Integrated Perception and Planning for Autonomous Vehicle Navigation: An Optimization-based Approach

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

Abstract

We propose an optimization-based integrated perception and planning framework for autonomous vehicle navigation that achieves real-time state estimation and path planning with high accuracy and robustness. Our Simultaneous Localization And Mapping (SLAM) module is based on Error-State Extended Kalman Filter (ES-EKF) for LiDAR-Inertial sensor fusion. The SLAM system generates a cost map using Euclidean Distance Transform (EDT) that directly encodes environmental constraints as a cost map. A non-linear trajectory optimization problem is formulated with the cost function and solved in real-time using the direct collocation approach. Our results on the KITTI dataset demonstrate the effectiveness of our framework.

1. Introduction

One of the key challenges in autonomous vehicle navigation is to find collision-free trajectories in unstructured or cluttered environments in real-time while respecting the constraints on the dynamics of the vehicle. Though there exist efficient algorithms for machine perception [1–4] and motion planning [5–7] individually, there is still lack of navigation solutions that can deal with both vehicle non-holonomic [8] constraints and environmental constraints simultaneously and efficiently. The present work addresses this shortcoming by implementing the whole navigation pipeline as an optimization-based integrated perception and planning solution for autonomous vehicles.

In this work, we implement the localization and mapping module using the Light Detection And Ranging (LiDAR) sensor, which is a fast and accurate sensor that can operate under a wide range of environmental conditions [9–12]. When the LiDAR SLAM incorporates Inertial Measuring Unit (IMU) measurements, it is called a LiDAR-Inertial SLAM system. Our LiDAR inertial fusion is implemented using the Error-State Extended Kalman Filter (ES-EKF) [13], which relies on linear error state dynamics for optimal state prediction. Robust localization and state estimation can be achieved using ES-EKF [14, 15]. In addition, we incorporate a loop closure module based on a non-histogram-based global descriptor for LiDARs, called *scan context* [16]. *Scan context* exhibited superior loop-detection capabilities across multiple datasets [16]. The fused odometry and loop closure constraints are optimized in the back-end by performing the global pose graph optimization using miniSAM [17] library. Our LiDAR-Inertial SLAM directly encodes the obstacles or environmental constraints in a cost map using the Euclidean Distance Transform (EDT) technique. The EDT represents the distance to the nearest obstacle and has been an efficient tool to represent environmental constraints for optimization-based planning techniques [18, 19].

The planning module solves a non-linear trajectory optimization problem to minimize the overall cost arising from dynamics constraint, environment cost map, and terminal cost or tracking error. The optimizer generates dynamics-compliant and obstacle-free feasible trajectories and the necessary control inputs for the autonomous vehicle.

The complete pipeline of the integrated perception and planning system is shown in Figure 1 and described in Section 3. Experimental results on the performance of the proposed approach on the KITTI odometry benchmark [20] are presented in Section 4.

2. Background and Related Work

We review below the literature related to individual modules used in our integrated perception and planning approach.

2.1. Mapping and Localization

SLAM is broadly divided into two categories: LiDAR-based and visual-based. Research in both directions is still active, primarily focusing on developing a real-time, high-fidelity environmental representation and accurate odome-

try. ORB-SLAM [2], RTABMap [21], Kimera [22], LSD-SLAM [1] are some of the well-known open-source techniques that can deliver real-time performance, and have been successfully deployed in many mobile robotics applications. However, visual SLAM inherits the limitations of visual sensors. Its accuracy is affected by poor lighting conditions, lack of optical texture in the environment, and inaccurate depth estimation via indirect methods. On the contrary, LiDAR SLAM offers more robust localization and mapping by leveraging the directly captured 3D spatial data as point clouds. Some well-known works based on LiDAR odometry include: LOAM [10], LeGO-LOAM [11], and HectorSLAM [23]. However, these techniques do not provide a loop closure module and are susceptible to localization drifts due to compounding errors. While LiDAR provides robust localization in a wide range of environments, it fails in scenarios lacking structural information such as tunnels, open-fields, etc. 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. [24] investigated a loosely coupled Extended Kalman-Filter-based IMU-ICP-fusion. Similarly, authors Ye et al. [25] introduced a tightly coupled LiDAR-IMU fusion method by jointly minimizing the cost derived from LiDAR and IMU measurements. There are also deep learning-based sensor fusion techniques such as DEEPLIO [26] where a neural network learns an appropriate fusion function by considering different modalities of its input latent feature vectors.

