Automatic Hyper-Parameter Tuning for Black-box LiDAR Odometry

Kenji Koide[1], Masashi Yokozuka[1], Shuji Oishi[1], and Atsuhiko Banno[1]

Abstract— **LiDAR odometry algorithms are complex and involve a number of hyper-parameters. The choice of hyper-parameters can substantively affect the performance of odometry estimation, and it is necessary to carefully fine-tune the hyper-parameters depending on the sensor, environment, and algorithm to achieve the best estimation results. While odometry estimation algorithms are often tuned manually, this is time-consuming and may also result in a sub-optimal parameter set. This paper presents an automatic hyper-parameter tuning approach for LiDAR odometry estimation. By taking advantage of the sequential model-based optimization (SMBO) approach, we automatically optimize the hyper-parameter set of a black-box odometry estimation algorithm without detailed knowledge of the algorithm. In addition, a LiDAR data augmentation approach is also proposed to prevent overfitting. Through evaluation, we show that the combination of SMBO-based parameter exploration and data augmentation enables us to efficiently and robustly optimize the hyper-parameter set for several different odometry estimation algorithms. We also demonstrate that the optimized parameter set exhibits superior performance with respect to KITTI dataset and in a real use scenario.**

I. INTRODUCTION

LiDAR odometry estimation and SLAM are crucial for many functions in intelligent autonomous systems, such as localization, mapping, and navigation. A number of odometry estimation and SLAM algorithms have been proposed [1], [2], [3], and the SLAM community has benefited from these advances. Those algorithms are, however, often surprisingly complex and involve many hyper-parameters. Their performance can depend substantively on the choice of hyper-parameters. It is necessary to carefully fine-tune the hyper-parameters depending on the sensor, environment, and algorithm to achieve the best estimation results. Without appropriate hyper-parameter selection, their accuracy can drastically deteriorate when we try to apply them in a new environment or we use a new sensor model [4].

Despite their importance, hyper-parameters tend to be manually calibrated in odometry estimation algorithms. Manual tuning through trial and error, however, requires a large amount of human effort to maximize the estimation performance for large datasets. Furthermore, it requires a thorough mechanistic understanding of the algorithm to be tuned; insufficient understanding and cursory tuning can lead to a parameter set that results in sub-optimal performance. Although a few studies have proposed automatic hyper-parameter tuning methods for LiDAR and visual odometry estimation methods [5], [6], [7], they are all dedicated to specific algorithms or environments.

[1] All the authors are with the Department of Information Technology and Human Factors, the National Institute of Advanced Industrial Science and Technology, Umezono 1-1-1, Tsukuba, 3050061, Ibaraki, Japan, `k.koide@aist.go.jp`

In the context of machine learning, sequential model-based optimization (SMBO), which optimizes a set of parameters without detailed modeling of the behavior of the system to be tuned, has attracted attention and has been applied to automatic hyper-parameter tuning [8], [9], [10]. The SMBO approach explores the parameter space through an automatic parameter-selection-and-trial loop with a surrogate function of an expensive evaluation function. It has been shown that this approach can efficiently optimize the hyper-parameters of models that have high evaluation cost and involve many hyper-parameters, such as deep neural networks [11].

The purpose of this paper is to put forward an improved approach for hyper-parameter tuning that can be applied to all odometry estimation algorithms. We first propose an automatic hyper-parameter optimization method for odometry estimation algorithms based on the SMBO approach. The proposed approach does not exploit detailed knowledge of the algorithm to be tuned but instead treats the system as a *black-box*. We also propose a data augmentation method for LiDAR sequences to prevent overfitting of the optimized hyper-parameters and find a hyper-parameter set that is robust to environment changes. Through evaluation, we show that the proposed hyper-parameter optimization approach considerably improves the performance of LiDAR odometry algorithms for a publicly available dataset, KITTI [12].

The contribution of this paper is three-fold. First, we propose an automatic and general hyper-parameter tuning approach for odometry estimation algorithms. Second, a data augmentation method to prevent the parameter tuning process from overfitting is proposed. Third, we have released the code, as open source, to automate the odometry tuning process [1].

II. RELATED WORK

SLAM and odometry estimation algorithms are inherently very complex and use various hyper-parameters (e.g., point cloud resolution, feature matching threshold, and keyframe interval). It is known that some popular SLAM frameworks like Google cartographer require fine-tuning many hyper-parameters that affect each other to achieve the best results [13]. This tuning process requires a thorough understanding of its inner behavior and a large amount of trial and error.

