2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

FORM: Fixed-Lag Odometry with Reparative Mapping utilizing Rotating LiDAR Sensors

Easton R. Potokar, Taylor Pool, Daniel McGann, and Michael Kaess

***Abstract*— Light Detection and Ranging (LiDAR) sensors have become a de-facto sensor for many robot state estimation tasks, spurring development of many LiDAR Odometry (LO) methods in recent years. While some smoothing-based LO methods have been proposed, most require matching against multiple scans, resulting in sub-real-time performance. Due to this, most prior works estimate a single state at a time and are "submap"-based. This architecture propagates any error in pose estimation to the fixed submap and can cause jittery trajectories and degrade future registrations. We propose Fixed-Lag Odometry with Reparative Mapping (FORM), a LO method that performs smoothing over a densely connected factor graph while utilizing a single iterative map for matching. This allows for both real-time performance and active correction of the local map as pose estimates are further refined. We evaluate on a wide variety of datasets to show that FORM is robust, accurate, real-time, and provides smooth trajectory estimates when compared to prior state-of-the-art LO methods.**

I. Introduction

Since their introduction, Light Detection and Ranging (LiDAR) sensors have become a popular sensor for many robot state estimation tasks due to their precision, long range capabilities, and high sensor rate. They have made their way into applications such as unmanned aerial vehicles, quadrupeds, and autonomous ground vehicles, with significant improvements in state estimation accuracy and mapping fidelity over prior sensor modalities [1].

For other modalities such as cameras, most state-of-the-art (SOTA) odometry methods estimate a window of previous poses simultaneously, a process known as smoothing [2]. Smoothing enables systems to use new information to refine past states, resulting in more accurate and consistent odometry estimates. Some have applied smoothing to LO, but generally attempt to perform Iterative Closest Point (ICP) against multiple previous scans individually, which results in an over-representation of the environment and generally sub-real-time performance. Others have utilized GPU-acceleration to achieve real-time smoothing performance, or are part of a large sensor fusion pipeline, but, to the author's knowledge, there has been no real-time LiDAR-only smoothing odometry methods.

To overcome these challenges, most current LiDAR Odometry (LO) methods adopt designs akin to filtering methods in that they estimate a single pose at a time. To utilize information from multiple previous scans, these approaches aggregate multiple scans into a single "submap" or "local map" [3], [4]. This provides more context for the ICP step, but remains real-time due to requiring only a single round of matching. Unfortunately, these submaps inherit any errors that may occur in state estimation and are never repaired. This, along with the submap downsampling techniques, can propagate error and cause irregular trajectories with significant jumps in pose estimation. These choppy trajectories can lead to inaccurate velocity estimation and can be disastrous for downstream tasks such as planning or control.

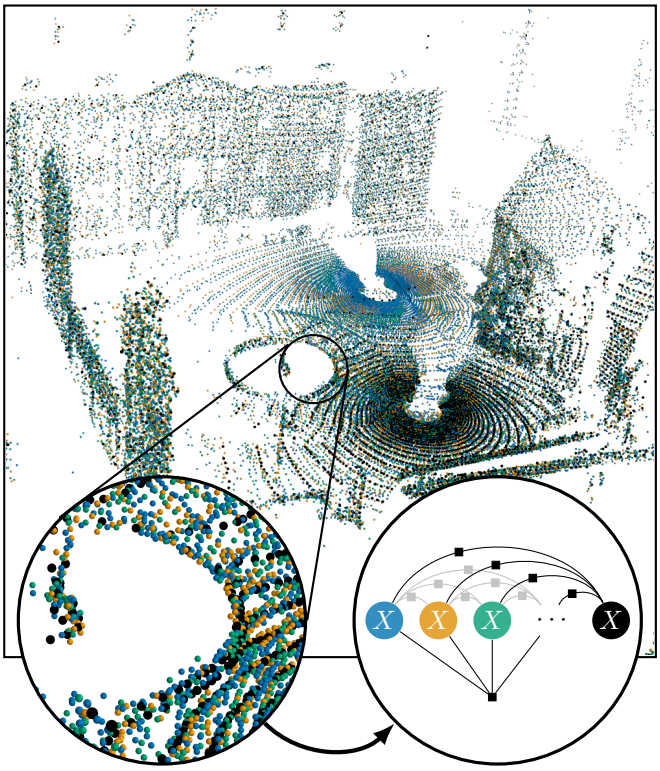

Fig. 1: An example map built using FORM from the Oxford Spires [5] dataset. Point colors represent which scan the point originated from with black points being from the current scan. The point matches are added as factors to a dense factor graph over a window of prior poses and are optimized in a dense optimization. At each timestep, the map is regenerated with the new, optimized poses. This scheme allows for reparative mapping to occur and minimizes overall pose error.

In this work, we propose **F**ixed-Lag **O**dometry with **R**eparative **M**apping (FORM), a novel LO method that performs fixed-lag smoothing to improve pose estimation

ERP and MK are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA USA,{potokar, kaess}@cmu.edu. TP is with Main Street Autonomy and DM with FieldAI.

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.

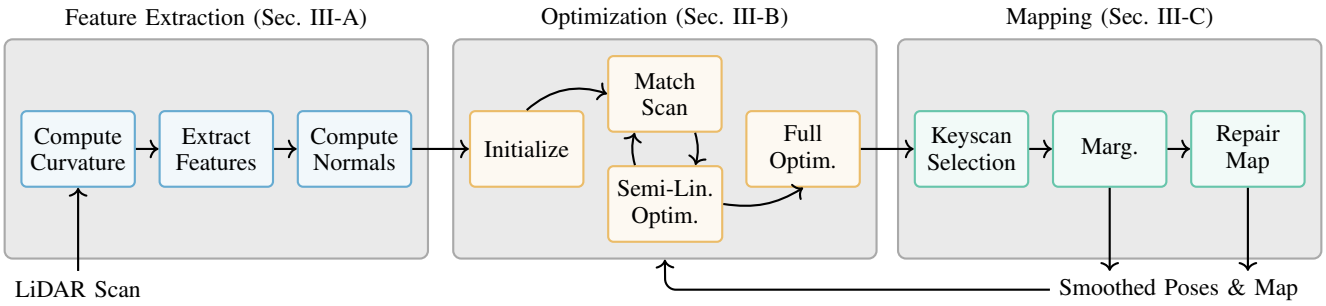

Fig. 2: The components that make up the FORM pipeline as described in Section III. First, point and planar features are extracted. Next, the pose is initialized, ICP iterations are performed utilizing a semi-linearized dense optimization over past poses, and a full nonlinear optimization follows. Finally, keyscan management is done, poses are marginalized, and a new map is generated from the smoothed poses.