2.2. Trajectory Optimization

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. There are two ways to integrate perception data as collision avoidance constraints in trajectory optimization: hard or soft constraints.

There are multiple ways to formulate obstacles as hard constraints. Work in [27] convexifies the free space and solves a Sequential Quadratic Programming (SQP) problem. [28] builds a piece-wise flight corridor (PFC) that describes a piece-wise convex feasible flight zone for drone applications. The FaSTrack [29] method computes an upper bound with which to inflate the obstacles. These approaches can be over-conservative in practice. In general, while hard constraints guarantee safety, they treat the entirety of the free space equally, making the solution space more sensitive to noise in perception and state estimation. In comparison, soft constraints consider distance to obstacles and formulate collision cost via an EDT of the map [30–32]. In our application, we follow the soft constraint-based approach for ease of implementation and its intuitive appeal.

2.3. Integrated Perception and Planning

There are two primary approaches to integrated perception and planning in autonomous vehicles: the modular approach and the end-to-end deep learning approach [33]. An example of the latter is the ALVINN [34] model which uses a three-layer fully connected network to predict the steering wheel angle using a camera and radar images. [35] shows a large-scale convolutional neural network that is capable of steering a commercial vehicle along a highway or a smaller residential road. Although the above Imitation Learning (IL) approach performs well for simpler tasks such as lane-following, it falls short in handling complex and infrequently occurring long-tail traffic scenarios due to the distribution shift problem. Various methods such as data augmentation and diversification [36, 37] have been implemented to address this issue.

While IL frequently suffers from limited exposure to diverse driving scenarios, Reinforcement Learning (RL) is less prone to this issue. An RL agent learns from its interactions with the environment to find out a sequence of actions that will maximize the total reward received from the environment without actually requiring expert ground truth actions upfront [38]. RL is applied in challenging scenarios using deep Q-learning [39], Deep Deterministic Policy Gradients [40], and Proximal Policy Optimization [41]. However, RL is more data-intensive than IL and presents difficulties when applied to real-world scenarios. Exploration of driving policies may result in expensive damages to either the vehicle or the environment. There is an increase in research on sim-to-real transfer techniques [42], but generalization may still be problematic due to challenges in realistic simulation and domain adaptation. Proposed solutions to address sim-to-real transfer issues include supervised fine-tuning [43] and unsupervised learning [44].

Although the end-to-end approach has to deal with only fewer components, it is plagued by two major issues: the absence of hard-coded safety measures and interpretability. In contrast, the modular approach uses self-contained and inter-connected modules like perception, planning, and control, and relies on explicitly defined intermediary representations and deterministic rules, which ensure predictability within their established capabilities [45].

3. The Proposed System

Figure 1 gives an overview of our proposed system. The inputs to the system are LiDAR and IMU data frames. We follow the multi-sensor fusion approach to state estimation, where the measurements from LiDAR and IMU are fused together using the Error State Extended Kalman Filter for the LiDAR-Inertial Odometry (LIO). The IMU data gives

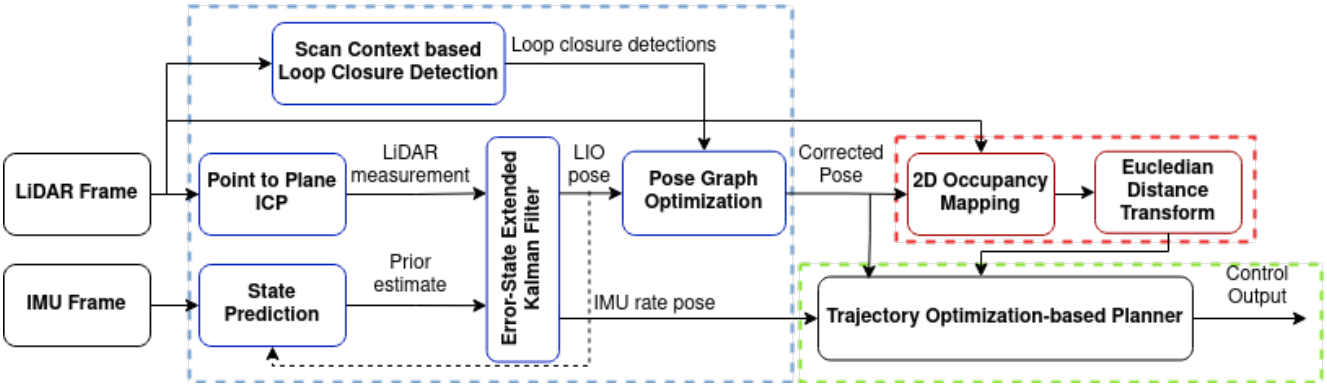


Figure 1. Pipeline of our proposed approach. Multi-sensor fusion is in blue, mapping is in red, and trajectory optimization is in green.