Despite the importance of hyper-parameter tuning, only a few studies have considered automatic tuning of SLAM-related algorithms. Nobili *et al.* proposed a tuning method for ICP algorithms to improve the LiDAR localization accuracy for a humanoid robot [6]. This method automatically

[1]`https://github.com/SMRT-AIST/automatic_tuning/tree/devel`

optimizes outlier removal based on the estimated overlap rate between point clouds. Permeleau *et al.* performed an exhaustive parameter-by-parameter exploration to investigate how changes in parameter values affect ICP registration results [5]. Zheng optimized the hyper-parameters for a visual SLAM algorithm (e.g., the number of features, patch size, and edge threshold) using a grid search [7]. There are also several case studies that propose techniques to tune SLAM-related algorithms for specific contexts [14]. However, the above-described methods are all dedicated to specific sensors and task configurations. We argue the necessity of an automatic and general hyper-parameter tuning process that is potentially beneficial to every SLAM-related algorithm.

In the context of machine learning, hyper-parameter tuning has been recognized as an important step to maximize the performance of learning models [15]. Machine learning algorithms commonly use a certain number of hyper-parameters to control their behavior, and their performance can substantively depend on the choice of the hyper-parameter set. In classic algorithms that use only a few parameters (e.g., RBF SVM parameter selection [16]), parameter tuning is often undertaken by an exhaustive grid search or a random search [17]. However, when the number of parameters is large or computation of the evaluation function is expensive, it is difficult to obtain a sufficient solution with these naive methods in a reasonable time. To efficiently optimize many parameters in an expensive model, the SMBO approach has been widely used in recent years [18], [19]. This approach uses a light surrogate function to approximate the costly evaluation function and effectively explore the parameter space with the model response prediction calculated with the surrogate function. Many SMBO algorithms have been proposed, and they are popular in the machine learning domain because automatic parameter tuning confers clear advantages over manual alternatives [8], [9], [10]. In particular, deep learning-based methods, which require tuning many hyper-parameters for a very costly training model, have benefited from SMBO-based automated hyper-parameter tuning [11].

In this work, we apply the SMBO approach to hyper-parameter tuning of odometry estimation algorithms. Although we focus on LiDAR odometry estimation, the proposed approach is general and can also be applied to visual odometry algorithms.

III. METHODOLOGY

A. Problem Definition

Let us assume that we have an odometry estimation function $\mathcal{F}$ that takes $\mathcal{D}$ as input data sequences and $\boldsymbol{x}$ as a set of hyper-parameters, and then outputs estimated sensor trajectories $\mathcal{T}$; $\mathcal{T} = \mathcal{F}(\mathcal{D}, \boldsymbol{x})$. Our goal is to find the parameter set $\tilde{\boldsymbol{x}}$ that minimizes the evaluation function $\mathcal{G}$ for the estimated trajectories:

$$\tilde{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \mathcal{G}\left(\mathcal{F}(\mathcal{D}, \boldsymbol{x})\right). \tag{1}$$

We use the notation $\mathcal{G}(\boldsymbol{x}) = \mathcal{G}\left(\mathcal{F}\left(\mathcal{X}, \boldsymbol{x}\right)\right)$ in the rest of this paper for simplicity.

Algorithm 1 Sequential Model-Based Optimization [18]

```
1: $\mathcal{H} \leftarrow []$
2: for $i \in [1, \cdots, N]$ do
3:     $\boldsymbol{x}^* \leftarrow \arg\max_{\boldsymbol{x}} \mathcal{A}\left(\mathcal{S}\left(\boldsymbol{x}, M_{i-1}\right)\right)$
4:     Evaluate $\mathcal{G}\left(\boldsymbol{x}^*\right)$
5:     $\mathcal{H} \leftarrow \mathcal{H} \cup (\boldsymbol{x}^*, \mathcal{G}(\boldsymbol{x}^*))$
6:     Fit $M_i$ to $\mathcal{H}$
   return $\mathcal{H}$
```

The most naive and common ways to find $\tilde{\boldsymbol{x}}$ are an exhaustive grid search and a random search [17]. They are straightforward to implement and are often used when the number of parameters is small. As the number of parameters increases, however, the required number of evaluations grows exponentially. The evaluation of a parameter set $\mathcal{G}(\boldsymbol{x})$ is quite expensive in LiDAR odometry estimation, and these naive methods cannot sufficiently explore the parameter space and find a good parameter set in a reasonable time.

