Simultaneous Localization and Mapping with Moving Object Tracking in 3D Range Data

Peng Mun Siew* and Richard Linares†

Department of Aerospace Engineering and Mechanics, University of Minnesota, MN, 55455

Vibhor L. Bageshwar‡

Advanced Technology, Honeywell Aerospace Honeywell International, Golden Valley, MN, 55405

A Bayesian framework is designed for simultaneous localization and mapping (SLAM) with detection and tracking of moving objects (DATMO) using only 3D range data. Bayesian formulated occupancy grid maps are used to store and represent the occupancy probability of the environment. Two separate maps (static occupancy grid map and dynamic occupancy grid map) are generated and updated at each instance. The static occupancy grid map functions as the global map and is used to localized the platform using iterative closest point, whereas the dynamic occupancy grid map contains all the information of possible dynamic objects which are used by the Probability Hypothesis Density (PHD) filter for multiple target tracking. The robustness of the PHD filter is leveraged to enable the usage of a more aggressive dynamic voxel detection algorithm when constructing the dynamic occupancy grid map. Data augmentation is introduced to compensate for “infinity return” to further improve the framework’s robustness. The proposed framework was tested on mid-end HDL-32E and high-end HDL-64E LiDAR data obtained from Velodyne LiDAR and KITTI Dataset respectively, and has shown promising results for both cases.

I. Introduction

Operations of autonomous unmanned vehicles in a dense urban environment has always been a challenging issue due to the presence of multiple moving and stationary objects (obstacles). The vehicle can pose a safety hazard to itself as well as to its environment. For the vehicle to be able to safely navigate itself in a dense and dynamic environment, it needs to be able to “see” and understand what is happening around it. In other words, the vehicle needs to be able to detect, track and estimate the motion characteristics of these obstacles that might pose a collision hazard and reroute its trajectory as it sees fit. This was traditionally achieved using visual cues obtained from images and video. However with the recent popularity and advancement in the Light Detection and Ranging (LiDAR) scanner, much of the attention has been shifted to the use of LiDAR systems as it is now possible to equip a puck size LiDAR scanner aboard a small unmanned aerial vehicles and obtain direct range measurement of objects in the surroundings.

LiDAR scanners function by emitting laser pulses to the environment and measure the time of flight for these lasers to return back to the sensor after being reflected by the surrounding objects. Figure 1 shows a snapshot of a LiDAR scan obtained from a mid-tier HDL-32E LiDAR scanner with the different color representing laser return intensity. The main advantages of using LiDAR data over traditional visual cues are higher resolution data and the dimension, shape and size of the surrounding objects can be easily recovered. Nevertheless, working with LiDAR scanner poses a different set of issues, such as each target can produce multiple returns (in the range of 100s depending on the LiDAR sensor resolution and proximity to the scanner) and extended object tracking (the detected shape of the same object changes depending on the viewing angle from the scanner). Furthermore, the high resolution of the data results in a large amount of data being generated at each scan, in the range of 100,000 points. There is also the issue of having different
level of detection sensitivity and resolution depending on the distance of the object to the scanner and little
to no texture information is provided in LiDAR data. Figure 2 shows an example of how all these issues are
compounded and acts on the measurement scan of a vehicle as it is approaching and then moving away from
the scanner. All these are further complicated with new mid to low tier LiDAR scanner due to issues such
as more noises, sensor artifacts and biases, and lower resolution scans of objects, which make detection and
classification difficult.

Figure 1. Point cloud obtained from a mid-tier HDL-32E LiDAR scanner.

(a) Vehicle moving toward the scanner from a distance. (b) Vehicle near the scanner. (c) Vehicle moving away from the scanner.

Figure 2. Measurement scans of the same vehicle Using HDL-32E LiDAR scanner from different view and distance.

Much of the early works dealing with LiDAR data try to project the 3D data into a lower dimensional
space, projecting 3D position data into 2D images or working on multiple 2D slices of the original 3D point
cloud. The 2D data are then processed using previously established techniques that have shown promising
results on visual cues.\textsuperscript{1–7} Projecting to a lower dimensional space greatly reduces the amount of processing
required, however this causes loss in useful information and lack in robustness of the proposed algorithm.

Target detection and tracking using 3D LiDAR point cloud can also be broadly divided into the same
4 branches; feature based, machine learning based, contextual based and filtering sampling based. Due to
the enormous amount of data generated at each scan, the raw data are normally preprocessed before being
fed into a target detection and tracking algorithm. For a static scanner with known background point cloud
distribution, a background subtraction algorithm is first applied to remove points cloud originating from
background and the ground, whereas for a mobile scanner platform, a ground point removal algorithm is
normally used to remove points clouds due to the ground. This is because not much information can be
obtained from the ground point clouds and an equally good representation can be obtained by using a fitted
ground plane. This greatly reduces the amount of data that need to be processed and stored at each time
step, and also simplify the task of data segmentation. Random Sample Consensus (RANSAC) is one of
the preferred method for ground plane fitting due to its robustness, it is able to produce a high degree of accuracy even when there is a large number of outlier within the data.\textsuperscript{8,9} Some of the previous works also down-sampled the raw point cloud using 3D grids or voxels.\textsuperscript{8–12} The down-sampled point clouds are normally then segmented to form distinct groups to be identified and tracked later on.\textsuperscript{8–16}

For feature based LiDAR approaches, features or descriptors are extracted from isolated objects at each frames and are matched across scans to track the transition of these objects. Numerous work has been carried out on expanding 2D feature based method used on visual cue to be able to work on 3D point cloud. Some of the more prominent feature (descriptor) extraction includes Normal Aligned Radial Feature (NARF), 3D-SIFT, Point Feature Histograms (PFH), and Signature of Histograms of Orientation (SHOT).\textsuperscript{17}

In the last few decades, machine learning based LiDAR object tracking approaches have been gaining traction among the communities. Machine learning based approaches, such as Support Vector Machine (SVM), Random Forest (RF) and Convolution Neural Network (CNN) are used to identify and match target between sequential scans. Some of the researchers was also able to extract contextual information of the isolated object using machine learning, which are then used to assign motion models for the tracked targets, resulting in better state predictions and data association.\textsuperscript{1,2,4,10,11,13,15,18–20}

Filtering based approaches formulate the problem as a state estimation problem, where the objective is to estimate the multiple target’s state from a sequence of noisy and cluttered measurement sets. Some of the more widely used multiple target tracking (MTT) filters include Multiple Hypothesis Tracking (MHT), Joint Probabilistic Data Association (JPDA), Global Nearest Neighbor (GNN), and Probability Hypothesis Density (PHD). The main challenges in multiple target tracking is the problem of assigning measurements from each scan to the correct tracks, while handling false measurements from clutter, misdetection, birth of new targets (new target entering scanning area) and death of target (target leaving scanning area), while having an unknown and time varying number of targets in a clutter and uncertain environment. Each filter algorithm uses a different approach to generate hypothesis that associate measurements to the tracks and target’s states are then estimated based on these hypothesis.

