Mask-SLAM: Robust feature-based monocular SLAM by masking using semantic segmentation

Masaya Kaneko Kazuya Iwami Toru Ogawa Toshihiko Yamasaki Kiyoharu Aizawa
The University of Tokyo
{kaneko, iwami, t_ogawa, yamasaki, aizawa}@hal.t.u-tokyo.ac.jp

Abstract

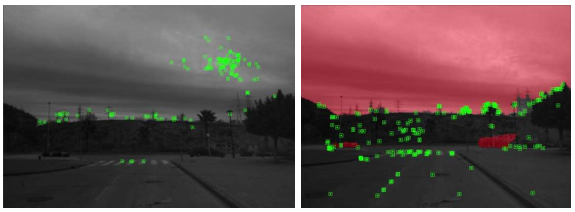
In this paper, we propose a novel method that combines monocular visual simultaneous localization and mapping (vSLAM) and deep-learning-based semantic segmentation. For stable operation, vSLAM requires feature points on static objects. In conventional vSLAM, random sample consensus (RANSAC) [5] is used to select those feature points. However, if a major portion of the view is occupied by moving objects, many feature points become inappropriate and RANSAC does not perform well. Based on our empirical studies, feature points in the sky and on cars often cause errors in vSLAM. We propose a new framework to exclude feature points using a mask produced by semantic segmentation. Excluding feature points in masked areas enables vSLAM to stably estimate camera motion. We apply ORB-SLAM [15] in our framework, which is a state-of-the-art implementation of monocular vSLAM. For our experiments, we created vSLAM evaluation datasets by using the CARLA simulator [3] under various conditions. Compared to state-of-the-art methods, our method can achieve significantly higher accuracy.

1. Introduction

In order for machines to recognize the real world, camera pose estimation is a crucial task that accurately estimates where a machine is located in the real world.

Visual simultaneous localization and mapping (vSLAM) is one of the most promising approaches to localization. It is simple and requires only a single camera to capture image sequences. Using a sequence of images, vSLAM is not only able to perform estimation of camera location and pose, but also reconstruct 3D scenes. Compared to other sensors, such as LiDAR, vSLAM is much less costly and can obtain a larger amount of data regarding surrounding environments. However, vSLAM is not particularly robust.

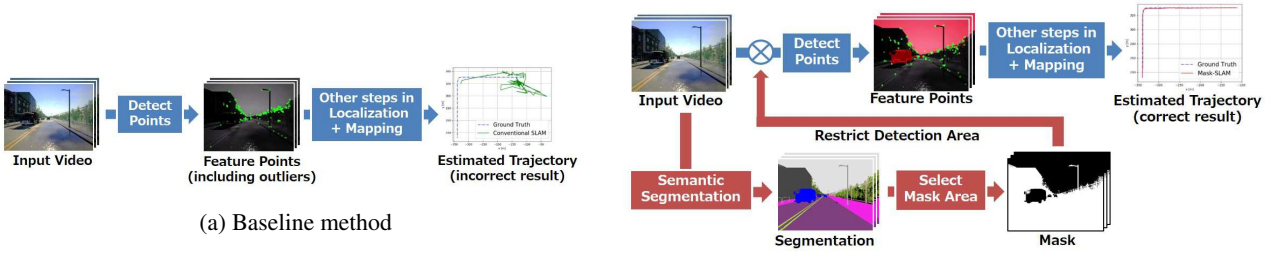
In this paper, we focus on feature-based vSLAM using the ORB-SLAM implementation by Mur-Artal et al. [15], which operates relatively stably among monocular vSLAM implementations [2, 10]. Feature-based vSLAM first extracts many feature points from an image, then compares descriptors of each point between images in a sequence to search for correspondences, and finally estimates the camera pose from those correspondences. Feature-based methods are resistant to image distortion and can perform highly accurate pose estimation for many kind of devices. However, these methods utilize only local information extracted as feature points and are not able to discriminate between feature points of stationary regions in the world coordinate system. If feature points are in moving regions, vSLAM produces estimation errors. In order to select feature points from stationary regions, random sample consensus (RANSAC) [5] is typically used in feature-based vSLAM.



(a) Existing method (b) Proposed method

Figure 1: Difference in feature point extraction between the existing method and proposed methods. By masking regions of feature points, feature points become well distributed in the image without concentrating on regions that are not suitable for vSLAM.