B. Sequential Model-Based Optimization

SMBO [18] is a technique to efficiently explore the parameter space with fewer evaluations. SMBO uses a lightweight surrogate function $\mathcal{S}(\boldsymbol{x}, M_i)$ that approximates the costly evaluation function $\mathcal{G}\left(\boldsymbol{x}\right)$ based on a model M_i fitted to the evaluation history $\mathcal{H}$. It then explores the parameter space by sampling the parameter set $\boldsymbol{x}^*$ that maximizes an acquisition criterion $\mathcal{A}$ (see Algorithm 1).

The expected improvement (EI) [20] has been widely used as an acquisition criterion. EI is the expectation that $y = \mathcal{G}(\boldsymbol{x})$ negatively exceeds a threshold y^*:

$$EI_{y^*} := \int_{-\infty}^{\infty} \max\left(y^* - y, 0\right) p_{M_i}(y|\boldsymbol{x}) dy. \tag{2}$$

A parameter set sample $\boldsymbol{x}^*$ that maximizes EI_{y^*} can be found using the tree-structured Parzen estimator (TPE); this is popular in automatic hyper-parameter tuning frameworks [9], [10]. TPE models $p(\boldsymbol{x}|y)$ instead of directly modeling $p(y|\boldsymbol{x})$. It defines $p(\boldsymbol{x}|y)$ using two density functions $l(\boldsymbol{x})$ and $g(\boldsymbol{x})$:

$$p(\boldsymbol{x}|y) = \begin{cases} l(\boldsymbol{x}) & \text{if } y < y^* \\ g(\boldsymbol{x}) & \text{otherwise,} \end{cases} \tag{3}$$

where $l(\boldsymbol{x})$ is composed of the evaluation results where $\mathcal{G}(\boldsymbol{x}) < y^*$, and $g(\boldsymbol{x})$ is composed of the other evaluation results. The quantile γ for the evaluation results such that $p(y < y^*) = \gamma$ is chosen as y^*, and Kernel density estimation is used to model $l(\boldsymbol{x})$ and $g(\boldsymbol{x})$.

By substituting Eq. (3) into Eq. (2), we obtain the following equation (see [18] for the detailed derivation):

$$EI_{y^*}(\boldsymbol{x}) \propto \left(\gamma + \frac{g(\boldsymbol{x})}{l(\boldsymbol{x})}(1-\gamma)\right)^{-1}. \tag{4}$$

Eq. 4 indicates that we can obtain $\boldsymbol{x}^*$ by finding $\boldsymbol{x}$ that maximizes $g(\boldsymbol{x})/l(\boldsymbol{x})$. Sampling-based techniques are used to find $\boldsymbol{x}^*$ in the parameter space.

TABLE I: Averaged translational RTEs [%] for KITTI

		Training set							Test set					
Method	Config	00	01	02	03	04	05	Avg.*	06	07	08	09	10	Avg. *
LeGO-LOAM	Baseline	1.77	39.9	6.78	1.50	1.02	1.20	9.90	0.99	1.11	1.75	1.47	1.92	1.53
	Tuned	3.01	3.76	2.39	2.09	1.01	1.65	2.62 (-7.28)	1.30	1.45	2.41	1.93	2.25	2.03 (+0.50)
hdl_graph_slam	Baseline	1.78	17.3	2.45	1.62	2.34	1.38	4.60	1.00	0.84	1.73	2.01	2.54	1.69
	Tuned	1.36	6.88	1.58	1.06	1.18	0.93	2.29 (-2.31)	0.83	1.47	1.64	1.38	3.08	1.61 (-0.07)
SuMa	Baseline	1.12	13.2	1.52	1.55	0.53	1.44	3.37	0.68	0.77	1.72	1.36	2.21	1.45
	Tuned	1.06	10.5	1.40	1.25	0.60	1.13	2.80 (-0.57)	0.66	0.71	1.86	0.98	1.97	1.39 (-0.06)

* Values in parentheses are with respect to the baseline.