Multiple Hypothesis Tracking (MHT) is one of the earliest proposed MTT filter. It produces an optimal state estimation as it generates a hypothesis (track) for each possible track-measurement pairs. The resulting target states from each hypothesis are then estimated using a Kalman Filter. At the subsequent scan, the new measurements will generate a new set of hypothesis for each track and the probabilities of these joint hypothesis are updated recursively. The track with the highest likelihood is selected to be the target’s state, however the entire track hypothesis are maintained. The tracks can be visualized as a branching tree with the number of branches for each node being the number of new measurements. The tree can grows exponentially, therefore the number of hypothesis need to be pruned by eliminating unlikely hypothesis and merging hypothesis with similar target estimates.\textsuperscript{21–24}

Joint Probabilistic Data Association (JPDA) was first proposed in the early 1980s. Instead of associating all new measurements to each hypothesis track, it associate the new measurements to existing tracks based on their joint probabilistic score. The score measures how well is the measurement to track association for each assignment. After calculating the score for each possible pair, the tracks are updated with the sum of measurement weighted by their respective scores. The traditional JPDA formulation requires that the number of targets/tracks to be known beforehand and remains fixed.\textsuperscript{23,25}

Both the MHT and JPDA suffers from high computational and memory cost, and with an increasing number of targets and measurements, the operation becomes intractable and is not feasible for real time applications. Various work has been done on approximating or simplifying part of the formulation to make both filter tractable at the expense of accuracy.\textsuperscript{24,25}

Global Nearest Neighbor (GNN) is another form of a single hypothesis tracking approach similar to that of the JPDA, where only a single hypothesis is maintained for each target at each time step. Instead of using a joint probabilistic score for pair evaluation, the best pair is selected to be the one that minimizes a particular cost function. The cost function can be in the form of a Mahalanobis distance between each possible pair, a similarity measure in terms of size and shape, or a similarity measure between features of the measurement to the target. Instead of evaluating the cost for all possible target-measurement pairs, gating is sometime used in GNN to first isolate out possible matching observations before evaluating and selecting the best matching pair.\textsuperscript{26}

GNN has lower computational complexity compared to MHT and JPDA, however it performs poorly when there is crossing between the targets’ path and GNN lacks the robustness demonstrated in MHT and JPDA. The PHD filter has the advantage of low computational complexity as well as robustness.
The Probability Hypothesis Density (PHD) filter was first proposed by Mahler in 2003. Instead of solving for an explicit associations between measurements and targets, the targets and measurements are modeled as random finite sets (RFS) and a mixture of probability density functions are used to represent the target states. Using RFS allows the filtering problem to be formulated in a Bayesian framework to jointly estimate the number of targets and their states. By using linear Gaussian multi target model, a closed form solution was able to be formulated by Vo and Ma in 2006. The Gaussian Mixture Probability Hypothesis Density (GMPHD) filter proposed by Vo and Ma has been successfully implemented in numerous real time application with promising results. Successful implementation of the PHD filter to LiDAR data on a static platform with known local map for MTT was also demonstrated in some of the previous work.

Early works on DATMO assumes that the exact pose and position of the scanner remains stationary or is known beforehand, and the map of the environment is known perfectly a priori. However in real life applications, the precise location of the platform or even the surrounding map is not available. For an unmanned vehicle to be truly autonomous, there is a need to have a Simultaneous Localization and Mapping (SLAM) component to complement the DATMO algorithm. SLAM encompasses learning and updating a map of an unknown environment while simultaneously localizing itself within the newly constructed map.

Much of the previous work on SLAM with DATMO using LiDAR data shares a similar framework. The raw point clouds are first preprocessed by using ground point extraction and voxel downsampling to reduce its size to be more manageable for real time application. The downsampled point clouds are then localized to the global navigation frame using SLAM approaches. In past literature, this is normally achieved by using Extended Kalman Filter (EKF) on GPS data compensated with IMU and wheel speed sensors or using an iterative closest point algorithm (ICP). More established SLAM techniques such as particle filter SLAM are normally not used in these applications due to computational cost restrictions. After localization, voxels belonging to dynamic objects are extracted by inspecting for change in voxels’ in occupancy state. The dynamic voxels are then clustered and tracked using classical MTT approaches, such as JPDA and GNN. However, these approaches depend heavily on pre-filtering to clean up the measurement data before feeding into the tracking algorithm, which causes a reduction in their tracking robustness.

In this paper, a Bayesian framework for SLAM with DATMO using only LiDAR data is proposed. The proposed framework does not require any additional sensor information such as IMU or GPS, however the framework can be adapted to fuse these additional sensor information in a Bayesian manner for improved accuracy and robustness. To the best of the author’s knowledge, there has not been any work on a fully integrated framework for SLAM with DATMO using PHD Filter. A Bayesian formulation is used to generate and update a voxelized map of the environment that stores the occupancy probability of each voxel. When a new measurement arrives, the platform is first localized to the map using iterative closest point algorithm (ICP). Dynamic voxels are then extracted using a Maximum Likelihood (ML) method by detecting inconsistencies between the newest measurement to the ML states of the map. The map is then updated with the new measurements and measurements belonging to dynamic voxels are updated at a lower confidence level. The dynamic voxels are then clustered based on their relative distance using mean shift clustering before being fed into a PHD Filter for tracking. By exploiting the robustness of PHD filter to clutter and noise, a more aggressive dynamic voxel extraction algorithm can be used. The general data flow for the proposed framework is as shown in figure 3.

In this paper, a Bayesian framework for SLAM with DATMO using only LiDAR data is proposed. The proposed framework does not require any additional sensor information such as IMU or GPS, however the framework can be adapted to fuse these additional sensor information in a Bayesian manner for improved accuracy and robustness. To the best of the author’s knowledge, there has not been any work on a fully integrated framework for SLAM with DATMO using PHD Filter. A Bayesian formulation is used to generate and update a voxelized map of the environment that stores the occupancy probability of each voxel. When a new measurement arrives, the platform is first localized to the map using iterative closest point algorithm (ICP). Dynamic voxels are then extracted using a Maximum Likelihood (ML) method by detecting inconsistencies between the newest measurement to the ML states of the map. The map is then updated with the new measurements and measurements belonging to dynamic voxels are updated at a lower confidence level. The dynamic voxels are then clustered based on their relative distance using mean shift clustering before being fed into a PHD Filter for tracking. By exploiting the robustness of PHD filter to clutter and noise, a more aggressive dynamic voxel extraction algorithm can be used. The general data flow for the proposed framework is as shown in figure 3.

The proposed Bayesian framework was first tested on LiDAR data collected from a static scanner for preliminary validation. Data collected from a static Velodyne HDL-32 laser scanner provided from the manufacturer was used for this purpose. The Bayesian framework was able to successfully localized itself using only LiDAR data and construct a map of the surrounding environment. Pedestrians and moving vehicles captured by the scanner was also successfully detected and tracked by the Bayesian framework. The framework is then tested on LiDAR data obtained from the KITTI Vision Benchmark Suite. The KITTI Vision Benchmark Suite was collected by driving around in a mid size city, in rural area and in highways, and contains data collected from two high resolution stereo camera and a Velodyne HDL-64E laser scanner.
The dataset provided contains manually annotated classes of Car, Van, Truck, Pedestrian, Sitting Person, Cyclist, Tram, and Misc (i.e. Trailers, Segways) with bounding boxes, as well as ground truth from a state of the art localization system to be used for performance analysis.

The main contribution of this paper are as followed:

- Allowing the application of GM-PHD filter for multiple target tracking on moving platforms with the use of a modified occupancy grid mapping.
- Introducing a more aggressive dynamic voxel detection algorithm by using target’s motion model and then leveraging the robustness of the GM-PHD filter to handle any additional clutter that are introduced.
- Introducing the idea of visitation grid to compensate for “infinity return” in LiDAR data augmentation which increases robustness toward detecting moving objects.

The paper is organized as follows: The Bayesian Simultaneous Localization and Mapping (SLAM) algorithm for LiDAR point clouds is presented in section II, whereas the Dynamic object extraction algorithm is presented in section III, followed by the PHD filter for MTT in section IV. Section V describes the data pre-processing and data augmentation based on the idea of visitation grid. Experimental results for static and moving platform are presented in section VI, meanwhile conclusion and future works are covered in section VII.

II. Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) involves both learning the environment (mapping) and estimating the platform’s current pose and position (localization). SLAM can be said to be a chicken and egg problem; you need to first have a good map in order to arrive at a good pose and position estimate, and at the same time, you need to have a good pose and position estimate in order to generate an accurate map. This makes the localization and mapping component to be equally important to arrive at a truly optimum solution. SLAM is an essential part of a robust DATMO algorithm as it enables the separation of environment from moving objects. It allows sequential LiDAR scans to be associated and it presents a framework to analyze the evolution of the environment. At the same time, it produces a useful byproduct, the local map of the environment, which can be used for latter stages of guidance and navigation, such as path planning and collision avoidance. In the proposed framework, a Bayesian formulation is used for mapping, as it provides an effective mean to combine prior information with new measurements. The Bayesian formulation for mapping is based on the work of Moravec and Elfes back in 1985. They proposed the use of occupancy grids which discretizes the environment into grids. An occupancy probability value is assigned to each grid based on the respective grid’s probability of being occupied. It was introduced to solve the problem of generating maps from noisy and uncertain sensor measurement data. Moravec represented the environment as a 2D grid with probability of being traversable by the robot. The traditional occupancy grid mapping are formulated using the following assumptions:

1. Occupancy of individual cells is independent of each other.
2. New measurements are independent of previous measurement.
3. The robot position and pose are exactly known.
4. The map is static.

By representing the environment using an occupancy grid map, it allows the raw data to be used directly without any preprocessing (i.e. trying to detect and identify landmarks). Furthermore, no information is discarded as the raw data are used to generate and update the map. Although in reality, cells are not independent of adjacent cells (i.e. some neighboring cells can represent the same object), the assumption that the occupancy of individual cells is independent of each other greatly simplifies the mapping algorithm without a significant loss in accuracy. Under this assumption, the probability of a particular map can be considered as a product of the individual probabilities of each cells and the occupancy value of the map depends on the robot’s location history, $x^t = x_0, x_1, ..., x_t$ and the sensor readings, $z_t$ at each time step, $t$. The probability density function (pdf) of the map at time $t$ is represented by the following equation

$$p(m_t | x^t, z_t) = \prod_{x,y,z} p(m_{t[x,y,z]} | x^t, z_t)$$

1 of 24

American Institute of Aeronautics and Astronautics
The occupancy probability of a cell is determined by whether it is observed to be occupied or unoccupied by the robot. All the sensor readings over the robot’s entire history need to be taken into account when generating the occupancy grid map. Under the assumption that the map is static, the occupancy probability of each cells can be updated recursively. Under the second assumption, the measurements are conditionally independent, given the occupancy value of the map.

\[ p(z|m_k) = \prod_{n=1}^{N} p(z_n|m_k) \]  

(2)

At each time step, the map cells are updated iteratively based on the position and sensor reading. However, only observed cells need to be updated. The measurement, \( z_t \) is a binary value of either 0 or 1. The occupancy probability of a cell given the measurement and robot pose is given by equation 3.

\[ p(m|x^t, z^t) = \frac{p(z_t|m, x_t)p(m|x_{1:t-1}, z_{1:t-1})}{p(z_t|x^t, z_{1:t-1})} \]

\[ = \frac{p(m|x_t, z_t)p(z_t|x_t)p(m|x_{1:t-1}, z_{1:t-1})}{p(m|x_t)p(z_t|x^t, z_{1:t-1})} \]

(3)

Instead of updating the occupancy probability of each cell, it is easier to keep track of and update the log odd ratio of the cells at each time step, as we can eliminate terms that are independent of \( m \), the log odd update function is given by equation (4).

\[ \ell(p(x)) = \log \left( \frac{p(x)}{1-p(x)} \right) \]

(4)

Therefore, the log odd ratio of each cell, given the robot’s position and measurements are as followed.

\[ \ell(m)_{t,x,y,z} = \log \left( \frac{p(m|x,y,z|x^t, z^t)}{p(-m|x,y,z|x^t, z^t)} \right) \]

\[ = \log \left( \frac{p(m|x_t, z_t)p(m|x_{1:t-1}, z_{1:t-1})p(-m|x_t)}{p(-m|x_t, z_t)p(-m|x_{1:t-1}, z_{1:t-1})p(m|x_t)} \right) \]

(5)

\[ \ell(m)_{t,x,y,z} = \ell(p(m|x,y,z|x_t, z_t)) + \ell(m)_{t-1,x,y,z} - \ell(p(m|x,y,z)) \]

(6)

where \( p(m|x,y,z) \) is the constant prior occupancy probability (i.e. information obtained from a generated map from a previous experiment), \( p(m|x,y,z|x_t, z_t) \) is the inverse measurement model. It is called an inverse measurement model as we are trying to estimate the environment (map) from the measurement through the sensor, instead of extracting measurement from the environment. For a LiDAR scanner, the dispersion of laser is not significant in the lengthwise direction, hence it can be modeled as a 1D model as shown in figure 4. The contribution due to the constant prior occupancy probability to the log odd ratio equation can be eliminated by setting \( p(m|x,y,z) \) to be 0.5 (i.e. having equal probability of being occupied and unoccupied). For the case of a simple LiDAR sensor, the returned measurements are the sensor’s distance to the nearest objects. Using the robot’s pose at each time step, we can compute the absolute coordinate of each measurements in the global frame. Occupancy grids located between the sensor’s line of sight to each measurements are highly probable unoccupied cells. These cells can be obtained using the “Bresenham line equation.” Both the occupied (“measurement”) cells and unoccupied cells are then updated using equation 6.

The occupancy probability of each cell can be recovered using equation 7.

\[ p(x) = \frac{\exp \{\ell(p(x))\}}{1 + \exp \{\ell(p(x))\}} \]

(7)

The assumption that the map is static can be relaxed by setting upper and lower bound on the log odd ratio of the occupancy grid. The bounds on the log odd ratio enable more rapid transition between inferred states of the grids (occupied and unoccupied), which allows for flexibility in the occupancy grid map to capture transitioning dynamic objects.
Localization to the occupancy grid map is done using the Iterative Closest Point algorithm (ICP). ICP was first introduced by Besl & McKay in ’92. It is a data driven method that is primarily used in aligning different 3D models of the same object, based on their geometry, color or meshes. ICP iteratively refines the transformation between different 3D models or meshes such that a mean-squared distance metric is minimized. If the precise correspondence between two point clouds are known, the optimum relative transformation that aligns them can be easily obtained using method such as minimum mean square error estimator (MMSE). However in practical applications, the exact correspondence between the point clouds, or in our case between the occupancy grid map and new measurement, is not known and is hard to resolve precisely. Under the ICP formulation, it is assumed that the closest point between the two point clouds correspond to each other, the optimum rotation and translation that align the two point clouds are then resolved using these estimated correspondence. The second point cloud is transformed using this calculated transformation and the whole process of finding correspondence based on global nearest neighbor, finding optimum transformation and transforming of the second point cloud is repeated again until convergence.

The ICP algorithm can generally be divided into 5 stages: selection, matching, weighting, rejecting and minimization of error metric. During the selection stage, points to be used for matching are selected, this is normally done to filter out outliers and reducing point cloud sampling for faster computation. Points in the two point clouds are then matched during the matching stage based on a distance metric. In the weighting stage, different weightage can be assigned to each matched pair based on certain parameters, such as their relative distance, curvature, tangent normal and similarity in reflectivity. During the rejecting stage, worst matched pairs can be chosen to be rejected and this is particularly useful when dealing with point clouds that only overlap partially as in our scenario. In the minimization of error metric stage, the transformation that minimizes the selected error metric is estimated. Two of the most widely used error metrics are the point to point error metric (equation 8) and the point to plane error metric (equation 9). After transforming the point cloud using the estimated transformation, the whole iterative process is repeated again until a convergence criteria is reached. Additional external sensors can be used to augment the ICP algorithm and its convergence speed and accuracy by providing a better initial estimate of the platform’s pose.

\[ E = \sum_{i=1}^{N} \| Tp_i - q_i \|^2 \]  
\[ E = \sum_{i=1}^{N} [(Tp_i - q_i)^T n_{q_i}]^2 \]

### III. Dynamic Object Extraction

In past literature, Wolf and Sukhatme proposed the use of conditioned inverse measurement model based on the inferred state of the map. The conditioned inverse measurement model is then leveraged to maintain two separate occupancy grids and isolating out dynamic objects from the static environment; a static environment occupancy grid to maintain the static part of the environment and a dynamic environment occupancy grid to capture the dynamic part of the environment. A complete description of the environment could then be reconstructed by merging these two occupancy grids. They uses a more conservative approach in isolating out dynamic object by treating all objects that are observed for the first time as static objects. In their proposed approach, only the static occupancy grid is continuously maintained and updated with each new measurement. The dynamic map is only used to capture the information of the dynamic object at the current time, hence a new dynamic map is generated at each time step using the maintained static occupancy grid map. The dynamic occupancy grid map is solely used a measurement input for target tracking and is neither maintained nor used for dynamic object extraction across consecutive time step.

