Radar-based Angular Correction for Tightly-coupled LiDAR Odometry

Hyesu Jang[1], Minwoo Jung[1], Sangwoo Jung[1] and Ayoung Kim[1*]

***Abstract*— Combining various types of sensors enables a robot to obtain plenty of information on input and exploit values in different ways. This information can be utilized to increase the accuracy of the simultaneous localization and mapping (SLAM) in unstructured environments. In this paper, we introduce a SLAM framework for tightly-coupled radar-LiDAR odometry via smoothing and mapping, which can redeem each data from radar and LiDAR. Our framework initially processes radar and LiDAR in parallel. Phase correlation can estimate relative pose between two radar polar images. LiDAR scans are marginalized and keyframes are selected for pose optimization. With the keyframe features, the relative poses between the adjacent scenes are optimized. After estimating relative poses of each sensor, radar odometry is synchronized and concatenated with LiDAR keyframes to exploit a factor graph. Finally, a keyframe pointcloud and optimized poses generate 3D map. We evaluate our SLAM framework's performance with LiDAR-based SLAM on five sequences based on two datasets.**

I. Introduction

Efforts to raise the accuracy of odometry in SLAM were conducted with various sensors such as camera, LiDAR, radar, and thermal camera. With these achievements, existing SLAM frameworks offer low odometry error in general environment. However, errors still exist, and higher accuracy is required for 3D map production or autonomous driving. One way to obtain accurate odometry is to exploit various types of sensors simultaneously. Among the autonomous driving sensors, we focus on radar sensors and LiDAR.

For the LiDAR sensor-based SLAM framework, the surrounding 3D environment is scanned with high accuracy and generates a dense point cloud. Moreover, calibration is less critical than vision sensors, and the framework has the advantage of being largely invariant to illumination change. For these reasons, there has been much research into performing SLAM based on LiDAR sensors, such as Lidar Odometry and Mapping (LOAM) [1] and Lightweight and Ground-Optimized Lidar Odometry and Mapping (LeGO-LOAM) [2]. However, the LiDAR sensor fails to detect the surrounding environment in harsh situations such as foggy weather. This causes a high error in the sight of odometry, which leads to a low uncertainty in points. On the other hand, a radar sensor works better in a harsh environment by using wider wavelengths. In particular, the recent development of the radar odometry method [3] has improved accuracy in foggy environments. Although the radar sensor seems to be a great solution for map construction, a problem still exists due to the limitation in resolution.

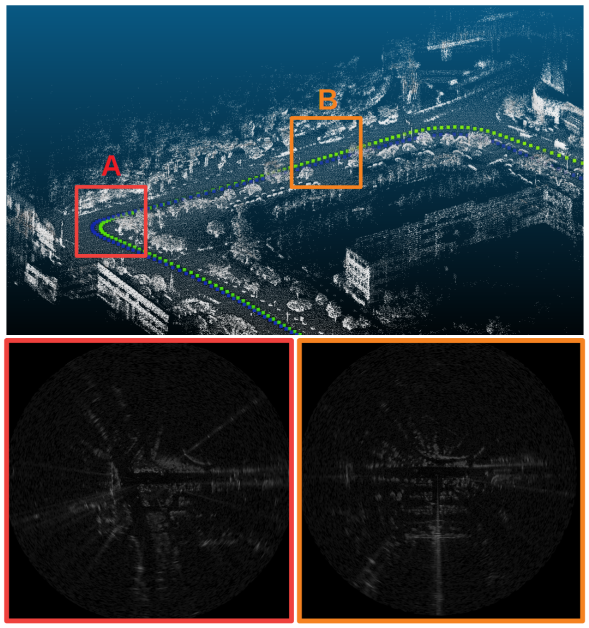


Fig. 1: We visualize the results of the proposed method and the two raw radar images scanned at poses A and B highlighted on the 3D map. Exploiting each raw radar image, we can estimate the x, y, and yaw values between pose A and B. After that, the relative pose between A and B can concatenate adjacent LiDAR keyframes if time synchronization is confirmed. The trajectory of our method (green) is more accurate and robust to z-axis rotation than the trajectory of LeGO-LOAM (blue).