TABLE II: Averaged translational RTEs [%] without KITTI 01 and 02

		Training set					Test set					
Method	Configuration	00	03	04	05	Avg. *	06	07	08	09	10	Avg. *
LeGO-LOAM	Baseline	1.77	1.50	1.02	1.20	1.52	0.99	1.11	1.75	1.47	1.92	1.52
	Tuned	1.60	1.51	1.14	0.98	1.37 (-0.15)	1.03	1.07	1.71	5.51	2.10	2.42 (+0.90)
	Tuned w. D.A.	1.80	1.52	1.03	0.99	1.47 (-0.05)	0.99	1.07	1.73	1.46	1.84	1.51 (-0.01)
hdl_graph_slam	Baseline	1.78	1.62	2.34	1.38	1.67	1.00	0.84	1.73	2.01	2.54	1.69
	Tuned	1.37	1.64	1.90	1.05	1.32 (-0.35)	0.98	0.60	1.37	1.61	1.78	1.34 (-0.35)
	Tuned w. D.A.	1.48	1.91	1.87	1.16	1.43 (-0.24)	0.96	0.66	1.51	1.54	1.69	1.37 (-0.32)
SuMa	Baseline	1.12	1.55	0.53	1.55	1.22	0.68	0.77	1.72	1.36	2.21	1.45
	Tuned	1.05	1.24	0.61	1.24	1.08 (-0.17)	0.63	0.72	1.55	0.94	1.70	1.21 (-0.24)
	Tuned w. D.A.	1.13	1.28	0.57	1.14	1.11 (-0.11)	0.67	0.82	1.62	1.30	1.96	1.37 (-0.08)

* Values in parentheses are with respect to the baseline.

SMBO algorithms are available in several hyper parameter optimization frameworks [8], [9], and in this work, we utilize the TPE implementation in *optuna* [10], a hyper parameter optimization library, to find the best hyper-parameter set $\tilde{\boldsymbol{x}}$.

C. Data Augmentation

Although the SMBO approach enables us to efficiently explore the parameter space, it tends to find an aggressive hyper-parameter set to minimize the trajectory errors in the training set and suffer from overfitting. In particular, for odometry estimation, the number of training sequences is much smaller compared to usual machine learning tasks, and overfitting can be a critical problem. Inspired by the recent success for data augmentation in the context of deep learning [21], we propose the following three data augmentation techniques to avoid overfitting:

1) Random range noise: For each point in an input point cloud, we sample a random range of noise in a typical noise range for a 3D LiDAR: $r \sim \mathcal{N}(\mu = 0, \sigma = 0.025)$ [m] and add it to the point.
2) Random transformation: For each input frame, we randomly sample a transformation, where the translation $\|t\| \sim \mathcal{N}(0, 0.05)$ [m] and the rotation angle $\theta \sim \mathcal{N}(0, 0.25)$ [°], and apply it to the point cloud and its inverse to the ground truth pose.
3) Reversing order: We reverse the data order for half of the augmented sequences.

By introducing perturbations with these data augmentation techniques, we aim to let SMBO find a parameter set that can robustly deal with noise and environment changes. We generate four augmented sequences for each input data sequence.

IV. Experiment

A. Evaluation on KITTI

1) Evaluation setting: To show that there is scope to improve the estimation accuracy with hyper-parameter tuning for many odometry estimation methods, we carried out an evaluation on the KITTI dataset [12]. Following [12] and [2], we used relative trajectory errors (RTEs) [22] averaged over sub-trajectories of 100 to 800 m lengths as the metric for evaluation. We utilized *evo* [2] to calculate the RTEs. The sequences 00 to 05 were used for hyper-parameter optimization, while the remaining sequences were used for validation.

We ran LeGO-LOAM [1], hdl_graph_slam [3], and SuMa [2] on the KITTI dataset and evaluated their RTEs. We used manually tuned parameter sets as the baselines. It is worth mentioning that the baseline parameter set of SuMa is fine-tuned for the KITTI dataset in [2]. Then, we optimized the parameter sets for these frameworks with the TPE-based SMBO approach. The list of tuned parameters and their ranges are available on our project page [3]. The number of SMBO evaluation trials was set to 128. For the CPU-based frameworks (LeGO-LOAM and hdl_graph_slam), we ran eight trials in parallel to speed up the optimization process. Parameter optimization was completed in about four hours. However, optimization of SuMa, which utilizes GPU computation, took about 12 hours. This is because its processing speed decreases significantly when multiple trials are run in parallel due to GPU scheduling issues, and we thus had to run a single trial at a time.