the prior for the filter-based odometry and also provides vehicle localization at the IMU frame rate. The LiDAR measurements, on the other hand, are used for state correction and loop closure detection. The estimated poses from filter-based odometry and the constraints from loop closure are optimized in the back-end using pose graph optimization. The drift-corrected poses are then fed to the mapping module to generate the EDT cost map. The optimization framework uses both the EDT cost map and localization as inputs to generate vehicle control outputs that are feasible and can account for vehicle dynamic constraints, while also avoiding collisions with environmental obstacles. The following subsections give the necessary details.

3.1. Multi-sensor Fusion for Vehicle State Estimation

The high-rate noisy measurements from IMU are given as inputs to the motion model which is integrated over time to predict the state at the next time instant. These predictions are updated whenever a new IMU measurement flows into the system. In parallel, LiDAR generates more accurate position measurements but at a slower rate than IMU. Both the model-based predicted state and LiDAR observations are fed into Error-State Extended Kalman Filter (ES-EKF) [13] to obtain the corrected predicted state. This prediction-correction mechanism for state estimation is repeated whenever a new LiDAR measurement becomes available. The trajectory of the LiDAR (and hence, the position of the vehicle at each instant) is estimated by associating scan points across the point clouds of multiple scans using the ICP algorithm that finds the correct translation and rotation matrices such that the current point cloud matches best with a reference cloud. For loop closure, we use the structural information-based *scan context* [16] to recognize places accurately and apply miniSAM [17] to solve the underlying graph optimization. Each step is described in detail below.

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 sum of squared distances between the source point and the tangent plane at its corresponding target point, as the error metric for minimization. That is, in the point-to-plane ICP, the problem is to find the transformation matrix $\boldsymbol{M}^*$ such that

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

where $\boldsymbol{p}_i \in \mathbb{R}^3$ and $\boldsymbol{q}_i \in \mathbb{R}^3$ are the i-th source and its corresponding target respectively. $\boldsymbol{n}_i \in \mathbb{R}^3$ is the normal at the target. In terms of rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$, $\boldsymbol{M}$ can be written as $\boldsymbol{M} = [\boldsymbol{R}|\boldsymbol{t}]$.

$$\boldsymbol{R}(\alpha, \beta, \gamma) = \boldsymbol{R}_z(\gamma).\boldsymbol{R}_y(\beta).\boldsymbol{R}_x(\alpha)$$

$$\boldsymbol{R}(\alpha, \beta, \gamma) \approx \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix} = \hat{\boldsymbol{R}}$$

Assuming small values for α, β, γ, the trigonometric functions involved in $\boldsymbol{R}_x(\alpha), \boldsymbol{R}_y(\beta), \boldsymbol{R}_z(\gamma)$ are simplified in $\hat{\boldsymbol{R}}$ above to convert problem (1) into a linear least squares problem of minimizing the error in the N correspondence pairs. 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

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} ||\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}||^2 \qquad (2)$$

The solution is computed using singular value decomposition method. Thus, $\boldsymbol{x}^* = (\alpha^*, \beta^*, \gamma^*, t_x^*, t_y^*, t_z^*)$. Since $\hat{\boldsymbol{R}}(\alpha^*, \beta^*, \gamma^*)$ is not a valid rotation matrix, $\boldsymbol{R}(\alpha^*, \beta^*, \gamma^*)$ 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 briefly introduce the methodology here without delving into details. A detailed treatment on error state extended Kalman filter can be found in [14]. Considering the following state and control vectors,

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

the motion model can be written as follows:

$$\boldsymbol{p}_k = \boldsymbol{p}_{k-1} + \Delta t \boldsymbol{v}_k + \frac{\Delta t^2}{2}(\boldsymbol{C}_{ns}\boldsymbol{f}_k - g) \tag{4}$$

$$\boldsymbol{v}_k = \boldsymbol{v}_{k-1} + \Delta t((\boldsymbol{C}_{ns}\boldsymbol{f}_k - g) \tag{5}$$

$$\boldsymbol{q}_k = \boldsymbol{\Omega}(\boldsymbol{q}(\boldsymbol{\omega_k}\Delta t))\boldsymbol{q}_{k-1} \tag{6}$$

