A Comprehensive Evaluation of LiDAR Odometry Techniques

Easton R. Potokar and Michael Kaess

***Abstract*— Light Detection and Ranging (LiDAR) sensors have become the sensor of choice for many robotic state estimation tasks. Because of this, in recent years there has been significant work done to find the most accurate method to perform state estimation using these sensors. In each of these prior works, an explosion of possible technique combinations has occurred, with each work comparing LiDAR Odometry (LO) "pipelines" to prior "pipelines". Unfortunately, little work up to this point has performed the significant amount of ablation studies comparing the various building-blocks of a LO pipeline. In this work, we summarize the various techniques that go into defining a LO pipeline and empirically evaluate these LO components on an expansive number of datasets across environments, LiDAR types, and vehicle motions. Finally, we make empirically-backed recommendations for the design of future LO pipelines to provide the most accurate and reliable performance.**

I. Introduction

Over the last few decades, Light Detection and Ranging (LiDAR) sensors have become the sensor of choice for many robot state estimation and mapping tasks due to their high accuracy, long range capabilities, and complementary nature to other sensors. They have found applications in many platforms from ground vehicles, quadrupeds, and even aerial vehicles, with significant improvement in accuracy and fidelity of the resulting pose estimates and maps.

With this popularity has come a significant increase in LiDAR Odometry (LO) techniques, ranging from methods in feature detection [1], initialization [2], point cloud dewarping [3], map aggregation, optimization residual definitions [4], and many more. With this large number of potential techniques to choose from comes a combinatorial number of potential LO "pipelines" that can be defined, each with their own strengths and weaknesses.

The current state-of-the-art work usually implements a number of novel techniques along with choosing methods from the before-mentioned list to define a novel LO pipeline. This new pipeline is then generally benchmarked against *other pipelines* on a relatively small subset of data to demonstrate an improvement in accuracy, compute efficiency, mapping capabilities, etc. However, there is limited ablation work being done on many of the base choices that underlie these complex state estimation pipelines.

We empirically perform these core-level ablation studies to shed light on what the effects of many of these LO components are and provide some recommendations on what techniques should be chosen as a foundation for future LO pipelines. Namely, our contributions are:

- We provide an introduction to common LO techniques to guide a more general audience and provide context and motivation for our later experiments.
- We empirically evaluate a number of core LO techniques *individually* across a *large* set of trajectories, environments, and LiDAR sensors. This is in contrast to the current prevalent approach of comparing entire LO pipelines as a unit on a smaller sample of data.
- Based on these results, we supply recommendations on the LO components that will provide the best odometry performance for various different scenarios, LiDAR sensors, etc. While empirically-backed and intuitive, some of these results run contrary to many current preferred choices for LO pipelines.

Finally, we release our python-C++ codebase as open-source[1] to allow for repeatability and future ablations.

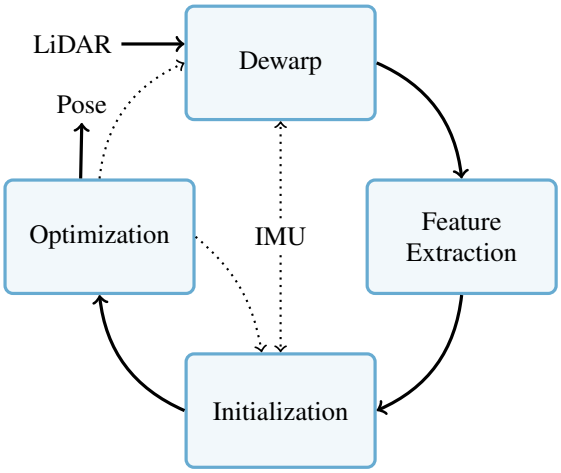


Fig. 1: The basic building blocks that make up a LiDAR Odometry (LO) pipeline. Dotted lines represent optional connections. Few ablation studies have been done comparing these core-level components of LiDAR Odometry (LO) pipelines.

II. Related Works

Due to the popularity of LiDAR sensors, numerous LiDAR-based techniques for robot pose estimation have been developed with a combinatorial number of core-level building blocks. There are a number of surveys and reviews that cover these building blocks, but none perform significant ablation studies. Some reviews are analyses of the various pipelines, but they lack any form of empirical results [5],

Easton R. Potokar and Michael Kaess are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA USA {potokar, kaess}@cmu.edu

This material is based upon work supported by the U.S. Army Research Office and the U.S. Army Futures Command under Contract No. W911NF20-D-0002. The content of the information does not reflect the position or the policy of the government and no official endorsement should be inferred.

[1] Available at https://github.com/rpl-cmu/lidar-evaluations/

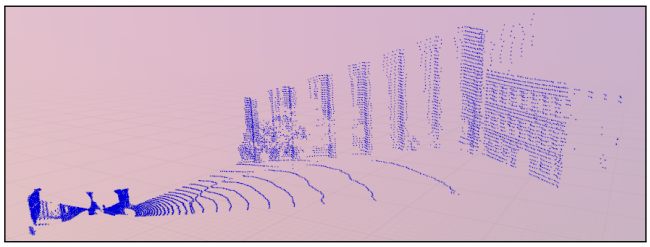


Fig. 2: An example LiDAR scan from the Newer College Stereo-Cam dataset. Note the obvious scanline curves on the ground.

[6]. Others are focused on general Iterative Closest Point (ICP) rather than LiDARs [7] which removes many of the important building blocks such as dewarping and initialization. Finally, some do perform valid empirical experiments, but only compare the LO pipelines directly from prior work without any form of ablation studies [8], [9].

The initial works presenting these LiDAR pipelines also lack clear ablation studies on these components despite the significant amount of diversity in choices such as feature definition and usage, dewarping, initialization, and map aggregation, amongst others.

