Integrated Perception and Planning for Autonomous Vehicles

Shubham Kedia
University of Illinois
Urbana Champaign
skedia4@illinois.edu

Yu Zhou
University of Illinois
Urbana Champaign
yuzhou7@illinois.edu

Sambhu H. Karumanchi
University of Illinois
Urbana Champaign
shk9@illinois.edu

May 11, 2022

1 Introduction

One of the key challenges in autonomous vehicle is navigation. Autonomous vehicles are required to find collision free trajectories in real-time for an unstructured or cluttered environment while satisfying the dynamic constraints of the vehicle. While there are reasonably efficient algorithms for machine perception [1, 2, 3, 4] and planning [5, 6, 7] independently, there is still lack of navigation solutions that are able to deal with vehicle non-holonomic [8] constraints and environmental constraints simultaneously and efficiently. The present work is structured to address these shortcomings by implementing the whole navigation pipeline as an integrated perception and planning solution for autonomous vehicles.

In the present work, the localization and mapping module is implemented using The Light Detection And Ranging (LiDAR) sensor. LiDAR is both fast and accurate sensor and can be used for a wide range of environmental conditions [9]. LiDAR Simultaneous Localization and Mapping (SLAM) has been extensively studied for machine perception [10]. In the present work, a naive attempt to develop a LiDAR SLAM system from scratch is made. The LiDAR SLAM incorporates an Inertia Measuring Unit (IMU) measurements a concept so called LiDAR inertial SLAM system. The localization is performed by Iterative closest point (ICP) point to plane metric [11] . The localization module generates a camera poses which is fed to the mapping module. The mapping module also incorporates the raw point cloud data to incrementally build an two dimension (2D) occupancy grid [12]. The 2D occupancy grid is be to generate a cost map or environmental constraints by a technique called Euclidean Distance Transform (EDT). The EDT encodes the information of the distance to the nearest obstacle and has been efficient tool to represent environmental constraints for optimization-based planning techniques [13, 14].

The planning module solves a non-linear trajectory optimization problem. The objective function is based upon minimization of cost from dynamic constraints, Environment cost map, and terminal cost or tracking error. The optimizer generates obstacle free dynamically feasible trajectories in real-time which can be easily tracked using any non-linear tracking controller.

The entire pipeline of the integrated perception and planning system is tested with KITTI odometry benchmark [15]. The current objective of this navigation module is to safely navigate to the target goal without any collisions with environmental obstacles. We made the source code available at BOX.

2 Background and Related Work

2.1 Mapping and Localization

SLAM can be roughly divided into two categories: LiDAR based and visual based. Researchers have been actively working in both of these techniques to develop a real-time and high fidelity environmental representation and accurate odometry. ORB-SLAM [2], RTABMap [16], Kimera [17], LSD-SLAM [1], are some of the well known open-source techniques which is able to deliver real-time performance and has been deployed for mobile robotics applications. However, visual SLAM suffers from limitation of visual sensor such as lighting condition, lack of optical texture in environment, and inaccuracies in depth estimation via indirect methods.
On the contary, LiDAR SLAM can often provide more robust localization and mapping by leveraging the directly captured 3D spatial data as laser point clouds. The LiDAR SLAM systems employs scan to scan matching algorithm the ICP [18], and has many variants such as [11, 19]. Some of the well known works based on LiDAR SLAM techniques

are: LOAM [10], LeGO-LOAM [20], Gmapping [21], and HectorSLAM [22]. LOAM extracts feature points of edges and planar surfaces based on local smoothness, and matches these feature points by minimizing the point-to-edge and the point-to-plane residuals. It achieves a translation error of less than 0.55 percent and ranks third on the KITTI odometry benchmark [15]. Gmapping constructs the SLAM on the basis of Rao-Blackwellized particle filters (RBPF) where each particle represent a pose of the robot and carries the individual map of the environment. Though, the pose representation is limited to only 2D applications.
While the LiDAR provides efficient and robust localization and mapping, it is prone to motion distortion which create in inadvertent drifts. This is addressed by introducing multi-modal sensor fusion techniques such as point cloud registration with integrated state estimation from IMU. Authors Tang et al. [23] investigated a loosely coupled Extended Kalman-Filter- based IMU-ICP-fusion. Similarly, authors Ye et al. [24] introduces a tightly coupled lidar-IMU fusion method by jointly minimizing the cost derived from lidar and IMU measurements. They have showed robust pose estimation with fast-motion, lidar-degraded, and limited overlapping cases. There are also deep learning-based sensor fusion techniques. DEEPLIO [25] is such a SLAM technique, where a neural network learns an appropriate fusion function by considering different modalities of its input latent feature vectors.

2.2 Planning

Optimization-based trajectory planning is widely used for local planning, in applications ranging from unmanned aerial vehicles to autonomous cars. One fundamental challenge in optimization-based trajectory planning is the appropriate formulation of collision avoidance constraints, which are generally non-convex and computationally demanding. In general, there are two ways of integrating perception data as collision avoidance constraints in trajectory optimization: either as hard or soft constraints.