where,

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

where q_ω represents the scalar part of quaternion and $\boldsymbol{q_v}$ represents the vector or imaginary part. $[\boldsymbol{q_v}]_X$ above is the skew-symmetric operator of $\boldsymbol{q_v}$. In ES-EKF, the nominal state is predicted by Equations 4, 5, and 6 using the IMU inputs. It is the best guess about what the true state could be based on knowledge about the vehicle's motion model. However, the predicted state $\boldsymbol{x}_k := [\boldsymbol{p}_k, \boldsymbol{v}_k, \boldsymbol{q}_k]$ fails to account for noise and perturbations in the motion. The errors build up over time as the motion model is integrated to estimate the state. The motion model needs to be augmented with the error-state dynamics model to facilitate correction. Denote the error state at time k by $\delta\boldsymbol{x}_k = [\delta\boldsymbol{p}_k, \delta\boldsymbol{v}_k, \delta\boldsymbol{q}_k]$. The non-linear dynamics are linearized using the first-order approximation given by Taylor's series. The linearized error-state dynamics can thus be written as:

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

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 $\boldsymbol{n}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k)$ is the IMU measurement noise with covariance $\boldsymbol{Q}_k$. 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 $\boldsymbol{\nu}_k \sim \mathcal{N}(0, \boldsymbol{R}_k)$ is the LiDAR noise with covariance $\boldsymbol{R}_k$. The LiDAR measurement $\boldsymbol{y}_k$ helps obtain the Kalman gain needed to correct the prediction. The corrected state and state covariance are used recursively over time to carry out the state estimation.

3.1.3 Scan Context for Place Recognition

Recognizing a previously visited place is essential for the *loop closure* step to correct any drift errors while building a consistent map of the environment. For effective place recognition, we apply the *scan context* [16] method that extracts structural information present in 3D point clouds. Scan context is an egocentric place descriptor built by splitting the whole set of points in 3D scan into mutually exclusive bins formed using N_r and N_s equally spaced *rings* and *sectors* in the sensor coordinates. Let $\mathcal{P}_{ij}$ denote the number of points in the bin formed by ring i and sector j. Then, an encoding function ϕ associates a real value to each bin so that each scan can be given a compact matrix representation. In our case, the value is set to the maximum height of the points to effectively summarize the vertical shape of surrounding structures. That is,

$$\phi(\mathcal{P}_{ij}) = max_{p\in\mathcal{P}_{ij}} z(\boldsymbol{p}) \tag{8}$$

where $z(\boldsymbol{p})$ is the z coordinate of point $\boldsymbol{p}$. The similarity between any two places q and c is measured using the average cosine distance between the associated scan context matrices $\boldsymbol{I}^q$ and $\boldsymbol{I}^c$, respectively. The similarity metric helps identify the previously visited locations on the map.

3.1.4 Pose Graph Optimization

In SLAM, a back-end is required to refine the map and the poses constructed in its front end. This is posed as a graph optimization problem where the nodes in the graph represent key-frames or landmark points, and edges represent the geometric relationships between the nodes. The optimization aims to find an optimal estimation of the values of the nodes which minimizes the errors determined by the constraints. We used miniSAM library [17] to solve the underlying non-linear least squares graph optimization efficiently by using the factor graph representation for pose graphs; each pose graph can be viewed as the product of a prior factor, odometry factors, and a loop closure factor.

3.2. Mapping

Recall that $\boldsymbol{x}_k$ is the state vector of the autonomous vehicle at time k representing its location and orientation. Let $\boldsymbol{x}_{1:k}$ denote the states at times 1 to k. The state estimation at time k gives the pose matrix $\boldsymbol{T}_k \in SE(3)$ in the global world coordinates. 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 $\boldsymbol{T}_1, \boldsymbol{T}_2, ..\boldsymbol{T}_k$, the mapping problem pertains to estimating a globally 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 in the world coordinate frame using the odometry pose $\boldsymbol{T}_k$ at that timestamp. Mathematically, we define:

$$\mathcal{M}_k = \mathcal{M}_{k-1} \cup \{\boldsymbol{T}_k \mathcal{P}_k\} \tag{9}$$

Since the above formulation is memory intensive and autonomous vehicle trajectory planning is a 2D navigation problem, we project the 3D map onto a 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.

