InEKF Localization and Semantic Mapping on the KITTI Dataset

Mu-Ti Chung, James Cooney, Tony Pan, Shoutian Wang, Haoxiang Wu

April 30, 2020

***Abstract*—Localization and mapping are the foundations for the movement of any mobile agent. The uncertainty in sensors prohibits accurate state estimation. Traditional methods such as the EKF may not always converge to the optimal solution. We have implemented a Left-Invariant Extended Kalman Filter that can better model the nonlinear uncertainty in robot motion and generate a more accurate trajectory. Furthermore, using our trajectory, lidar points, and images, we have built semantic maps that can encode more useful information for mobile robots.**

I. Introduction

A. SLAM

A mobile agent needs a map to know its location while it also needs its location to build a map. Simultaneous Localization and Mapping (SLAM) is the processing of incrementally constructing a spatial representation of the environment and locating the robot within the map at the same time.

A mobile robot typically has some sensors like motor encoders or inertial measurement units (IMUs) that track its movement, so a motion model can be constructed. It can also carry some sensors like cameras, lidar, and GPS for gathering information about its surrounding environment, and use it to build a sensor model. The problem is the sensor measurements are not precise, and the drift over time can cause the maps to become degenerate. The three main SLAM methods are Extended Kalman Filter (EKF), Particle Filter (PF), and Pose-Graph SLAM.

B. Extended Kalman Filter

Robot motion and observation is a Markov process because given the present state and observations, the future state is conditionally independent of the past. A recursive Bayes' Filter can thus be constructed. The current uncertainties about the robot state can first be propagated through the motion model. Afterward, the estimated state can be corrected by the sensor model. When the uncertainty is modeled as a Gaussian, such a Bayes' Filter is called a Kalman filter (KF). Since robot motion is most likely nonlinear, an extended version of KF, the extended KF (EKF), linearizes the model via first-order Taylor expansion. However, neglecting the higher-order terms of the nonlinearity may cause the filter to converge locally, in contrast to the guarantee of global convergence by KF.

C. Invariant Extended Kalman Filter

To better preserve the nonlinearities of the model while benefiting from the global convergence of KF, the theory of invariant extended Kalman filter (InEKF) was first proposed by Bonnabel et al [1]. It makes use of the fact that the geometry of robot motion is a manifold, the linearized model becomes a better approximation to the nonlinear one with a bigger set of trajectories. That way, InEKF would converge with any slow-varying permanent trajectory rather than equilibrium points as those with the traditional EKF.

The InEKF has been implemented in various applications, including robot state estimation, localization, odometry, and so on. Barczyk et al designed an invariant observer with IMU and magnetometer-plus-GPS sensors to estimate the states of a Helicopter UAV [2]. Hartley et al [3] developed a contact-aided InEKF for legged robots under the $SE_{N+2}(3)$ group. They used the points as landmarks for correction and propagated the robot's odometry through IMU sensors.

D. Pose-graph SLAM

Unlike the filtering methods, the graph-based SLAM method formulates SLAM into a graph construction and optimization problem. Each robot pose is represented as a node in a graph. During robot motion, edges connect its poses. An edge can either be an odometry constraint during movement, or a feature constraint for the previously observed area (loop closure). Because the graph is overdetermined, an optimization algorithm can correct the node poses.

E. ORB-SLAM

This SLAM algorithm uses ORB feature tracking between poses to find loop closures. ORB features are a method of key point detection where high dimensional descriptors that highlight unique features in an image. ORB features are similar but more efficient alternatives to SURF and SIFT features. They are used for SLAM by finding and saving some number of ORB features at each pose. When a new pose is added the newly detected ORB features are compared to those saved off in past poses. If they match than a loop closure is detected [4].

F. Semantic Mapping

A mobile agent needs a spatial representation of its surroundings to navigate through an environment. Traditionally, occupancy grid maps are widely used for spatial representation. As mobile agents like autonomous vehicles enter more complex environments, they require maps with more information to understand the significance of the scene and the objects within. A semantically labeled occupancy grid map not only represents the space around the mobile agent but also maintains for each cell a set of probabilities of semantic classes [5]. Knowing the semantic labels of each cell can help mobile agents better use these maps in real-life scenarios.

Robot mapping with LiDAR is common because LiDAR can provide high-frequency range measurements with relatively high precision [6]. However, LiDAR has several disadvantages compared to an RGB-D camera and stereo vision. LiDAR point clouds are sparse. More importantly, LiDAR points lack meaning within the entire point cloud. Meanwhile, while RGB-D cameras can provide semantic labels, they lack depth information, which is crucial to the safety of autonomous vehicles. A combination of LiDAR points and photos can be used to generate semantic maps.