There are several ways of formulating obstacles as hard constraints. Work in [26] convexifies the free space and performs Sequential Quadratic Programming (SQP) to converge to a fast solution. Work in [27] builds a piecewise flight corridor (PFC) that describes a piecewise convex feasible flight zone for drone applications. By introducing PFC, they are able to iteratively solve a series of convex optimization problems until the system reaches the goal position. The FaSTrack [28] method computes an upper bound with which to inflate the obstacles. However, this approach can be over-conservative in practice.

In general, while hard constraints guarantee safety, by treating the entirety of the free space equally, it can lead the solution space to be more sensitive to noise in perception and state estimation. In comparison, soft constraints, which are formulated as terms in the objective function of an optimization problem, lift this assumption by incorporating information such as distance to obstacles. Previous works often formulate a collision cost via a EDT of the map [29, 30, 31]. For our application, we choose to formulate obstacles as soft constraints using the EDT approach for its ease of implementation and intuitive understanding.

3 Methods

3.1 Multi-sensor Fusion for Vehicle State Estimation

An autonomous vehicle must locate itself with high precision to navigate independently within the environment. We follow the multi-sensor fusion approach to state estimation where the measurements from LiDAR and IMU are integrated into the motion dynamics model to estimate the state of the vehicle. The overall approach involves the following steps. The high-rate noisy measurements from IMU are given as inputs to the motion model that predicts at any time, the state at the next time instant. These predictions are updated every time a new IMU measurement flows into the system. In parallel, LiDAR generates more accurate position measurement but at a slower rate than IMU. Both the predicted state and LiDAR observations are fed into a variant of Kalman filter, called Error-State Extended Kalman Filter (ES-EKF) [32], to correct the predicted state. The prediction-correction mechanism for state estimation is repeated every time a new LiDAR measurements become available. The LiDAR generates high resolution point clouds using its laser-based scan of the environment. The trajectory of the LiDAR (and hence, the position of the vehicle in time) is estimated by associating scan points between the point clouds of multiple scans. This is accomplished through an ICP algorithm which finds the right translation and rotation matrices such the the current point cloud matches best with a reference cloud. In the following, we first describe the ICP point cloud matching algorithm and subsequently, the ES-EKF based prediction-correction scheme in detail.

3.1.1 ICP

Matching between any two LiDAR scans is done through the Iterative Closest Point algorithm. We consider the point-to-plane ICP with the error metric for minimization being the sum of squared distances between the source point and the tangent plane at its corresponding target point. That is, in the point-to-plane ICP, the problem is to find the transformation matrix M^* such that

$$M^* = \arg\min_M \sum_i ((\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t} - \boldsymbol{q}_i) \cdot \boldsymbol{n}_i)^2 \quad (1)$$

where p_i and q_i are the i-th source and its corresponding target respectively. n_i is the normal at the target.

In terms of rotation and translation matrices T and R, M can be written as

$$M = T.R$$

$$R \approx \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix}$$

and

$$R(\alpha, \beta, \gamma) = R_z(\gamma).R_y(\beta).R_x(\alpha)$$

for rotation angles α, β, γ about the axes x, y, z respectively. Assuming small values for α, β, γ, the trigonometric functions involved in $R_x(\alpha), R_y(\beta), R_z(\gamma)$ can be simplified to convert problem (1) into a linear least squares problem of minimizing the error in the N correspondence pairs. To ensure that the approximation $\hat{R}$ for R satisfies the condition of a rotation matrix, the error function E is approximated by

$$\begin{aligned} E &\approx \sum_i [(\boldsymbol{p}_i - \boldsymbol{q}_i).\boldsymbol{n}_i + \boldsymbol{t}.\boldsymbol{n}_i + \alpha(p_{i,y}n_{i,z} - p_{i,z}n_{i,y}) \\ &\quad + \beta(p_{i,z}n_{i,x} - p_{i,x}n_{i,z}) + \gamma(p_{i,x}n_{i,y} - p_{i,y}n_{i,z})]^2 \\ &\sim \Sigma_i[(\boldsymbol{p}_i - \boldsymbol{q}_i).\boldsymbol{n}_i + \boldsymbol{t}.\boldsymbol{n}_i + \boldsymbol{r}.(\boldsymbol{p}_i \times \boldsymbol{n}_i]^2 \end{aligned}$$

where $\boldsymbol{r} = [\alpha, \beta, \gamma]^T$. Then, solving (1) reduces to solving

$$x^* = \arg\min_x ||Ax - b||^2 \quad (2)$$

for which we use the singular value decomposition: $A = USV^T$. The solution is given by

$$x^* = A^+ b$$

where A^+ is the pseudo-inverse of A. Thus, $x^* = (\alpha^*, \beta^*, \gamma^*, t_x^*, t_y^*, t_z^*)$. Since $\hat{R}(\alpha^*, \beta^*, \gamma^*)$ is not a rotation matrix, $R(\alpha^*, \beta^*, \gamma^*)$ is used instead.

