Semantic Lidar Odometry

Ali Badreddine[1], Austin Jeffries[2], Tyler Glenn[3]

Abstract— **In this paper, we present the results of our investigation into the use of lidar point clouds with semantic labels for performing odometry of a moving robot. We compare trajectories calculated when using the semantic label information as we perform point cloud registration against the trajectories calculated without using the semantic labels. We also present initial results for the effects of removing points that correspond to moving objects from point clouds and discuss the effects on the transformations generated through point cloud registration and the resulting trajectories. Finally, we simulate loop closures and show how this lidar odometry could make up part of a complete Simultaneous Localization and Mapping (SLAM) solution.**

I. INTRODUCTION

An accurate odometry system is a key component of a complete SLAM solution. There are many different algorithms for performing odometry, and many different sensors that can be used. 3D Lidar sensors are increasingly used in robotics applications, especially self-driving cars. Lidar sensors can provide accurate data of the environment around the vehicle at a high scan rate, often around 10 Hz. Point cloud registration can be performed between successive lidar scans to estimate the 3D translation and rotation of the robot between the two scans. This point cloud registration is often done via the Iterative Closest Point (ICP) algorithm [1] [2], or a modification of ICP known as Generalized Iterative Closest Point (GICP) [3]. By transforming the estimate of the robot pose at the previous time step by the rotation and translation results of GICP applied between the lidar scan of the previous and current poses, an estimate of the current pose can be obtained. Performing this calculation iteratively over of a set of lidar scans will produce an estimate of the robot's trajectory. Going beyond this technique of only making use of transformations between consecutive poses, a smoothing technique can be implemented to incorporate transformations between non-consecutive poses as well.

The KITTI Odometry benchmark [4] is commonly used to evaluate the performance of visual and lidar-based odometry systems. The data in the KITTI dataset [4] include point clouds from a lidar sensor and two cameras mounted on top of a moving passenger car. Last year, the SemanticKITTI dataset [5] was released, in which the lidar data of the KITTI odometry [4] data was hand-labeled with 34 semantic classes. The goal of this paper is to compare performance of odometry performed with lidar point cloud registration when the additional information of semantic class labels is used. An extension for GICP that incorporates semantic labels, Semantic Iterative Closest Point (SICP), is presented in [6]. This paper describes the application of the SICP algorithm [6] to the KITTI Odometry benchmark [4] data with the use of the semantic labels provided in the SemanticKITTI dataset [5] in order to perform Semantic Lidar Odometry (SLO). All of our code is open-source and available on our github page. All the dependencies are open source as well.

[1]Ali Badreddine is an MS student, University of Michigan abadredd@umich.edu
[2]Austin Jeffries is an MS student, University of Michigan ajeffr@umich.edu
[2]Tyler Glenn is a Robotics MS student, University of Michigan tyglenn@umich.edu

II. Related Work

In just the past two years, researchers have started looking into using semantic point cloud information for odometry and SLAM. In [8], the semantic point clouds are approximated by a Normal Distributions Transform, and then the point clouds are registered using the NDT Histograms. They call this method Semantic-assisted NDT.

In addition, researchers have recently been using semantic labels to filter out dynamic objects and further improve results. In [9], researchers used a neural network to extract semantic labels and projected them on a spherical projection. This data was used to find dynamic objects and then filter them in a SLAM application.

III. Our Approach

The data pipeline for a real-time implementation of Semantic Lidar Odometry is illustrated in 1. For each lidar scan, the new point cloud would be have motion compensation applied using some estimate of the robot velocity, then the point cloud would be passed through a neural network designed for semantic segmentation. Following that, the point cloud would be aligned to the previous point cloud via SICP [6] to generate an estimate of the motion over the time step. Lastly, this new transformation would be added to an iterative smoother to update the estimated trajectory. The new pose estimate could then be fed back and used as the motion estimate for motion compensation of the next lidar scan.