In our proposed framework, instead of using an inverse measurement model to completely separate the
static and dynamic part into two disjoint maps, dynamic part of the environment are allowed to exist on both the dynamic and static map. This approach allows for more aggressive dynamic object extraction without risking discarding too much useful information from the static map and resulting in failure in the platform’s simultaneous localization and mapping algorithm. The dynamic occupancy grid map from the previous time step is leveraged to better estimated the state of objects that are observed for the first time.

For the static map update, equation 6 in the previous section is used, however the inverse measurement model is no longer constant but changes based on the previous state of the static map voxels. Let \( S_{t-1} \) and \( S_t \) respectively be the static map voxel state in the previous time step and current time step. An inference can be made on the static map voxel state based on their log odd ratio, whether it is \textit{free}, \textit{unobserved} or \textit{occupied}. The static part of the environment does not change with time, hence possible moving object can be differentiated by comparing the current set of measurements with the inferred static map voxel state. Only the information of dynamic objects at the current time step is used as a measurement input for the MTT algorithm, hence the history of the dynamic objects (ie, the previous position of the dynamic object) are not important. The dynamic objects’ states at the current time step can be identified by looking for voxel grids that has either a static map state of \textit{free} at the previous time step \( (S_{t-1} = \text{free}) \) and an \textit{occupied} measurement state for the current time step \( (z_t = \text{occupied}) \) or a static map state of \textit{unobserved} at the previous time step \( (S_{t-1} = \text{unobserved}) \) and an \textit{occupied} measurement state for the current time step \( (z_t = \text{occupied}) \).

Instead of pegging all voxels that are observed for the first time as dynamic voxels (voxels with \( S_{t-1} = \text{unobserved} \) and \( z_t = \text{occupied} \)), the dynamic map from the previous time step is used to help classify these previously unobserved voxels. A motion model is imposed onto the dynamic voxels in the dynamic map to capture all possible motion of previously detected object, the union of these voxels would then be identified as dynamic voxels. Voxels with \( S_{t-1} = \text{unobserved} \) and \( z_t = \text{occupied} \) but do not lie within the imposed dynamic map are assumed to be static until proven otherwise by subsequent measurements.

The inverse measurement model used to update the static map is as shown in table 1. The first and second column lists all the possible combination of the static map state in the previous time step \( (S_{t-1}) \) and the current measurement state \( (z_t) \). If the log odd ratio of the static map is below a negative threshold, the static map voxel state is considered to be \textit{free}, whereas if the ratio is above a positive threshold, the voxel is considered to be \textit{occupied}. Voxels with ratio between the negative and positive threshold are considered to be \textit{unobserved}. The current measurement state can either be \textit{free} or \textit{occupied} based on the LiDAR return. Voxels containing the new measurement point clouds are considered to have an \textit{occupied} measurement state, whereas any grids that lie between the sensor’s line of sight to each measurement are considered to have a \textit{free} measurement state. For simplicity, the values of the inverse measurement model are represented by low, medium and high. Medium and high values are greater than 0.5 and this will in turn increase the occupancy probability value of that particular voxel, whereas low value has a value lower than 0.5, which will results in a decrease in the occupancy probability value. The first three rows show the trivial case where no obstacle is detected and the occupancy probability of the voxel is then subsequently reduced by using a low inverse measurement value. Obstacle observed at a previously occupied voxel reconfirms the existence of an obstacle, hence a high inverse measurement value is assigned to further increase the occupancy probability of that voxel. Voxels belonging to dynamic objects are assigned a medium inverse measurement value to slightly increase the occupancy probability. This is done to compensate for misclassification of static objects as dynamic objects and to capture dynamic objects that are transitioning or have transitioned to static. Previously unobserved voxel that lies outside of the imposed dynamic map are considered belonging to static objects, hence a high inverse measurement value is used.

The inverse measurement model used to update the occupancy grid identified by the ray tracing method are as shown in figure 5. The width of the peak are identified based on the LiDAR scanner return distance accuracy specification, in the case of the HDL-32E, it has a distance accuracy of 2cm. Based on the static map state from the previous time step, a slightly different inverse measurement model is used to update the static occupancy grid map. Occupancy grids that are previously unobserved will use either of the two inverse measurement model depending on whether it lies within the imposed dynamic map. If it lies within the imposed dynamic map, it has a higher probability of belonging to a dynamic object and will be updated using the lower weighted inverse measurement model.

The dynamic voxel map serves to capture the current states of the dynamic objects and is used as a measurement input to the MTT algorithm. Hence, if a voxel was previously occupied by a dynamic object in the previous time step and is currently free, no additional distinction is given and the voxel will just be
Static Map State, Measurement State, Inverse Meas. Value,

| \(S_{t-1}\) | \(z_t\) | \(p(S_t|z_t)\) |
|---|---|---|
| Free | Free | Low (< 0.5) |
| Unobserved | Free | Low (< 0.5) |
| Occupied | Free | Low (< 0.5) |
| Free | Occupied | Medium (> 0.5, < High) |
| Unobserved | Occupied | \(\begin{cases} 
\text{Medium, if } \in \text{imposed dynamic map} \\
\text{High, otherwise}
\end{cases}\) |
| Occupied | Occupied | High |

Table 1. Inverse measurement model for static map.

The amount of points in the dynamic point cloud can still be quite large (~12k points) depending on the scene. The dynamic point cloud is preprocessed using a method known as mean shift clustering to cluster nearby points. Bounding boxes encompassing each clusters and the clusters’ centroids are then used to represent the dynamic point cloud. Mean shift clustering is an iterative non-parametric feature-space analysis technique that uses the concept of kernel density estimation (KDE). KDE is used to estimate the underlying distribution (also called the probability density function) for a set of data. It works by placing a kernel on each point in the data set. A probability surface is then generated by adding up all of the kernel. An uphill method is then used to locate the local maxima of the density function and points in the proximity of the local maxima are clustered together to form individual clusters. One of the advantages of the mean shift clustering algorithm is that it is simple and flexible. Additional data, such as return intensity and color, can be easily incorporated into the data for better clustering. Furthermore, it is only controlled by a single parameter, bandwidth, which dictates the kernel function. The use of a small kernel bandwidth is more time consuming and can cause an object to be wrongly divided into a few

Table 2. Inverse measurement model for dynamic map.

Figure 5. Inverse measurement model for static map with laser return measured distance, \(d_t\) at 100cm.
small clusters due to formation of more distinct modes, whereas a large kernel bandwidth results in faster convergence but might wrongly group small objects to form a large cluster.\textsuperscript{36} In our framework, the kernel bandwidth was manually tuned to allow good separation between pedestrians and vehicles. Besides that, the mean shift clustering algorithm is easily parallelizable with GPUs and multicore processing.