In this paper, we propose a SLAM framework for tightly-coupled radar-LiDAR odometry via smoothing and mapping to redeem each LiDAR and radar sensor's aforementioned shortcomings. Radar and LiDAR odometry are processed in parallel. Exploiting phase correlation and feature extraction, the relative pose between each sensor's frame can be estimated. Then, radar factors are synchronized with LiDAR keyframes to estimate which frames have to be linked. Pointcloud is transformed to global coordinate from LiDAR coordinate through the exploitation of odometry which is optimized via factor graph and the transformed pointcloud constructs a 3D map. Fig. 1 shows the output and radar images used to estimate pose. The novelty of this paper includes the following:

[†]This study is a part of the research project, "Development of core machinery technologies for autonomous operation and manufacturing (NK230G)", which has been supported by a grant from National Research Council of Science Technology under the RD Program of Ministry of Science, ICT and Future Planning.

[1]H.Jang, M.Jung, S.Jung and A.Kim are with Depart. Mechanical Engineering, SNU, Seoul, S. Korea [dortz, moonshot, dan0130, ayoungk]@snu.ac.kr

- We suggest a SLAM framework that integrates LiDAR and radar sensors using a factor graph.
- Keyframes selected from LiDAR are concatenated with radar odometry using time synchronization between LiDAR and radar.
- Our framework is evaluated under two datasets[4][5] and outperforms the LiDAR-based method.

II. Related works

A. LiDAR-Based SLAM

In the case of LiDAR SLAM, the relative positions of adjacent pointclouds are compared using Iterative Closest Point (ICP) [6], Generalized Iterative Closest Point (GICP) [7], and Trimmed Iterative Closest Point (TrICP) [8] to track their locations. Although high accuracy can be obtained by comparing the entire pointcloud, it is impossible to perform SLAM in real time due to the disadvantage of high time complexity. Therefore, Yokozuka et al. [9] and Dellenbach et al. [10] attempted to redeem time complexity for LiADR mapping. Similar to detection and use of features in Visual SLAM [11][12][13], points, lines, and planes are detected and exploited for real-time LiDAR SLAM. In LOAM [1], planar and edge features extracted from the pointcloud are used to estimate the relative pose. However, there is a disadvantage in that it takes a long time to detect the features. Behley and Stachniss [14] proposed surfel-based LiDAR mapping system with efficient loop closure detection. An improved version of LOAM is LeGO-LOAM [2]. The overall method is similar to that of LOAM, but it develops feature extraction by segmenting each point. Calculation time is also reduced by exploiting the Levenberg-Marquardt algorithm, and the stability of the SLAM is increased by adding loop closing. Some of our framework is similar to LeGO-LOAM. However, in the case of pointcloud, since distortion can easily occur due to external vibration or rotation, the detected features also have distortion.

Another sensor that can compensate for the distortion is needed, as the error can have a significant impact on the result if it continues to accumulate. In the Robocentric Lidar-Inertial State Estimator (R-LINS) [15], the Tightly Coupled Lidar Inertial Odometry and Mapping (LIOM) [16], the Tightly Coupled Lidar-Inertial-Ranging Odometry (LIRO) [17], and the Tightly Coupled Lidar-Inertial Odometry via Smoothing and Mapping (LIO-SAM) [18], accuracy is greatly improved through more active use of the IMU. IMU preintegration is used to calculate IMU values between two frames as a single relative position change, and this helps to solve computational complexity while reducing the repetition of calculations and unnecessary variables such as multiple point clouds. Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping (LVI-SAM) [19] additionally exploits camera sensor to improve accuracy. Our framework utilizes radar sensors with LiDAR to reduce error.