For example, some methods perform scan matching via vanilla point-to-point ICP [10]. Other work attempts to model these surfaces as edges [1] and planar points [3] to perform feature-based ICP. Others define additional feature classes using PCA [11]. Despite this large diversity in options, and while benchmarks were done comparing pipelines to pipelines, none of these odometry methods or techniques perform comprehensive ablations against the various combinations of potential feature types.

Additionally, there is a large variance in how to define planes and edges in a LiDAR scan. Some use heuristics by summing vectors [1] along scan lines while others use a more expensive eigenvalue threshold to define planes [12], [13]. These are a smaller part of the prior works, but still no experiments investigating the impact of these choices have been done.

Removing motion distortion in a LiDAR scan is also missing empirical comparisons, with common options being a constant velocity model for LiDAR-only methods [10], IMU integration for LiDAR-Inertial Odometry (LIO) methods [2], or using a continuous-time optimization scheme for even higher accuracy [3], [12]. Again, while the original authors may have performed ablation studies, little of those experiments have made it into the final published work.

Initialization is also an important part of any ICP method, but limited studies have been done on its impact. Most LO methods use a constant velocity model [1], [10] while LIO methods integrate IMU integration to generate an initial estimate for ICP [2], [13]. While IMU integration is assumed to be more accurate, little published work has been done to confirm this, as far as the authors know.

While most of these choices are smaller parts of a larger LO pipeline, their joint impact can be significant on the final LO pipeline. Due to the lack of a consensus on which methods generally perform the best, we clearly expound on the differences between all these techniques and perform a comprehensive set of empirical studies to shed light on what methods generally result in the highest quality LO.

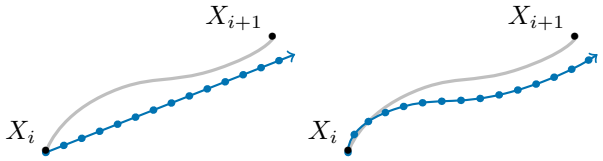


Fig. 3: Approximating poses between X_i and X_{i+1} for dewarping. A constant velocity model is shown on the left, with each dot representing a X_j^i. IMU dewarping, which more fully captures the motion, is shown on the right.

III. LiDAR Odometry Techniques

Most LO methods rely upon some variant of ICP for pose estimation between received LiDAR scans. This can be done via scan-to-scan matching or some form of scan-to-map matching, where a map is an aggregate of multiple scans often down-sampled using a voxel grid or kd-tree.

In order to make each execution of ICP perform independently of previous executions, and to reduce the number of techniques to search over, we focus exclusively on scan-to-scan matching and its related components, and note that the techniques and conclusions found herein should also extend to, or at least guide development of, scan-to-map matching.

There is a significant number of options when implementing a LO pipeline and even more for a LIO pipeline, a large subset of which apply to scan-to-scan ICP. We briefly cover a number of these, both as a precursor to our later experiments and as an introduction for new readers. We also note that while the majority of these techniques are specifically for rotating LiDARs, a number of conclusions and techniques also apply to solid-state LiDAR sensors as well.

A. Dewarping

Spinning LiDAR sensors operate by continuously rotating while collecting range measurements along a roughly vertical column. Each element of the vertical column produces a generally horizontal "scanline" upon rotation completion, as can be seen in Fig. 2. At the end of a full rotation, they return all the scanlines, collected into a single "scan".

During this rotation, the sensor may undergo motion that can cause a distortion, also called a warp or skew, to be present in the final scans. A common technique is to estimate the motion of the vehicle during the scan and use the motion estimate to "dewarp" or "deskew" the LiDAR scan.

Assuming a LiDAR scan was stamped at time i, and a point p in the scan was taken at time j and has points represented in the LiDAR frame, the deskewed point $\tilde{p}$ can be had by using the transformation $X_j^i \in \mathrm{SE}(3)$ that maps from the LiDAR pose at i to the LiDAR pose at j,

$$\tilde{p} = X_j^i p. \tag{1}$$

The following techniques for estimating X_j^i are shown geometrically in Fig. 3. Estimating X_j^i is often done by assuming a constant velocity computed using the previous change in pose over change in time and integrating this constant velocity to produce the necessary poses for dewarping.

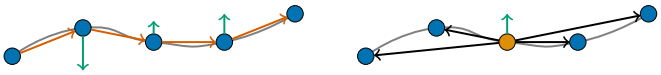


Fig. 4: Discrete curvature of a 3D parametrized curve shown in gray, of which a LiDAR scanline is a discrete sampling. On the left, the approximated tangents $\hat{T}_i$ are orange and curvature-scaled normals $\hat{\kappa}_i\hat{N}_i$ green. On the right, summing the black vectors formed from tangent approximations over a window to estimate $\hat{\kappa}_i\hat{N}_i$. When $\|\hat{\kappa}_i\hat{N}_i\|$ is small, the point is classified as planar and when it is large an edge.

Another option for estimating X_j^i is to integrate the high rate IMU measurements into $\mathrm{SE}(3)$ poses and interpolate between their timestamps. This requires some form of body velocity and IMU bias estimation method to perform the IMU integration, often done in a LIO pipeline [1].

Beyond the scope of this work, another technique is to perform continuous-time pose estimation [12], [3], which then provides the necessary poses continuously for dewarping.

B. Initialization

ICP is known to be extremely sensitive to initialization [14]. The two most common initialization schemes are constant velocity and IMU integration.