IV. PHD Filter for MTT

The PHD filter was developed based on the theory of Finite Set Statistics (FISST) and Random Finite Set (RFS). Under the PHD filter framework, the multi targets and observations are modeled as RFS with their relative uncertainties being characterized by using FISST. The FISST is a variant of the theory of point process that allows for multi target systems to be treated as visualizable images while maintaining the Bayesian formalism. The main idea of the FISST was to generalize the probability densities and the corresponding calculus methods such that ideas from statistics and information theory can be extended to RFS. It allows for traditional concepts such as derivatives, integrals, probability mass function and likelihood functions to be generalized and applicable for set operations. By utilizing FISST, the formal Bayesian framework for single target tracking can be extended to nontraditional information and to multi sensor multi targets problems. This enables the RFS to be developed in a Bayesian framework that allows for the recursive update of the multi target posterior density. A RFS is a finite, set-valued random variable where both the number of targets and the target’s states are unknown. A RFS can be completely specified by the number of targets and the target’s states. The PHD filter framework for single target tracking can be extended to nontraditional information and to multi sensor multi targets problems. This enables the RFS to be developed in a Bayesian framework that allows for the recursive update of the multi target posterior density. A RFS is a finite, set-valued random variable where both its element and the set’s cardinality are random. This formalism is useful for multi target tracking as both the number of targets and the target’s states are unknown. A RFS can be completely specified by the distribution of its cardinality and a family of symmetric joint distributions that characterize the distributions of the points conditioned on the cardinality of the set.\textsuperscript{37, 38}

Under the FISST framework, the Bayesian recursion for the RFS can be formulated using belief mass function. For any closed subsets $S \subseteq E_s$ and $T \subseteq E_o$, where $E_s$ and $E_o$ are the target and observation space respectively, let the posterior belief mass function of the RFS $X_t$ given all the observation sets $(Y_{1:t})$, the belief mass function of the RFS $X_t$, and the belief mass function of the observation set be:

$$
\beta_t(S|Y_{1:t}) \equiv P(X_t \subseteq S|Y_{1:t})
$$

$$
\beta_t(S|Y_{t-1}) \equiv P(X_t \subseteq S|Y_{t-1})
$$

$$
\beta_t(T|X_t) \equiv P(Y_t \subseteq T|X_t)
$$

The FISST multi target posterior density $\pi_t(X_t|Y_{1:t})$, FISST multi target transition density $\phi_{t|t-1}(X_t|X_{t-1})$ and FISST multi target likelihood $\rho_t(X_t)$ are the set derivatives of $\beta_t(S|Y_{t-1})$, $\beta_t(S|Y_{t-1})$ and $\beta_t(T|X_t)$ respectively. The FISST multi target Bayesian filter is then given by:

$$
\pi_{t|t-1}(X_t|Y_{1:t-1}) = \int \phi_{t|t-1}(X_t|X) \pi_t(X|Y_{1:t-1}) \delta X
$$

$$
\pi_t(X_t|Y_t) = \frac{\rho_t(Y_t|X_t) \pi_{t|t-1}(X_t|Y_{1:t-1})}{\int \rho_t(Y_t|X) \pi_{t|t-1}(X|Y_{1:t-1}) \delta X}
$$

A. PHD Filter Formulation

Say for example at a particular time step $t$, there are $N_t$ targets that need to be tracked and $M_t$ measurements were detected by the sensor at that time step. Note that the number of targets can vary at each time step and the number of measurements might not necessary be equal to the number of target as some target may pass by undetected by the sensor (“misdetection”) and some measurements may be false alarm due to clutter and errors in the observation process (“false detection”). Therefore, it is natural to model both the target states and measurements as RFS, where the order in which the targets and measurements are listed has no importance. The target state, $(x_{t,i})_{1 \leq i \leq N_t}$, contains all the information on each target, which can be their kinematic properties such as attitude, position and velocity or other information such as target label, whereas the measurement state, $(y_{t,i})_{1 \leq i \leq M_t}$, contains each return from the observation process.

$$
X_t = \{x_{t,1}, \cdots, x_{t,N_t}\}
$$

$$
Y_t = \{y_{t,1}, \cdots, y_{t,M_t}\}
$$
At the subsequent time step, \( t + 1 \), some targets may have disappear or move outside of the observation space of the sensor or have spawned into numerous smaller target, while the states of the remaining surviving targets evolve according to their motion model and new targets may have entered the observation space. All of these possible occurrences need to be taken into consideration when formulating the RFS of the target state at time step \( t + 1 \). The survivability of each target can be modeled as a Bernoulli RFS, \( S_{t+1|t}(x_t) \), where it has a probability \( p_s,t(x_t) \) of surviving and transitioning to the new state with probability density \( f_{t+1|t}(x_{t+1}|x_t) \) or dying and assuming a null value with a probability of \( 1 - p_s,t(x_t) \). The RFS of a new target spawning from a previous existing target is represented by \( B_{t+1|t}(x_t) \), while \( \Gamma_{t+1} \) is used to represent new targets due to spontaneous birth. The RFS of targets at time step \( t + 1 \) is thus given by the union of the three previous RFS:

\[
X_{t+1} = (\{x_t \in X_t, S_{t+1|t}(x)\} \cup \{x_t \in X_t, B_{t+1|t}(x)\}) \cup \Gamma_{t+1} \tag{15}
\]

Similarly, a Bernoulli RFS, \( \Theta(x_t) \) can be used to model the detection probability of the targets at a given time step. Each target is either detected with a probability of \( p_d,t(x_t) \) or missed with a probability of \( 1 - p_d,t(x_t) \). The probability density of the measurement from \( x_t \) conditional on detection is given by the likelihood function \( g_t(\cdot|x_t) \). The sensor also produces a set of “false detection” which can be represented by the RFS, \( K_t \). Thus, the set of measurements at a given time step \( t \) can be represented as such:

\[
Y_t = (\{x_t \in X_t, \Theta(x)\} \cup K_t) \tag{16}
\]

The general PHD filter derivation assumes that each target is only responsible for the generation of at most one observation and each RFS are assumed to be mutually independent. The target RFS can then be updated in a Bayesian recursion manner using the FISST framework as discussed earlier. All of the target state and information are contained within the FISST multi target posterior probability density, \( \pi_t(X_t|Y_{1:t}) \). However, the FISST multi target recursion involves the evaluation of set integrals which are often intractable with numerical approximation that is only computationally feasible for a small number of targets. Instead of propagating the full multi target posterior probability density, the PHD filter only propagates the first moment of the multi target posterior probability density which greatly reduces the computational requirement while maintaining sufficient accuracy for multi target tracking.

The general PHD filter makes the assumption that the birth and clutter RFS are Poisson RFS that is independent of the other RFS that makes up the target RFS and measurement RFS, and the multi target probability density are approximated as Poisson RFS. These assumptions allow us to exploit a special behavior of the Poisson RFS class where each Poisson RFS are uniquely characterized by their intensity function and this allows higher order moments to be neglected. Furthermore, the expectation measure of a Poisson RFS coincides with their intensity measure. The expected number of targets can be recovered by integrating the Poisson RFS intensity function over its domain. Although these assumptions are not always valid in real life scenario, these approximations are still useful and have demonstrated satisfactory performance in various applications. Let \( \gamma_{t|t-1} \) and \( \gamma_{t|t-1} \) be the multi target posterior and predicted intensity functions respectively, the predicted multi target intensity function is then given by equation 17.

\[
\gamma_{t|t-1}(x) = \int_{E_x} \{p_{s,t}(u)f_{t|t-1}(x|u) + b_{t|t-1}(x|u)\}\gamma_{t-1|t-1}(u)du + \mu_t(x) \tag{17}
\]

where \( f_{t|t-1}(x|u) \) is the transition probability density, \( p_{s,t}(u) \) is the survival probability, \( b_{t|t-1}(x|u) \) is the spawn intensity for a new target spawning from a previously existing target and \( \mu_t(x) \) is the spontaneous birth intensity for new targets. Meanwhile, the posterior multi target intensity function is given by equation 18.

\[
\gamma_{t|t}(x) = (1 - p_d,t(x))\gamma_{t|t-1}(x) + \sum_{y \in Y_t} \frac{p_d,t(x)g_t(y|x)\gamma_{t|t-1}(x)}{h_t + \int p_d,t(u)g_t(y|u)\gamma_{t|t-1}(u)du} \tag{18}
\]

where \( p_d,t(x) \) is the target detection probability, \( g_t(y|x) \) is the single target likelihood function, and \( h_t \) is the clutter intensity.