accuracy while still maintaining real-time performance. Additionally, the map is iteratively updated with the smoothed poses to prevent errors from propagating. All together, this results in smoother trajectories, increased robustness, and more accurate estimates while maintaining real-time performance. More specifically, our contributions are:

- We present a novel real-time LO method that optimizes over a fixed window of poses and performs map corrections using the smoothed poses.
- To ensure FORM functions across environments, we develop a novel feature extraction method that utilizes both planar and point features from scans.
- We perform ablation studies to demonstrate the various important components of our method, including feature types and smoothing.
- We empirically validate our method against other state-of-the-art LO methods on a variety of datasets and show that FORM improves accuracy, reduces irregularities, and increases robustness while still retaining real-time performance.

Finally, we release our joint Python-C++ codebase as open-source[1], which includes the FORM implementation and code to easily recreate all experimental results for our method and all prior works.

II. Related Works

Over the past years, a large amount of LO work has been performed with a large variance in techniques. They can be split into roughly two categories: smoothing-based and filtering-based.

Fixed-lag smoothing is a popular technique for state estimation that estimates multiple timesteps of poses jointly and results in more accurate results than filtering-based approaches. Due to the time-complexity of the matching step in ICP, smoothing approaches for LO are more uncommon. Some have simply performed multiple ICPs that match the current scan against a set of prior scans, with resulting ICP pose solutions being put into a pose graph at the end [6]. However, this essentially linearizes each scan pair and lacks results that demonstrate real-time performance. Most similar to ours, others [7] utilize a hierarchical approach that matches against a submap with scan-indexed points and also generates a dense factor graph, but to the author's knowledge, still utilize a fixed submap.

Others have leveraged GPU acceleration to achieve real-time smoothing performance [8] or down-sampling residuals using coresets [9]. Unfortunately, not all vehicles have GPUs and both these methods are utilized solely with IMU fusion.

To maintain real-time performance, most current methods perform filtering, only actively estimating a single pose at any given time. Generally, most of these will perform ICP [10] on the current scan against a submap comprised of an aggregation of prior scans. Naturally, there are many variants of this process. Some utilize features extracted from the scans, such as planar or edge features [11], [12], [13], PCA feature extraction [14], or even learned semantic features [15]. Rather than perform feature extraction, others utilize the raw points from the scans [3], [4]. These result in excellent computational speed and robustness, but often have jagged trajectories due to usage of point-to-point residuals and errors accumulating in the sub-map.

Some filtering-based methods utilize more complex techniques, such as continuous-time estimation over the scan [16], [17], or utilizing novel kd-tree structures to compute normals and matching the normals to previous scans [18]. While these do result in increased accuracy for many datasets, they often are sub-real-time, require parameter tuning, or still have degradation due to submap aggregation.

Our proposed method achieves the best of both these categories; it has the accuracy and degradation-free results of smoothing while still remaining real-time like prior filtering-based approaches.

III. Fixed-Lag Odometry with Reparative Mapping LO

FORM is comprised of a number of modular parts, starting with feature extraction, followed by ICP steps and optimization, and finally keyscan management and mapping. These are summarized in Fig. 2 and we cover each in depth here. Note that integer system parameters are denoted with a subscripted N, and thresholds with a subscripted δ.

[1]https://github.com/rpl-cmu/form/

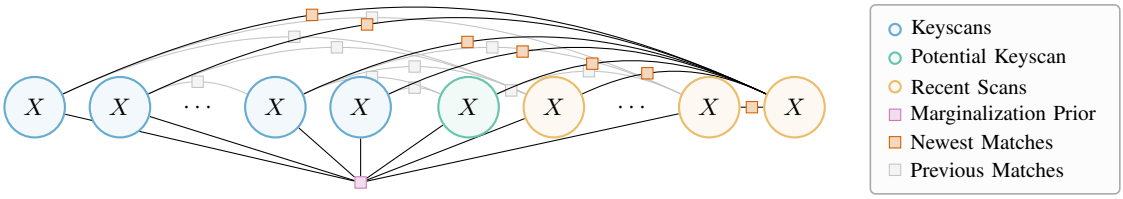

Fig. 3: Example of the dense factor graph used in FORM. Shown are keyscans, recent scans, and the recent scan that is being considered to become a keyscan. During semi-linearized optimization, all the previous matches are linearized to use less compute. The newest matches are also recomputed at every ICP step.

A. Feature Extraction

While the feature extraction used in FORM is not the main source of novelty, we have found features to have a significant impact on odometry performance and cover it as a demonstration of the joint planar and point feature extraction. We also note that the core components of FORM, the smoothing optimization and map corrections, are feature invariant and arbitrary features can be utilized.

For our base feature extraction, FORM utilizes classical LOAM [11] planar features that are extracted by leveraging the scanline structure of rotating LiDAR sensors. The first step is to mark any points outside of the minimum or maximum range and points along the edge of scanlines as invalid, along with N_{neighbor} of their neighbors. We note that since we are not extracting edge features, we do not perform the occlusion or parallel checks that the original LOAM paper used.