For this paper we implemented an off-line version of SLO that uses motion-compensated point clouds from the KITTI Odometry benchmark [4] along with the corresponding labels from the SemanticKITTI dataset [5]. The first step was to downsample the point clouds to 20% of their original size, by removing 8 out of every 10 points, sequentially. We did this to reduce the processing time for SICP. Each lidar scan contains up to 100,000 points[4], many of which are very close to each other. Next, we generated the transformations between poses using the downsampled and labeled point clouds in an off-line batch-processing step utilizing SICP

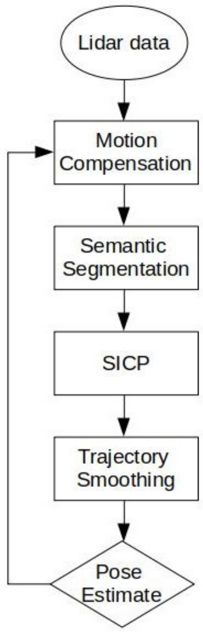


Fig. 1: Real-time SLO data pipeline

[6]. The output of SICP for our problem is a 4x4 matrix that belongs to the 3D special Euclidean matrix group, SE(3). We calculated these matrix transformations between each pose and up to five subsequent poses. Since a vehicle like the one used to generate the data in the KITTI dataset[4] has high inertia and the time step between successive lidar scans was short (0.1 seconds), we chose to use the SICP-calculated transformation between poses $i-1$ and i as the initial estimate in the SICP solver for calculating the transformation between poses i and $i+1$. We found that this reduced the solution time for SICP by just over a factor of two. We then fed these transformations as edges into a factor graph and solved them using Iterative Smoothing And Mapping 2 (ISAM2) [7] to get the final estimate of the vehicle trajectory. We also ran the same process using transformations generated via GICP [3] for comparison to see if there was improvement in the trajectory through use of the semantic information. For comparison, we also ran SLO with only transformations between consecutive poses and calculated the trajectory by right-multiplying the previous pose (an SE(3) matrix) by the current transformation to get the current pose estimate.

We also investigated whether we could make use of the semantic information and improve the estimated trajectory by removing points from the lidar point clouds that corresponded to moving vehicles. The SemanticKITTI [5] classes differentiated between non-moving cars and moving cars. In addition, there were two classes, 'unlabeled', and 'outliers' that we removed as well. These are all the classes for which we removed points:

- unlabeled
- outlier
- moving-car
- moving-bicyclist
- moving-person
- moving-motorcyclist
- moving-on-rails
- moving-bus
- moving-truck
- moving-other-vehicle

Lastly, we investigated how SLO could fit into an overall SLAM system by generating simulated loop closures between poses using the KITTI ground truth data [4] and feeding those into the ISAM2 smoother along with the other transformations.

IV. Results

A brief video of our results can be seen here

A. Odometry

We ran our offline SLO method on the 07 and 00 KITTI [4] lidar data with the corresponding SemanticKITTI [4] class labels. Using the sequence of ground truth pose data, we calculated the ground truth transformations between each pose according to this formula:

$$T_{gt,i} = P_{gt,i-1}^{-1} * P_{gt,i} \tag{1}$$

To calculate the translation error of the transformations, we calculated the Euclidean norm of the difference between the $[x, y, z]$ translation of the transformation and that of the ground truth transformation. We calculated the rotation error of the transformations according to (2), where $||\cdot||_F$ is the Frobenius norm.

$$\epsilon_r = ||R_{SICP} * R_{gt}^{-1} - I_{3x3}||_F \tag{2}$$

Figure 2 shows boxplots for the rotation and transformation errors for both SICP and GICP for all transformations between consecutive poses in the 07 dataset. There is no significant difference between the SICP results and GICP results in this case.

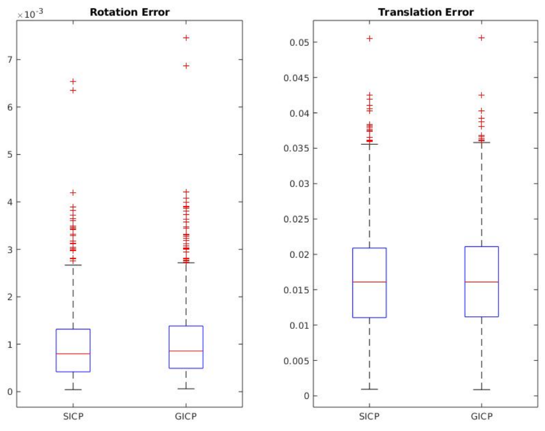


Fig. 2: SICP vs GICP Transformation Errors

Figure 3 shows the SICP and GICP trajectories calculated without smoothing by using only consecutive poses. The current pose is estimated by applying the current estimated transformation to the previous pose estimate. These trajectories are shown compared to the ground truth.