G. KITTI

We are using the KITTI Vision Benchmark Suite for our project. KITTI is collected in the mid-size city Karlsruhe by a station wagon equipped with two high-resolution color and grayscale video cameras, Velodyne laser scanner and a GPS localization system. In the dataset, up to 15 cars and 30 pedestrians are visible per frame. The benchmark suite also provides evaluation metric and ground truth poses that we can compare with in the Results section of the report. [7]

H. Cityscapes

We are using the Cityscapes dataset for training the semantic labels used for mapping. Cityscapes is a benchmark suite and large-scale dataset to train and test approaches for pixel-level and instance-level semantic labeling. Cityscapes is comprised of a large, diverse set of stereo video sequences recorded in streets from 50 different cities. [8]

I. OctoMap

The OctoMap library is an efficient probabilistic 3D mapping framework that implements a 3D occupancy grid mapping approach. It displays a full 3D model including representation for unknown space. The map can be updated probabilistically. The algorithms are highly memory and time efficient [9]. We have used OctoMap in our software implementation of semantic mapping.

II. Methodology

A. Left-Invariant EKF

In this project, IMU and GPS sensor data are used for state estimation. The left-InEKF is chosen to fit the left-invariant observation of the GPS data because GPS uses a global frame of reference. In this section, the error dynamics and the equations for the filter will be derived with the inclusion of the IMU bias.

1) IMU Bias: The IMU biases are represented as an additive element to the measurements:

$$\begin{aligned}\tilde{\omega}_t &= \omega_t + \mathbf{b}_t^g + \mathbf{w}_t^g, \quad \mathbf{w}_t^g \sim \mathcal{GP}\left(\mathbf{0}_{3,1}, \mathbf{\Sigma}^g \delta\left(t - t'\right)\right) \\ \tilde{\mathbf{a}}_t &= \mathbf{a}_t + \mathbf{b}_t^a + \mathbf{w}_t^a, \quad \mathbf{w}_t^a \sim \mathcal{GP}\left(\mathbf{0}_{3,1}, \mathbf{\Sigma}^a \delta\left(t - t'\right)\right)\end{aligned} \tag{1}$$

Then, the bias-corrected inputs are denoted as $\bar{\omega}_t \triangleq \tilde{\omega}_t - \bar{\mathbf{b}}_g^t$ and $\bar{\mathbf{a}}_t \triangleq \tilde{\mathbf{a}}_t - \bar{\mathbf{b}}_t^a$. Also, we denote $\theta_t \triangleq \begin{bmatrix}\mathbf{b}_t^g & \mathbf{b}_t^a\end{bmatrix}^T \in \mathbb{R}$.

2) Error Dynamics: Since we are deriving the Left-InEKF, the left-invariant error along with the IMU bias augmented can be written as

$$\mathbf{e}_t^l \triangleq \left(\mathbf{X}_t^{-1}\bar{\mathbf{X}}_t,\ \bar{\theta}_t - \theta_t\right) \triangleq \left(\eta_t^l,\ \zeta_t\right) \tag{2}$$

The system dynamics can be represented using the IMU measurements:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{R}_t &= \mathbf{R}_t(\tilde{\omega}_t - \mathbf{w}_t^g)_\times \\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{v}_t &= \mathbf{R}_t(\tilde{\mathbf{a}}_t - \mathbf{w}_t^a) + \mathbf{g} \\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p}_t &= \mathbf{v}_t\end{aligned} \tag{3}$$

Moreover, the IMU bias dynamics are modeled using the Brownian motion model:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{b}_t^g &= \mathbf{w}_t^{bg}, \quad \mathbf{w}_t^{bg} \sim \mathcal{GP}\left(\mathbf{0}_{3,1}, \mathbf{\Sigma}^{bg}\delta\left(t - t'\right)\right) \\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{b}_t^a &= \mathbf{w}_t^{ba}, \quad \mathbf{w}_t^{ba} \sim \mathcal{GP}\left(\mathbf{0}_{3,1}, \mathbf{\Sigma}^{ba}\delta\left(t - t'\right)\right)\end{aligned} \tag{4}$$

Thus, the deterministic system dynamics can be written as

$$f_{\mathbf{u}_t}\left(\bar{\mathbf{X}}_t, \bar{\theta}_t\right) = \begin{bmatrix}\bar{\mathbf{R}}_t\left(\bar{\omega}_t\right)_\times & \bar{\mathbf{R}}_t\bar{\mathbf{a}}_t + \mathbf{g} & \bar{\mathbf{v}}_t \\ \mathbf{0}_{1,3} & 0 & 0 \\ \mathbf{0}_{1,3} & 0 & 0 \\ \mathbf{0}_{1,3} & 0 & 0\end{bmatrix} \tag{5}$$