Next, an estimated curvature $\hat{\kappa}_i$ is computed for valid point p_i utilizing N_{neighbor} neighbors in either direction along the scanline [11],

$$\hat{\kappa}_i = \left\| \frac{1}{N_{\text{neighbor}}} \sum_{0<j\leq N_{\text{neighbor}}} (p_{i+j} - 2p_i + p_{i-j}) \right\| \quad (1)$$

Each scanline is split into N_{sectors} sectors to ensure features are distributed geometrically, and the N_{planar} points with the smallest curvature less than a threshold δ_{planar} in that sector are selected as planar keypoints, making sure to select keypoints that are spaced by at least N_{neighbor}.

Once features are selected, a normal for each is estimated by solely using the scan. We found utilizing a single scan to be more reliable, does not require re-computation of normals due to the adaptive map described later, and reduces the amount of correlation between poses. Normals are computed by searching adjacent scanlines for the nearest point to the feature f, and selecting N_{neighbor} points within a radius δ_{radius} of said nearest point. A normal is then extracted via the smallest eigenvector of the covariance of the selected neighborhood $\mathcal{N}$, making sure $|\mathcal{N}| > 5$,

$$\Sigma_f = \sum_{p\in\mathcal{N}} (p-f)(p-f)^\top \quad (2)$$

Additionally, we select a number of point features to be utilized. Point features have previously been shown to add robustness when operating in unstructured environments where planar features are less abundant [19]. To ensure our features represent the geometry of the entire scan, we select points features where planar features are not present. We select N_{point} point features in each sector in a scanline, making sure they are spaced N_{neighbor} points away from other planar or point features and are not points outside the minimum or maximum range. An example map built using these features can be seen in Fig. 4. Moving forward, we denote the set of features for scan i as $\mathcal{P}_i$.

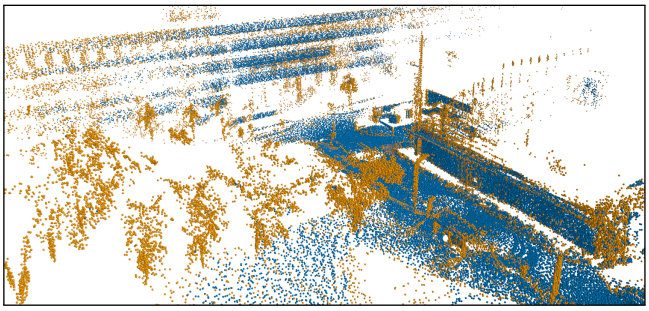

Fig. 4: An example FORM map, with planar features in blue and point features in orange. Note the trees on the left captured by point features, and walls and ground captured by planar features.

B. Optimization

Once features are extracted, the optimization stage is performed as shown in the second portion of Fig. 2. First, an initial pose estimate is computed for the scan using a constant velocity model, with velocity extracted using the previous two poses.

This is followed by ICP iterations, with slight variation to the standard method. We match each keypoint p_i in $\mathcal{P}_i$ to the closest point p_k from scan k in the map $\mathcal{M}$ within threshold distance δ_{match}, with all distances computed using the standard Euclidean metric. The generation of $\mathcal{M}$ is described in more detail in the following section. For each match of p_i and p_k, we add a factor from pose X_i to X_k, resulting in a densely connected factor graph that can be seen in Fig. 3. The residuals used are the standard point-to-plane metric for planar matches and point-to-point metric for point matches as follows [26] where R_k is the rotation of pose X_k and n_k the estimated normal at point p_k,

$$r_{\text{planar}}(X_k, X_i) = (R_k n_k)^\top (X_i p_i - X_k p_k) \quad (3)$$

$$r_{\text{point}}(X_k, X_i) = X_i p_i - X_k p_k \quad (4)$$

With these new factors, we re-optimize the smoothing window using Levenberg-Marquadt. It should be noted, unlike many factor graph problems such as visual-inertial odometry, the only variables to be estimated are the poses. Since all

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.

Dataset	Acronym	Year	Setting	Device	Lidar	Beams	Ground Truth
Newer Stereo-Cam [20]	N20	2020	Outdoor Campus	Handheld	Ouster OS-1	64	Laser Scanner & ICP
Newer Multi-Cam [21]	N21	2021	Outdoor Campus	Handheld	Ouster OS-0	128	Laser Scanner & ICP
Hilti 2022 [22]	H22	2022	Indoor Campus	Handheld	Hesai PandarXT	32	Laser Scanner & ICP
Oxford Spires [5]	OS	2024	Campus	Backpack	Hesai QT64	64	Laser Scanner & ICP
Multi-Campus [23]	MC	2024	Outdoor Campus	Handheld/ATV	Ouster OS-1	64/128	Laser Scanner & Cont-Opt.
Botanic Garden [24]	BG	2024	Trail Road	ATV	Velodyne VLP16	16	Laser Scanner & ICP
CU-Multi [25]	CU	2025	Outdoor Campus	ATV	Ouster OS-0	64	RTK-GPS Fusion

poses are likely connected, this results in an extremely dense and low variable dimension factor graph. The largest cost of the optimization is actually the high dimensional residual and jacobian computations [8].

Due to this, and since previous poses in the smoothing window will not change significantly, before beginning the ICP loop we linearize all factors not connected to the current scan about the current pose estimates. This results in a "semi-linearized" optimization that significantly speeds up the ICP process and allows more time for additional matching iterations to be performed. Also, this optimization is initialized with smoothed pose results from the previous full optimization, not from the previous semi-linearized optimization, in order to avoid poor pose estimates from initial matching that can degrade estimates.

The matching and optimization loop continues until a maximum iteration threshold N_{icp} is hit, or until the change in the most recent pose is less than a threshold δ_{icp}. Once the ICP iterations are completed, a full nonlinear optimization is performed without the fixed linearization utilized during ICP.