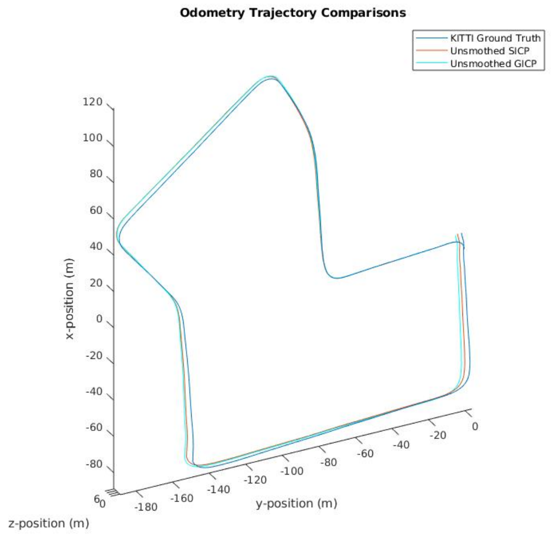


Fig. 3: SICP and GICP Trajectories Without Smoothing Vs Ground Truth, 07 dataset

We also solved for the trajectory by using transformations between consecutive poses, and between poses separated by two time steps. We put all of the transformations as edges into a factor graph and solved using ISAM2 [7]. Figure 4 shows the results for SICP and GICP vs the ground truth. Figure 5 compares the translational error of SICP and GICP for both our smoothed and unsmoothed tests.

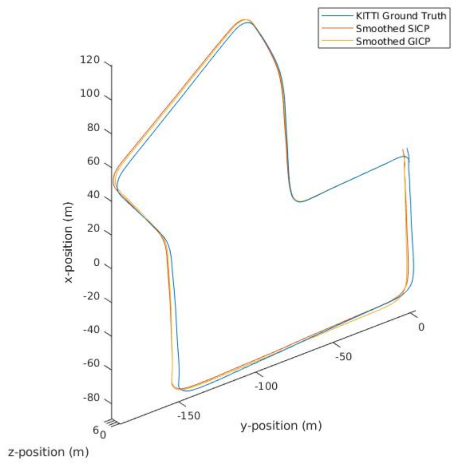


Fig. 4: SICP and GICP Trajectories With Smoothing Vs Ground Truth, 07 dataset

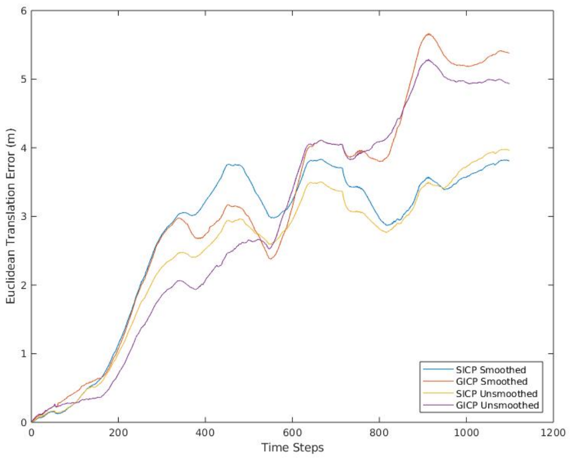


Fig. 5: SICP and GICP Translational Error Over the Entire 07 Trajectory

The KITTI Odometry benchmark [4] uses the percent translation error, $\epsilon_{t,\%}$ as one of the metrics for ranking algorithms, which is defined in (3), where p_i and $p_{i,gt}$ are the $[x, y, z]$ position vectors of the estimated pose and ground truth pose, respectively, at pose i, and d_i is the length of the ground truth trajectory at pose i.

$$\epsilon_{translation,\%} = \frac{||p_i - p_{i,gt}||}{d_i} \tag{3}$$

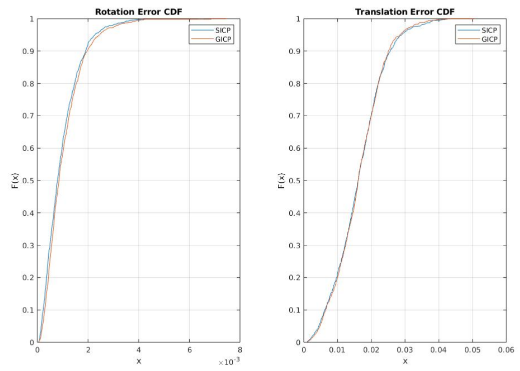
The error, $\epsilon_{t,\%}$, for our SICP and GICP trajectories are shown in table I. In both the unsmoothed and smooth tests, SICP had slightly lower were than GICP. Interestingly, the addition of the non-consecutive transformations led the smoothed results to have a higher error than the unsmoothed trajectory estimates. We tried adding even more non-consecutive edges and that increased the error even further.