Constant velocity initialization uses the previous optimized pose delta over the change in time to compute an estimated velocity, and forward propagates this velocity to predict the next pose. If scans come in at a constant rate, this results in using the optimized pose delta at the previous timestep as initialization for the following timestep. Beyond the scope of this work, more recent work utilizes leasts-squares and a window of past poses to solve for the constant velocity [**?**]

IMU integration is generally done in a LIO pipeline, where body velocity and IMU biases are also being estimated. Then, using the estimated body velocity and biases, IMU measurements can be integrated from the previous timestep to the current timestep to provide an initial estimate.

C. Selecting Features

LiDAR features are the method with the highest variance of options used that we will cover. This choice will heavily influence optimization and results, as will be seen in Section IV. It should be noted, before entering the LO pipeline, preprocessing must be done to remove points that are invalid. These include points below the minimum range, above the maximum range, on surfaces parallel to the point ray direction, and when object discontinuity may occur [1].

A LiDAR scan is essentially a scan of the 3D surfaces in the environment. Basic LiDAR features attempt to identify the kind of geometry that a point lies on, generally either an edge or a surface. This is usually done by locally estimating some form of the curvature at each point, which defines how curved the geometry is at a given point. Minimally curved geometries correspond to planar points, while maximally curved geometries can correspond to edges. Again, there are a variety of ways to estimate this curvature.

1) Classical Curvature: One popular technique popularized by LOAM [1] is to treat each scanline as a discrete sampling from a 3D parametrized curve, also known as a space curve. A simple differencing scheme can then be used

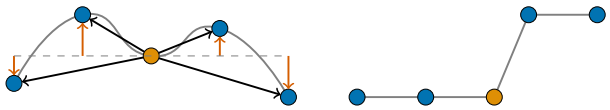


Fig. 5: Various failures of curvature computation. On the left, the vectors horizontally and vertically sum to 0, thus $\hat{\kappa} = 0$ despite it not being a plane. On the right, eigen-based approaches fail to detect the orange point as an edge. Nearest-neighbor eigenvalue will likely miss the far wall as part of the neighborhood, and scanline-based eigenvalue will have two small eigenvalues if the gap is large enough and will detect it as planar.

to approximate the curvature $\kappa \in \mathbb{R}$ of the space curve. Given a space curve p parametrized by arc length s,

$$\begin{aligned} p''(s) = T'(s) &= \kappa(s)N(s) \\ \implies \kappa(s) &= \|p''(s)\|, \end{aligned} \tag{2}$$

where T is the 3D unit tangent vector and N is the 3D unit normal. While there is no exact discrete analog to κ [15], a rough estimate can be done using finite difference methods. With discrete points $p(i) = p_i$, we can define estimates to the space curve derivatives $\hat{p}_i'$ and the curvature $\hat{\kappa}_i$,

$$\begin{aligned} \hat{p}_i' &= p_i - p_{i-1} \\ \hat{p}_i'' &= \hat{p}_{i+1}' - \hat{p}_i' \\ &= p_{i+1} - 2p_i + p_{i-1} \\ \implies \hat{\kappa}_i &= \|p_{i+1} - 2p_i + p_{i-1}\|. \end{aligned} \tag{3}$$

These vector estimates can be seen on the left of Fig. 4. In practice, this is done over a window of points in a scanline to add robustness, which ends up looking like

$$\begin{aligned} \hat{\kappa}_i &= \Big\| \frac{1}{n} \sum_{0<j\le n} (p_{i+j} - 2p_i + p_{i-j}) \Big\| \\ &= \Big\| \frac{1}{n} \sum_{-n\le j\le n} (p_{i+j} - p_i) \Big\|. \end{aligned} \tag{4}$$

This is visualized in the right of Fig. 4 where each of the difference vectors can be seen to be summing up to compute $\kappa_i N_i$. Small values of κ_i then correspond to planar points, while large values correspond to edge points.

There are some assumptions here that are not necessarily always valid, namely that the curve is parameterized by arc length, and that our discrete curvature necessarily converges to the continuous curvature as the number of points increases. However, this is a well-tested heuristic and empirically has been shown to function well.

2) Scanline Eigenvalue: An alternative method is to simply compute the covariance of points around a given neighborhood on a scanline,

$$\Sigma_i = \frac{1}{2n} \sum_{-n\le j\le n} (p_{i+j} - p_i)(p_{i+j} - p_i)^\top. \tag{5}$$

The eigenvalues of Σ_i will correspond to the main directions the points are spread in. There will always be one small eigenvalue along the scanline, thus we can differentiate planar and edge points with low and high values of the second eigenvalue, respectively.

Dataset	Color	Year	Setting	Device	Lidar	Beams	IMU	Ground Truth
Newer Stereo-Cam [16]		2020	Outdoor Campus	Handheld	Ouster OS-1	64	TDK ICM-20948	Laser Scanner & ICP
Newer Multi-Cam [17]		2022	Outdoor Campus	Handheld	Ouster OS-0	128	TDK ICM-20948	Laser Scanner & ICP
Hilti 2022 [18]		2022	Indoor Campus	Handheld	Hesai PandarXT	32	Bosch BMI085	Laser Scanner & ICP
Oxford Spires [19]		2024	Campus	Backpack	Hesai QT64	64	Bosch BMI085	Laser Scanner & ICP
Multi-Campus [20]		2024	Outdoor Campus	ATV & Handheld	Ouster OS-1	128	VectorNav VN-200	Laser Scanner & Cont-Opt.
HeLiPR [21]		2024	Urban Road	Car	Ouster OS-2	128	Xsens MTi-300	INS Fusion
Botanic Garden [22]		2024	Trail Road	ATV	Velodyne VLP16	16	Xsens Mti-680G	Laser Scanner & ICP

TABLE I: Overview of datasets used in evaluation. The majority of sequences from each dataset are included, with the exception of some with motion beyond what is common in a robot trajectory. The color is the color the dataset will use throughout in figures.