3.1.2 Error-State Extended Kalman Filter for State Prediction

The state of the vehicle ($\boldsymbol{x}$) is assumed to be composed of a large *Nominal* state ($\hat{\boldsymbol{x}}$) and a small *Error* state ($\delta\boldsymbol{x}$). The nominal state is updated by integrating the motion model and the error state is estimated using the Kalman filter, which is subsequently used to correct the nominal state. We make these notions precise below. To this end, let the vehicle state be described by its position, velocity, and orientation parameterized by unit quaternion, and the controls (inputs) by the force to accelerate and rotation rates. That is,

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{v}_k \\ \boldsymbol{q}_k \end{bmatrix} \quad \textit{and} \quad \boldsymbol{u}_k = \begin{bmatrix} \boldsymbol{f}_k \\ \boldsymbol{\omega}_k \end{bmatrix} \quad (3)$$

The motion model can be written as follows:

$$\begin{aligned} \boldsymbol{p}_k &= \boldsymbol{p}_{k-1} + \Delta t \boldsymbol{v}_k + \frac{\Delta t^2}{2}(\boldsymbol{C}_{ns}\boldsymbol{f}_k - g) \quad (4) \\ \boldsymbol{v}_k &= \boldsymbol{v}_{k-1} + \Delta t((\boldsymbol{C}_{ns}\boldsymbol{f}_k - g) \quad (5) \\ \boldsymbol{q}_k &= \boldsymbol{\Omega}(\boldsymbol{q}(\boldsymbol{\omega_{k-1}}\Delta t))\boldsymbol{q}_{k-1} \quad (6) \end{aligned}$$

where,

$$\begin{aligned} \boldsymbol{C}_{ns} &= \boldsymbol{C}_{ns}(\boldsymbol{q}_{k-1}) \\ &= \boldsymbol{\Omega}\left(\begin{bmatrix} q_\omega \\ \boldsymbol{q}_v \end{bmatrix}\right) \\ &= q_\omega \boldsymbol{I} + q_\omega I_4 + \begin{bmatrix} 0 & -\boldsymbol{q_v}^T \\ \boldsymbol{q_v} & -[\boldsymbol{q_v}]_X \end{bmatrix} \\ \boldsymbol{q}(\theta) &= \begin{bmatrix} \sin\frac{|\theta|}{2} \\ \frac{\theta}{|\theta}\cos\frac{\theta}{2} \end{bmatrix} \end{aligned}$$

$[\boldsymbol{q_v}]_X$ above is the skew-symmetric operator of $\boldsymbol{q_v}$. In ES-EKF, the nominal state is predicted by the equations (2)-(4) using the IMU inputs. However, the predicted state $\check{\boldsymbol{x}}_k := [\check{\boldsymbol{p}}_k, \check{\boldsymbol{v}}_k, \check{\boldsymbol{q}}_k]$ fails to account for noise and perturbations in the motion and needs to be augmented with the error-state dynamics model to accommodate the same. Denote the error state at time k by $\delta\boldsymbol{x}_k = [\delta\boldsymbol{x}_k, \delta\boldsymbol{v}_k, \delta\boldsymbol{q}_k]$. The non-linear dynamics are linearized using the first order approximation given by the Taylor's series. The linearized error- state dynamics can thus be written as:

$$\delta\boldsymbol{x}_k = \boldsymbol{F}_{k-1}\delta\boldsymbol{x}_{k-1} + \boldsymbol{L}_{k-1}\boldsymbol{n}_{k-1} \quad (7)$$

where $\boldsymbol{F}_{k-1}$ is the Jacobian of the motion model, $\boldsymbol{L}_{k-1}$ is the Jacobian of the motion model noise in the Taylor's series expansion, and $\boldsymbol{n}_k \sim N(\boldsymbol{0}, \boldsymbol{Q}_k)$ is the IMU measurement noise with covariance $\boldsymbol{Q}_k$.

These are given by:

$$\begin{aligned}
\boldsymbol{F}_{k-1} &= \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{I}_3\Delta t & 0 \\ 0 & \boldsymbol{I}_3 & -[\boldsymbol{C}_{ns}\boldsymbol{f}_{k-1}]_X\Delta t \\ 0 & 0 & \boldsymbol{I}_3 \end{bmatrix} \\
\boldsymbol{L}_{k-1} &= \begin{bmatrix} 0 & 0 \\ \boldsymbol{I}_3 & 0 \\ 0 & \boldsymbol{I}_3 \end{bmatrix} \\
\boldsymbol{Q}_k &= \Delta t^2 \begin{bmatrix} \boldsymbol{I}_3\sigma^2_{acc} & 0 \\ 0 & \boldsymbol{I}_3\sigma^2_{gyro} \end{bmatrix}
\end{aligned}$$

The prediction uncertainty in the state propagates by the state covariance matrix:

$$\check{\boldsymbol{P}}_k = \boldsymbol{F}_{k-1}\boldsymbol{P}_{k-1}\boldsymbol{F}^T_{k-1} + \boldsymbol{L}_{k-1}\boldsymbol{Q}_{k-1}\boldsymbol{L}^T_{k-1}$$