Formally, we define $p_k^{(m,n)}$ as a probabilistic occupancy grid with each cell index represented by the tuple (m, n) where $m \in M$ and $n \in N$. M, N denote grid cells segmentation of the 3D point cloud. We compute $\tilde{p}_k^{(m,n)} = \mathbb{1}[|\mathcal{M}_k \downarrow \mathcal{W}| > \delta]$ where $\mathcal{W} \in \mathbb{R}^2$ denotes the grid coordinates (m, n), δ denotes the minimum threshold count required to consider a cell as occupied. By adjusting δ, it is possible to filter the noise and ghost points in the LiDAR scans which can create false positives in the obstacle cost map. $\tilde{p}_k^{(m,n)} \in \{0, 1\}$ represents occupancy of a cell based on the evidence from current LiDAR scan. $\downarrow$ denotes the projection for a range $[z_{initial}, z_{final}]$ along the Z- axis in the world coordinate frame, which represents a subset of the configuration space $\mathcal{C}$ of the vehicle relevant to the 2D navigation problem. Tuning the parameters $z_{initial}, z_{final}$ allows building 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 10.

$$p_k^{(m,n)} = p_{k-1}^{(m,n)} + \alpha(\tilde{p}_k^{(m,n)} - p_{k-1}^{(m,n)}) \tag{10}$$

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 into an obstacle cost map with Euclidean Distance Transform (EDT). For computing the EDT, the probabilistic occupancy map is converted into the binary image: $I_{\boldsymbol{i}} = \mathbb{1}[p_k^{(m,n)} > 0.5]$, 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) [46].

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 from the goal configuration x_g in planning horizon time-steps N:

$$\begin{aligned} \min_{\boldsymbol{x}[\cdot], \boldsymbol{u}[\cdot]} \quad & \lambda_{N,a} f_a(x_N) + \lambda_c f_c(x_N) \\ & + \sum_{k=0}^{N-1} \lambda_a f_a(x_k) + \lambda_c f_c(x_k) + \lambda_e f_e(u_k) \\ \text{s.t.} \quad & \forall k \in \{0, \ldots, N-1\}, \\ & x_{k+1} = f\left(x_k, u_k\right), \\ & x_k \in \mathcal{X}, \;\; u_k \in \mathcal{U} \end{aligned} \tag{11}$$

where $f_a(x_k) = (x_k - x_g)^T Q_k (x_k - x_g)$ 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:

$$f_c(x_k) = \begin{cases} \left(d\left(x_k\right) - d_c\right)^2 & d\left(x_k\right) \leq d_c \\ 0 & d\left(x_k\right) > d_c \end{cases}, \tag{12}$$

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 car to stay away from obstacles. Note that we offset $d(x_k)$ by a constant value with the assumption that the car is inflated by a certain radius and bilinear interpolation is applied on the EDT map to extract the continuous values. One example of the collision cost function is shown in Figure 2.

The direct collocation method is chosen here, which discretizes the trajectory optimization problem itself, converting the original trajectory optimization problem into a nonlinear program. There are mainly two benefits of the direct collocation approach. The first is that having x_k as explicit decision variable makes it easy to add additional state constraints. The solver effectively reuses the computation of each constraint. The second is that, for larger problems, evaluating the constraints can have a substantial computation cost. In direct transcription, one can evaluate these constraints in parallel.

An example of solving the trajectory optimization with and without considering environmental constraints is shown in Figure 3.

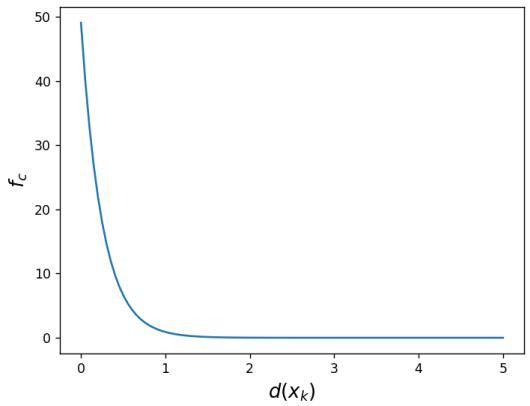
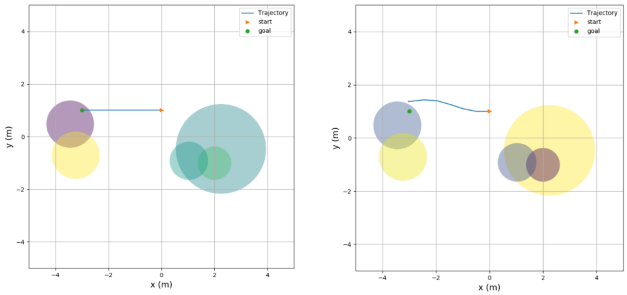


Figure 2. Collision cost function.