2) Optimizing parameters using the entire KITTI: Table I shows the RTEs for the baseline and tuned parameter sets. We can see that the RTEs for all the methods for the training set were significantly improved after fine-tuning (LeGO-LOAM: -7.28%, hdl_graph_slam: -2.31%, SuMa:

[2]`https://github.com/MichaelGrupp/evo`

[3]`https://github.com/SMRT-AIST/automatic_tuning/blob/devel/parameters.md`

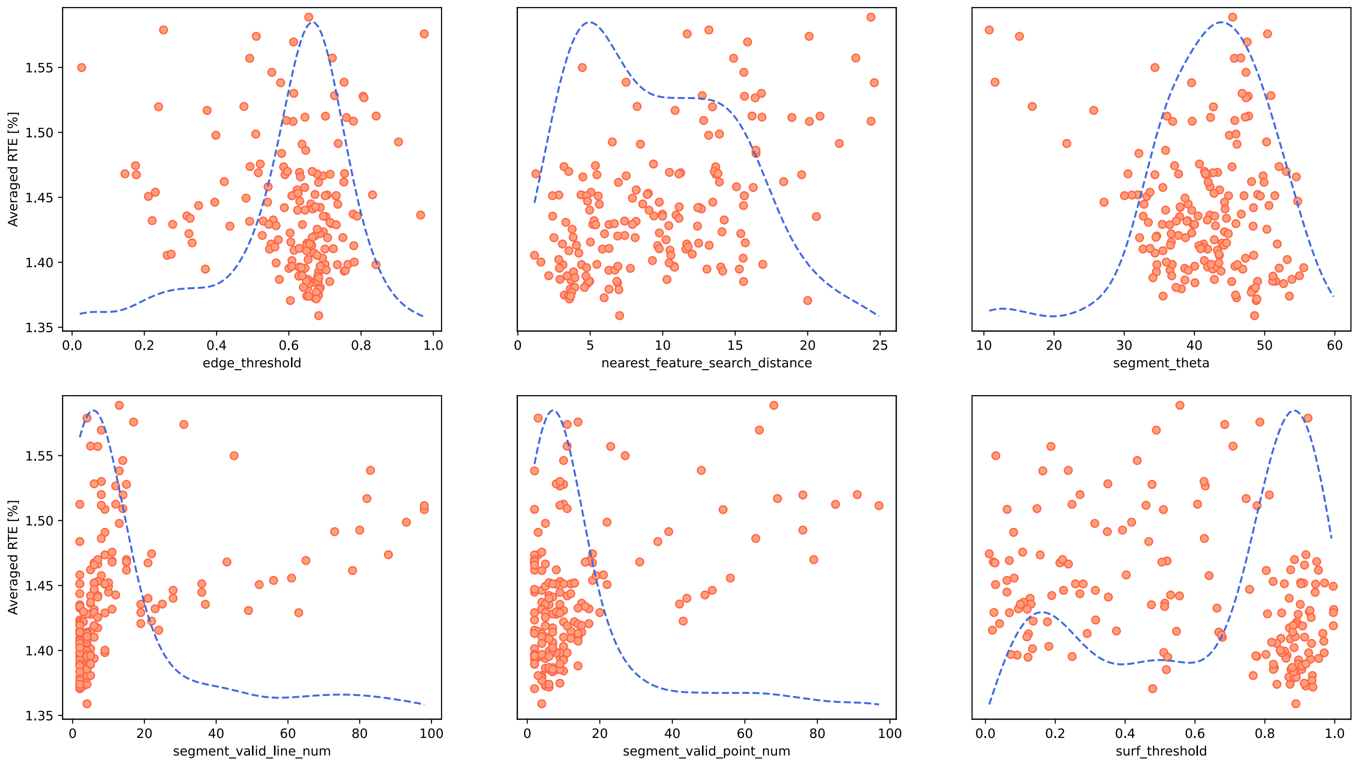


Fig. 1: Parameters for LeGO-LOAM sampled by TPE-based SMBO during parameter optimization. The blue dashed lines indicate the sample densities.