The uncertainty grows with time and is corrected using the measurements obtained either concurrently or intermittently. The measurement process is modeled as:

$$\begin{aligned}
\boldsymbol{y}_k &= \boldsymbol{p}_k + \boldsymbol{\nu}_k \\
&= \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{\nu}_k
\end{aligned}$$

where $\boldsymbol{\nu}_k$ is the LiDAR noise $\sim N(0, \boldsymbol{R})$. The measurement $\boldsymbol{y}_k$ helps obtain Kalman gain needed to correct the prediction which is given by

$$\boldsymbol{K}_k = \check{\boldsymbol{P}}_k\boldsymbol{H}_k(\boldsymbol{H}_k\check{\boldsymbol{P}}_k\boldsymbol{H}^T_k + R)^{-1}$$

and the error state $\boldsymbol{H}_k = \boldsymbol{K}_k(\boldsymbol{y}_k - \check{\boldsymbol{p}}_k)$. Now, the corrected state and state covariance given below are used recursively over time to carry out the state estimation.

$$\begin{aligned}
\check{\boldsymbol{p}}_k &= \check{\boldsymbol{p}}_k + \Delta\boldsymbol{p}_k && (8) \\
\check{\boldsymbol{v}}_k &= \check{\boldsymbol{v}}_k + \Delta\boldsymbol{v}_k && (9) \\
\check{\boldsymbol{q}}_k &= \boldsymbol{q}(\delta\boldsymbol{\phi}_k) \otimes \check{\boldsymbol{q}}_k && (10) \\
\hat{\boldsymbol{P}}_k &= (I - \check{\boldsymbol{K}}_k\check{\boldsymbol{H}}_k)\check{\boldsymbol{P}}_k && (11)
\end{aligned}$$

3.2 Mapping

Previously, we have defined $\boldsymbol{x}_k$ as the state vector of the autonomous vehicle describing it's location and orientation. Also, $\boldsymbol{x}_{1:k}$ describing the states as different timestamp. Now we can now use the state estimation to define a pose matrix $\boldsymbol{T}_k \in SE(3)$ in global world coordinate. The problem we are trying to address here is given a sequence of 3D lidar point cloud scans $\mathcal{P}_1, \mathcal{P}_2, ..\mathcal{P}_k$ in lidar coordinate frame and a sequence of vehicle poses $\boldsymbol{T}_1, \boldsymbol{T}_2, ..\boldsymbol{T}_k$, estimate a global consistent map $\mathcal{M}_1, \mathcal{M}_2, ..\mathcal{M}_k$ in the world coordinate frame. We implement this by registering the lidar scans $\mathcal{P}_k$ incrementally to the world coordinate frame using the odometry pose $\boldsymbol{T}_k$ at that timestamp. Mathematically, we define:

$$\mathcal{M}_k = \mathcal{M}_{k-1} \cup \{\boldsymbol{T}_k\mathcal{P}_k\} \tag{12}$$

Since, the above formulation is memory intensive and the autonomous vehicle trajectory planning is a 2D navigation problem. We project the 3D map constructed to 2D probabilistic occupancy grid map representation. This allows us to account for uncertainty and noise in the lidar point cloud scans and to store and process the environmental representation in a memory and computationally efficient format.

More formally, we define $p^{(m,n)}_k$ as an probabilistic occupancy grid with each cell index is represented by tuple (m,n) where $m \in M$ and $n \in N$. M, N denotes grid cells segmentation of the 3D point cloud. We compute $\tilde{p}^{(m,n)}_k = \mathbb{1}[|\mathcal{M}_k \downarrow \mathcal{W}| > \delta]$. Where $\mathcal{W} \in \mathbb{R}^2$ shows the grid coordinate (m,n), δ shows the minimum threshold of count to consider a cell as occupied. Modulating the value of δ allows us to filter the noise and ghost points in the lidar scans which can create false positives in the obstacle cost map. $\tilde{p}^{(m,n)}_k \in \{0, 1\}$ represents occupancy of a cell based on the evidence from current lidar scan. $\downarrow$ show the projection for a range $[z_{initial}, z_{final}]$ along the Z axis in world coordinate frame, which represents the subset of configuration space $\mathcal{C}$ of autonomous vehicle relevant to the 2D navigation problem. Tuning the parameters $z_{initial}, z_{final}$ allows to build the occupancy map without the roads, street lights, trees, etc. which are generally not an obstacle in the $\mathcal{C}$-SPACE of the vehicle.

Finally, we update the occupancy map using a probabilistic recursive update Equation 13.

$$p^{(m,n)}_k = p^{(m,n)}_{k-1} + \alpha(\tilde{p}^{(m,n)}_k - p^{(m,n)}_{k-1}) \tag{13}$$

Where $\alpha \in (0, 1]$ is a constant hyper-parameter which controls the exponential decay of history or previous observations. This formulation is especially useful for handling dynamic obstacles.

The probabilistic occupancy map is transformed to an obstacle cost map with Euclidean distance Transform (EDT). For computing the EDT, the probabilistic occupancy map is converted into binary image: $I_{\boldsymbol{i}} = \mathbb{1}[p^{(m,n)}_k > 0.5]$. Where $\boldsymbol{i} \in \mathbb{R}^2$ shows the image coordinate. The EDT for the binary image is defined as: $\min_j\{||\boldsymbol{i} - \boldsymbol{j}||_2; I_{\boldsymbol{j}} = 1\}$. This computation can be implemented with O(n) complexity with two passes on the image (top left to bottom right and back) [33].

3.3 Trajectory Optimization-Based Planning

We introduce collision avoidance into a trajectory optimization framework by means of a soft environmental constraint. We choose trajectory optimization for its ease of integrating costs, dynamics and constraints.