Combining (3) and (4), the error dynamics can be written as

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{e}_t^l = \left(\frac{\mathrm{d}}{\mathrm{d}t}\eta_t^l,\ \begin{bmatrix}\mathbf{w}_t^{bg} \\ \mathbf{w}_t^{ba}\end{bmatrix}\right) \tag{6}$$

and we compute the first-order approximation of the error dynamics:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{R}_t^T\bar{\mathbf{R}}_t\right) &\approx \left(\bar{\mathbf{R}}_t\left(\mathbf{w}_t^g - \zeta_t^g\right)\right)_\times \\ \frac{\mathrm{d}}{\mathrm{d}t}\left(\bar{\mathbf{v}}_t - \mathbf{R}_t^T\bar{\mathbf{R}}_t\mathbf{v}_t\right) &\approx (\mathbf{g})_\times\xi_t^R \\ &\quad + \left(\bar{\mathbf{v}}_t\right)_\times\bar{\mathbf{R}}_t\left(\mathbf{w}_t^g - \zeta_t^g\right) \\ &\quad + \bar{\mathbf{R}}_t\left(\mathbf{w}_t^a - \zeta_t^a\right) \\ \frac{\mathrm{d}}{\mathrm{d}t}\left(\bar{\mathbf{p}}_t - \mathbf{R}_t^\top\bar{\mathbf{R}}_t\mathbf{p}_t\right) &\approx \xi_t^v + \left(\bar{\mathbf{p}}_t\right)_\times\bar{\mathbf{R}}_t\left(\mathbf{w}_t^g - \zeta_t^g\right)\end{aligned} \tag{7}$$

Finally, the error dynamics in (7) can be constructed as a linear system

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\xi_t \\ \zeta_t\end{bmatrix} = \mathbf{A}_t^l\begin{bmatrix}\xi_t \\ \zeta_t\end{bmatrix} + \mathbf{w}_t \tag{8}$$

where

$$\mathbf{A}_t^l = \begin{bmatrix}-(\bar{\omega}_t)_\times & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & -\mathbf{I}_3 & \mathbf{0}_{3,3} & \\ -(\bar{\mathbf{a}}_t)_\times & -(\bar{\omega}_t)_\times & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} & -\mathbf{I}_3 & \mathbf{0}_{3,3} \\ \mathbf{0}_{3,3} & \mathbf{I}_3 & \mathbf{0}_{3,3} & -(\bar{\omega}_t)_\times & \mathbf{0}_{3,3} & \mathbf{0}_{3,3} \\ & & \mathbf{0}_{6,15} & & & \end{bmatrix} \tag{9}$$

3) Propagation: In each iteration of the filtering process, we first construct a motion model that uses the IMU measurements to predict the robot pose at each state. The system dynamics are discretized with the assumption of zero-hold between two samples:

$$\begin{aligned}\mathbf{R}_k^- &= \mathbf{R}_{k-1}\Gamma_0(\bar{\omega}_k\Delta t) \\ \mathbf{v}_k^- &= \mathbf{v}_{k-1} + \mathbf{R}_{k-1}\Gamma_1(\bar{\omega}_k\Delta t)\bar{\mathbf{a}}_k\Delta t + \mathbf{g}\Delta t \\ \mathbf{p}_k^- &= \mathbf{p}_{k-1} + \mathbf{v}_{k-1}\Delta t \\ &\quad + \mathbf{R}_{k-1}\Gamma_2(\bar{\omega}_k\Delta t)\bar{\mathbf{a}}_k\Delta t^2 + \frac{1}{2}\mathbf{g}\Delta t^2\end{aligned} \tag{10}$$

where

$$\begin{aligned}\Gamma_0(\phi) &= I + \frac{\sin(\|\phi\|)}{\|\phi\|}\left(\phi_\times\right) + \frac{1 - \cos(\|\phi\|)}{\|\phi\|^2}\left(\phi_\times\right)^2 \\ \Gamma_1(\phi) &= I + \frac{1 - \cos(\|\phi\|)}{\|\phi\|^2}\left(\phi_\times\right) \\ &\quad + \frac{\|\phi\| - \sin(\|\phi\|)}{\|\phi\|^3}\left(\phi_\times\right)^2 \\ \Gamma_2(\phi) &= \frac{1}{2}I + \frac{\|\phi\| - \sin(\|\phi\|)}{\|\phi\|^3}\left(\phi_\times\right) \\ &\quad + \frac{\|\phi\|^2 + 2\cos(\|\phi\|) - 2}{2\|\phi\|^4}\left(\phi_\times\right)^2\end{aligned} \tag{11}$$