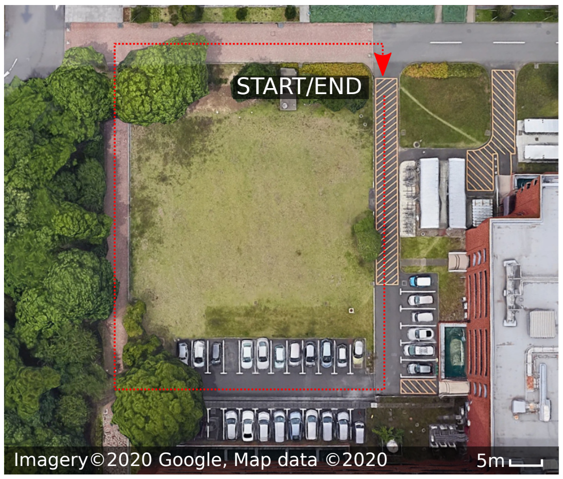
-0.57%). In particular, the RTEs for sequences 01 and 02, which were the most difficult sequences in KITTI, involving forest-like environments with baselines that tend to yield corrupted estimates, were greatly improved. However, for some of the other sequences in the training set and the sequences in the test set, the RTEs deteriorated. This is because, to improve the RTEs for the difficult sequences (01 and 02), the SMBO selected more conservative parameters, which resulted in worse RTEs for the other sequences.

This finding testifies to the importance of the metric used for optimization. Although the SMBO approach enables us to improve the performance of a system for a specific metric, the results do not always reflect intuitive impressions of the *goodness* of the system (Discussed later in Sec V).

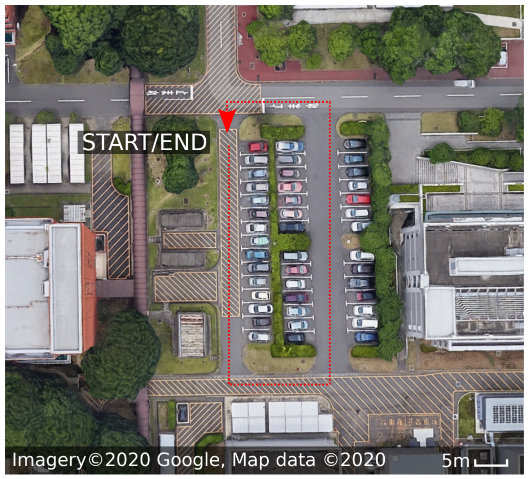
Fig. 1 shows the parameters for LeGO-LOAM sampled during parameter optimization. The blue dashed lines show the sampling densities estimated using kernel density estimation. We can see that the SMBO took more samples from regions where the RTE was expected to be small and efficiently explored the parameter space to find the best parameter set. While some parameters have a clear relationship between the parameter value and the RTE value and appear straightforward to optimize (e.g., *edge_threshold* and *segment_valid_point_num*), some other parameters produce relatively complex RTE distributions (e.g., *nearest_feature_search_distance* and *surf_threshold*) that would be more difficult to optimize manually.

3) Optimizing parameters for urban scenes: To evaluate the peak performance of these frameworks in terms of accuracy in a typical urban environment, we removed sequences 01 and 02, which involve forest-like environments, from the training set and ran the parameter optimization again. The results are shown in Table II. In this case, SMBO chose the parameters that fitted better to the typical urban scenes in KITTI, and the RTEs for hdl_graph_slam and SuMa were greatly improved for both the training and test sets (hdl_graph_slam: $(-0.35\%, -0.35\%)$, SuMa: $(0.17\%, 0.24\%)$ for the training and test sets, respectively). While the RTE for LeGO-LOAM was also improved for the training set (-0.15%), it deteriorated substantively for the test set $(+0.90\%)$. This suggests that an overly aggressive parameter set was selected to improve the RTE for the training set, and resulted in overfitting.

Next we generated four data sequences for each training sequence with the proposed data augmentation techniques and optimized the parameter sets for the augmented training dataset (see Table II). Although the parameter sets optimized with data augmentation have slightly inferior RTEs compared to those optimized without data augmentation, we can see that the proposed data augmentation enables all the methods including LeGO-LOAM to improve the RTEs for both the training and test sets (LeGO-LOAM: $(-0.05\%, -0.01\%)$ hdl_graph_slam: $(-0.24\%, -0.32\%)$, SuMa: $(-0.11\%, -0.08\%)$). This suggests that perturbations injected in the augmented dataset lead SMBO to choose a robust parameter set and help to prevent overfitting.