Some of the advantages of the PHD filter is low computational complexity of \( O(mn) \) compared to other multi target tracking approaches, furthermore it enables an easy integration of misdetection, false detection, and the restriction due to the sensor field of view in the formulation as well as easy integration for target appearing, disappearing and spawning. Besides that, the number of targets do not need to be specified.
beforehand nor fixed and the framework will produce an estimated number of targets. One of the more notable disadvantage of the general PHD filter is that there is a high variance on the estimated number of target, this is due to the nature of a Poisson RFS where its variance has the same magnitude as its mean. A big amount of information is lost as only the first moment of the full multi target posterior probability density is propagated at each time step. Furthermore, the final recursion equations (17 and 18) still involves multiple set integrals and often have no closed form solution. The general PHD filter also does not produce any peak to track association. Various variant of the general PHD filter was proposed to overcome these disadvantages of the general PHD filter while maintaining its Bayesian nature and low computational complexity. Among these variants are the Sequential Monte Carlo PHD (SMCPHD) filter, Cardinalized PHD (CPHD) filter and the Gaussian Mixture PHD (GMPHD) Filter.

B. Gaussian Mixture PHD Filter

Under the GMPHD filter, the target dynamics and measurement model are assumed to be linear Gaussian as shown in equation 19 and 20 respectively, while the intensities of the RFS take the form of a Gaussian mixture distribution. Under these assumptions, a closed form solution to the general PHD filter recursion equation can be found.

\[
f_{t|t-1}(x_t|x_{t-1}) = \mathcal{N}(x_t; F_{t-1}x_{t-1}, Q_{t-1})
\]

where \( F, Q, H, R \) are the state transition matrix, process noise covariance matrix, measurement matrix and the measurement noise covariance matrix respectively.

For the intensity of the propagated multi state probability density function to remain as a Gaussian mixture at each time step, the birth intensity and spawn intensity are assumed to be Gaussian mixture. The predicted multi target intensity function, \( \gamma_{t|t-1} \) is given by equation 21.

\[
\gamma_{t|t-1}(x_t) = \gamma_{S,t|t-1}(x_t) + \gamma_{B,t|t-1}(x_t) + \mu_t(x_t)
\]

where \( \gamma_{S,t|t-1}(x_t) \) is the survival intensity function, \( \gamma_{B,t|t-1}(x_t) \) is the spawn intensity function and the \( \mu_t(x_t) \) is the birth intensity function given by equation 22, 23 and 24 respectively.

\[
\gamma_{S,t|t-1}(x_t) = p_{s,t} \sum_{j=1}^{J_{s,t}} w_{s,t-1}^{(j)} \mathcal{N}(x_t; m_{s,t|t-1}^{(j)}, P_{s,t|t-1}^{(j)})
\]

\[
m_{s,t|t-1}^{(j)} = F_{t-1}m_{t-1}^{(j)}
\]

\[
P_{s,t|t-1}^{(j)} = Q_{t-1} + F_{t-1}P_{t-1}F_{t-1}^T
\]

Note, equation 22b and 22c are the Kalman Filter prediction equations.

\[
\gamma_{B,t|t-1}(x_t) = \sum_{l=1}^{J_{B,t}} \sum_{j=1}^{J_{B,t}} w_{B,t-1}^{(l)} w_{B,t}^{(j)} \mathcal{N}(x_t; m_{B,t|t-1}^{(j)}, P_{B,t|t-1}^{(j)})
\]

\[
m_{B,t|t-1}^{(j,l)} = F_{t-1}m_{t-1}^{(l)} + d_{B,t-1}^{(j,l)}
\]

\[
P_{B,t|t-1}^{(j,l)} = Q_{B,t-1} + F_{B,t-1}P_{B,t-1}^{(j,l)}F_{B,t-1}^T
\]

\[
\mu_t(x_t) = \sum_{\mu=t}^{J_{\mu,t}} w_{\mu,t}^{(i)} \mathcal{N}(x_t; m_{\mu,t}^{(i)}, P_{\mu,t}^{(i)})
\]

where \( J_{B,t}, w_{B,t}^{(i)}, F_{B,t-1}^{(j,l)}, Q_{B,t-1}^{(j,l)} \) and \( J_{\mu,t}, w_{\mu,t}^{(i)}, m_{\mu,t}^{(i)}, P_{\mu,t}^{(i)} \) are the model parameter which determine the spawn and birth intensities.

The posterior intensity function is also in the form of a Gaussian mixture given by equation 25.

\[
\gamma_t(x_t) = (1 - p_d,t)\gamma_{t|t-1}(x_t) + \sum_{y \in Y_t} \gamma_{D,t}(x_t, y_t)
\]
plane that has an elevation angle of greater than 30°. Pavement and road divider are safely removed by the ground plane removal algorithm. As a fail-safe, fitted planes from the fitted plane are removed from the full point cloud. A height threshold of 10" is selected such that points within each quadrant are used to fit a plane using RANSAC. Points that lie within a height of 10" plane, the point cloud is first segmented into four equal horizontal quadrants and the lowest 40 percentile ground plane are first removed using a RANSAC plane fitting algorithm. For better modeling of the ground plane, the centroid of each voxels are used instead of the raw point clouds for faster computation. This also serves to reduce the effect of measurement noise.

The maximum range of the Velodyne LiDAR Scanner is 100m; when there is no object located within 100m of the laser’s path, no return would be obtained by the scanner (“infinity return”). The voxel grids passed by these laser path has a high probability of being unoccupied and thus can be updated as such. At each time step, the azimuth and elevation angle corresponding to the laser with no return can be recorded and a dummy point is then projected at a distance of 100m along the laser’s path. Grids transversed by these laser can then be recovered using the Bresenham’s line equation on these dummy points and be updated as unoccupied.

However in the KITTI dataset, the raw LiDAR measurement data have been preprocessed into 3D coordinates at the platform’s local frame. Instead of solving for the exact azimuth and elevation angle of the laser ray that produced the “infinity returns” and using the Bresenham’s line equation to identify the unoccupied grids, a visitation grid is used to identify these grids. Based on the vertical and horizontal angular resolution of the scanner, a visitation grid can be formulated such that there is always at least a single ray passing through each grid. Due to the restricted vertical field of view of the scanner and the removed ground plane, the visitation grid would have a form similar to that of a swept sector with a flatten base. A ray is projected along each point in the point cloud to the radius of the swept sector. Any visitation grids that is not visited by the projected rays are then updated as unoccupied.

For the case of the HDL-32E laser scanner, it has a vertical resolution of 1.33° and based on the static map resolution of 0.5m, the radius of the visitation grid need to be less than 20.89m to ensure that there is...
Algorithm 1 Pseudo-Code for the GMPHD Filter

Given \( \{w_{t|t-1}^{(i)}, m_{t|t-1}^{(i)}, P_{t|t-1}^{(i)}\}_{i=1}^{J_{t|t-1}} \) and the measurement set \( Y_t \):

Step 1. Prediction of birth targets

\[ i = 0 \]

for \( j = 1, \ldots, J_{t|t-1} \) do

\[ i = i + 1 \]

\[ w_{t|t-1}^{(i)} = w_{t|t-1}^{(i)}, \quad m_{t|t-1}^{(i)} = m_{t|t-1}^{(i)}, \quad P_{t|t-1}^{(i)} = P_{t|t-1}^{(i)} \]

for \( j = 1, \ldots, J_{t|t-1} \) do

\[ i = i + 1 \]

\[ w_{t|t-1}^{(i)} = w_{t|t-1}^{(i)} w_{B,t-1}^{(i)}, \quad m_{t|t-1}^{(i)} = d_{B,t-1}^{(j)} + P_{B,t-1}^{(j)} m_{t|t-1}^{(j)}, \quad P_{t|t-1}^{(i)} = Q_{t|t-1}^{(i)} + F_{t|t-1}^{(i)} P_{t|t-1}^{(i)} F_{t|t-1}^{T} \]