Additionally, the IMU biases are assumed to be slow-varying, so the propagation simply goes

$$\theta_{k+1} = \theta_k \tag{12}$$

and propagation of covariance is derived with the linearized error dynamics and exponential mapping:

$$\Sigma_k^- = \Phi_k^l \left(\Sigma_k + \mathbf{Q}_k \Delta t\right) {\Phi_k^l}^T \tag{13}$$

where $\Phi_k^l = \text{expm}(\mathbf{A}_k^l \Delta t)$.

4) Correction: We then use the GPS readings to correct our propagation. The H matrix in the correction step is derived as

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{0}_{3,6} & \mathbf{I}_3 & \mathbf{0}_{3,6} \end{bmatrix} \tag{14}$$

Then the remaining steps follow the ones in KF.

$$\begin{aligned}
\mathbf{S}_k &= \mathbf{H}_k \Sigma_k^- \mathbf{H}_k^T + \mathbf{R}_k^- \mathbf{N} {\mathbf{R}_k^-}^T \\
\mathbf{L}_k &= \Sigma_k^- \mathbf{H}_k^T \mathbf{S}_k^{-1} \\
\nu_k &= \left(\bar{\mathbf{X}}_k^-\right)^{-1} \mathbf{Y}_k - \mathbf{b} \\
\bar{\mathbf{X}}_k &= \bar{\mathbf{X}}_k^- \text{expm}\left[(\mathbf{L}_k \nu_k)^\wedge\right] \\
\bar{\theta}_k &= \bar{\theta}_k^- + \mathbf{L}_k \nu_k \\
\Sigma_k &= (\mathbf{I} - \mathbf{L}_k \mathbf{H}_k) \Sigma_k^- (\mathbf{I} - \mathbf{L}_k \mathbf{H}_k)^T \\
&\quad + \mathbf{L}_k \mathbf{R}_k^- \mathbf{N} {\mathbf{R}_k^-}^T \mathbf{L}_k^T
\end{aligned} \tag{15}$$

B. Semantic Mapping

In this section, we will first discuss image segmentation with U-net. We will then explain our technique for semantic mapping with LiDAR points and labels obtained from the first part.

1) U-Net: We have selected the U-net architecture because it performs well on multi-class image segmentation. As seen in Figure 1, the size of the input RGB images are 512x1024x3, and the size of the output is 512x1024x34. There are 34 classes that we can classify each pixel to.

For training our neural network, we are using the cross-entropy loss and the Adam optimizer to predict the pixel-wise category by corresponding scores. To evaluate our model, we consider the Intersection over Union (IoU) metric of main categories such as road, sidewalk, vegetation, car, and building.

Our network architecture consists of a contracting path (downsampling) and an expansive path (upsampling). The contracting path consists of repeated 3x3 convolutions (padding=1) followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation. The expansive path consists of an upsampling of the feature followed by a 2x2 transpose convolution (stride=2). Then it is followed by a concatenation with the corresponding feature map from the contracting path and two 3x3 convolutions each followed by ReLU [10].

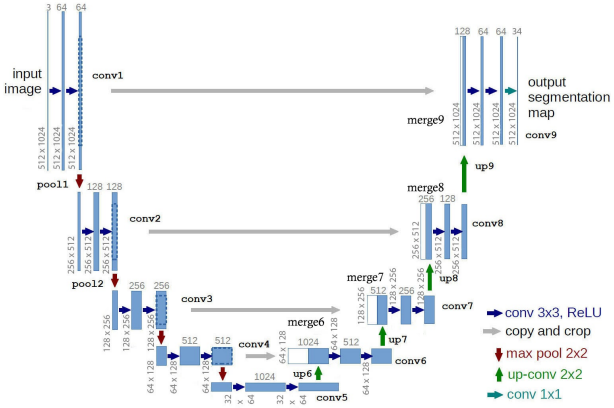


Fig. 1: U-Net Architecture

2) LiDAR Mapping: After extracting the semantic labels for each pixel in the input image, LiDAR point clouds are projected onto the camera frame based on LiDAR-camera extrinsic spatial transformation matrices. Since the Cityscapes dataset lacks LiDAR data, input images from the KITTI dataset are trained in our U-net to produce the label of each pixel.