B. Radar-Based SLAM

Radar-based SLAM has developed with the two main types of radars. The first is automotive radar, which has the advantage of giving Doppler information and radial velocity, but has the disadvantage of low accuracy and sparseness. The second is scanning radar, which gives a raw power-range image. Scanning radar has the high angular and range resolution, however, output images are noisy and the radar does not give velocity information. Due to the differences between the two sensors, various papers have addressed each of the shortcomings. For automotive radar, Holder et al. [20] proposed static target extraction and radar motion estimation. Radar pointcloud-based scan matching and pose graph optimization enhanced the accuracy of odometry. Park et al. [3] utilized radial velocity information to estimate 3d-ego motion. Unlike automotive radar based methods, most scanning radar-based methods utilize image processing. Therefore, various studies using the scanning radar have been attempted[21][22][23]. Among the studies, the currently highest performing studies are proposed by Barnes et al. [24] and Adolfsson et al. [25]. The learning-based method [24] shows a masking network that enhances radar images for odometry prediction. There is a filtering method [25] that computes the k-strongest points and oriented surface points. Since the angular information from scanning radar retains high accuracy, our radar factor for LiDAR SLAM stands on the basis of scanning radar.

III. Methodology

A. System Overview

To enable odometry estimation, we combine two individual sensors: radar and LiDAR. Since the wavelengths of the radar and LiDAR are different, we can reduce the estimation failure for various environments. The entire process can be divided into three parts: 2D pose estimation with a radar factor, 3D pose estimation and refinement with a LiDAR factor, and optimization with time synchronization exploiting the pose graph. Fig. 2 illustrates the entire framework. The robustness of our odometry relies on the radar. By calculating the phase correlation of the radar data, we directly estimate the relative motion between the keyframes. We generate a radar factor for both the adjacent frame and the frame located far from the current frame to obtain a reliable relative pose. The radar factor compensates for the error from the LiDAR feature-based pose refinement. To compute accurate odometry, we sync the radar and LiDAR factor values and optimize the pose graph.

B. Radar Factor

To calculate the ego motion between two radar scans, we conduct a phase correlation method, introduced by Park et al. [26]. When the polar image of the radar is published, we first down-sample the image to make coarse cartesian and log-polar images. Each coarse image produce an initial $\Delta x, \Delta y$ information and $\Delta\theta$ by Fourier Mellin Transform (FMT). Noise elimination and context emphasizing are conducted during phase correlation operations. Calculated terms