Step 2. Prediction of existing targets

for \( j = 1, \ldots, J_{t|t-1} \) do

\[ i = i + 1 \]

\[ w_{t|t-1}^{(i)} = p_{s,t} w_{t|t-1}^{(i)}, \quad m_{t|t-1}^{(i)} = F_{t|t-1}^{(j)} m_{t|t-1}^{(j)}, \quad P_{t|t-1}^{(i)} = Q_{t|t-1}^{(i)} + F_{t|t-1}^{(i)} P_{t|t-1}^{(i)} F_{t|t-1}^{T} \]

\[ J_{t|t-1} = i \]

Step 3. Construction of PHD Update Components

for \( j = 1, \ldots, J_{t|t-1} \) do

\[ \nu_{t|t-1}^{(j)} = H_{t} m_{t|t-1}^{(j)}, \quad s_{t|t-1}^{(j)} = H_{t} P_{t|t-1}^{(j)} H_{t}^{T} + R_{t}, \quad K_{t|t}^{(j)} = P_{t|t}^{(j)} H_{t}^{T} s_{t|t}^{(j)} - 1, \quad P_{t|t}^{(j)} = (I - K_{t|t}^{(j)} H_{t}) P_{t|t}^{(j)} \]

Step 4. Update

for \( j = 1, \ldots, J_{t|t-1} \) do

\[ i = i + 1 \]

\[ w_{t|t}^{(i)} = 1 - p_{d,t} w_{t|t-1}^{(i)}, \quad m_{t|t}^{(i)} = m_{t|t-1}^{(i)}, \quad P_{t|t}^{(i)} = P_{t|t-1}^{(i)} \]

\[ l = 0 \]

for each observation \( y \in Y_t \) do

\[ l = l + 1 \]

for \( j = 1, \ldots, J_{t|t-1} \) do

\[ w_{t}^{(lJ_{t|t-1}+j)} = p_{d,t} w_{t|t-1}^{(i)} N(z; \eta_{t|t-1}^{(j)}, S_{t}^{(j)}), \quad m_{t|t}^{(lJ_{t|t-1}+j)} = m_{t|t}^{(lJ_{t|t-1}+j)} + K_{t}^{(j)} (z - \eta_{t|t-1}^{(j)}), \]

\[ J_{t} = l J_{t|t-1} + J_{t|t-1} \]

\[ \text{output } \{w_{t}^{(i)}, m_{t}^{(i)}, P_{t}^{(i)}\}_{i=1}^{J_{t}} \]
at least a single laser ray passing through all of the visitation grid. Visitation grids located below the fitted
ground plane of the respective time step are removed. The swept sector of the visitation grid is as shown in
figure 6(a). The top figure in 6(b) shows the projected line of sight of the point cloud at a particular scan
cycle, the line of sight of each point is projected from the origin to the radius of the visitation grid. The
unvisited grids shown in the bottom figure of 6(b) is obtained by subtracting the element of the projected
grids from the visitation grid. The visitation grid can be reform by the union of the unvisited grids and the
projected grids. The unvisited grids are then updated as unoccupied in the static occupancy map.

![Figure 6. Update of unvisited grids.](image)

**VI. Experimental Results**

**A. Stationary platform**

Sample LiDAR data collected from a static Velodyne HDL-32E laser scanner positioned in the middle a
cross junction was used for preliminary validation and feasibility test of the proposed framework. Only raw
LiDAR data was used as an input to our proposed framework. Vehicles stopping, entering and exiting the
junction, and pedestrians crossing the junction was captured by the laser scanner as shown in figure 7.

![Figure 7. Point cloud generated by the static laser scanner at a time step.](image)
For the experimental setup, a voxel resolution of 0.5m was used for the SLAM algorithm. The ground points are first removed using RANSAC as detailed in section V. During localization, the LiDAR measurements are voxelated according to the resolution of the occupancy grid map and the centroid of each voxels are used for localization instead of the raw point clouds. Voxelization of the raw point cloud not only reduces the computational cost but also serves to enable better and more accurate correspondence matching for the ICP algorithm. The raw point clouds within each voxels are retained and used later on for clustering. A nearest neighbor kD tree search is used to match the voxelated measurement point cloud to the generated static voxel map. In our application, the contribution of all matched points to the error metric are weighted equally. As no outliers were removed during the selection stage, a higher worst match rejection criteria (10%) is used to reject matched points with unreasonable error metric score. A point to plane error metric is used as it allows sliding of the point cloud and this error metric has shown from previous works to produce faster and better convergence compared to the traditional point to point error metric.

Figure 8 shows the change in attitude and Cartesian coordinates that is proposed by the solution. As the platform is stationary, both the change in attitude and Cartesian coordinates should be close to zero. During the first 5 frames, error in attitude change and coordinate change are observed, this is caused by the framework not having access to a prebuilt map to accurate localize the platform. As the number of frames increases, a more comprehensive map of the surrounding is constructed and the error in rate of change for attitude and coordinate drops to close to zero. Here, the author would like emphasize that the objective of the SLAM algorithm is not to build a map with high accuracy and precision, but to enable sequential LiDAR scans to be localized for environment evolution analysis. Hence, more focus was placed on minimizing error in rate of change with a reasonable computational usage.

The localized point clouds are then used to update the static occupancy map, the update of each grids are weighted based on the number of rays passing through that particular grid. The data is augmented to compensate for “infinity return” as described in section V. Two separate threshold are used to isolate dynamic voxels that transitioned from free to occupied and from unknown to occupied. The former threshold is first manually tuned by disabling the second threshold and slowly increasing the threshold until a satisfactory performance is achieved in tracking a known dynamic object. The latter threshold is then tuned to enable the earliest detection of a known dynamic object leaving the scene and with the least amount of false detection. The measurement point clouds located within the each isolated dynamic voxel are then fed into the mean shift clustering algorithm to generate clusters for the PHD filter. A bandwidth of 2.2 was used and clusters with less than 5 returns (points) are rejected. The GM-PHD filter generated target tracks are then filtered to remove stationary targets as the filter is capable of tracking both stationary and moving targets. This is done by analyzing the change in Cartesian coordinates within a number of continuous frame for each target tracks and removing those that fall below a cutoff threshold from being displayed. Once a track is identified to originate from a moving target, any future solutions that are associated to the track will also be displayed. The stationary target tracks are still stored within the PHD propagated state as these targets
have the probability to transition into a moving object in future scans.

Figure 9 shows the generated moving target tracks by the GM-PHD filter in black scatter plots with bounding boxes around each tracked objects over a period of 200 frames. The bounding boxes are formed by the mean shift clustering algorithm when pre-processing the dynamic point cloud. The light gray scatter plot are the measurement point clouds from the current time step. Dynamic objects (pedestrian in bright red bounding box, and vehicles in green and dark red bounding boxes) was successfully detected and tracked entering and exiting the junctions.

The proposed framework has shown satisfactory performance in the preliminary feasibility and validation test. The static platform was successfully identified as stationary and dynamic objects (both pedestrian and vehicles) in the environment was successfully isolated and tracked by the proposed Bayesian framework.

B. Moving platform

Two different segments of LiDAR and GPS data was taken from the KITTI dataset to test the performance of the proposed framework. The first segment consists of 600 frames of data taken from a residential area with some moving targets. Meanwhile, the second segment consists of 200 frames of data taken on-board a moving platform in an urban environment with vehicles traveling in both direction and some pedestrian on the sidewalk. GPS and IMU data provided by the KITTI dataset was not used as an input for the proposed framework but is only used as a ground truth for SLAM performance evaluation. The same setup from the static scenario was used for the moving platform scenario with only changes to the dynamic voxel isolation threshold.
Segment 1: Residential Environment

The main purpose of segment 1 was to study the effect of using different SLAM parameters on map and localization accuracy. The parameters that were varied include varying occupancy map resolution, using unvoxelated point cloud (unvoxelated) or voxelated with point cloud centroid as input (centroid), using full point cloud (full) or point cloud with ground point removed (object), varying percentage of worst matched pair rejection (WR) and using different error metric (point to point (point) or point to plane (plane)).