Semantic labels in the camera frame and LiDAR points are associated with each other via a LiDAR-Image projection. Equation 16 demonstrates how each camera pixel $y = (u, v, 1)^T$ can be transformed from a 3D LiDAR point $x = (X, Y, Z, 1)^T$.

$$y = P_{rect}^{(i)} R_{rect}^{(0)} x \tag{16}$$

$$P_{rect}^{(i)} = \begin{bmatrix} f_u^{(i)} & 0 & c_u^{(i)} & -f_u^{(i)} b_x^{(i)} \\ 0 & f_v^{(i)} & c_v^{(i)} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{17}$$

$P \in \mathbb{R}^{3\times4}$ is the i^{th} projection matrix shown in equation 17, where $b_x^{(i)}$ denotes the baseline with respect to reference photo 0. $R_{rect}^{(0)} \in SE(3)$ is a homogeneous transformation mapping from the image plane to camera coordinates [7]. After associating every LiDAR point with some pixel within the image frame, a semantically labeled LiDAR point cloud can be generated.

There are many LiDAR point clouds generated from stacking point clouds based on vehicle poses. Due to the large scale of the data, it is challenging to process the point clouds in real-time with limited memory. To represent the point cloud efficiently, we are using OctoMap.

For mapping, every LiDAR point is transformed to pose frame as described by Equation 18

$$point_{pose} = pose \times point_{LiDAR} \tag{18}$$

in which $point_{LiDAR} \in \mathbb{R}^4$, and $pose \in SE(3)$.

III. Results

The link to our GitHub repository is https://github.com/tonypan2000/EECS568_final. Our final presentation video can be found at https://youtube.com/watch?v=A9tSE8NMWzA.

A. Left-InEKF

Our Left-Invariant EKF has shown promising results when estimating trajectories. We evaluated the performance of our left-InEKF with "2011_10_03_drive_0027" of a residential area from the KITTI dataset. Figure 2 compares the trajectory generated by our algorithm (shown in yellow) against the ground truth trajectory provided by KITTI (in blue). As you can see, our trajectory almost exactly matches that of the ground truth. Especially around corners where multiple trajectories intersect, our trajectory distinctly shows the overlapping paths where the car drove by repeatedly. Similarly, in corners with loop closures, our trajectory looks approximately the same as the ground truth.

While the trajectory overall is highly representative of the actual movement of the vehicle, the entire trajectory seems rotated clockwise for a bit compared to the ground truth trajectory.

To show the trajectory building process, we have made a video https://youtu.be/7E5PInxk9EU. The video plays at 10 fps and displays the pose and the image taken by one of the color video cameras. The top subplot shows the trajectory in blue, with a red '*' indicating the current pose. The bottom subplot shows the photo associated with the present state. Figure 3 shows the last frame of our video.