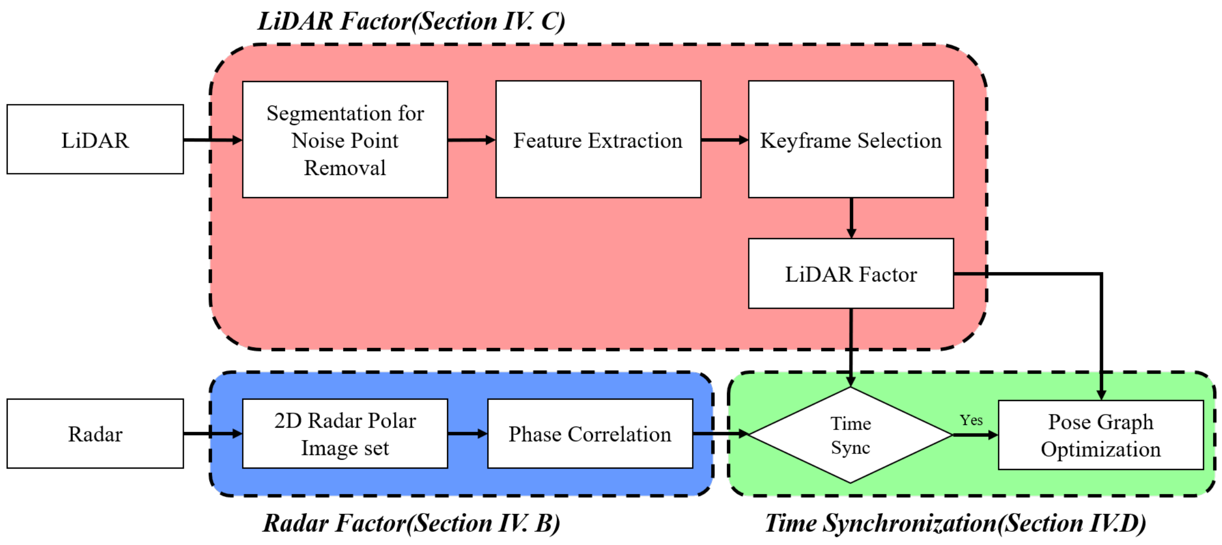


Fig. 2: A block diagram of the proposed method. It is divided into three parts: Radar Factor, LiDAR Factor, and Time Synchronization. The system starts with radar sensor and LiDAR inputs. In the case of radar, utilizing phase correlation, the system calculates the most accurate relative pose between current frame and a frame from the past, but not adjacent frame (see Section III-B). Simultaneously, the system preprocesses the LiDAR point cloud by removing noise points and extracting features in a way similar to LeGO-LOAM and obtaining relative pose between previous and current keyframes by optimizing the poses of each feature point (see Section III-C). The LiDAR factor directly passes to the pose graph. However, it should be confirmed whether radar factor is time synchronized with the LiDAR factor for passing to the pose graph (see Section III-D). Finally, the Keyframe pose can be exploited to construct a 3D map.

are used for translation refinement in full cartesian images. Finally, we receive scanning radar-based ego motion values. We omit the radar keyframing process, since the LiDAR SLAM controls the entire system. Thus, radar factor is given as

$$F^r_{ki} = (t^r_k, t^r_i, x_{ki}, y_{ki}, \theta^{yaw}_{ki}) \tag{1}$$

t^r_i is the time of the current frame index and t^r_k is the time of the previous frame index. x_{ki}, y_{ki}, and θ^{yaw}_{ki} are the ego motion values between two frames, k and i. Since the scanning radar generates a 2D top-down view image, the radar rotation factor only contains yaw information. Generated radar factors are inserted in the LiDAR keyframe to optimize the pose graph.

C. LiDAR Factor

Let $P_t = \{p_1, p_2, \ldots, p_n\}$ be the pointcloud scanned by LiDAR at time t, and let p_i be the individual point of pointcloud P_t. In P_t, there are unnecessary points when the system compares each frame's points. For example, small objects such as leaves may not be found at the same place in consecutive frames, so these points are not reliable. Moreover, points reflected by the ground are not uniform, so they should not be included in pose estimation. P_t must consider only reliable points such as tree trunks. To select certain points $\tilde{P}_t$, two methods are exploited: (1) Ground segmentation, (2) Point clustering. Ground points can be segmented by constructing a range image based on r_i, which is the euclidean distance between sensor and p_i and conducting a column-wise evaluation[27]. After that, non-ground points are clustered. Point clusters are selected unless the cluster has fewer than 30 points.

Planar and edge features at time t, S^p_t and S^e_t are extracted by calculating the roughness in $\tilde{P}_t$. Roughness is a curvature information of each point. If the roughness of points is larger than threshold, it can be segmented as planar features. On the contrary, points which have a roughness smaller than threshold can be segmented as edge features. S_t is defined as frame at time t and includes features $S_t = \{S^p_t, S^e_t\}$. Because we select a way similar to [2], a more detailed procedure for feature extraction can be found in [2].

Computation complexity is too high when feature matching is performed on all frames. Therefore, a frame that satisfies our condition is selected as a keyframe, and frames between keyframes are ignored. The frame is selected as keyframe if it has moved more than τ_d or passed time τ_t from the lastest keyframe. In this paper, we set the $(\tau_d, \tau_t) = (1m, 1s)$. If a frame at time j is selected as keyframe index i, $K_i = S_j$ and K_i represents the current robot's status x_i. In this framework, a frame is selected when no keyframe is generated within $1m$.

After a keyframe is selected, relative pose can be estimated between the current keyframe and adjacent previous keyframes. The transformation between two keyframes is found by performing point-to-edge and point-to-plane matching. The correspondences from F^e_i and F^p_i to F^p_{i+1} and F^e_{i+1} are confirmed by the way in [1] and labels clustering in the segmentation stage.