TABLE I: $\epsilon_{t,\%}$ for SICP and GICP, 07 dataset

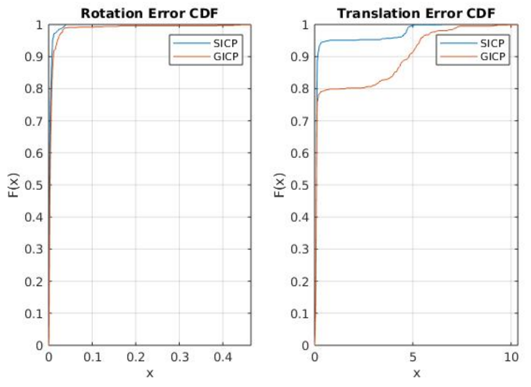
	SICP	GICP
Unsmoothed	0.81	0.93
Smoothed	0.89	1.03

We ran the same tests on the first 3010 poses of the 00 dataset. We investigated further into the accuracy of SICP and GICP transformations by calculating the translation and rotation error (2). Figure 6 shows how pose separation affected rotation and transformation error for the 00 dataset. Figure 7 shows the unsmoothed SICP and GICP trajectories compared to the ground truth. Table II shows the error, $\epsilon_{t,\%}$. The mean $\epsilon_{t,\%}$ is much larger than for the 07 dataset. Figure 8 breaks down the error for the unsmoothed SICP trajectory into the x, y, and z coordinates. The error for the z coordinate tended to be larger than that for x and y, even though most of the displacement is in x and y. We saw this unexpectedly large z-coordinate error consistently in the 00 dataset, and to a lesser degree in the 07 dataset.

$$\epsilon_T = ||log(T * T_{gt}^{-1})^{\vee}|| \tag{4}$$



(a) CDF of the Transformation Error for Poses 1 Time Step Apart



(b) CDF of the Transformation Error for Poses 5 Time Steps Apart

Fig. 6: Rotation and Translation Error CDFs for Point Clouds Separated by 1 and 5 Time Steps