C. Mapping

Once optimization is finalized, keyscanning and map management are performed as in the right of Fig. 2. Keyscanning ensures the map $\mathcal{M}$ continues to cover sufficient geometric area. At any given time, a maximum of N_{key} keyscans and N_{recent} recent scans are kept. If there are more than N_{recent} recent scans, the oldest recent scan i is considered to become a keyscan based on the metric,

$$\frac{|\text{matches to recent scans}|}{|N_{\text{recent}}||\mathcal{P}_i|} > \delta_{\text{key}} \tag{5}$$

This roughly measures how important the scan is to the other recent scans, weighted by the number of recent scans and keypoints extracted. If scan i does not meet this threshold, it is marginalized out of the optimization.

A check is also performed to see if any keyscans are no longer needed. If a keyscan has not been matched to a recent scan in the last N_{marg} iterations, it is marginalized out. Additionally, to make sure an unlimited number of keyscans is not possible, the oldest keyscan is marginalized out if the number of keyscans exceeds N_{key}. In practice we have found this limit is rarely hit.

Next, a new map $\mathcal{M}$ is generated. Any keypoints p_i from the current scan $\mathcal{P}_i$ with a match distance above a threshold δ_{map} are kept to be inserted into the map. We found adjusting this parameter to provide intuitive results; lower values resulted in a denser map and increased accuracy at a runtime cost, with higher values providing the opposite.

Map points are all stored in their respective scan's local frame. Upon map generation, all of these points are transformed into the world frame using the most recent pose estimates and stored with each point's respective scan index to form the map $\mathcal{M}$. Since the map is generated with smoothed poses, many previous registration errors are removed resulting in a repaired or corrected map. Experiments were done with performing this map generation after each ICP step, but it was found that initial matches resulted in inaccurate pose estimates and caused degradation in the map, hence the map is only generated at the end of the pipeline.

Finally, we note that FORM does have a decent number of parameters. However, we have found a large majority of them to have limited impact on performance or are feature extraction related and not a core component of FORM. The parameters used are all shown along with their subjective significance to performance in Table II

IV. Experiment Setup

We design our experiments with carefully chosen metrics, datasets, and prior works to demonstrate the performance of FORM. All data-loading and prior work methods were run using the open-source `evalio` [19] library.

A. Datasets

The list of seven datasets chosen for evaluation can be seen in Table I, with a total of 64 trajectories. To ensure that our method generalizes well, we have prioritized maximizing the variety of LiDAR brands, LiDAR beam-size, and vehicle motion while selecting only datasets that provide accurate ground truth. As is common for many open-source datasets, most align LiDAR scans to a map created from a high precision laser scanner to generate ground truth. All datasets have LiDAR sensor rates of $10\,\text{Hz}$, with the exception of CU-Multi which is $20\,\text{Hz}$.

The trajectories in our chosen datasets range anywhere from around $1\,\text{min}$ to $25\,\text{min}$ long. Due to the number of experiments ran, we limit experiments to running on the first $10\,\text{min}$ of longer trajectories, noting this results in about 6,000 LiDAR scans and is a sufficient sample size for our chosen metrics.

B. Metrics

The most common metric used for evaluating LO drift is translational windowed Relative Trajectory Error (RTE_j) with window size j in meters. To compute this, from the

TABLE II: FORM System Parameters

Significant Parameters

Parameter	Value	Description
N_{neighbor}	5	Neighbor points in curvature computation
δ_{map}	0.1 m	Map insertion threshold
δ_{match}	0.8 m	Maximum match distance
N_{recent}	10	Number of recent scans
δ_{key}	0.1	Keyscan criteria threshold

Insignificant Parameters

Parameter	Value	Description
N_{sectors}	6	Extraction scanline sector size
N_{point}	3	Point features extracted per scanline sector
N_{planar}	50	Planar features extracted per scanline sector
δ_{planar}	1.0	Threshold for planar classification
δ_{radius}	1.0 m	Neighbor radius for normal computation
N_{icp}	30	Number of ICP iterations
δ_{icp}	1e-4	Stopping pose threshold for ICP
N_{key}	50	Maximum number of keyscans
N_{marg}	10	Unconnected keyscan marginalization steps

ground truth pose set $\mathcal{X}$, we place all subsequences of metric length j defined by a change in pose δ into the set $\mathcal{X}_j$. We additionally skip any subsequences with an identical ending pose to ensure times when the vehicle is stationary are not overly represented. The metric is then computed by, denoting our trajectory estimate as $\hat{\mathcal{X}}$,

$$\text{RTE}_j(\mathcal{X}, \hat{\mathcal{X}}) = \left(\frac{1}{|\mathcal{X}_j|} \sum_{\delta_i \in \mathcal{X}_j, \hat{\delta}_i \in \hat{\mathcal{X}}_j} \|\text{trans}(\delta_i \ominus \hat{\delta}_i)\|^2 \right)^{1/2} \tag{6}$$

Importantly, varying the window size can provide different context about the results; a smaller window size can capture the local smoothness of a trajectory, while longer window sizes represent longer-term drift quantities. This can be seen in Fig. 5, where RTE_j is plotted over window size in meters. Notice the choppy trajectory of prior works, which is well-captured by the small window size RTE_j.

Rotational RTE_j can also be computed, but we found it produces similar trends to translational RTE_j so it is omitted for conciseness.

Finally, we also measure real-time performance by timing the performance of adding a new LiDAR scan to the pipelines, making sure to exclude any time spent on data loading, result saving, etc.

C. Prior Works

We compare against a number of SOTA LO methods that have been open-sourced. This includes the point-to-point submap-based KiSS-ICP [3], joint point-to-point and point-to-plane submap-based GenZ-ICP [4], the novel kd-tree matcher MAD-ICP [18], and the continuous-time CT-ICP [16]. Unfortunately, at this time, to the author's knowledge, there is no open-source CPU-based smoothing LO methods available for comparison.