The Levenberg-Marquardt(L-M) method is applied to find the transformation which minimizes feature distances. Exploiting the point correspondence, distances between each keyframe's feature can be expressed.

$$d_{e_a} = \frac{|(p^e_{i+1,a} - p^e_{i,b}) \times (p^e_{i+1,a} - p^e_{i,c})|}{|p^e_{i,b} - p^e_{i,c}|} \tag{2}$$

$$d_{p_a} = \frac{|(p^p_{i+1,a} - p^p_{i,b}) \cdot \{(p^p_{i,b} - p^p_{i,c}) \times (p^p_{i,b} - p^p_{i,d})\}|}{(p^p_{i,b} - p^p_{i,c}) \times (p^p_{i,b} - p^p_{i,d})} \tag{3}$$

$p_{i+1,a}$ is a feature point in the current keyframe, while $p_{i,b}$, $p_{i,c}$ and $p_{i,d}$ are feature points in the previous keyframe. $[z, \theta_{roll}, \theta_{pitch}]$ can be optimized by forming a line from $p_{i,b}$ and $p_{i,c}$ in (2). On the other hand, $[x, y, \theta^{yaw}]$ can be optimized by representing a plane from $p_{i,b}$, $p_{i,c}$ and and $p_{i,d}$ in (3). After initially performing an optimization to equation (3), (2) is optimized while using $[z, \theta^{roll}, \theta^{pitch}]$ as constraints. Then, the relative transformation which is LiDAR factor (4) can be achieved. The detailed description is found in [2].

$$F^L_{i,i+1} = (t^L_i, t^L_{i+1}, x_{i,i+1}, y_{i,i+1}, z_{i,i+1}, \theta^{roll}_{i,i+1}, \theta^{pitch}_{i,i+1}, \theta^{yaw}_{i,i+1}) \tag{4}$$

D. Time Synchronization

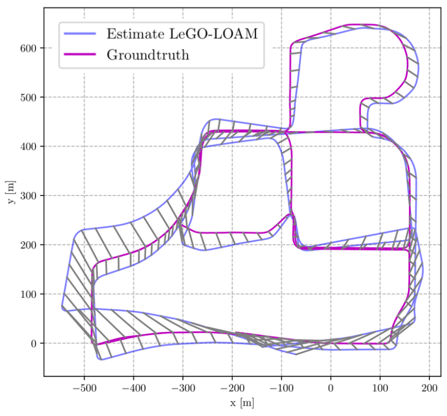
To achieve a more accurate pose in the keyframe, F^L should be concatenated with F^r. If each factor's time is closer, synchronization will be reliable. However, because of the sensor frequency of the scan, factors have time differences with another factor. Therefore, in user-defined thresholds, c_{th} and c_{sum} are set to synchronize F^r with F^L. If c_{th} is smaller than the time difference between two factors, F^r at that time is neglected. We had to set an appropriate c_{th} value. If c_{th} is too small capturing F^r adjacent to F^L is challenging. Moreover, an opposite case becomes worthless for synchronization. Thus, we limited the time difference condition as (5).

$$|t^r_j - t^L_i| < c_{th} \;\; and \;\; |t^r_k - t^L_{i+1}| < c_{th} \; with \; F^r_{jk}, F^L_{i,i+1} \tag{5}$$

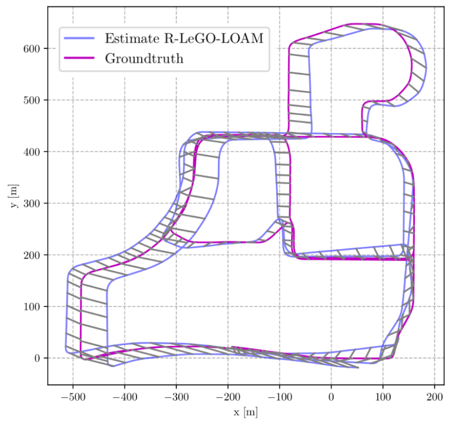
This being so, to alleviate the harsh condition mentioned earlier, we add another condition so that the sum of the difference between the start and end of each factor's time difference does not exceed c_{sum}.

$$|t^r_j - t^L_i| + |t^r_k - t^L_{i+1}| < c_{sum} \; with \; F^r_{jk}, F^L_{i,i+1} \tag{6}$$

Passing conditions (5) and (6), the factor graph receives F^r and optimizes. The features S in K transform to $\tilde{S}$ exploiting the pose from the factor graph, and the final map is composed of $\tilde{S}$.