3) Nearest Neighbor Eigenvalue: This is nearly identical to the prior approach, except rather than using the neighborhood defined by the scanlines, nearest neighbors are found throughout the entire scan. Since there will no longer be a naturally small eigenvalue along the scanline, thresholding can be done using the first eigenvalue, where planar points will have small values once again. Care must be taken to ensure points are on separate scanlines, as points on the same scanline are nearly collinear and will always have a small first eigenvalue.

All of these techniques have various drawbacks. The classical curvature approach can falsely identify a plane if points happened to be symmetrical, as seen in the left of Fig. 5. The eigenvalue-based approaches are naturally more computationally heavy and fail to identify edge points due to discontinuities as shown on the right of Fig. 5. The nearest neighbor eigenvalue approach should be the most robust due to utilizing points across scanlines to check for planarity, but it does require more parameter tuning across LiDAR beam sizes to ensure neighbors are found on adjacent scanlines and may not function with LiDARs with fewer beams.

Additionally, some LO methods do not bother with features and treat all the points in a LiDAR scan as points. This results in fewer heuristics for computing features and potentially more points to guide the optimization, but it is more sensitive to initialization, as will be seen in Section IV.

D. Optimization

Once features are selected and then matched, the relative pose $X \in \mathrm{SE}(3)$ that transforms the source scan to the target scan is then optimized for in a nonlinear least squares optimizer with objective,

$$X^\star = \mathrm{argmin}_X \sum_k r_k(X)^\top \Lambda_k(X) r_k(X) \tag{6}$$

where each k is a pair of matched features, $\Lambda_k(X)$ is an optional symmetric weighting matrix, and r_k the residual. Λ_k and r_k vary based on the type of feature chosen, with definitions as follows. Moving forward, we drop the k subscript for conciseness.

1) Point Features: Given point p_i in the target scan and p_j in the source scan, point-to-point residual is defined as

$$r(X) = p_i - X p_j \tag{7}$$
$$\Lambda(X) = I. \tag{8}$$

2) Edge Features: Edge features are optimized by minimizing the portion of the residual that is orthogonal to the estimated edge direction given by d_i. This can easily be done using the annihilator $A = I - d_i d_i^\top$ that removes all vector components that are parallel to d_i. Since A is a projection matrix $A = A^2$, we have,

$$r(X) = p_i - X p_j \tag{9}$$
$$\Lambda(X) = A = I - d_i d_i^\top. \tag{10}$$

3) Planar Features: The planar residual has a number of variants [4]. The basic form of point-to-plane can be thought of as the distance of p_j from the plane given by p_i and its estimated normal n_i. Similar to edge features, this just requires a custom weighting matrix,

$$\begin{aligned} r(X)^\top \Lambda(X) r(X) &= (n_i^\top r)^2 = (r^\top n_i)(n_i^\top r) \\ &= r^\top (n_i n_i^\top) r \\ \implies r(X) &= p_i - X p_j \qquad (11) \\ \Lambda(X) &= n_i n_i^\top \qquad (12) \end{aligned}$$

where $n_i n_i^\top$ is the projection matrix that projects onto n_i. Normal estimation generally occurs by fitting a plane to a subset of points nearby p_i, taking care that points from separate scanlines are used, as points on a single scanline are collinear and induce a degeneracy.

Similarly, a "pseudo-point-to-plane" residual can be used to constrain the two orthogonal directions to n_i as well. The annihilator $I - n_i n_i^\top$ can be used for the projection to these extra dimensions,

$$\begin{aligned} \Lambda(X) &= n_i n_i^\top + \epsilon_i (I - n_i n_i^\top) \\ &= (1-\epsilon) n_i n_i^\top + \epsilon I \end{aligned} \tag{13}$$

for $\epsilon \in [0, 1]$. Notice that for $\epsilon = 0$, the standard point-to-plane equation is recovered, while for $\epsilon = 1$, point-to-point is recovered.

Additionally, a plane-to-plane residual can also be used [4]. It essentially functions as a second residual constraining the target point to source plane given by n_j. Standard plane-to-plane then uses the weighting matrix,

$$\Lambda(X) = n_i n_i^\top + R n_j n_j^\top R^\top \tag{14}$$

where R is the rotation in X. A "pseudo-plane-to-plane", popularized by Generalized-ICP [4] can then be defined

$$\begin{aligned} \Lambda(X) = n_i n_i^\top &+ \epsilon (I - n_i n_i^\top) \\ &+ R n_j n_j^\top R^\top + \epsilon (I - R n_j n_j^\top R^\top) \\ = (1-\epsilon)(n_i n_i^\top &+ R n_j n_j^\top R^\top) + 2\epsilon I. \end{aligned} \tag{15}$$

As one can see, there is a combinatorial number of options of feature combinations that can be used, with additional ϵ parameters to choose from as well. We explore these, and all other given options, in the following sections.

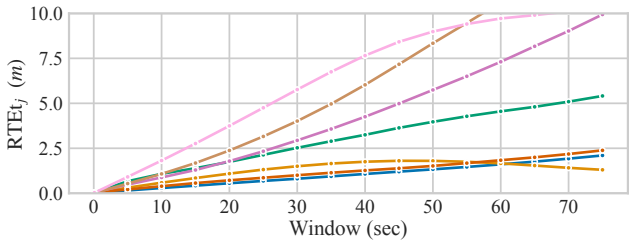


Fig. 6: Results of basic scan-to-scan LiDAR odometry on a number of sequences from across datasets. As the window size increases, RTE_j generally increases linearly until trajectory effects such as loops begin to have an impact. Dataset colors are as in Table I.