The problem formulation includes an environmental cost f_c, in addition to penalizing controls and deviation to the user command x_h:

$$\begin{aligned}\min_{\boldsymbol{x}[\cdot],\boldsymbol{u}[\cdot]} \quad & \lambda_{N,a} f_a(x_N) + \lambda_c f_c(x_N) \\ & + \sum_{k=0}^{N-1} \lambda_a f_a(x_k) + \lambda_c f_c(x_k) + \lambda_e f_e(u_k) \\ \text{s.t.} \quad & \forall k \in \{0,\ldots,T-1\}: \\ & x_{k+1} = f(x_k, u_k), \\ & x_k \in \mathcal{X}, u_k \in \mathcal{U},\end{aligned} \tag{14}$$

where $f_a(x_k) = (x_k - x^h)^T Q_k (x_k - x^h)$ is the attraction potential ($Q_k := Q$ for $k \in \{0,\ldots,N-1\}$), $f_e(u_k) = u_k^T R u_k$ the stage energy cost and $f_c(x_k)$ is stage collision repulsive potential, defined as:

$$f_c(x_k) = \begin{cases} (d(x_k) - d_c)^2 & d(x_k) \le d_c \\ 0 & d(x_k) > d_c \end{cases}, \tag{15}$$

where $d(x_k)$ is the distance between the state x_k and the nearest obstacle, d_c is the minimum path clearance, i.e., the minimum distance for the robot to stay away from obstacles. Note that we subtract the robot is inflated by a certain radius (which offsets $d(x_k)$ by a constant value) and bilinear interpolation is applied on the EDT map to extract the continuous values. One example of the collision cost function is shown in Fig. 1.

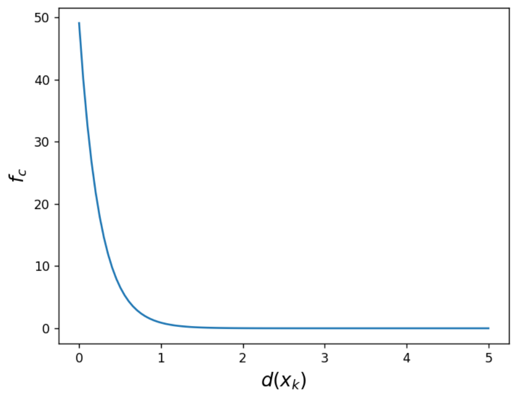
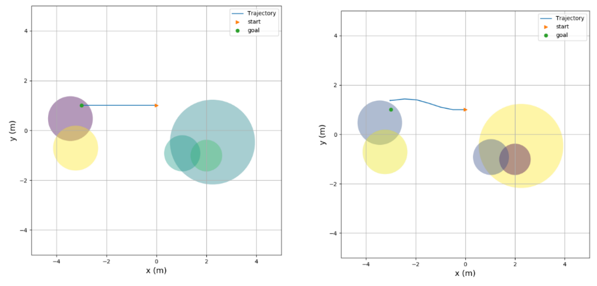


Figure 1: Collision cost function.

An example comparison between solving the trajectory optimization with and without considering environmental constraints is shown in Fig. 2.



(a) Plan without collision cost (b) Plan with collision cost

Figure 2: Comparison between solving the trajectory optimization problem without and with the environmental cost. The starting point is shown in orange and the goal is shown in green. The planned trajectory is shown as the blue curve. On the left, we observe that the path would end up going into the obstacle if it does not consider environmental constraints, while on the right the planned path is able to avoid obstacles along its way to the goal point and stop at a nearby point close to the target.

4 Experiments

We have evaluated our perception and planning pipeline on KITTI benchmark dataset. [15]. This dataset contains 3D point clouds coming from a Velodyne 64E installed on the top of the car. It also provides acceleration and angular rates of the vehicle from an OXTS RT3003 inertial and GPS navigation system. To reduce the computational load we have down-sampled the points with Open3D [34] voxel down-sampling of size 0.07. As the odometry ground truth is available for only the first 11 sequences, the sequences selected for odometry evaluation and mapping are: 03, 05, and 07, whereas planning is tested for sub-sections of sequence 05.

4.1 Localization

Figure 3 depicts the estimated trajectory against the ground-truth trajectory for the KITTI sequence 05 which is used to test the performance of the planning algorithm subsequently. The trajectories can be seen to agree well but for some drifts in a few locations which may be attributed to the IMU measurement bias and approximations used in the ICP algorithm.