(a) Lego-Loam



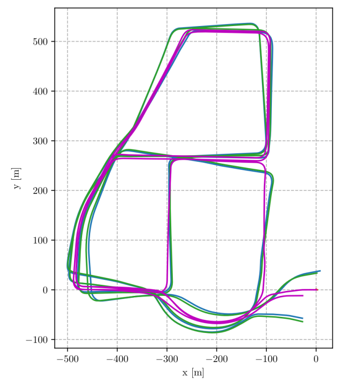
(b) Radar-Lego-Loam

Fig. 3: (a) LeGo-LOAM odometry estimation result with ground truth. (b) Proposed odometry estimation result with ground truth. The proposed method depicts a more reliable odometry result for rotation.

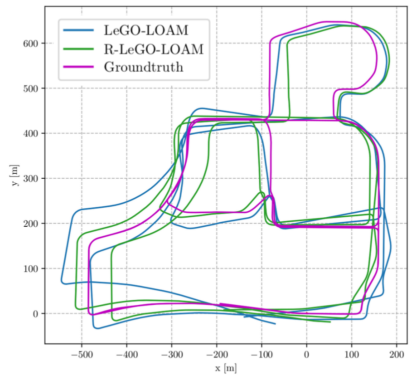
IV. Experiments and Results

A. Datasets

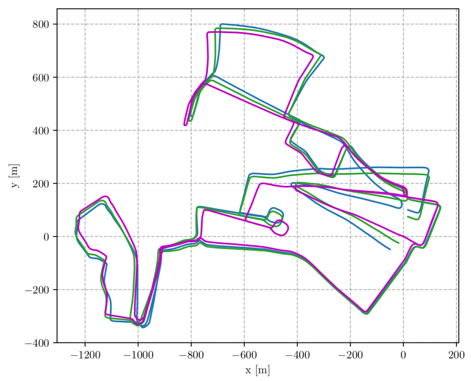
We conducted our odometry estimation method with public radar-LiDAR datasets: Oxford Radar Robotcar[5], and MulRAN[4]. The Oxford radar robotcar dataset is the most popular and widely used dataset for radar navigation research. The dataset contains four point-grey cameras, one GPS receiver, four LiDARs, and one imaging radar. Data is acquired from one location, and the time interval between sequences is short. The MulRan dataset is a long-term driving dataset for radar and LiDAR. Unlike the Oxford dataset, MulRan provides a multiple urban environment, and the time interval between sequences is large. Verification proceeded with five different sequences. The sequences in the Oxford radar dataset are acquired from a single space. However, MulRan data is obtained from multiple places. For verification in a variety of environments, we selected four MulRan dataset sequences (DCC, KAIST, Riverside1, Riverside2), and one Oxford radar dataset sequence. The sample odometry estimation result is depicted in Fig. 3. The



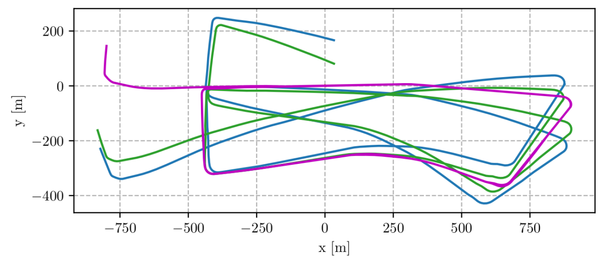
(a) DCC



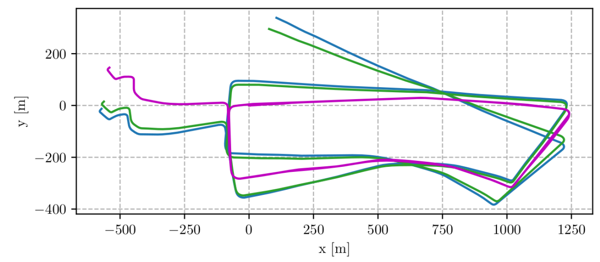
(b) KAIST



(c) Oxford



(d) Riverside1



(e) Riverside2

Fig. 4: All odometry estimations and comparison for each sequence.

trajectory for Fig. 3 is the KAIST region in the MulRan dataset, which contains various road types with curves.