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

Figure 3. 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 evaluated our perception and planning pipeline on KITTI benchmark dataset. [20]. 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 a voxel down-sampling of size 0.07. As the odometry ground truth is available for only the first 11 sequences, we carried out odometry evaluation and mapping on Sequences 03, 05, and 07, and tested the planning on sub-sections of sequence 05.

4.1. Localization

Figure 4 depicts the estimated trajectory against the ground-truth trajectory for the KITTI sequence 05 which

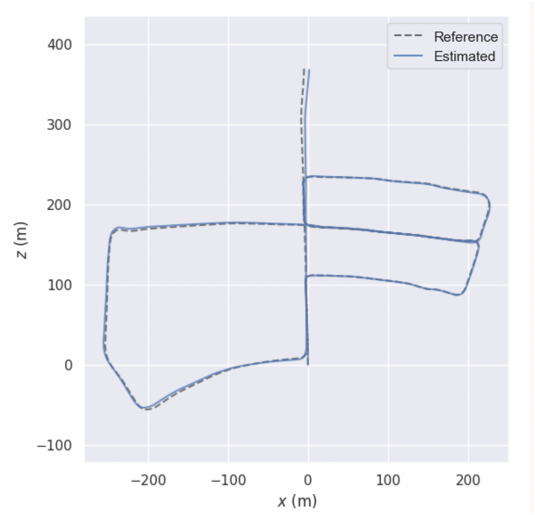


Figure 4. Qualitative odometry result on kitti dataset sequence 05 using evo [47]

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 $\boldsymbol{T}_i^e \in SE(3)$ 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. The RPE matrices when split into translation (t_{rel}) and rotational (r_{rel}) components, the respective average errors are given by:

$$t_{rel}(\Delta) = (\frac{1}{m}\sum_{i=1}^{m}||trans(F_i)||_2)$$

$$r_{rel}(\Delta) = \frac{1}{m}\sum_{i=1}^{m}\angle(rot(F_i))$$

In our experiment, matching is done between every two successive frames, and hence, 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 with loop closure 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 5, 7 shows the occupancy mapping for sequences 3, 5, and 7. The black pixels represent the obstacles that need to be avoided by the planner. The generated maps are able to represent the traversable regions in roads and also street sidewalks, other cars, etc. as obstacles. Figure 6 shows a zoomed-in sub-section of sequence 5. The implemented mapping approach is able to define the cars parked at the sideways and road boundaries as obstacles. Also, the trees' canopy and road surface, visible in the bird's-eye view is correctly not marked as obstacles.

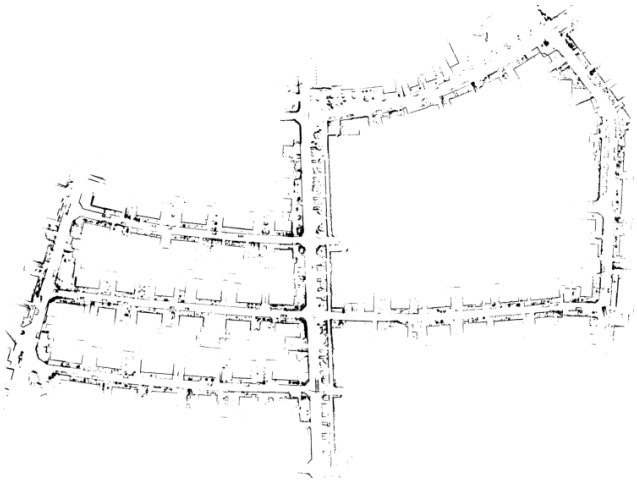


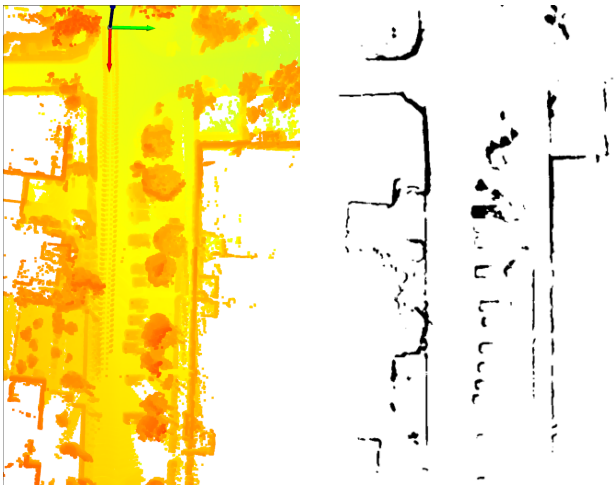
Figure 5. Occupancy grid mapping on KITTI dataset sequence 05