RANSAC is used to select the most reliable value from many hypotheses. Hypotheses are computed from random samples of a large number of correspondence pairs. vSLAM should exclude feature points in moving areas, which likely contain incorrect pairs and can lead to errors. However, consider a case where most of the correspondence pairs are incorrect. RANSAC is unable to select exclusively correct pairs and vSLAM is unable to solve the problem correctly. In order to handle these situations that are unsuitable for RANSAC, we use a mask obtained through semantic segmentation based on deep neural networks (DNNs) [13].



(a) Baseline method

(b) Proposed method

Figure 2: Difference in structure between baseline method and proposed method. In the proposed method, by masking based on semantic segmentation result in the form of preliminary knowledge, we restrict the regions to extract feature points. This makes it possible to reduce the number of incorrect matches among the correspondences selected by RANSAC [5] and improve the overall performance of vSLAM.

Semantic segmentation accounts for global information in an image and is complementary to feature-based vSLAM, which utilizes only local information in an image.

This method facilitates flexible and highly accurate image recognition that considers not only local, but also global features of an image. Feature-based vSLAM using only local information in an image implicitly assumes static areas, which are strongly dependent on circumstances. However, using semantic segmentation, which is one of the most active research topics in deep learning, can remedy the issues in feature-based vSLAM. This is accomplished by removing feature points in inappropriate regions using masks produced by semantic segmentation. The masks ensure that hypotheses do not concentrate on inappropriate regions, such as moving objects.

We focus on videos captured from a moving vehicle in our experiments. In our preliminary experiments, areas whose feature points were incorrect often included pedestrians, vehicles, and the sky. In the most famous benchmark dataset, called KITTI [7], conditions are too optimal and there are very few inappropriate areas. In the real world, there is extreme diversity in weather, moving objects (cars), pedestrians, etc. Therefore, we decided to use a simulated environment generated by the CARLA simulator [3] to evaluate our proposal. With the CARLA simulator, one can freely move cars and change weather, and it's possible to create as much data as needed. We evaluated the effectiveness of the proposed method using a large amount of dataset obtained in a simulated environment as close to the real world as possible.

The main contributions of this paper are as follows:

- We propose a method that can easily improve the performance of monocular vSLAM by introducing deep-learning-based semantic segmentation. Shortcomings in the local information of vSLAM can be addressed by introducing global information from semantic segmentation.
- The performance achieved by the proposed method is nearly the same as the performance achieved when traditional vSLAM operates ideally.
- We make use of a driving simulator to produce a number of environments that are not available in existing datasets. Based on experiments using the simulated video, the proposed method achieves significantly better performance than the existing method.

2. Related work

There have been several attempts at solving the problem of connecting deep learning with vSLAM or accomplishing the function of conventional SLAM using deep learning. One of the most famous approaches is CNN-SLAM [17], which uses depth values inferred by deep learning as an initial solution for SLAM estimation. Recently, unsupervised approaches have been proposed to estimate depth [6] and egomotion [19], and perform 3D reconstruction [18]. These methods can learn only from tag-less (raw) movies and their accuracy is poor when compared to existing vSLAM methods. We thought that a part of these method can help improve existing vSLAM method. In particular the authors of [19], introduced an "explainability mask," which is a mask for image regions areas that do not fit mainstream motion-estimation techniques. However, their method is not able to exclude regions of moving objects that occupy the view because their training data did not contain such images. It is necessary to have preliminary knowledge regarding which parts in an image tend to move and which parts are stationary.

In our proposal, we make use of semantic segmentation to improve vSLAM. Semantic segmentation is a method that divides an image into regions of semantic categories, such as people, cars, sky, etc. It is reasonable to combine vSLAM and semantic segmentation because the former relies on local information and the latter produces regional in-