(a) Environment 1



(b) Environment 2

Fig. 2: Experimental environments.

TABLE III: Odometry drift errors for LeGO-LOAM in real environments

	Training set (Env1)		Test set (Env1)		Test set (Env2)	
Configuration	Trans. [m]	Rot. [°]	Trans. [m]	Rot. [°]	Trans. [m]	Rot. [°]
Baseline	0.104 ± 0.029	**0.027** ± 0.009	0.102 ± 0.006	0.030 ± 0.006	0.170 ± 0.053	**0.015** ± 0.003
Tuned	**0.078** ± 0.035	0.028 ± 0.010	0.214 ± 0.058	0.060 ± 0.015	0.135 ± 0.046	0.022 ± 0.003
Tuned w. D.A.	0.098 ± 0.042	0.033 ± 0.009	**0.092** ± 0.020	**0.021** ± 0.009	**0.083** ± 0.029	0.022 ± 0.008

B. Evaluation in a Real Environment

To show that the proposed approach can be applied to a practical situation, we recorded five LiDAR data sequences with Velodyne VLP16 in environment 1 shown in Fig. 2 and optimized the LeGO-LOAM parameter set. For validation, we recorded two more sets of three sequences in environment 1 and 2, respectively.

To evaluate the odometry drift, we aligned the last frame of each sequence with the first frame using ICP, and then compared the obtained relative pose with the last sensor pose of the estimated sensor trajectory. In this evaluation, we made SMBO minimize the sum of the translation and rotation pose errors: $\mathcal{G}(\boldsymbol{x}) = error_t[\mathrm{m}] + error_r[\mathrm{rad}]$.

Table III shows the evaluation results. Although the estimation accuracy for the training set was greatly improved by hyper-parameter optimization, the accuracy for the test set in environment 1 deteriorated due to overfitting. With the proposed data augmentation, although the errors for the training set were slightly worse compared to the case without data augmentation, we achieved the best results for the test sets. This shows that the proposed combination of SMBO-based hyper-parameter tuning and data augmentation is effective for improving the accuracy of LiDAR odometry while retaining its robustness in practical situations.

V. Discussion

A. Evaluation Metric

As shown in Sec. IV-A, the SMBO approach enables us to improve the performance of odometry estimation algorithms with respect to a specific metric. However, the optimization results do not always reflect the intuitive *goodness* of the system. While the averaged RTE is widely used for odometry evaluation in the literature, it does not represent some important aspects of odometry estimation (e.g., the balance between accuracy and robustness). We believe that it is necessary to create a new metric that reflects other aspects of odometry estimation (e.g., robustness to sensor motion and environment change) to better capture the performance of odometry estimation systems.

B. Data Augmentation

While we proposed basic data augmentation techniques for LiDAR odometry datasets, there is scope to investigate more advanced techniques. When we injected a large amount of noise into the augmented dataset (e.g., translation noise $\|t\| \sim \mathcal{N}(0, 0.25)$ [m] and rotation noise $\theta \sim \mathcal{N}(0, 2.5)$ [°]), that made SMBO choose a very conservative parameter set and resulted in deteriorated RTEs on not only the test set but also the training set. It is desirable to develop data augmentation techniques to safely inject large perturbations without jeopardizing the accuracy of odometry estimations. We consider that techniques used in deep learning (e.g., randomly erasing some parts of input data [23], data generation with adversarial networks [24]) could be a way to safely increase the variety in the training dataset. We also consider that recent simulation-based LiDAR data generation techniques [25] are promising for enlarging the training dataset and striking a balance between the accuracy and robustness of odometry estimation.

C. Optimization Time

Parameter optimization using SuMa [2] for the augmented dataset took about two days. It would require a much longer amount of time if we optimized the hyper-parameters on a large dataset (e.g., a city-scale dataset). As described in Sec. IV-A, we can process each trial of the SMBO in parallel, and this significantly reduces the optimization time. However, with GPU-based algorithms like SuMa, it is difficult to run multiple trials in parallel due to GPU scheduling issues, and thus it is not possible to reduce the optimization time in the same way.