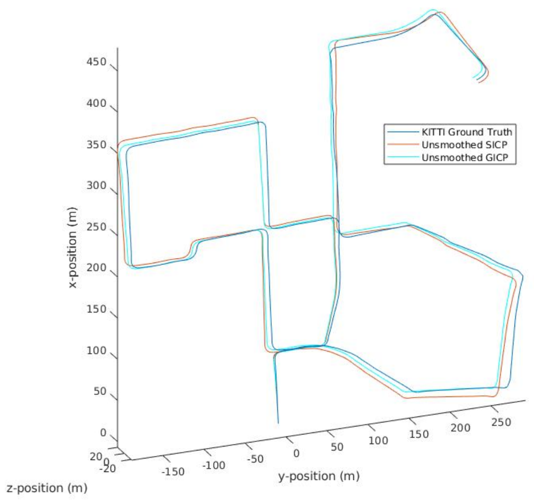


Fig. 7: SICP and GICP Unsmoothed Trajectories Vs Ground Truth, 00 dataset

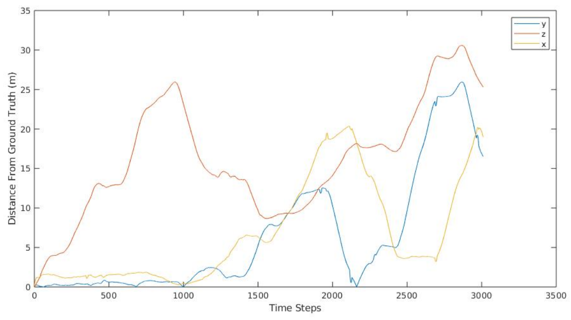


Fig. 8: Coordinate Translation Errors of SICP Unsmoothed Trajectory, 00 Dataset

TABLE II: $\epsilon_{t,\%}$ for SICP and GICP, 00 dataset

	SICP	GICP
Unsmoothed	2.28	2.10

B. Odometry with Moving Classes Removed

Our next tests were to see if our odometry results could be improved by removing points labeled as moving objects. The 00 dataset had very few such points, but the 07 dataset had more. We calculated the transformation error using the SE(3) transformation error metric described in [6] and (4). We did this for the entire trajectory for point clouds with and without points removed. Figure 9 shows the boxplots of the difference in ϵ_T for the original points clouds versus those with points removed. Positive values indicate that the transformation with points removed had lower error. Overall, the mean is near zero, indicating that for the majority of poses, there was not much change to the transformation error. This make sense as the majority of lidar scans in the 07 dataset contain few, if any, points corresponding to the classes removed. The outliers in these plots are skewed positive, indicating that when there was a difference in the transformation it was more likely to be an improvement than a detriment. The spread of the outliers increases as we move from consecutive poses to poses separated by two or three time steps.

Figure 10 shows the difference in ϵ_T between the original point clouds and those with points removed for all poses in the 07 dataset. Again, positive numbers indicate that transformation error after removing points was lower. The spike around time step 635 corresponds to a large truck driving directly in front of the ego vehicle, as seen in Figure 11.

Figure 12 compares the results of odometry performed on the 07 dataset with and without removing points corresponding to moving objects.

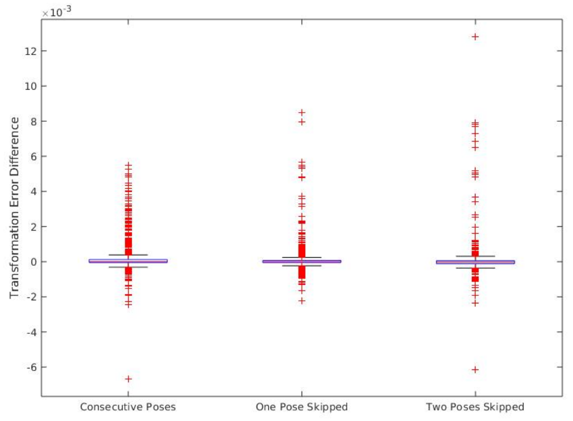


Fig. 9: Boxplot of difference in transformation error for original point clouds vs clouds with moving points removed

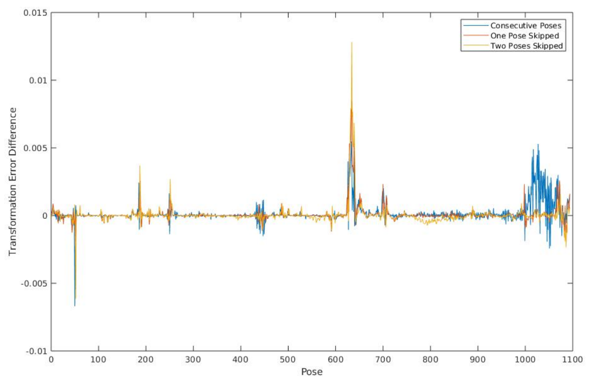
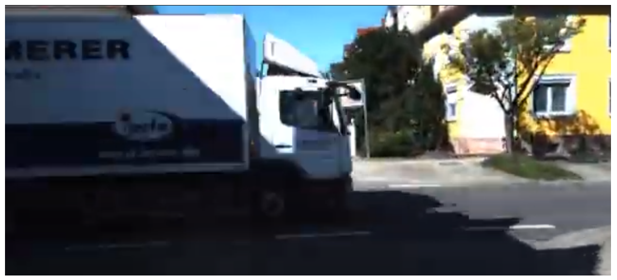


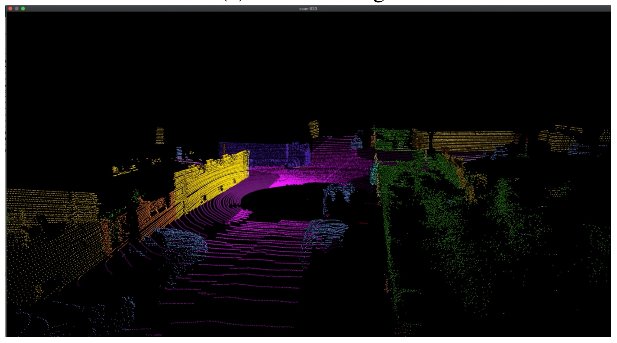
Fig. 10: Difference in transformation error for original point clouds vs clouds with moving points removed for all poses