4.3. Planning

We use Dubins' car model to model the dynamics. The state is defined as: $\boldsymbol{x} = [x_c \quad y_c \quad \psi]^T$, 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 the velocity and $\dot{\psi}$ is the heading change. The dynamics are:

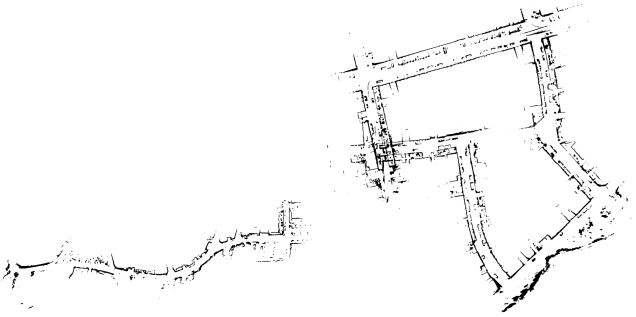
$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{x_c} \\ \dot{y_c} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} v\cos\psi \\ v\sin\psi \\ \dot{\psi} \end{bmatrix}. \tag{13}$$

The state and control spaces are set as $\mathcal{X} = \mathcal{R}^2 \times [-100, 100]$ and $\mathcal{U} = [-5, 5] \times [-3, 3]$. The cost matrices are $R = I_2$ for controls and $Q = Q_N = \text{diag}([1\ 1\ 0.1])$



(a) point cloud map (b) occupancy grid map

Figure 6. A zoomed sub-section of KITTI sequence 5

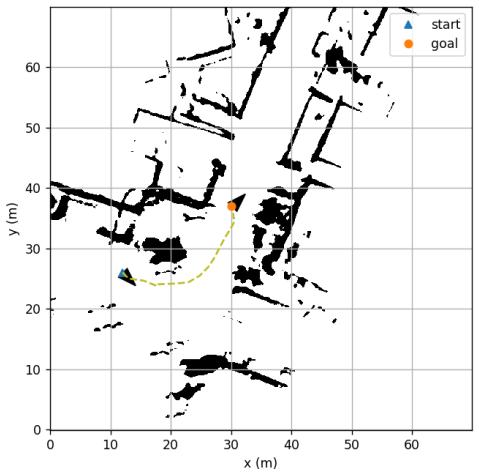


(a) Occupancy map on sequence 03 (b) Occupancy map on sequence 07

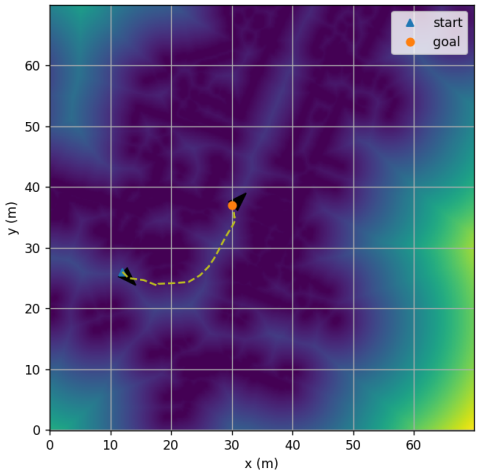
Figure 7. Qualitative Mapping results on other KITTI sequences

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 follow the direct collocation approach to solve the optimization problem and apply the 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 optimization model is solved via CasADi [48] with the open-source solver Ipopt [49]. We used the "limited-memory" option to perform the quasi-Newton method for Hessian approximation for computational speed. The planning results for multiple targets in subsections of KITTI sequence 5 are shown in Fig. 8 and Fig. 9. 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 the arrowhead indicating the front side.