For parameter tuning of deep neural networks, trial pruning based on learning curve evaluation is often used [26]. We consider that a similar approach to terminate unpromising trials by evaluating only a few of the first sequences would be effective for accelerating the entire parameter optimization process.

VI. Conclusion

This paper proposed an automatic hyper-parameter tuning approach for LiDAR odometry estimation algorithms. The proposed approach utilized the TPE-based SMBO technique to efficiently explore the parameter space to maximize an evaluation metric without detailed knowledge of the algorithm to be tuned. The experimental results show that the proposed data augmentation techniques enable us to prevent overfitting during parameter optimization and obtain a parameter set for robust and accurate odometry estimation.

Although we explored the possibility of hyper-parameter tuning to improve odometry estimation performance, as discussed in Sec. V, we consider that there is scope to make the automatic hyper-parameter tuning more efficient and practical (e.g., developing new metric, data augmentation, and trial pruning strategies). Furthermore, we would be able to obtain greater performance gains by adaptively changing the hyper-parameters depending on the circumstances instead of using fixed pre-optimized parameters; we plan to focus on this in future work.

ACKNOWLEDGMENT

*This work was supported in part by a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, oct 2018.

[2] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3d laser range data in urban environments," in *Robotics: Science and Systems*. Robotics: Science and Systems Foundation, jun 2018.

[3] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 172988141984153, mar 2019.

[4] K. Koide, J. Miura, M. Yokozuka, S. Oishi, and A. Banno, "Interactive 3d graph SLAM for map correction," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 40–47, jan 2021.

[5] F. Pomerleau, S. Magnenat, F. Colas, M. Liu, and R. Siegwart, "Tracking a depth camera: Parameter exploration for fast ICP," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2011.

[6] S. Nobili, R. Scona, M. Caravagna, and M. Fallon, "Overlap-based ICP tuning for robust localization of a humanoid robot," in *IEEE International Conference on Robotics and Automation*. IEEE, may 2017.

[7] Z. Zheng, "Feature based monocular visual odometry for autonomous driving and hyperparameter tuning to improve trajectory estimation," *Journal of Physics: Conference Series*, vol. 1453, p. 012067, jan 2020.

[8] F. Madrigal, C. Maurice, and F. Lerasle, "Hyper-parameter optimization tools comparison for multiple object tracking applications," *Machine Vision and Applications*, vol. 30, no. 2, pp. 269–289, oct 2018.

[9] J. Bergstra, D. Yamins, and D. Cox, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Python in Science Conference*. SciPy, 2013.

[10] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna," in *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, jul 2019.

[11] X. Liang, "Image-based post-disaster inspection of reinforced concrete bridge systems using deep learning with bayesian optimization," *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 5, pp. 415–430, dec 2018.

[12] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2012.

[13] "Google cartographer: Tuning methodology." [Online]. Available: https://google-cartographer-ros.readthedocs.io/en/latest/tuning.html

[14] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[15] P. Probst, A.-L. Boulesteix, and B. Bischl, "Tunability: Importance of hyperparameters of machine learning algorithms." *Journal of Machine Learning Research*, vol. 20, no. 53, pp. 1–32, 2019.

[16] S. Han, Cao Qubo, and Han Meng, "Parameter selection in svm with rbf kernel function," in *World Automation Congress*, 2012, pp. 1–4.

[17] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. null, p. 281–305, Feb. 2012.

[18] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., 2011, pp. 2546–2554.

[19] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, jan 2016.

[20] D. Zhan and H. Xing, "Expected improvement for expensive optimization: a review," *Journal of Global Optimization*, vol. 78, no. 3, pp. 507–544, jul 2020.

[21] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, jul 2019.

[22] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, oct 2018.

[23] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 13 001–13 008, apr 2020.

[24] G. Douzas and F. Bacao, "Effective data generation for imbalanced learning using conditional generative adversarial networks," *Expert Systems with Applications*, vol. 91, pp. 464–471, jan 2018.

[25] T. Shimizu, K. Koide, S. Oishi, M. Yokozuka, A. Banno, and M. Shino, "Sensor-independent pedestrian detection for personal mobility vehicles in walking space using dataset generated by simulation," in *IAPR International Conference on Pattern Recognition*, 2020.

[26] A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter, "Learning curve prediction with bayesian neural networks," in *International Conference on Learning Representations*, 2017.