In order to obtain trajectory level accuracy, we use the estimated trajectory pose matrices $\boldsymbol{T}_i^e \in SE(3)$

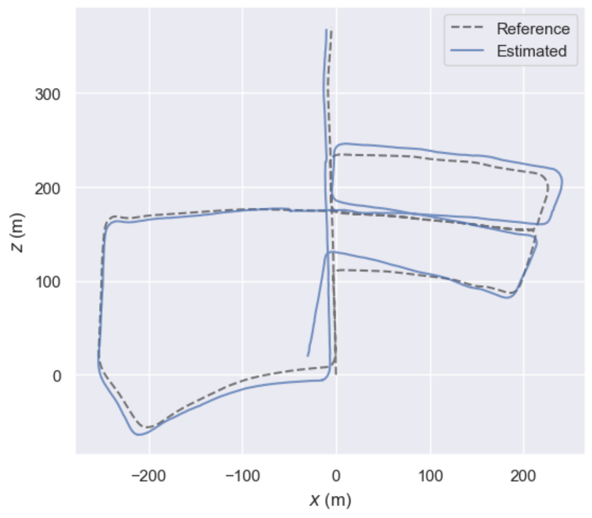


Figure 3: Qualitative odometry result on kitti dataset sequence 05

and the ground truth trajectory pose matrices $\boldsymbol{T}_i^g \in SE(3)$ at time instant $i, i = 1, \ldots, n$, where n is the total length of the time sequence. For evaluation, we consider the Relative Pose Error (RPE) which measures the local accuracy of the trajectory over a fixed time interval Δ and considers both rotational and translational errors. Defining the error matrix at instant i by

$$F_i^\Delta := ({\boldsymbol{T}_i^{g}}^{-1}\boldsymbol{T}_{i+\Delta}^g)^{-1}({\boldsymbol{T}_i^{e}}^{-1}\boldsymbol{T}_{i+\Delta}^e),$$

we obtain $m = n - \Delta$ individual relative pose error matrices from a sequence of n poses along the sequence. The RPE matrices, being $SE(3)$ matrices, can be split into translation (t_{rel}) and rotational (r_{rel}) components. The respective average errors are given by:

$$\begin{aligned} t_{rel}(\Delta) &= (\frac{1}{m}\sum_{i=1}^{m}||trans(F_i)||^2)^{\frac{1}{2}} \\ r_{rel}(\Delta) &= \frac{1}{m}\sum_{i=1}^{m}\angle(rot(F_i)) \end{aligned}$$

In our experiment, because we match successive frames, the time interval Δ is set equal to 1. The values of t_{rel}(as % of the total sequence length) and r_{rel}(in $deg/100m$) for the KITTI sequences 03, 05, and 07 are given in Table 1. Overall, the ES-EKF and ICP based method is found to perform well with low vehicle state estimation errors.

Table 1: Relative Pose Error in %

Sequence	03	05	07
t_{rel}	1.05	1.213	1.019
r_{rel}	2.2	3.1	4.00

4.2 Mapping

Figure 4, 6 shows the occupancy mapping for sequence 3, 5, and 7. The black pixels represents the obstacles which needs to avoided by the planner. The generated maps are able to represent the traversable regions in road and also street sidewalks, other cars, etc. as obstacles. Figure 5 shows a zoomed in subsection of sequence 5. The implemented mapping approach is able to define the cars parked on the sideways and road boundaries as obstacle. Also, the trees canopy and road surface, visible in the point cloud bird eye view are correctly not marked as obstacles. Though, due to inaccuracies in odometry module compounded over long distances created some inaccuracies. Figure 4 shows the middle road (traversed twice in this sequence) is mapped incorrectly with two projections side-by side due to a drift in the odometry. This is also due to lack of loop closure detection and map optimization technique in current framework.

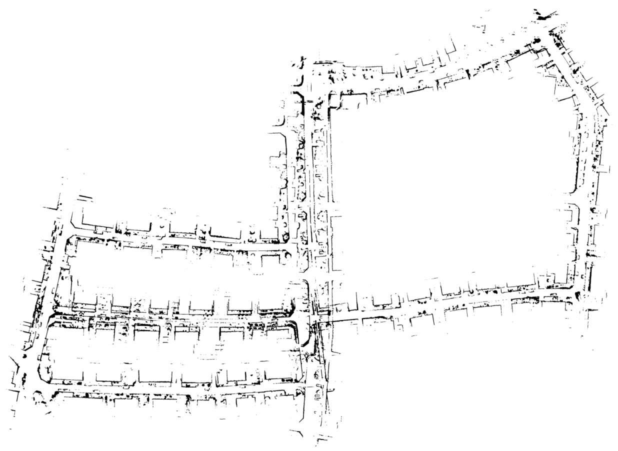
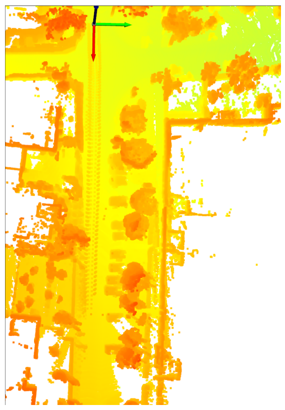


Figure 4: Occupancy grid mapping on KITTI dataset sequence 05

4.3 Planning

The dynamics model used for our planning is the differential drive model. The state is defined as:


(a) point cloud map

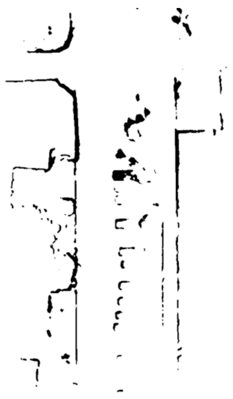
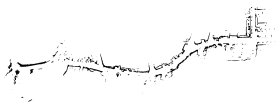

(b) occupancy grid map

Figure 5: A zoomed sub-section of KITTI sequence 5


(a) Occupancy map on sequence 03

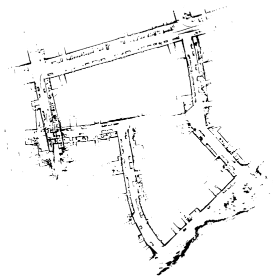

(b) Occupancy map on sequence 07

Figure 6: Qualitative Mapping results on other KITTI sequences

$\boldsymbol{x} = [x_c \quad y_c \quad \psi]^T$, and the control is defined as $\boldsymbol{u} = [v \quad \dot{\psi}]^T$, where x_c and y_c are coordinates of the car, ψ is the heading, v is velocity and $\dot{\psi}$ is the heading change.