B. Evaluation

Our algorithm was verified by comparing a pose from the SLAM framework LeGO-LOAM[2]. Since our method focuses on rotation correction, a single LiDAR odometry method could depict a perceptible difference. We enhanced the LeGO-LOAM algorithm with a radar rotation factor and time synchronization. By plotting the odometry outputs in one figure, we evaluated the qualitative performance of the proposed method. This will allow us to check how the odometry tendency has improved. For quantitative analysis, we utilized the RPG trajectory evaluation method[28], which calculates the absolute trajectory error (ATE) for each sequence. We obtained both the translation and rotational error with regard to GPS ground truth.

C. Qualitative results

Fig. 4 depicts our odometry results for each sequence. Compared to LeGO-LOAM, the overall result is refined with regard to GPS ground truth. All odometry information is neatly arranged after the radar factor is added. A detailed analysis can be observed in Fig. 5. Although the translation values have some differences from ground truth in small patch, the proceeding rotation consistently shows high accuracy. In particular, the given radar factor enables the maintenance of parallel odometry with ground truth. These small-scale differences in accuracy converge to reduce the overall translation error.

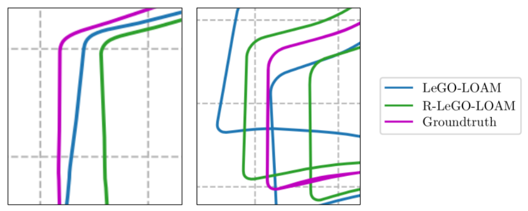


Fig. 5: Partial odometry result. LeGO-LOAM was confirmed that errors occurred in the rotation angle, while the proposed method retains parallelism with the ground truth value.

D. Quantitative results

For quantitative analysis, Table. I and Table. II depict RMSE translation and rotation error. By refining the rotation of the original LiDAR odometry, the shape of radar-enhanced odometry becomes more correlative to the ground truth value. This alteration brings the reduction of the entire translation error, as shown in Table. I. Unlike in the translation error, we can observe marginal improvements in the rotational error. However, as mentioned above, the proposed method is robust for a curvy road, and this enhancement is depicted in the KAIST sequence in Table. II. As the result of the RMSE error, we can observe that small flaws from every moment generate a big difference in the result.

V. Conclusion

We have proposed a methodology for refining radar angular factor-based LiDAR odometry. We estimated the rotation between keyframes with radar image phase correlation and

TABLE I: RMSE translation error w.r.t GPS [m]

	DCC	KAIST	Oxford	River 1	River 2
LeGO-LOAM	25.817	38.356	40.352	152.488	112.116
Proposed	24.870	31.855	28.951	124.497	94.965

TABLE II: RMSE rotation error w.r.t GPS [deg]

	DCC	KAIST	Oxford	River 1	River 2
LeGO-LOAM	7.834	8.672	28.593	15.289	170.058
Proposed	8.26	5.534	28.485	14.174	171.776

tightly coupled with the LiDAR factor. The radar factor and LiDAR factor synchronized with the time threshold, and the factor graph optimization enhanced the resulting odometry. Our result effectively prevented rotation warp, and reduced the translation error indirectly. For future work, experiments should be conducted in a harsh environment to verify our contributions. To evaluate organized accuracy, data has to be acquired in two completely different environments, i.e. a sunny day and a foggy day. Also, we will complement our methodology and quantitative analysis. We published high-cost radar information; however, the system will generate a more precise result when the radar keyframe is added. We will apply our approach to various methods for the estimation of LiDAR odometry.

References

[1] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 2014.

[2] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[3] Yeong Sang Park, Young-Sik Shin, Joowan Kim, and Ayoung Kim. 3d ego-motion estimation using low-cost mmwave radars via radar velocity factor for pose-graph slam. *IEEE Robotics and Automation Letters*, 6(4): 7691–7698, 2021.