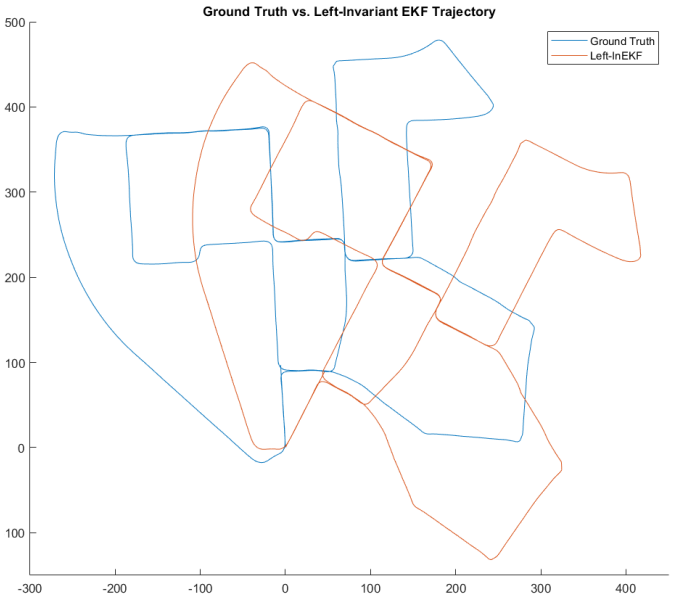


Fig. 2: Ground Truth vs. Left-Invariant EKF Trajectory

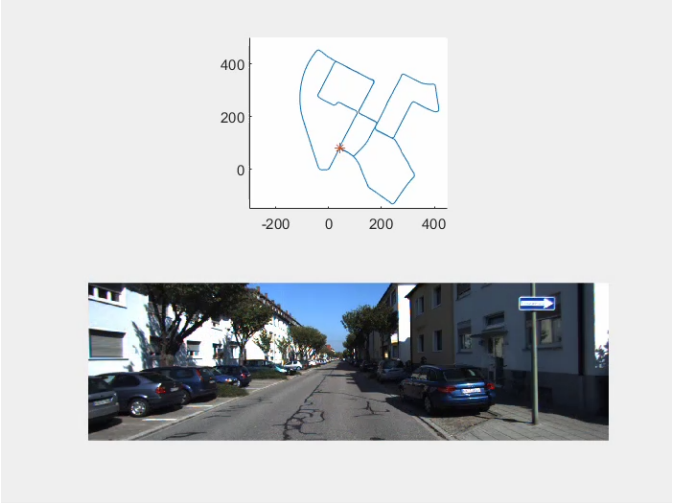


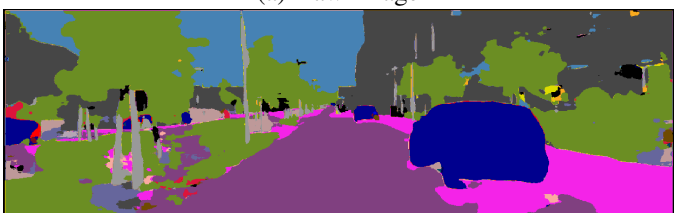
Fig. 3: A Frame of the Trajectory Building Video

B. Semantic Mapping

Input image from KITTI odometry sequence 06 and its corresponding semantic labels are shown in Figures 4a and 4b. From these two images, we can see the semantic map provides more important information about the complex environment and scenario in which mobile robots need to navigate.

After getting the semantic prediction of each pixel in the image, we can label the point cloud by

(a) Raw Image



(b) Semantically Segmented Image

projecting the LiDAR point cloud into the pixels in the image. The labeled point cloud is shown in Figure 5.

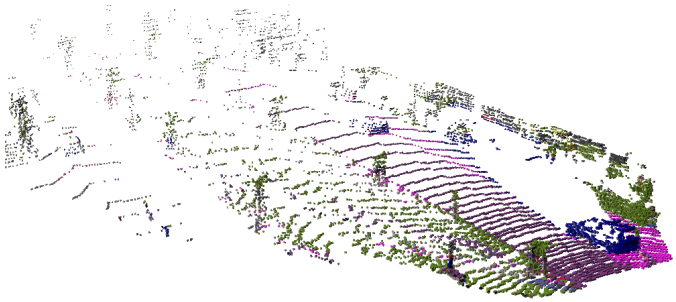


Fig. 5: Labeled Point Cloud

The evaluation of the LiDAR Mapping is done by comparing the 3D semantic maps constructed from the ground truth poses and those generated by the Left-InEKF in the visualization tool Octovis [9]. Figure 6 and Figure 7 show the results of the first 50 point clouds semantic mapping for KITTI sequence 00 constructed from ground truth poses and Left-InEKF poses.

IV. Discussion

A. Left-InEKF Localization

One of the approaches we attempted was loop closure with SURF features. Initially, we tried extracting SURF features from each photo. After building a trajectory with our Left-InEKF, we would also run a loop closing optimization. Our code could identify the loops and mark the connections as edges or constraints. However, while solving