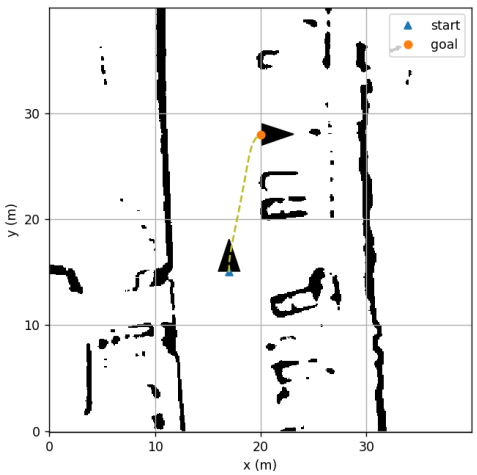


(a) Occupancy map

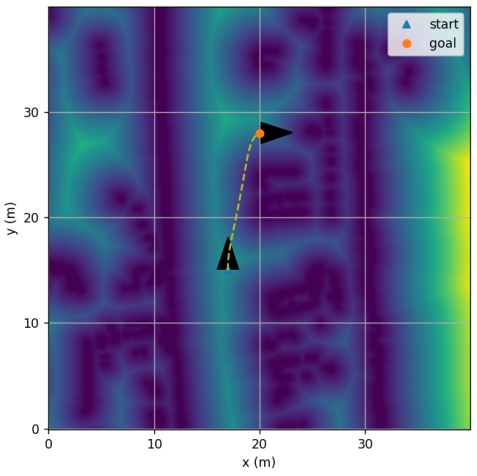


(b) EDT map

Figure 8. 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 9. Planning for $goal_2$ with initial state: [17, 15, π/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 reach the goal if there are no obstacles along the path. Further, the plans can avoid obstacles along the way to the goal 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 developed an integrated perception and planning pipeline for autonomous vehicles. We have implemented a loosely coupled LiDAR-Inertial SLAM with loop closure based on Error-State Extended Kalman Filter state estimation. Later, we mapped the 3D spatial information of the environment to an obstacle cost map using an occupancy grid and EDT. Our evaluation studies on Sequences 3, 5, and 7 of KITTI dataset demonstrate that our proposed planner can successfully offer navigation along obstacle-free trajectories for a non-holonomic system. In our future work, we will include uncertainty quantification of the perception system as a constraint to trajectory optimization and study its implications. We also plan to compare our optimization-based planner with sampling-based planners, potential fields, and learning-based 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

[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, 2

[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] 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. 1, 2

[12] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 5135–5142. IEEE, 2020. 1

[13] Stergios I Roumeliotis, Gaurav S Sukhatme, and George A Bekey. Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 2, pages 1656–1663. IEEE, 1999. 1, 3

[14] 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)*. 1, 4

[15] Venkatesh Madyastha, Vishal Ravindra, Srinath Mallikarjunan, and Anup Goyal. Extended kalman filter vs. error state kalman filter for aircraft attitude estimation. In *AIAA Guidance, Navigation, and Control Conference*, page 6615, 2011. 1

[16] Giseop Kim and Ayoung Kim. Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4802–4809. IEEE, 2018. 1, 3, 4

[17] Jing Dong and Zhaoyang Lv. miniSAM: A flexible factor graph non-linear least squares optimization framework. *CoRR*, abs/1909.00903, 2019. 1, 3, 4

[18] 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

[19] 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

[20] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 1, 6

[21] 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. 2

[22] 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. 2

[23] 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

[24] 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

[25] 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

[26] 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

[27] 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

[28] 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

[29] 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

[30] 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

[31] 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

[32] 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

[33] Ardi Tampuu, Tambet Matiisen, Maksym Semikin, Dmytro Fishman, and Naveed Muhammad. A survey of end-to-end driving: Architectures and training methods. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1364–1384, 2020. 2

[34] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988. 2

[35] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 2

[36] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019. 2

[37] Brady Zhou, Philipp Krähenbühl, and Vladlen Koltun. Does computer vision matter for action? *Science Robotics*, 4(30):eaaw6661, 2019. 2

[38] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998. 2

[39] Peter Wolf, Christian Hubschneider, Michael Weber, André Bauer, Jonathan Härtl, Fabian Dürr, and J Marius Zöllner. Learning how to drive in a real world simulation with deep q-networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 244–250. IEEE, 2017. 2

[40] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019. 2

[41] Błażej Osiński, Adam Jakubowski, Paweł Ziecina, Piotr Miłoś, Christopher Galias, Silviu Homoceanu, and Henryk Michalewski. Simulation-based reinforcement learning for real-world autonomous driving. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 6411–6418. IEEE, 2020. 2

[42] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*, 2018. 2

[43] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014. 2

[44] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017. 2

[45] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020. 2

[46] Gunilla Borgefors. Distance transformations in digital images. *Computer vision, graphics, and image processing*, 34(3):344–371, 1986. 5

[47] Michael Grupp. evo: Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`, 2017. 6

[48] 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

[49] 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