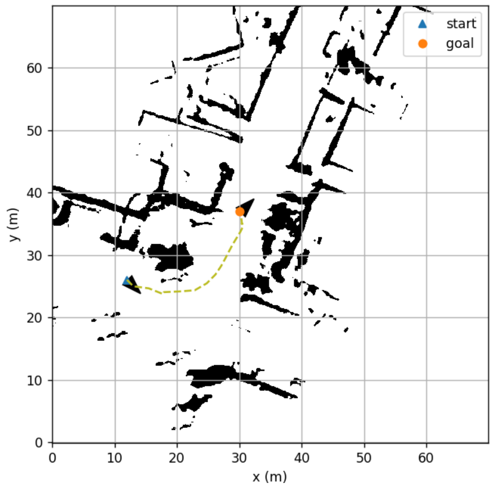
The dynamics are:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{x_c} \\ \dot{y_c} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ \dot{\psi} \end{bmatrix}. \tag{16}$$

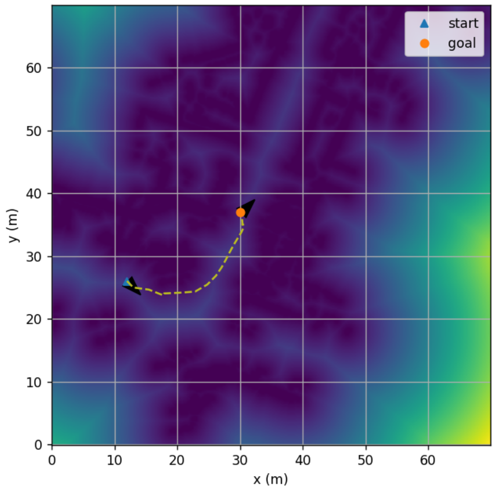
The state and control spaces for it are chosen to be $\mathcal{X} = \mathcal{R}^2 \times [-100, 100]$ and $\mathcal{U} = [-5, 5] \times [-3, 3]$. The cost matrices are $R = I_2$ for controls and $Q = Q_N = \text{diag}([1 \;\; 1 \;\; 0.1])$ for state deviation. The weighting coefficients are $\lambda_{N,a} = 10, \lambda_a = 0.001, \lambda_e = 0.01$ and $\lambda_c = 100$. The optimization time step is $h = 0.2s$ and horizon $N = 30$.

We chose the direct collocation approach to solve the optimization problem and apply explicit Euler integration method between each collocation point. The trajectory guess was initialized with a straight line in state-space between the initial and final states.

The problem is solved via CasADi [35] with the open-source solver Ipopt [36]. Note that we used the "limited-memory" option to perform quasi-Newton approximation for hessian approximation to increase computation speed. The planning result for multiple targets in subsections of KITTI sequence 5 is shown in Fig. 7 and Fig. 8. In both figures, the map on the left side is the occupancy grid map, and on the right is its corresponding EDT map. The starting point is shown in blue, and the targets are shown as circles with different colors. The planned trajectories are shown in yellow. The arrows represent the car with arrow head being the front side.

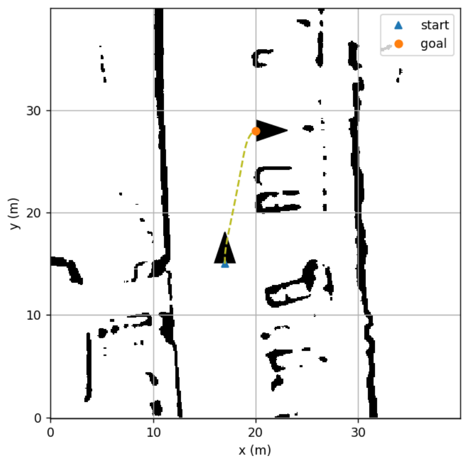


(a) Occupancy map

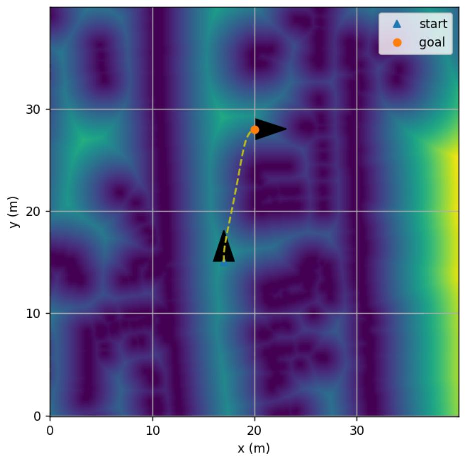


(b) EDT map

Figure 7: Planning for $goal_1$ with initial state: $[12, 26, -\pi/4]$ and goal state: $[30, 37, \pi/4]$. The offset of height and width in the map is $[60, 530]$.