C. SLAM with Simulated Loop Closures

The final test we did was to simulate loop closures to see how SLO would perform as part of a complete SLAM solution. To generate the loop closures, we calculated transformations between ground truth poses where the vehicle crossed or came very close to a previous pose on the trajectory. We added those as factors between poses in the factor graph that we fed into ISAM2 [7]. Figure 13 compares the results for the 07 dataset, and Figure 14 compares the results for the first 2,190 poses of the 00 dataset. We used this reduced set of poses for 00 because we chose to start and end the trajectory at loop closure points so the loop closures could remove error from the entire trajectory. Table III shows the trajectory errors for both datasets. For 07, the GICP mean $\epsilon_{t,\%}$ was lower than without loop closures, and the SICP mean $\epsilon_{t,\%}$ was about the same. For 00, the error was reduced for both SICP and GICP. Some of this may be attributable to solving over just a subset of the trajectory. For both, having a mean $\epsilon_{t,\%}$ value around or below 1% is promising.



(a) Camera image



(b) Semantically-labeled lidar point cloud

Fig. 11: Large truck passing in front of ego vehicle around time step 635

TABLE III: $\epsilon_{t,\%}$ for SICP and GICP

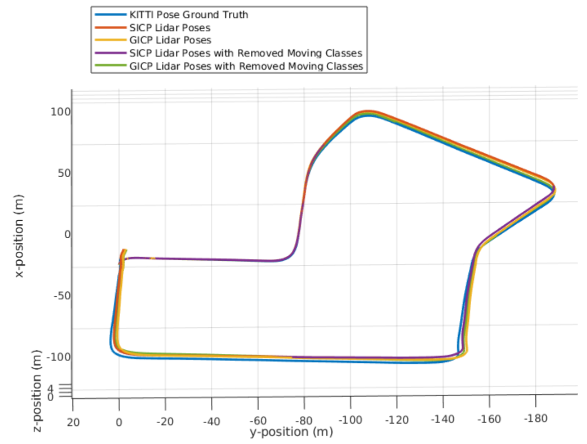
	SICP	GICP
mean $epsilon_{t,\%}$, 07	0.81	0.59
mean $epsilon_{t,\%}$, 00	1.06	1.25

V. Discussion and Conclusion

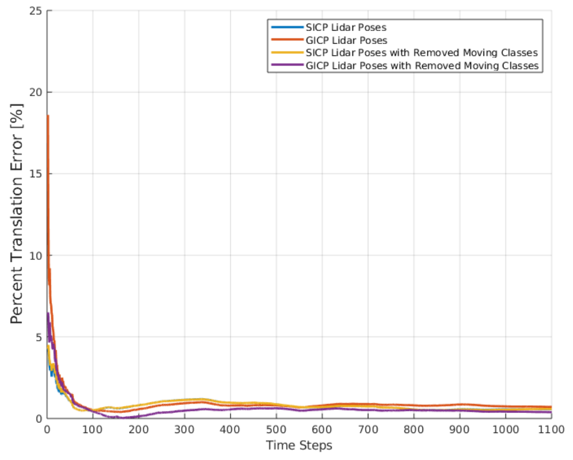
Generating the SICP and GICP transformations, we noticed GICP and SICP had comparable rotation and translation errors as seen in Figure 2. This surprised us due to the results presented in [6]. When we investigated further by looking at how the accuracy of SICP and GICP changed as the point clouds to be aligned were spaced further apart. Figure 6 shows that for the 00 dataset, SICP and GICP performed similarly when the poses were separated by 1 time step (0.1 seconds), but SICP clearly has lower error, especially translation error, when the poses are separated by 5 time steps (0.5 seconds). This could explain the discrepancy between the results we showed in Figure 2 and those presented in [6].

Because we found that our odometry results were made made worse by using additional transformations, we stuck mostly to adding only 1 or 2 transformations per pose to the factor graph. We expect that this is why we did not find a clear advantage for SICP in the simple odometry tests we ran, since SICP and GICP performed similarly for transformations between consecutive edges.