IV. Experimental Setup

We seek to explore the effects of each individual building block independently, and thus attempt to remove as many independent variables as possible in each experiment. Hence, we focus on scan-to-scan matching, noting that the results should transfer to the more accurate scan-to-map matching.

To this end, we also remove the impact of poor body velocity and bias estimation. To do this, we compute bias and body velocity estimates offline for the full trajectory in a factor graph with IMU preintegration [23] factors and ground truth poses as prior factors. This gives us a consistent and usable bias and velocity estimates for usage in IMU initialization and dewarping.

Also to remove other independent variables, we use ground truth data to compute velocities any time a constant velocity assumption is used for initialization or dewarping. We also use ground truth initialization in some experiments to remove the impact of poor convergence due to an inaccurate initialization. Further, we evaluate on a large number of datasets and choose a final metric that represents the goal of LO well, namely minimizing drift.

A. Datasets

Our list of chosen datasets can be seen in Table I. To ensure that the results generalize well, we prioritized ground truth accuracy as well as maximizing the variety of environments, LiDAR brands and LiDAR beam-size, and motion based on type of vehicle. All data was loaded using our open-source data-loading library evalio[2]

Trajectories range anywhere from around 1 min to 25 min long. Due to the quantity of experiments done, we have run exclusively on the first 5 min of the longer trajectories, or about 3,000 LiDAR scans. We have found this to be a sufficient sample size for drawing conclusions about the various LiDAR approaches we evaluate.

B. Metric

There are a number of metrics that are commonly used to evaluate odometry methods, the choice of which can have a significant impact on the final results. One common choice is Absolute Trajectory Error (ATE), which roughly estimates the average global error of the entire trajectory. Given a ground truth pose at timestamp i X_i and an estimate given by $\hat{X}_i$, the translational ATE is given by,

[2]Available at https://github.com/contagon/evalio/

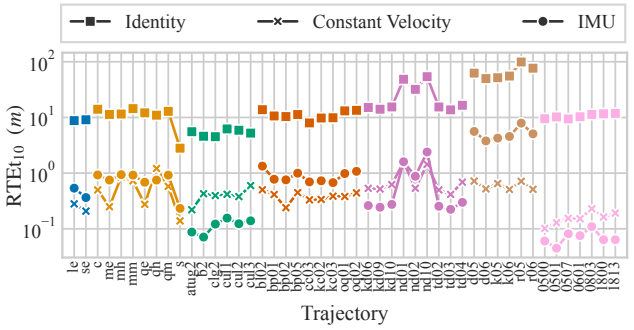


Fig. 7: Initialization accuracy with no optimization performed. Both constant velocity and IMU initialization are computed using ground truth poses and optimized velocities and biases from our offline factor graph. In this idealized scenario, constant velocity is generally more accurate for smooth motion, IMU initialization for more aggressive motion. Dataset colors are as in Table I.

$$\text{ATEt}(X, \hat{X}) = \left(\frac{1}{N} \sum_{i \in N} \left\| \text{trans}(X_i \ominus \hat{X}_i) \right\|^2 \right)^{1/2}. \quad (16)$$

While very intuitive, ATE has significant issues. One issue is that a small rotational error at the start of a trajectory will have a significantly larger impact than one at the end of a trajectory [24]. Further, when dealing with trajectories that revisit locations, a trajectory may have a low ATE by happenstance if it returns close to the original location regardless of the rest of the trajectory. This makes ATE better suited for evaluating results globally and is a poor choice for evaluating drift on local odometry methods.

A better option for measuring odometry drift is Relative Trajectory Error (RTE). Representing a change in pose from timestep i to $i+1$ by $\delta_{i,i+1}$, translational RTE is given by [24]

$$\text{RTEt}(\delta, \hat{\delta}) = \left(\frac{1}{N} \sum_{i=0}^{N} \left\| \text{trans}(\delta_{i,i+1} \ominus \hat{\delta}_{i,i+1}) \right\|^2 \right)^{1/2}. \quad (17)$$

However, most of the datasets that we have used have centimeter level ground truth precision which is near the precision of scan-to-scan odometry, and can result in noisy final metrics.

Instead, since most of the ground truth poses were gathered independently rather than sequentially, we utilize windowed Relative Trajectory Error (RTE$_j$) [**?**]. RTE$_j$ increases the drift error quantities to be larger than the error found in the ground truth, but still avoids the issues that can occur with ATE. The translational version is defined as

$$\text{RTEt}_j(\delta, \hat{\delta}) = \left(\frac{1}{N-j} \sum_{i=0}^{N-j} \left\| \text{trans}(\delta_{i,i+j} \ominus \hat{\delta}_{i,i+j}) \right\|^2 \right)^{1/2} \quad (18)$$

for some $j > 0$. Generally, one would expect the value of RTE$_j$ to increase relatively linearly as j increases. However, if j is large enough, decreases may appear due to tight curves occurring. An example of this can be seen in Fig. 6 for the Newer College Multi-Cam ■ and Botanic Gardens ■ trajectories, both of which contain a trajectory loop.