[4] Giseop Kim, Yeong-Sang Park, Younghun Cho, Jinyong Jeong, and Ayoung Kim. Mulran: Multimodal range dataset for urban place recognition. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6246–6253, 2020.

[5] Dan Barnes, Matthew Gadd, Paul Murcutt, Paul Newman, and Ingmar Posner. The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset. *arXiv preprint arXiv: 1909.01300*, 2019. URL `https://arxiv.org/pdf/1909.01300`.

[6] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. doi: 10.1109/34.121791.

[7] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[8] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. The trimmed iterative closest point algorithm. In *2002 International Conference on Pattern Recognition*, volume 3, pages 545–548 vol.3, 2002. doi: 10.1109/ICPR.2002.1047997.

[9] Masashi Yokozuka, Kenji Koide, Shuji Oishi, and Atsuhiko Banno. Litamin: Lidar-based tracking and mapping by stabilized icp for geometry approximation with normal distributions. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5143–5150. IEEE, 2020.

[10] Pierre Dellenbach, Jean-Emmanuel Deschaud, Bastien Jacquet, and François Goulette. Ct-icp: Real-time elastic lidar odometry with loop closure. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5580–5586. IEEE, 2022.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[12] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33833-8.

[13] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. pages 2564–2571, 11 2011. doi: 10.1109/ICCV.2011.6126544.

[14] Jens Behley and Cyrill Stachniss. Efficient surfel-based slam using 3d laser range data in urban environments. In *Robotics: Science and Systems*, volume 2018, page 59, 2018.

[15] Chao Qin, Haoyang Ye, Christian E. Pranata, Jun Han, Shuyang Zhang, and Ming Liu. R-lins: A robocentric lidar-inertial state estimator for robust and efficient navigation. 2019.

[16] Haoyang Ye, Yuying Chen, and Ming Liu. Tightly coupled 3d lidar inertial odometry and mapping. *2019 International Conference on Robotics and Automation (ICRA)*, pages 3144–3150, 2019.

[17] Thien-Minh Nguyen, Muqing Cao, Shenghai Yuan, Yang Lyu, Thien Hoang Nguyen, and Lihua Xie. Liro: Tightly coupled lidar-inertia-ranging odometry. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14484–14490, 2021.

[18] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142, 2020.

[19] Tixiao Shan, Brendan Englot, Carlo Ratti, and Daniela Rus. Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5692–5698, 2021.

[20] Martin Holder, Sven Hellwig, and Hermann Winner. Real-time pose graph slam based on radar. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1145–1151. IEEE, 2019.

[21] Sarah H Cen and Paul Newman. Radar-only ego-motion estimation in difficult settings via graph matching. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 298–304. IEEE, 2019.

[22] Ziyang Hong, Yvan Petillot, and Sen Wang. Radarslam: Radar based large-scale slam in all weathers. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5164–5170. IEEE, 2020.

[23] Keenan Burnett, David J Yoon, Angela P Schoellig, and Timothy D Barfoot. Radar odometry combining probabilistic estimation and unsupervised feature learning. In *Robotics: Science and Systems*, 2021.

[24] Dan Barnes, Rob Weston, and Ingmar Posner. Masking by moving: Learning distraction-free radar odometry from pose information. *arXiv preprint arXiv:1909.03752*, 2019.

[25] Daniel Adolfsson, Martin Magnusson, Anas Alhashimi, Achim J Lilienthal, and Henrik Andreasson. Cfear radarodometry–conservative filtering for efficient and accurate radar odometry. *arXiv preprint arXiv:2105.01457*, 2021.

[26] Yeong Sang Park, Young-Sik Shin, and Ayoung Kim. Pharao: Direct radar odometry using phase correlation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2617–2623. IEEE, 2020.

[27] M. Himmelsbach, Felix v. Hundelshausen, and H.-J. Wuensche. Fast segmentation of 3d point clouds for ground vehicles. In *2010 IEEE Intelligent Vehicles Symposium*, pages 560–565, 2010. doi: 10.1109/IVS.2010.5548059.

[28] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.