All prior works were run with the default parameters found in their open-source repositories. The only exception to this is increasing the number of ICP iterations and switching

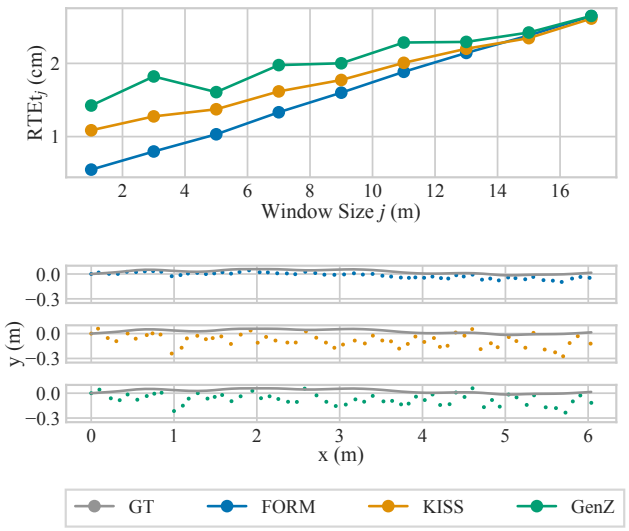

Fig. 5: RTE_j small window size demonstration on a trajectory in the Oxford Spires dataset. In the above plot, many filtering methods result in higher than expected RTE_j for small windows sizes. This can be seen in the jittery trajectories shown in the other plots, with the exception of FORM.

the solver to Ceres in CT-ICP. Additionally, MAD-ICP is ran with a "real-time" flag enabled, resulting in varying performance depending on compute power. While excellent for real-world deployment, it makes benchmarking slightly stochastic. When disabled, MAD-ICP performs at sub-real-time, so we leave it enabled. Moving forward we drop the "ICP" at the end of prior works for conciseness. Finally, dewarping was disabled on all prior works except CT-ICP, as it should provide a similar marginal improvement across methods and may even reduce accuracy of some prior works that produce jittery trajectories. As mentioned previously, FORM parameters for all experiments are shown in Table II.

V. Experiments

We designed a number of experiments to showcase the odometry performance of FORM and ablations to demonstrate the effectiveness of the various techniques utilized. Across all experiments, we utilize a RTE_1 to represent the smoothness of a trajectory, and RTE_{30} for odometry drift. All experiments were run on a 12th-Gen Intel i9 processor.

It should be noted that for FORM, the pose is saved immediately after being added to the pipeline, not at the time of marginalization. It is likely that FORM results on marginalized poses would be slightly improved due to the effects of continued smoothing.

A. Ablations

We perform an ablation to show the impact of optimizing a window of poses rather than just filtering the current pose. The rest of the pipeline, including feature extraction, map aggregation, etc., all remained the same. The summarized results can be seen in Table III. As expected, smoothing requires more compute, but does improve performance across all datasets, particularly for the longer windowed RTE_{30}

TABLE III: Ablation study showing the effects of smoothing in FORM. Values are averaged across dataset sequences, with results in bold representing the best results. Smoothing improves the drift performance of FORM but has limited impact on the smoothness of the trajectory.

D.	Filtered			Smoothed		
	RTE_1	RTE_{30}	Hz	RTE_1	RTE_{30}	Hz
N20	**0.08**	0.46	25.5	**0.08**	**0.45**	21.4
N21	**0.07**	0.62	12.5	**0.07**	**0.58**	11.0
H22	0.14	1.58	32.1	**0.13**	**1.24**	21.2
OS	**0.08**	0.69	26.1	**0.08**	**0.60**	21.0
MC	**0.09**	0.53	18.9	**0.09**	**0.51**	15.8
CU	0.04	0.31	39.1	**0.03**	**0.20**	31.8
BG	**0.05**	1.00	68.7	**0.05**	**0.82**	47.0

TABLE IV: Ablation study comparing features used in FORM. Values are averaged across dataset sequences, with results in bold representing the best results. While the point and planar variant of FORM does not significantly improve smoothness, it does help with odometry drift at a performance cost.

D.	Planar			Point & Planar		
	RTE_1	RTE_{30}	Hz	RTE_1	RTE_{30}	Hz
N20	0.09	0.64	35.5	**0.08**	**0.45**	21.4
N21	**0.07**	**0.58**	18.9	**0.07**	**0.58**	11.0
H22	**0.11**	**1.12**	28.1	0.13	1.24	21.2
OS	**0.08**	1.01	35.6	**0.08**	**0.60**	21.0
MC	0.10	0.67	30.8	**0.09**	**0.51**	15.8
CU	0.04	0.28	51.4	**0.03**	**0.20**	31.8
BG	0.06	0.86	58.3	**0.05**	**0.82**	47.0

metric. Interestingly, it had a more minimal impact on trajectory smoothness; this could be due to using the pose estimates immediately rather than upon marginalization as mentioned previously.

Additionally, we perform an ablation comparing FORM with only planar features to the version with both planar and point features. The results are summarized in Table IV. Generally, the addition of points features reduced drift amounts in the longer windowed RTE_j, particularly for datasets with unstructured environments such as Multi-Campus (MC) and Oxford Spires (OS), while having a limited impact on trajectory smoothness. The exception to this is the extremely structured datasets Hilti 2022 (H22) and Newer Multi-Cam (N21), where point features were less impactful.

B. Comparison with Prior Works

We additionally run on 64 trajectories across the seven datasets listed in Table I. The main results are summarized in Table VI

Representing odometry drift, RTE_{30} results can be seen in the right of Table VI. FORM is competitive in this category and is generally only a few centimeters away from the best method. Importantly, FORM never has a significant failure case and performs well across all environments with identical parameters. This is well-summarized in Fig. 6, which shows success rates for different RTE_{30} thresholds. In this figure we can see that FORM clearly has the largest area under the curve and is the only pipeline to successfully complete all trajectories at a success threshold of $3.08\,\mathrm{m}$.