We select a window time of 10 s. This ensures that odometric drift is typically above that of the ground truth accuracy,

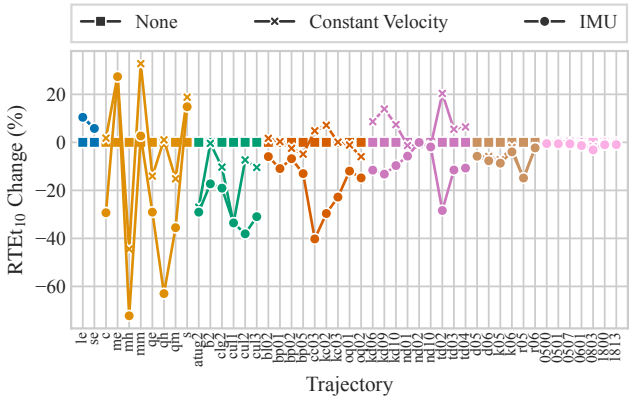


Fig. 8: Dewarping experiment showing percent change in RTE_j over no dewarping performed. Constant velocity and IMU dewarping are computed using ground truth values. Expectedly due to its high sensor rate, IMU dewarping generally improves performance the most. Dataset colors are as in Table I.

but still avoids large trajectory effects. Additionally, we only show the translational RTE_j moving forward for conciseness since the rotational RTE_j follows similar trends.

As the scale of RTE_j can vary from dataset to dataset due to factors such as environment, LiDAR beam size, and sensor motion, many figures report the percent change C in RTE_j to normalize for scale. Thus a change in RTE_j a over a base case RTE_j b is given by,

$$C = 100\left(\frac{a-b}{b}\right). \tag{19}$$

V. Experiments

In the following experiments, we draw conclusions about what feature types should be used, how they should be extracted, what dewarping method provides the best results, and how good of an initialization is needed for convergence to the ground truth optimum. Unless otherwise stated, the base techniques used for each experiment are no dewarping, ground truth initialization, and planar and edge features with the vanilla point-to-plane residual.

A. Initialization

To test initialization accuracy, we compute the RTE_j for each initialization scheme with no optimization, including using the identity, constant velocity, and IMU integration. For the constant velocity, we use ground truth poses from the prior timestep to compute, and the IMU integration integrates on top of a ground truth poses with velocity and biases from our offline factor graph optimization. These can be seen as the ideal scenario that will be achieved if the LO or LIO pipeline is functioning properly. Results can be seen in Fig. 7.

There are a few trends to note in this figure. First, the motion of the wheeled vehicles in the HeLiPR ■ and part of the Multi-Campus ■ datasets are likely notably smoother across timesteps. Due to this, constant velocity is particularly accurate. In contrast, the Newer College Multi-Cam ■ has some "medium" or "hard" trajectories that contain aggressive dynamic behavior. In these sequences, constant velocity was naturally worse, with the IMU integration not being impacted significantly by the aggressive motion.

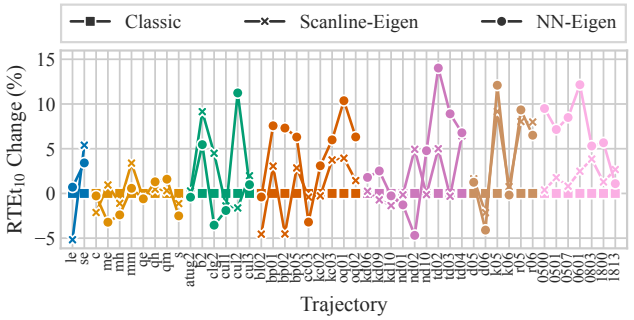


Fig. 9: Curvature computation comparison between the classical scanline curvature popularized by LOAM [1], scanline-eigenvalue, and nearest-neighbor eigenvalue methods, reported in percent change in RTE_j over classical curvature. All performed very similarly, with the classic method being less computationally expensive. Dataset colors are as in Table I.

While controlled for in this experiment, in real-world deployments it can be argued that there is more that could potentially go wrong with IMU integration particularly with a low-grade IMU, namely poor bias or velocity estimation, time-sync, etc. These can impact results, whereas the constant velocity model should remain functional as long as odometry is functioning properly. Thus in cases of smooth motion or a low-quality IMU, we recommend using constant velocity, and in other scenarios IMU initialization.

However, as will be seen in later sections, features can be chosen that are less sensitive to initialization and this choice will have a small impact. To reduce the number of changing variables, most of the following experiments are done with ground truth initialization; sensitivity to initialization will follow in its own experiment.

B. Dewarp

We compare dewarping strategies similarly, with ground truth IMU and constant velocity values, planar and edge features, and ground truth initialization. Experiments were ran with stamps at the start of scans, which can have a minor impact on constant velocity dewarping. Results can be seen in Fig. 8. We report the percent change in RTE_j between the dewarping methods and no dewarping.

Generally, we found that some form of dewarping was worthwhile, with IMU dewarping performing the best in the majority of cases. This is unsurprising due to the IMU measurements capturing higher-rate motion between LiDAR scans. Even the datasets with low-quality IMUs have a small performance improvement with IMU dewarping. If IMU fusion is being performed, IMU dewarping should definitely be used. If not and computational expense is a constraint, one should considering using no dewarping since the gains for using constant velocity dewarping are more minimal.

C. Curvature

Curvature computations methods, namely classical scanline based, scanline-eigenvalue, and nearest neighbor eigenvalue methods, are also compared across the trajectory set. We tune the various thresholds for each to emit roughly the same number of features and run experiments with only planar features. The exception is in datasets with small beam

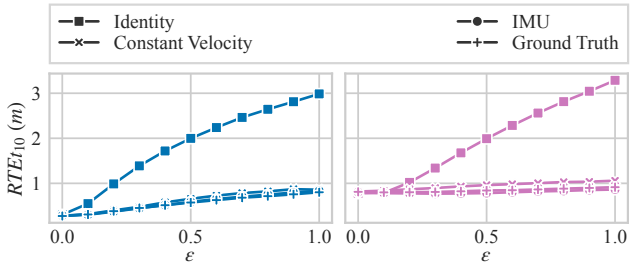


Fig. 10: Comparison of ϵ parameter in pseudo-point-to-plane optimization across varying initialization strategies. Introduction of any ϵ to constrain the orthogonal directions to the normal degrades odometry accuracy. Additionally, point-to-plane optimization is highly insensitive to initialization and the opposite holds for point-to-point optimization. Dataset colors are as in Table I.

sizes, where the nearest-neighbor eigenvalue approach failed to gather as many features due to the large gap across scanlines. Results can be seen in Fig. 9 in percent change from classic curvature.