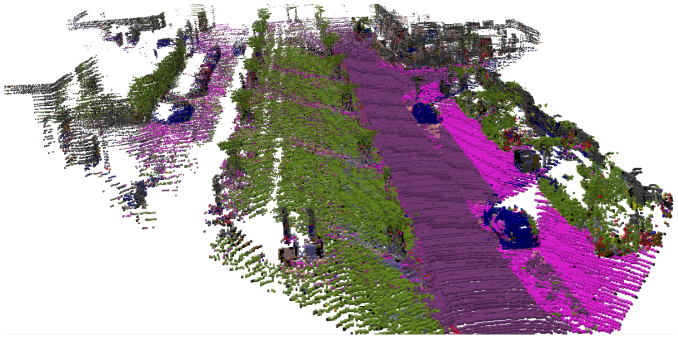


Fig. 6: Partial Semantic Mapping for KITTI 00 from Ground Truth Poses

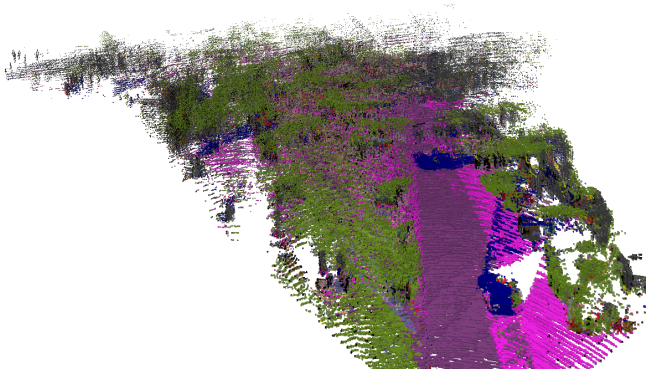


Fig. 7: Partial Semantic Mapping for KITTI 00 from Left-invariant EKF Poses

for the nonlinear least-squares of the pose graph to close the loops, our information matrix would blow up. The covariance matrix representation in Lie algebra makes it difficult for us to fix the issue. Nevertheless, the trajectory we generated without loop closure looks close enough to the ground truth, so we decided that loop closure was not necessary. Although we did not integrate a feature-based SLAM method, we successfully built semantic maps based on our Left-InEKF trajectories.

A discrepancy our plot has against the ground truth was the orientation. Our plot seems rotated around 45 degrees clockwise compared to the ground truth. We first thought the difference was caused by the initial mean. After tuning the initial values, we concluded that the initialization does not have any effect on the orientation of the trajectory. The other thing was attempted was to use different velocities to build our model. The rationale was that since all poses are off to the same amount of orientation, we should perhaps use a different

frame of reference. However, changing from the body frame of the car to the north-east frame also does not affect on our final trajectory. On the bright side, since our plot is consistent, and the relative change in poses is almost the same as the ground truth, it is still useful for autonomous vehicles on the road.

B. Semantic Mapping

The semantic map built from ground truth poses look very detailed. The structure of the road, car, vegetation, buildings, etc. is well-preserved. This proves that our semantic program is great at providing useful semantic maps for autonomous vehicles when given the ground truth poses.

From the result of semantic mapping, we can see there exists some difference between semantic mapping from ground truth poses and semantic mapping from Left-InEKF poses. Many points orient at the wrong position in the point cloud map. The difference in orientation is due to the difference between the $SO(3)$ rotation matrix from ground truth pose and Left-InEKF pose. The comparisons between ground truth poses and left-invariant EKF poses are shown in Figures 8, 9, and 10. We can see the translation part matches well, but the rotation parts are different from each other.