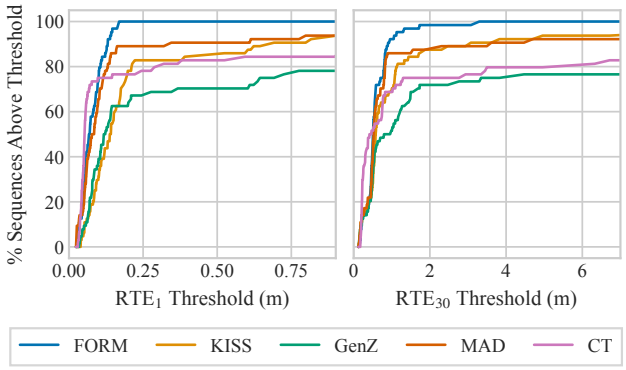

Fig. 6: Percentage of sequences for each pipeline that falls below the given RTE_1 and RTE_{30} thresholds. Note this does not take into account sequences with sub-real-time performance. FORM provides smooth trajectories as shown by RTE_1 below $20\,\mathrm{cm}$ on all sequences, and is the sole pipeline to have successfully completed all sequences.

Importantly, FORM is easily the leader in RTE_1, often by a significant margin. This is well summarized in the left of Fig. 6 and Table VI. For some sequences, such as bp01 in Oxford Spires, the RTE_1 of FORM is a mere fraction of that of KISS-ICP and GenZ-ICP despite having similar RTE_{30}. This is of crucial importance, as these trajectories will often be used by downstream tasks such as planning, control, or even initialization of the next step of the pipeline. Jittery trajectories, as seen in Fig. 5, can cause inaccurate estimates of velocity and position, and significantly degrade robustness of the overall systems.

Finally, we compare the rate of computation to ensure real-time performance. These rates are averaged across datasets, while excluding trajectories with RTE_{30} greater than ten meters, as often those sequences run at irregular rates due to failure. The results are summarized in Table V. Both KISS-ICP and GenZ-ICP are the fastest, which is to be expected since they are simple ICP methods. On the other side, MAD-ICP ran almost exactly at real-time due to its "real-time" mode. CT-ICP struggled to be real-time in trajectories it finished successfully, most notably Multi-Campus and CU-Multi. On other datasets, it often failed and caused the pipeline to run super-real-time. FORM falls somewhere in between these two camps; it does require more compute time than the simpler ICP methods, but runs comfortably real-time, particularly for 64-beam lidars datasets.

TABLE V: Average compute speed of the various pipelines in hertz, while excluding trajectories with RTE_{30} greater than ten meters. Real-time performance is achieved above the LiDAR sensor rate, $10\,\mathrm{Hz}$ for all datasets except CU-Multi which is $20\,\mathrm{Hz}$. FORM consistently provides real-time performance.

D.	FORM	KISS	GenZ	MAD	CT
N20	21.4	43.1	41.8	22.3	9.4
N21	11.0	72.1	45.9	27.3	13.5
H22	21.2	182.1	60.2	102.4	22.5
OS	21.0	85.8	47.9	39.8	15.3
MC	15.8	35.1	26.9	19.1	6.5
CU	31.8	87.0	46.0	25.6	10.2
BG	47.0	170.3	55.3	64.0	15.6

TABLE VI: Both RTE_1 and RTE_{30} for all datasets. Strike-throughs indicate a failure to complete, italics sub-real-time performance, **bold blue** the best real-time result, and light blue the second best. FORM has competitive drift performance while still maintaining a smooth trajectory and real-time speed.

		RTE_1				
D.	Seq.	FORM	KISS	GenZ	MAD	CT
N20	le	**0.04**	0.09	0.07	0.05	0.07
	pm	0.14	0.18	~~1.19~~	**0.14**	*1.83*
	se	0.05	0.17	0.08	**0.05**	*0.07*
N21	c	**0.04**	0.13	0.08	0.05	0.05
	me	0.03	0.05	0.04	**0.03**	0.06
	mh	*0.13*	0.21	1.26	0.61	**0.06**
	mm	**0.07**	0.10	0.10	0.07	0.28
	p	0.06	0.10	0.08	0.07	**0.05**
	qe	0.05	0.07	0.06	0.06	**0.04**
	qh	0.11	0.17	0.13	1.32	**0.08**
	qm	*0.07*	0.12	0.11	**0.10**	0.90
	s	*0.05*	0.80	0.06	**0.03**	3.88
H22	atug2	**0.17**	1.85	0.64	0.32	5.16
	b2	0.14	0.82	0.14	**0.11**	1.26
	clg2	**0.15**	1.55	0.20	~~0.13~~	1.00
	cul1	**0.10**	0.64	0.62	1.23	4.39
	cul2	**0.08**	0.90	0.69	0.78	1.09
	cul3	**0.16**	4.96	~~4.50~~	1.32	4.39
OS	bl02	0.10	0.21	0.35	0.10	**0.04**
	bp01	0.12	0.38	0.60	**0.12**	0.52
	bp02	0.05	0.11	0.14	0.06	**0.05**
	bp05	**0.13**	0.61	1.27	0.16	1.45
	cc01	0.07	0.21	0.13	0.08	**0.03**
	cc02	0.05	0.17	0.13	0.07	**0.04**
	cc03	0.05	0.10	0.08	0.06	**0.03**
	cc05	0.05	0.14	0.09	0.06	**0.05**
	kc02	**0.08**	0.19	0.12	0.09	0.22
	kc03	**0.07**	0.39	0.21	0.09	0.14
	kc04	0.08	0.59	0.72	0.10	**0.04**
	oq01	0.07	0.13	0.10	0.07	**0.04**
	oq02	**0.07**	0.14	0.20	0.08	0.29
MC	kd06	**0.10**	0.15	0.14	0.11	*0.05*
	kd09	**0.13**	0.20	~~0.20~~	0.14	*0.06*
	kd10	**0.10**	0.18	~~0.23~~	0.12	*0.37*
	kn01	**0.10**	0.15	0.14	0.11	*0.05*
	kn04	**0.10**	0.14	~~0.13~~	0.13	*0.04*
	kn05	**0.09**	0.16	~~0.14~~	0.10	*0.05*
	nd01	**0.07**	0.12	0.12	0.09	*0.06*
	nd02	**0.05**	0.08	0.08	0.06	*0.04*
	nd10	**0.06**	0.10	0.10	0.07	*0.06*
	nn04	**0.05**	0.09	0.08	0.06	*0.05*
	nn08	**0.06**	0.12	0.12	0.08	*0.05*
	nn13	**0.09**	0.15	0.24	0.10	*0.10*
	td02	**0.09**	0.14	0.12	0.14	*0.04*
	td03	**0.09**	0.14	~~0.11~~	0.09	*0.04*
	td04	**0.12**	0.19	~~0.14~~	0.13	*0.05*
	tn07	**0.11**	0.17	~~0.14~~	0.16	*0.05*
	tn08	0.10	0.15	~~0.12~~	**0.10**	*0.05*
	tn09	0.11	0.17	~~0.13~~	**0.09**	*0.04*
CU	klr1	0.03	0.04	0.04	**0.03**	*0.04*
	klr2	0.03	0.04	0.03	**0.02**	*0.04*
	klr3	0.03	0.05	0.04	**0.03**	*0.04*
	klr4	0.03	0.04	0.04	**0.03**	*0.03*
	mcr1	0.05	0.06	0.06	**0.05**	*0.06*
	mcr2	0.03	0.05	0.06	**0.03**	*0.03*
	mcr3	0.03	0.06	0.05	**0.02**	*0.04*
	mcr4	0.03	0.05	~~0.04~~	**0.02**	*0.04*
BG	0500	**0.05**	0.09	0.07	0.05	0.05
	0501	0.05	0.07	0.07	**0.04**	0.05
	0507	**0.05**	0.08	0.11	0.06	0.05
	0601	0.05	0.08	0.07	0.05	**0.05**
	0803	0.06	0.09	0.09	0.06	**0.05**
	1800	0.05	0.07	0.07	**0.05**	0.06
	1813	0.06	0.11	0.11	**0.05**	0.05