While they did not identify identical features, we found that they generally produced very similar results. For those using rotating LiDARs, we recommend the classical scanline based approach due to its lower computational overhead and ability to also detect edge features if desired.

D. Pseudo-Features

Additionally, we investigate the impact that pseudo-planar features may have on optimization, with results on a pair of trajectories seen in Fig. 10. More trajectories were tested, but the majority exhibited similar trends and are not included for figure clarity. LO was run with only planar features and with varying ϵ values. The main takeaway from this experiment is that the introduction of constraints on the orthogonal directions to the normal is detrimental to odometry accuracy.

Of note, at ϵ=0 when point-to-plane is recovered, initialization seemed to have a minimal impact, implying a much larger basin of attraction with all initializations converging to the same optimum. At $\epsilon = 1$ when point-to-point is recovered, the results are highly sensitive to initialization. This is a theme which will be repeated in Section V-F.

E. Plane-To-Plane

We also empirically test plane-to-plane features against the standard point-to-plane counterpart. Only planar features are used along with ground truth initialization. We present the results as the percent change over standard point-to-plane in RTE_j in Fig. 11.

This is our result with the most obvious conclusion; when normals can be computed for both target and source point clouds, plane-to-plane residuals will give the best performance, mimicking the results of Generalized-ICP [4]. This intuitively makes sense; it essentially introduces additional constraints and allows for the source cloud normals to help guide the optimization in the weighting matrix Λ.

F. Features

Finally, we also test the impact that features may have on the final results. We specifically test the combination of point features, planar features, and planar and edge features

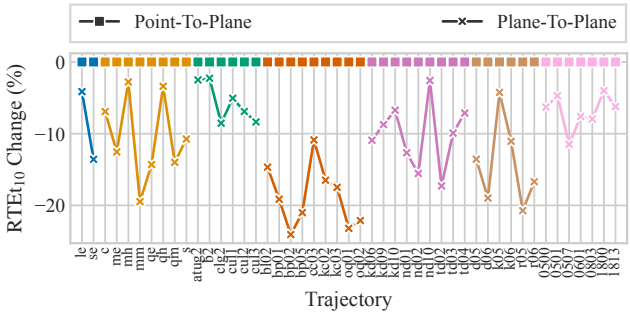


Fig. 11: Comparison of point-to-plane and plane-to-plane planar residuals reported in percent change over point-to-plane. Plane-to-plane always outperformed point-to-plane methods. Dataset colors are as in Table I.

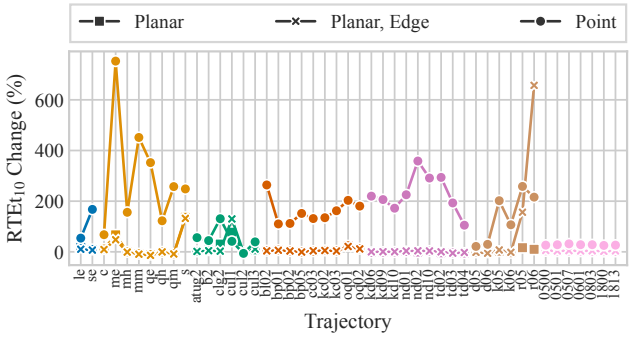


Fig. 12: Feature comparison. Results are the percent change from constant velocity to identity initialization. Higher values, like those of the point features, thus demonstrate a high sensitivity to initialization as they improved significantly with better initialization, whereas planar and planar and edge features showed little change. Dataset colors are as in Table I.

as these are the most common feature combinations used in prior works. Point features were chosen at the exact locations of edge and planar features to allow for similar quantities to be compared. In practice, there may be significantly more point features. However, this impact is likely to decrease if scan-to-map ICP is being used and is left for future work.

The first experiment was run to test the impact of initialization on the various forms of features. Fig. 12 shows the relative change from initialization using a constant velocity model to initializing with the identity. Feature types that are sensitive to initialization will result in a increase in RTE_j. It can clearly be seen if something goes poorly during odometry, point features are less likely to recover from a poor initialization than planar features or planar features with edge features. We also note that point features required more iterations to converge than did planar features, leading to a significant runtime increase.

We additionally test the performance of the various features outright initialized with a constant velocity model. Results are shown in Fig. 13. In most scenarios, edge features seemed to improve performance rather minimally and in some cases even degraded performance. The counterexample to this is in the HeLiPR ■ and Botanic Garden ■ trajectories, where the environments are extremely unstructured and lack any planar features other than the ground. In these scenarios, edge features detected along the edges of trees helped significantly by adding further constraints. In any structured

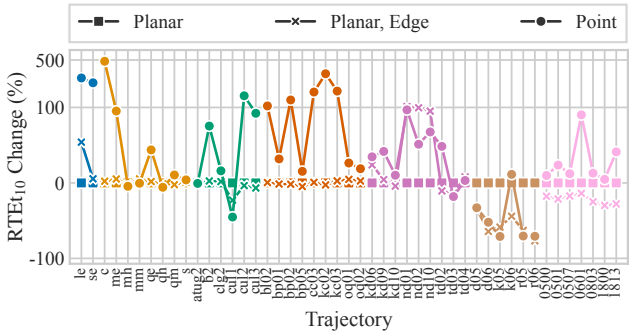


Fig. 13: Feature comparison showing relative change to using solely planar features, all initialized with constant velocity. Scale is linear from −100 to 100, and log outside that. In most scenarios, edge features had limited positive impact on results, with the exception being in the unstructured HeLiPR ■ and Botanic Garden ■ datasets. Dataset colors are as in Table I.

environment, we thus recommend using exclusively planar features; in more unstructured environments, using edge or point features as well is advisable to aid the optimization.