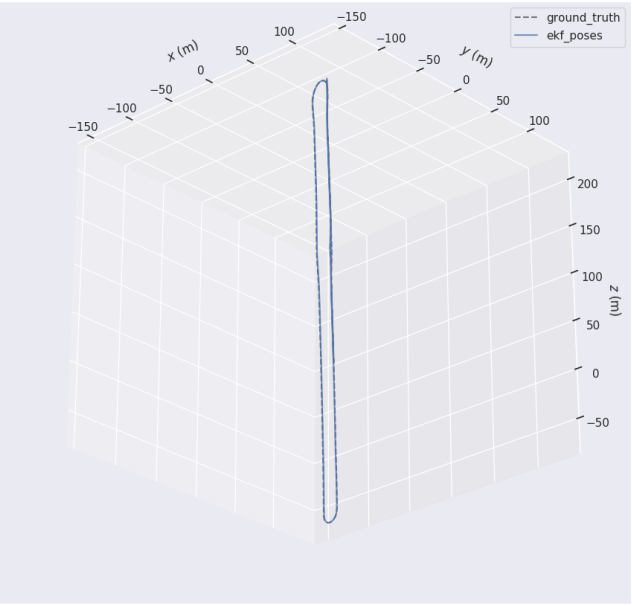


Fig. 8: Comparison Between Trajectories from KITTI Sequence 06

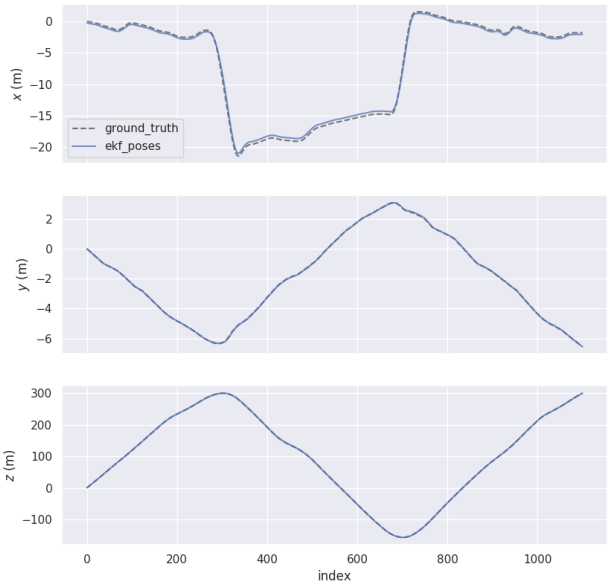


Fig. 9: Comparison Between Translation of Poses from KITTI Sequence 06

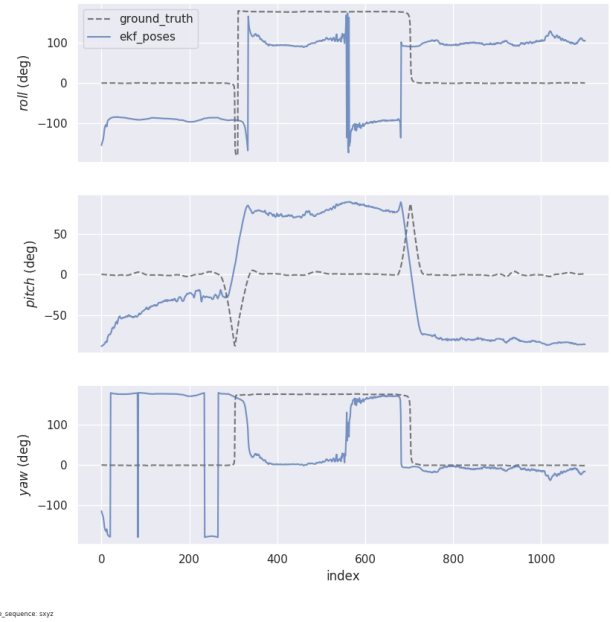


Fig. 10: Comparison Between Rotation of Poses from KITTI Sequence 06

C. Future Work

In the future, we need to first fix the rotation discrepancy between the Left-InEKF and the ground truth pose so that our semantic mapping program can build accurate 3D point cloud occupancy maps.

Because semantic 3D point cloud occupancy maps can give autonomous vehicles important knowledge of its surroundings that binary occupancy grid maps cannot offer, they can help make self-driving cars safer on the road. However, our project is currently limited to batch processing with a collected dataset. Ideally, our Left-InEKF and semantic mapping algorithms should function in real-time. In the future, we should integrate Left-InEKF with semantic mapping into SLAM. In each iteration, the Left-InEKF estimates a pose, identifies whether the scene has been observed before, and marks the same observations as constraints. An iterative smoothing algorithm can optimize the trajectory and close any loops. Using the pose, the photo and LiDAR points at that step, the 3D semantic occupancy map can be updated.

Currently, our localization code is written in MATLAB while the semantic mapping code is in C++ and Python. We could use the Robot Operating System (ROS) or Lightweight Communications and Marshalling (LCM) to send data packets among our programs for our SLAM integration.

V. Conclusion

In this project, we first took advantage of the fact that the trajectory of rigid body robot motion in $SE(3)$ is a manifold. Modeling a manifold in Lie groups allows our model to properly linearize the error dynamics and eventually converge to a globally optimal solution. We also used deep learning to segment more useful information from images and used the projection of LiDAR points to extract depth and maintain the structure of the 3D point cloud.

References

[1] S. Bonnabel, P. Martin, and E. Salaün, "Invariant Extended Kalman Filter: theory and application to a velocity-aided attitude estimation problem," in *48th IEEE Conference on Decision and Control*, Shanghai, China, Dec. 2009, pp. 1297–1304. [Online]. Available: https://hal-mines-paristech.archives-ouvertes.fr/hal-00494342

[2] M. Barczyk and A. F. Lynch, "Invariant observer design for a helicopter uav aided inertial navigation system," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 791–806, 2012.

[3] R. Hartley, M. Ghaffari Jadidi, J. Grizzle, and R. M Eustice, "Contact-aided invariant extended kalman filtering for legged robot state estimation," *Robotics: Science and Systems XIV*, Jun 2018. [Online]. Available: http://dx.doi.org/10.15607/RSS.2018.XIV.050

[4] R. Mur-Artal and J. D. Tardos, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, p. 1255–1262, Oct 2017. [Online]. Available: http://dx.doi.org/10.1109/TRO.2017.2705103

[5] L. Gan, R. Zhang, J. W. Grizzle, R. M. Eustice, and M. Ghaffari, "Bayesian spatial kernel smoothing for scalable dense semantic mapping," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, p. 790–797, Apr 2020. [Online]. Available: http://dx.doi.org/10.1109/LRA.2020.2965390

[6] D. Maturana, P.-W. Chou, M. Uenoyama, and S. Scherer, "Real-time semantic mapping for autonomous off-road navigation," in *Field and Service Robotics*, M. Hutter and R. Siegwart, Eds. Cham: Springer International Publishing, 2018, pp. 335–350.

[7] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597