		RTE_{30}				
D.	Seq.	FORM	KISS	GenZ	MAD	CT
N20	le	**0.44**	0.46	0.46	0.45	0.64
	pm	**0.54**	0.57	~~1.89~~	0.54	*29.69*
	se	0.38	0.39	0.37	**0.36**	*0.55*
N21	c	0.53	0.66	0.51	0.55	**0.37**
	me	0.14	0.14	0.14	**0.12**	0.32
	mh	*1.12*	1.73	14.58	11.95	**0.51**
	mm	0.21	0.23	0.22	**0.20**	2.77
	p	0.55	0.66	0.57	0.56	**0.47**
	qe	0.49	0.52	0.54	0.53	**0.37**
	qh	**0.93**	1.08	1.04	8.73	1.27
	qm	*0.79*	0.91	0.96	**0.87**	17.43
	s	*0.47*	6.72	0.39	**0.28**	132.51
H22	atug2	**3.08**	14.69	11.05	4.47	33.38
	b2	**0.78**	10.35	0.78	1.50	31.89
	clg2	**0.89**	3.61	1.20	~~—~~	13.25
	cul1	**0.56**	2.94	3.82	20.93	20.54
	cul2	**0.40**	2.30	9.37	3.49	3.34
	cul3	**1.71**	36.27	~~21.94~~	14.85	22.90
OS	bl02	0.70	0.83	1.00	0.60	**0.37**
	bp01	1.05	1.07	2.47	**0.69**	10.59
	bp02	0.42	0.39	0.39	**0.36**	1.23
	bp05	1.30	1.69	3.30	**0.83**	19.41
	cc01	0.48	0.60	0.52	0.46	**0.24**
	cc02	0.46	0.60	1.33	0.47	**0.41**
	cc03	0.46	0.50	0.50	0.47	**0.30**
	cc05	**0.41**	0.54	0.47	0.45	1.02
	kc02	0.49	0.82	0.49	**0.47**	8.50
	kc03	**0.51**	4.70	0.62	0.52	3.45
	kc04	0.55	0.99	1.16	0.55	**0.31**
	oq01	0.45	0.56	0.48	0.47	**0.23**
	oq02	0.48	0.55	0.54	**0.48**	6.32
MC	kd06	**0.49**	0.51	0.51	0.51	*0.19*
	kd09	**0.58**	0.62	~~0.62~~	0.60	*0.22*
	kd10	**0.67**	0.70	~~0.71~~	0.70	*4.96*
	kn01	**0.45**	0.47	0.46	0.48	*0.19*
	kn04	**0.46**	0.49	~~0.46~~	1.87	*0.21*
	kn05	**0.51**	0.57	~~0.48~~	0.53	*0.23*
	nd01	0.55	0.57	0.59	**0.51**	*0.72*
	nd02	0.36	0.34	**0.34**	0.34	*0.25*
	nd10	0.50	0.51	0.53	**0.49**	*0.30*
	nn04	**0.44**	0.44	0.45	0.45	*0.25*
	nn08	0.52	0.57	0.68	**0.50**	*0.25*
	nn13	**0.56**	0.59	1.73	0.57	*1.09*
	td02	**0.47**	0.49	0.49	0.50	*0.17*
	td03	**0.51**	0.54	~~0.52~~	0.53	*0.18*
	td04	**0.54**	0.58	~~0.56~~	0.82	*0.20*
	tn07	**0.58**	0.62	~~0.60~~	0.63	*0.21*
	tn08	**0.51**	0.54	~~0.54~~	0.52	*0.18*
	tn09	**0.49**	0.51	~~0.51~~	0.51	*0.20*
CU	klr1	0.30	0.25	**0.25**	0.26	*0.28*
	klr2	0.16	**0.13**	0.13	0.15	*0.21*
	klr3	0.20	0.18	**0.17**	0.19	*0.22*
	klr4	0.21	0.15	**0.15**	0.17	*0.20*
	mcr1	0.18	0.17	**0.17**	0.18	*0.21*
	mcr2	0.18	0.16	**0.16**	0.16	*0.20*
	mcr3	0.21	0.22	0.21	**0.20**	*0.22*
	mcr4	0.18	0.16	~~0.17~~	**0.15**	*0.22*
BG	0500	0.81	1.33	1.48	0.79	**0.75**
	0501	0.78	1.42	1.49	0.80	**0.77**
	0507	0.82	1.13	1.55	0.82	**0.76**
	0601	0.82	1.10	1.17	0.78	**0.74**
	0803	0.86	1.03	1.25	0.84	**0.74**
	1800	**0.80**	1.06	1.09	0.82	0.82
	1813	0.83	1.09	1.44	0.81	**0.77**