Accuracy in the X and Y direction seems to be comparable for all methods when the unvoxelated measurement point cloud with no worst pair rejection is used, except for the case of point to point matching using the full point cloud, as shown in 10. However, there is a build up in error in the Z direction as time progresses for most of the methods, with the point to plane matching using full point cloud having the best performance. The point to point matching has the second best performance in Z direction, followed by the point to point matching using the full point cloud, while the point to plane matching using object point cloud has the worst deviation in the final z position.

Instead of directly using the unvoxelated measurement point cloud as matching input, it is first voxelated based on the resolution of the static occupancy map. The centroid of each voxels are used to form the new pre-processed measurement point cloud. The static occupancy grid map is also modified to keep track and update the centroid of each grid, which is then used for ICP matching instead of using the center of occupied voxels. The X and Y direction accuracy are significantly improved when the new preprocessed measurement point cloud is used with worst matched pair rejection. Better SLAM solution was achieved when a lower static occupancy map of 0.5m and a worst matched pair rejection of 10% was used.

The error analysis of the best performing SLAM solution is as shown in figure 11, table 3 and table 4. The solution was able to successfully track the attitude and location of the platform across 450 frames.
The errors and oscillations within the solution can be further reduced by using a smoothing filter to post-process the ICP localization solution or by incorporating external sensor data such as GPS and IMU into the localization algorithm using a Kalman Filter framework.

Segment 2: Urban Environment

The same parameters tuned from segment 1 was used on another segment obtained from the KITTI dataset. The main purpose of segment 2 was to study the robustness of the tuned localization parameters to work under different scenarios and to examine the tracking capability of our proposed framework. Figure 12 shows the comparison between the rate of change in attitude and coordinate proposed by the framework and ground truth. Again, a strong correspondence in the rate of change for roll, pitch and yaw between the framework solution and the ground truth was observed, meanwhile the rate of change in coordinate proposed by the framework follows the general trend of the ground truth.

Table 5 shows the maximum absolute value, mean and standard deviation for the error in predicted change in rotation and translation, whereas table 6 shows the error in rotation and translation solution.

Figure 13 shows the moving target track solution generated by the GM-PHD filter. Vehicles traveling in both direction of the road as well as pedestrian on the sidewalk and vehicle entering from a junction was successfully detected and tracked by our proposed framework.
<table>
<thead>
<tr>
<th></th>
<th>Max Absolute Value</th>
<th>Mean Value</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error in Change in Roll, rad</td>
<td>0.0094</td>
<td>1.2576e-04</td>
<td>0.0024</td>
</tr>
<tr>
<td>Error in Change in Pitch, rad</td>
<td>0.0100</td>
<td>2.6802e-05</td>
<td>0.0023</td>
</tr>
<tr>
<td>Error in Change in Yaw, rad</td>
<td>0.0038</td>
<td>1.1397e-05</td>
<td>8.4344e-04</td>
</tr>
<tr>
<td>Error in Change in X Coordinate, m</td>
<td>0.1082</td>
<td>0.0014</td>
<td>0.0323</td>
</tr>
<tr>
<td>Error in Change in Y Coordinate, m</td>
<td>0.0895</td>
<td>-0.0033</td>
<td>0.0311</td>
</tr>
<tr>
<td>Error in Change in Z Coordinate, m</td>
<td>0.1121</td>
<td>0.0055</td>
<td>0.0310</td>
</tr>
</tbody>
</table>

Table 3. Error in change in attitude and coordinate for segment 1.

<table>
<thead>
<tr>
<th></th>
<th>Max Absolute Value</th>
<th>Mean Value</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error in Roll, rad</td>
<td>0.0918</td>
<td>0.0147</td>
<td>0.0397</td>
</tr>
<tr>
<td>Error in Pitch, rad</td>
<td>0.0386</td>
<td>0.0046</td>
<td>0.0164</td>
</tr>
<tr>
<td>Error in Yaw, rad</td>
<td>0.0164</td>
<td>0.0080</td>
<td>0.0028</td>
</tr>
<tr>
<td>Error in X Coordinate, m</td>
<td>1.1721</td>
<td>-0.1833</td>
<td>0.6653</td>
</tr>
<tr>
<td>Error in Y Coordinate, m</td>
<td>2.3808</td>
<td>-1.1915</td>
<td>0.7944</td>
</tr>
<tr>
<td>Error in Z Coordinate, m</td>
<td>3.3040</td>
<td>-0.9113</td>
<td>1.7520</td>
</tr>
</tbody>
</table>

Table 4. Error in attitude and coordinate for segment 1.

<table>
<thead>
<tr>
<th></th>
<th>Max Absolute Value</th>
<th>Mean Value</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error in Change in Roll, rad</td>
<td>0.0019</td>
<td>-4.3364e-05</td>
<td>5.0803e-04</td>
</tr>
<tr>
<td>Error in Change in Pitch, rad</td>
<td>0.0018</td>
<td>-3.3591e-05</td>
<td>5.8129e-04</td>
</tr>
<tr>
<td>Error in Change in Yaw, rad</td>
<td>0.0003</td>
<td>5.7978e-06</td>
<td>1.0226e-04</td>
</tr>
<tr>
<td>Error in Change in X Coordinate, m</td>
<td>0.0804</td>
<td>-0.0013</td>
<td>0.0363</td>
</tr>
<tr>
<td>Error in Change in Y Coordinate, m</td>
<td>0.0248</td>
<td>0.0071</td>
<td>0.0056</td>
</tr>
<tr>
<td>Error in Change in Z Coordinate, m</td>
<td>0.0383</td>
<td>-0.0023</td>
<td>0.0107</td>
</tr>
</tbody>
</table>

Table 5. Error in change in attitude and coordinate for segment 2.

<table>
<thead>
<tr>
<th></th>
<th>Max Absolute Value</th>
<th>Mean Value</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error in Roll, rad</td>
<td>0.0088</td>
<td>-0.0045</td>
<td>0.0030</td>
</tr>
<tr>
<td>Error in Pitch, rad</td>
<td>0.0068</td>
<td>-0.0033</td>
<td>0.0016</td>
</tr>
<tr>
<td>Error in Yaw, rad</td>
<td>0.0026</td>
<td>7.2384e-04</td>
<td>7.9103e-04</td>
</tr>
<tr>
<td>Error in X Coordinate, m</td>
<td>0.4404</td>
<td>-0.1347</td>
<td>0.2067</td>
</tr>
<tr>
<td>Error in Y Coordinate, m</td>
<td>1.4061</td>
<td>0.6018</td>
<td>0.4636</td>
</tr>
<tr>
<td>Error in Z Coordinate, m</td>
<td>0.6519</td>
<td>-0.3973</td>
<td>0.1991</td>
</tr>
</tbody>
</table>

Table 6. Error in attitude and coordinate for segment 2.
VII. Conclusion

A Bayesian framework that is capable of simultaneous localization and mapping with detection and tracking of moving object (SLAM-DATMO) using only LiDAR data collected from a moving platform was successfully demonstrated. Previous works depend heavily on pre-filtering to remove clutter from the data before feeding into the tracking algorithm, which reduces the robustness of the whole framework due to the possibility of accidentally removing useful information as clutter. In this proposed framework, the PHD filter is used to handle the clutter within the measurement data without much pre-processing. Future proposed works include improving the performance of the proposed framework by incorporating additional sensor data at both the PHD filter level as well as at the SLAM and ICP algorithm. Additional positioning data, such as encoders, IMU and GPS data can be used to provide a better initial estimate for the ICP algorithm and they can also be fused with the localization solution provided by the ICP algorithm using an Extended Kalman Filter (EKF). Visual cues can also be fused with LiDAR data to generate point cloud with additional data, such as color and texture, to allow for better ICP localization solution. Contextual information obtained from the visual cues can also be used at the PHD filter level for assignment of more accurate motion model, better data association and tracking.
Figure 13. Tracks generated for dynamic objects.

VIII. Acknowledgment

The first two authors would like to acknowledge the support of Honeywell International Inc. and the Robotics, Sensors and Advance Manufacturing MnDrive initiative at the University of Minnesota through the MnDrive Industrial Partnership Grant.
References