VI. Conclusion

In this work, we have empirically evaluated many of the common core techniques used in LO in current state-of-the-art methods across a large sample of data. Specifically, we have covered ICP initialization, LiDAR scan dewarping, curvature computation methods, various feature optimization residuals, and combinations of common features.

We have shown that planar features with a plane-to-plane residual perform the best in structured environments, with additional features required for unstructured environments. Further, we found little impact in the curvature computation method chosen. Additionally, we showed that IMU dewarping performs the best, with constant velocity also being worthwhile given there is sufficient compute. We found that constant velocity initialization is excellent if motion is smooth, but IMU initialization is preferred in cases of aggressive behavior. Finally, we observed that if planar features are used, initialization tends to have a much smaller effect than with edge features.

Future work may include evaluation of further core techniques. These include components such as map aggregation using voxel maps, kd-trees, or others; the optimal way to fuse with IMU measurements; and which method is most robust for computing planar normals.

References

[1] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Robotics: Science and Systems X*. Robotics: Science and Systems Foundation, July 2014.

[2] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 5135–5142.

[3] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "CT-ICP: Real-time Elastic LiDAR Odometry with Loop Closure," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 5580–5586.

[4] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: Science and Systems*, vol. 2. Seattle, WA, 2009, p. 435.

[5] L. Huang, "Review on LiDAR-based SLAM Techniques," in *2021 International Conference on Signal Processing and Machine Learning (CONF-SPML)*, Nov. 2021, pp. 163–168.

[6] Y. Zhang, P. Shi, and J. Li, "3D LiDAR SLAM : A survey," *The Photogrammetric Record*, vol. 39, no. 186, pp. 457–517, June 2024.

[7] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. Quebec City, Que., Canada: IEEE Comput. Soc, 2001, pp. 145–152.

[8] D. Lee, M. Jung, W. Yang, and A. Kim, "LiDAR odometry survey: Recent advancements and remaining challenges," *Intelligent Service Robotics*, vol. 17, no. 2, pp. 95–118, Mar. 2024.

[9] N. Jonnavithula, Y. Lyu, and Z. Zhang, "LiDAR Odometry Methodologies for Autonomous Driving: A Survey," Sept. 2021.

[10] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1029–1036, Feb. 2023.

[11] Y. Pan, P. Xiao, Y. He, Z. Shao, and Z. Li, "MULLS: Versatile LiDAR SLAM via multi-metric linear least square," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 633–11 640.

[12] M. Ramezani, K. Khosoussi, G. Catt, P. Moghadam, J. Williams, P. Borges, F. Pauling, and N. Kottege, "Wildcat: Online Continuous-Time 3D Lidar-Inertial SLAM," May 2022.

[13] S. Zhao, H. Zhang, P. Wang, L. Nogueira, and S. Scherer, "Super Odometry: IMU-centric LiDAR-Visual-Inertial Estimator for Challenging Environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2021, pp. 8729–8736.

[14] P. Li, R. Wang, Y. Wang, and W. Tao, "Evaluation of the ICP Algorithm in 3D Point Cloud Registration," *IEEE Access*, vol. 8, pp. 68 030–68 048, 2020.

[15] K. Crane, "Discrete differential geometry: An applied introduction," *Notices of the AMS, Communication*, vol. 1153, 2018.

[16] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. Fallon, "The Newer College Dataset: Handheld LiDAR, Inertial and Vision with Ground Truth," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 4353–4360.

[17] L. Zhang, M. Camurri, D. Wisth, and M. Fallon, "Multi-Camera LiDAR Inertial Extension to the Newer College Dataset," May 2022.

[18] L. Zhang, M. Helmberger, L. F. T. Fu, D. Wisth, M. Camurri, D. Scaramuzza, and M. Fallon, "Hilti-Oxford Dataset: A Millimeter-Accurate Benchmark for Simultaneous Localization and Mapping," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 408–415, Jan. 2023.

[19] Y. Tao, M. Á. Muñoz-Bañón, L. Zhang, J. Wang, L. F. T. Fu, and M. Fallon, "The Oxford Spires Dataset: Benchmarking Large-Scale LiDAR-Visual Localisation, Reconstruction and Radiance Field Methods," Nov. 2024.

[20] T.-M. Nguyen, S. Yuan, T. H. Nguyen, P. Yin, H. Cao, L. Xie, M. Wozniak, P. Jensfelt, M. Thiel, J. Ziegenbein, and N. Blunder, "MCD: Diverse Large-Scale Multi-Campus Dataset for Robot Perception," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2024, pp. 22 304–22 313.

[21] M. Jung, W. Yang, D. Lee, H. Gil, G. Kim, and A. Kim, "HeLiPR: Heterogeneous LiDAR dataset for inter-LiDAR place recognition under spatiotemporal variations," *The International Journal of Robotics Research*, vol. 43, no. 12, pp. 1867–1883, Oct. 2024.

[22] Y. Liu, Y. Fu, M. Qin, Y. Xu, B. Xu, F. Chen, B. Goossens, P. Z. H. Sun, H. Yu, C. Liu, L. Chen, W. Tao, and H. Zhao, "BotanicGarden: A High-Quality Dataset for Robot Navigation in Unstructured Natural Environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2798–2805, Mar. 2024.

[23] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual–Inertial Odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[24] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of SLAM algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, Nov. 2009.