As a note on prior works, GenZ-ICP failed to complete a few trajectories due to accumulating a map of over a million points. We considered tuning the voxel size to fix this, but didn't want to degrade performance on other sequences so default parameters from the repository were used. Additionally, when CT-ICP works, it performs very well. Unfortunately, it either diverged or was sub-real-time a significant amount of the time. We suspect this could be fixed on a per dataset scenario via parameter tuning; however, we feel it is an important benchmark that a LO pipeline works across datasets without additional tuning.

VI. Conclusion

In this work, we have proposed FORM, a novel LO method that smooths over a past window of poses, iteratively performs map corrections as poses are smoothed, and still provides real-time performance due to matching against a single-map. We provided ablations to validate our design choices, specifically that of smoothing and feature selection, to show they increase robustness and accuracy. Additionally, we empirically show across over 60 sequences and seven datasets that FORM provides competitive long-term drift accuracy, robust performance, and still maintains smooth trajectory estimates across all datasets.

We believe this line of work holds the potential for further improvements. The feature extraction was mostly chosen as a safe, well-trusted technique, and we believe there is still significant work that can be done to improve performance and make it functional with solid-state LiDARs. It should also be noted that sensor fusion is straightforward with FORM due to its underlying factor graph representation, an important feature as LiDAR sensor fusion is currently an active area of research. Due to this, we plan to pursue the trivial fusion in a factor graph with other sensors such as IMUs, wheel encoders, and others.

References

[1] D. Lee, M. Jung, W. Yang, and A. Kim, "LiDAR odometry survey: Recent advancements and remaining challenges," *Intelligent Service Robotics*, vol. 17, pp. 95–118, Mar. 2024.

[2] G. Huang, "Visual-Inertial Navigation: A Concise Review," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9572–9582, May 2019.

[3] 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, pp. 1029–1036, Feb. 2023.

[4] D. Lee, H. Lim, and S. Han, "GenZ-ICP: Generalizable and Degeneracy-Robust LiDAR Odometry Using an Adaptive Weighting," *IEEE Robotics and Automation Letters*, vol. 10, pp. 152–159, Jan. 2025.

[5] 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," Tech. Rep. arXiv:2411.10546, arXiv preprint, Nov. 2024.

[6] A. Kurda, S. Steuernagel, and M. Baum, "Lidar-only Odometry based on Multiple Scan-to-Scan Alignments over a Moving Window," Tech. Rep. arXiv:2503.21293, arXiv preprint, Mar. 2025.

[7] S. Liang, Z. Cao, C. Wang, and J. Yu, "Hierarchical Estimation-Based LiDAR Odometry With Scan-to-Map Matching and Fixed-Lag Smoothing," *IEEE Transactions on Intelligent Vehicles*, vol. 8, pp. 1607–1623, Feb. 2023.

[8] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "GLIM: 3D range-inertial localization and mapping with GPU-accelerated scan matching factors," *Robotics and Autonomous Systems*, vol. 179, p. 104750, Sept. 2024.

[9] K. Koide, A. Takanose, S. Oishi, and M. Yokozuka, "Tightly Coupled Range Inertial Odometry and Mapping with Exact Point Cloud Downsampling," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2642–2648, May 2025.

[10] P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, Feb. 1992.

[11] 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.

[12] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-LOAM : Fast LiDAR Odometry and Mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4390–4396, Sept. 2021.

[13] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, Oct. 2018.

[14] 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)*, pp. 11633–11640, IEEE, 2021.

[15] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "SuMa++: Efficient LiDAR-based Semantic SLAM," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4530–4537, Nov. 2019.

[16] 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)*, pp. 5580–5586, May 2022.

[17] M. Karimi, M. Oelsch, O. Stengel, E. Babaians, and E. Steinbach, "LoLa-SLAM: Low-Latency LiDAR SLAM Using Continuous Scan Slicing," *IEEE Robotics and Automation Letters*, vol. 6, pp. 2248–2255, Apr. 2021.

[18] S. Ferrari, L. D. Giammarino, L. Brizi, and G. Grisetti, "MAD-ICP: It is All About Matching Data – Robust and Informed LiDAR Odometry," *IEEE Robotics and Automation Letters*, vol. 9, pp. 9175–9182, Nov. 2024.

[19] E. Potokar and M. Kaess, "A Comprehensive Evaluation of LiDAR Odometry Techniques," Tech. Rep. arXiv:2507.16000, arXiv preprint, July 2025.

[20] 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)*, pp. 4353–4360, Oct. 2020.

[21] L. Zhang, M. Camurri, D. Wisth, and M. Fallon, "Multi-Camera LiDAR Inertial Extension to the Newer College Dataset," Tech. Rep. arXiv:2112.08854, arXiv preprint, May 2022.

[22] 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, pp. 408–415, Jan. 2023.

[23] 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), pp. 22304–22313, IEEE, June 2024.

[24] Y. Liu, Y. Fu, M. Qin, Y. Xu, B. Xu, F. Chen, B. Goossens, P. Z. 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, pp. 2798–2805, Mar. 2024.

[25] D. Albin, M. Mena, A. Thomas, H. Biggie, X. Sun, D. Woods, S. McGuire, and C. Heckman, "CU-Multi: A Dataset for Multi-Robot Data Association," Tech. Rep. arXiv:2505.17576, arXiv preprint, July 2025.

[26] 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), pp. 145–152, IEEE Comput. Soc, 2001.