The unsmoothed and smoothed trajectories for 07 are visually close to the KITTI Ground Truth. Further investigating the Euclidean translation error as seen in Figure 5 and Table I, we can see that SICP does performs better than GICP in

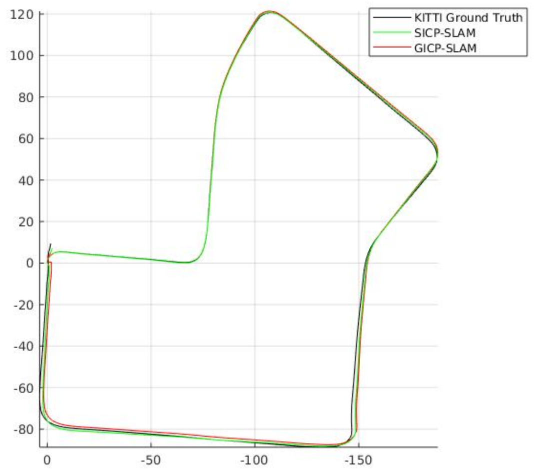


(a) Estimated trajectories vs KITTI ground truth trajectory

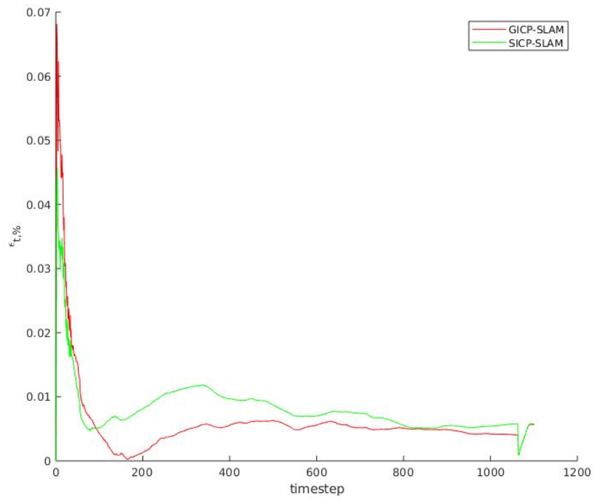


(b) Percent translation error comparison

Fig. 12: Odometry on 07 dataset with and without removing points corresponding to moving objects



(a) Estimated trajectories vs KITTI ground truth trajectory



(b) Percent translation error comparison

Fig. 13: SLAM on 07 dataset with simulated loop closures

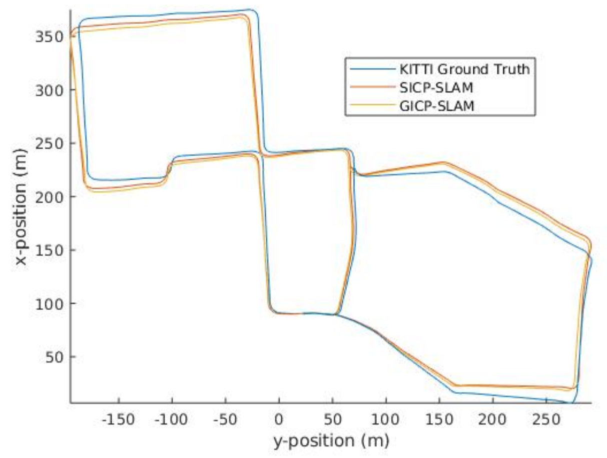
the unsmoothed case (0.81 vs 0.93 respectively) as well as in the smoothed case (0.89 vs 1.03 respectively).

Unexpectedly, we see the smoothed result performs slightly worse than the unsmoothed (0.89 vs 0.81 for SICP) and (1.03 vs 0.93 for GICP). This is likely due to the larger transformation error for both SICP and GICP when point clouds are separated by more time steps. We attempted many different ways of adjusting the covariance values for these edges when adding them to the factor graph, but did not find values that consistently led to better performance.

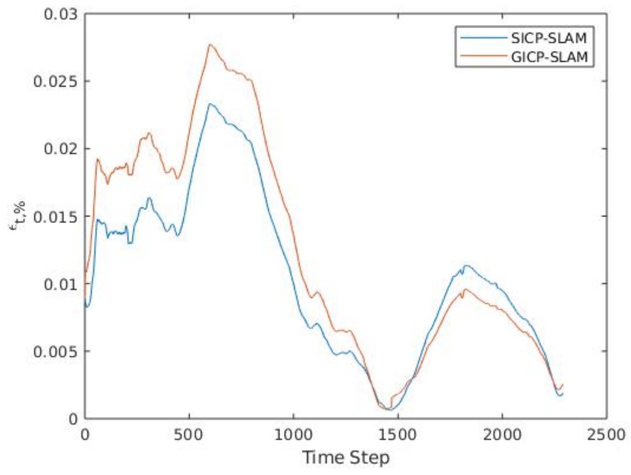
We also evaluate the trajectory for KITTI data set 00 and we see that in this case GICP performs slightly better than SICP (2.10 vs 2.28) although very close.

