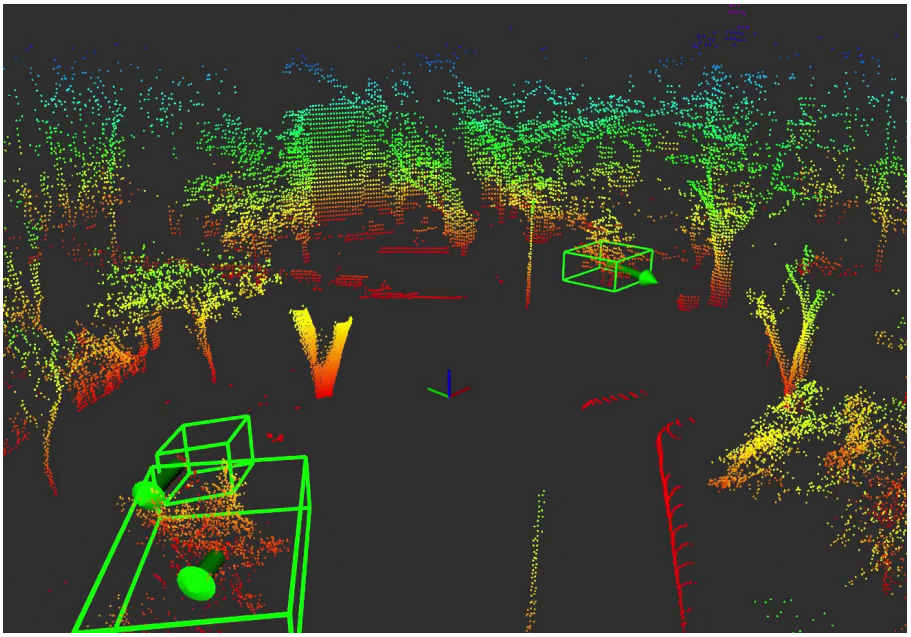


Figure 1.29: Results on 3D Lidar Data

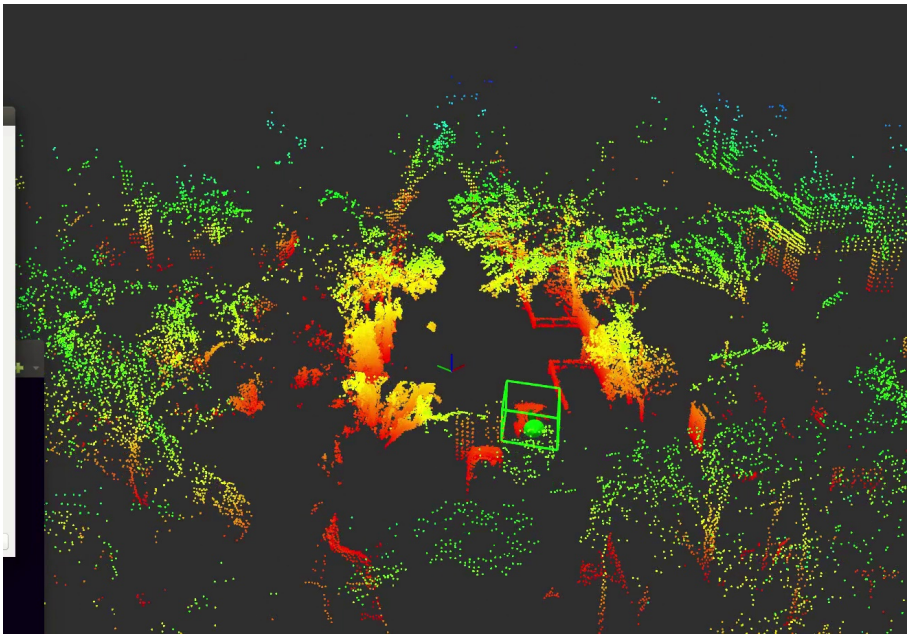


Figure 1.30: Results on 3D Lidar Data

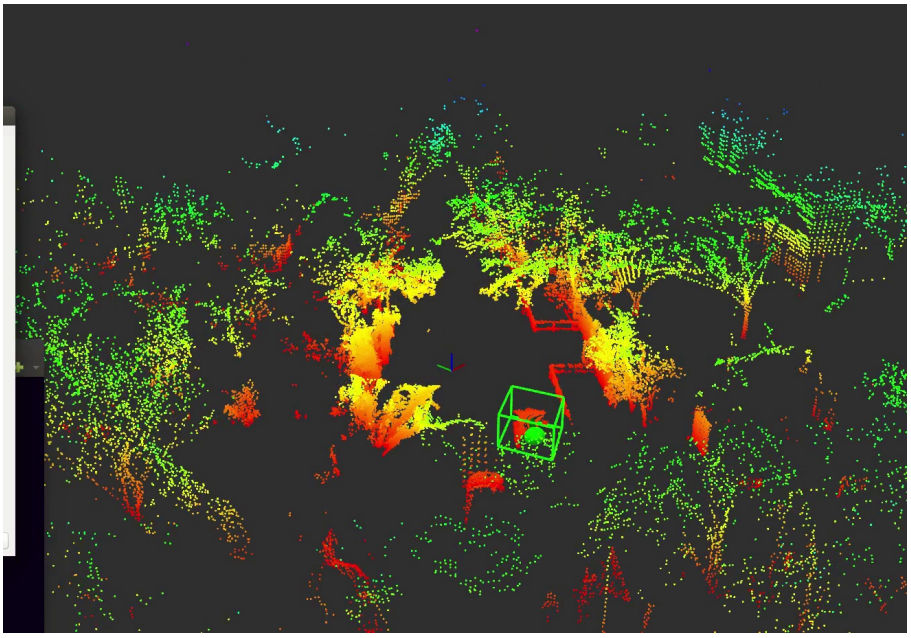


Figure 1.31: Results on 3D Lidar Data