(a) Occupancy map



(b) EDT map

Figure 8: Planning for $goal_2$ with initial state: $[17, 15, \pi/2]$ and goal state: $[20, 28, 0]$. The offset of height and width in the map is $[350, 330]$.

Table 2: Cost terms along the trajectory

Target	f_a	f_c	f_N	f_u	Total
goal$_1$	2.995	1.660	0.891	2.145	7.691
goal$_2$	1.276	0.004	0.001	1.911	3.192

The result shows that the planned paths are able to reach the goal if there are no obstacles along the path. Moreover, the planned paths are able to avoid obstacles along the way to the goal points and stop at nearby points with lower objective values. The cost terms along each trajectory are shown in Table. 2. In general, paths near obstacles have larger costs.

5 Conclusion

In this work, we explored a integrated perception and planning pipeline for autonomous vehicles. We have implemented a loosely coupled LiDAR-Inertial SLAM based on Error-State Extended Kalman Filter state estimation. Furthermore, we have successfully mapped the 3D spatial information of the environment to an obstacle cost map using an occupancy grid technique and Euclidean Distance Transform. We have quantitatively and qualitatively evaluated the localization and mapping on KITTI dataset (sequence 3, 5, and 7). With the generated cost map of environmental constraints, qualitative results for the proposed optimization based planner show the effective navigation through obstacle free trajectories for a non-holonomic system.

In future extension of this work, we will include loop closure and map optimization techniques to reduce the odometry drift and to develop a more consistent environmental representation. We will also explore quantitative evaluations for our optimization based planner and make some comparison with other popular planning techniques such as sampling based planners, potential fields, and learning based planning techniques.

References

[1] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. 1

[2] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014. 1

[3] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012. 1

[4] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, 13(3):108–117, 2006. 1

[5] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000. 1

[6] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013. 1

[7] Robert Bohlin and Lydia E Kavraki. Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 521–528. IEEE, 2000. 1

[8] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006. 1

[9] Pinliang Dong and Qi Chen. *LiDAR remote sensing and applications*. CRC Press, 2017. 1

[10] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, pages 1–9. Berkeley, CA, 2014. 1, 2

[11] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992. 1

[12] Hans Moravec and Alberto Elfes. High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, pages 116–121. IEEE, 1985. 1

[13] Frank Y Shih and Yi-Ta Wu. Three-dimensional euclidean distance transformation and its application to shortest path planning. *Pattern Recognition*, 37(1):79–92, 2004. 1

[14] Mohammadreza Radmanesh, Manish Kumar, Paul H Guentert, and Mohammad Sarim. Overview of path-planning and obstacle avoidance algorithms for uavs: A comparative study. *Unmanned systems*, 6(02):95–118, 2018. 1

[15] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 1, 2, 5

[16] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019. 1

[17] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696. IEEE, 2020. 1

[18] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992. 1

[19] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009. 1

[20] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018. 2

[21] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007. 2

[22] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *2011 IEEE international symposium on safety, security, and rescue robotics*, pages 155–160. IEEE, 2011. 2

[23] Jian Tang, Yuwei Chen, Xiaoji Niu, Li Wang, Liang Chen, Jingbin Liu, Chuang Shi, and Juha Hyyppä. Lidar scan matching aided inertial navigation system in gnss-denied environments. *Sensors*, 15(7):16710–16728, 2015. 2

[24] Haoyang Ye, Yuying Chen, and Ming Liu. Tightly coupled 3d lidar inertial odometry and mapping. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3144–3150. IEEE, 2019. 2

[25] Arash Javanmard-Gh, Dorota Iwaszczuk, and Stefan Roth. Deeplio: Deep lidar inertial sensor fusion for odometry estimation. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:47–54, 2021. 2

[26] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014. 2

[27] Hyungjoo Ahn, Junwoo Park, Hyochoong Bang, and Yoonsoo Kim. Model predictive control-based multirotor three-dimensional motion planning with point cloud obstacle. *Journal of Aerospace Information Systems*, pages 1–15, 2021. 2

[28] Mo Chen, Sylvia Herbert, Haimin Hu, Ye Pu, Jaime Fernandez Fisac, Somil Bansal, SooJean Han, and Claire J Tomlin. Fastrack: a modular framework for real-time motion planning and guaranteed safe tracking. *IEEE Transactions on Automatic Control*, 2021. 2

[29] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013. 2

[30] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pages 4569–4574. IEEE, 2011. 2

[31] Vladyslav Usenko, Lukas Von Stumberg, Andrej Pangercic, and Daniel Cremers. Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 215–222. IEEE, 2017. 2

[32] Kovac M. Milijas R. Car M. Bogdan S. Markovic, L. Error state extended kalman filter multi-sensor fusion for unmanned aerial vehicle localization in gps and magnetometer denied indoor environments. In *arXiv preprint arXiv:2109.04908 (2021)*. 2

[33] Gunilla Borgefors. Distance transformations in digital images. *Computer vision, graphics, and image processing*, 34(3):344–371, 1986. 4

[34] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 5

[35] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. 7

[36] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006. 7