Comparing the smoothed and unsmoothed trajectory results between 00 and 07 data sets, we can not draw a conclusion on which of GICP and SICP perform better. Even in the cases were SICP performs slightly better (07 data set), it is hard to justify the added computation time introduced by SICP when one can use the much faster GICP algorithm to generate the transformations needed for odometry. However, in an application that needs point cloud registration of point clouds with higher initial separation, the extra computational cost of SICP might be justified, seeing as our results suggest the benefits of SICP over GICP are increased when there is a larger degree of misalignment in the point clouds to be registered. This could be the case for a self-driving car that is driving faster than the low speeds in most of the KITTI dataset.

Our next investigation was to apply loop closure to the smoothing. When adding loop closures, simulated between poses 200 times apart and with a Euclidean distance of less than 0.5, we saw improved results for both SICP and GICP compared to the results without loop closures. SICP error improved from 0.89 to 0.81 for the 07 trajectory and from 2.28 to 1.06 for the 00 trajectory. GICP error also improved from 1.03 to 0.59 for the 07 trajectory and from 2.10 to 1.25



(a) Estimated trajectories vs KITTI ground truth trajectory



(b) Percent translation error comparison

Fig. 14: SLAM on 00 dataset with simulated loop closures

for the 00 trajectory. These results show that lidar odometry, with or without the added information of semantic labels, can form an integral piece of a complete SLAM system.

Lastly, Motivated by SUMA++ [9] and their findings, we decided to remove point labels corresponding to moving objects. Looking at the 07 data set, which had more moving labels, the transformation error after removing the labels remained similar to our observations in the original data. This made sense as the majority of lidar scans in 07 contained few points corresponding to the classes removed. In Figure 10, the outliers were skewed positive, which indicates that removing the points was more likely to improve the transformation than to make it significantly worse.

Our findings indicate that SICP is not a clear winner over GICP for lidar odometry, but might have advantages in certain applications. We found that both GICP and SICP are feasible for generating transformations necessary for odometry and SLAM techniques to yield a confident trajectory which is further improved by identifying loop closures.

ACKNOWLEDGMENT

We would like to thank our professor, Maani Ghaffari Jadidi for his overall guidance on this project and Steven Parkison for his support in using his SICP software library correctly.

REFERENCES

[1] P. J. Besl, N. D. McKay, A method for registration of 3-D shapes, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239-256, Feb. 1992.

[2] Y. Chen, G. Medioni, Object modeling by registration of multiple range images, in Conf. Rec. 1991 IEEE Int. Conf. Robotics and Automation, pp. 2724-2729.

[3] A. Segal, D. Haehnel, S. Thrun, Generalized-icp, Robotics: science and systems, vol. 2, no. 4, pp. 435, Jun. 2009.

[4] A. Geiger, p. Lenz, R. Urtasun, Are we ready for autonomous driving? the KITTI vision benchmark suite, in Conf. Rec. 2012 IEEE Int. Conf. Computer Vision and Pattern Recog.

[5] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, J. Gall, SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences, in Proc. 2019 IEEE Int. Conf. on Computer Vision.

[6] S.A. Parkison, L. Gan, M.G. Jadidi, and R.M. Eustice. Semantic Iterative Closest Point through Expectation-Maximization. In Proc. of British Machine Vision Conference (BMVC), page 280, 2018.

[7] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. The International Journal of Robotics Research (IJRR), 31:217–236, 2012.

[8] A. Zaganidis, L. Sun, T. Duckett, and G. Cielniak, "Integrating Deep Semantic Segmentation Into 3-D Point Cloud Registration," IEEE Robot. Autom. Lett., vol. 3, no. 4, pp. 2942–2949, Oct. 2018.

[9] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "SuMa++: Efficient LiDAR-based Semantic SLAM," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.