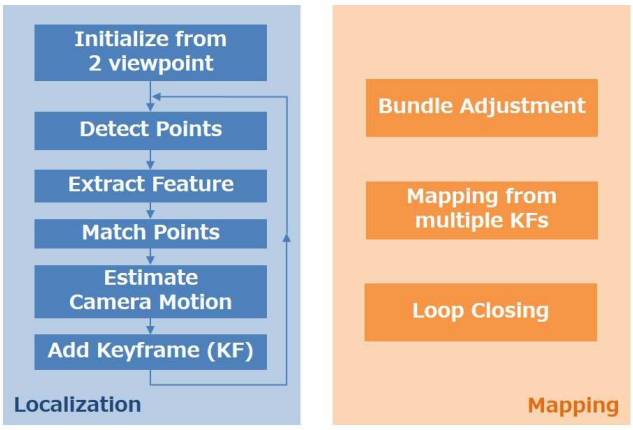


Figure 3: General structure of vSLAM. vSLAM utilizes a tracking thread that constantly estimates camera pose and a mapping thread that stores the tracked feature points as a 3D map.

formation. These two types of information are complementary and their proper combination can improve the accuracy of vSLAM.

3. Proposed method : Mask-SLAM

This section describes the proposed Mask-SLAM method. Our model consists of constructing a mask from an image using semantic segmentation and incorporating that information into the existing vSLAM pipeline. This pipeline is ilustrated in Fig. 2.

In the following sections, we describe the details of the vSLAM pipeline and semantic segmentation.

3.1. Visual SLAM

Feature-based vSLAM typically consists of "localization, that is tracking for camera pose estimation" and "mapping for reconstructing the surrounding 3D environment," where these two processes are executed simultaneously. The detailed algorithm is presented in Fig. 3.

vSLAM extracts feature points from an image, obtains corresponding pairs by comparing the descriptors of each point, and estimates camera motion from the correspondences. ORB-SLAM [15], which is a state-of-the-art implementation of vSLAM, uses ORB feature points [16] that can be extracted at high speed and compares these ORB features to obtain correspondence points. ORB-SLAM utilizes the strategy of obtaining a large number of correspondence pairs and selecting reliable pairs from among them. RANSAC [5] is the algorithm used for selecting the most reliable correspondences.

RANSAC sequentially derives mathematical parameters from data, including outliers. The RANSAC algorithm operates as follows:

1. Randomly extract a sufficient number of samples from the data.

2. Estimate a set of parameters to fit these samples. This set of parameters is called hypothesis.

3. Apply the obtained hypotheses to all data other than the extracted samples and compute the distances between the estimated parameters for each data sample.

4. Consider samples with small distances as inliers and let the number of inliers represent the correctness of a hypothesis.

5. The above operation is performed a number of times and the hypothesis with the largest number of inliers is adopted, while outlier data is excluded.

By implementing RANSAC as described above, one can obtain a correct estimation result that is not influenced by outliers. In general, in vSLAM, RANSAC can find the most correct pair from a large number of correspondences and can derive an accurate camera pose.

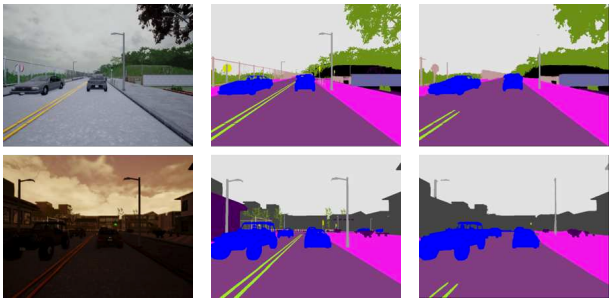
However, there is a limit to the usefulness of this method. vSLAM requires features from stationary objects for RANSAC to be able to select reliable correspondences. When the entire view is occupied by a moving object, RANSAC cannot select reliable correspondences. In vSLAM, RANSAC is executed on every frame of input video for estimation of the camera pose. If an area that is inappropriate for estimating camera pose appears for an extended duration, the probability of continuous detection of inappropriate correspondence points as inliers is greatly increased. This situation occurs frequently when utilizing vSLAM outdoors.

Therefore, we propose the use of semantic segmentation to compensate for the deficiencies of RANSAC. Semantic segmentation is used to produce a mask to exclude regions where correct correspondences are unlikely to be found. Specifically, in the general vSLAM pipeline, at the stage of detecting the feature points, the operation "do not detect feature points in masked area" is added. By simply adding this operation, it is possible to exclude most of the inaccurate correspondences obtained, which significantly reduces RANSAC error.

3.2. Semantic Segmentation

Semantic segmentation is the problem of assigning an object class to each pixel. VOC2012 [4] and MSCOCO [14] are representative datasets for semantic segmentaion. We define objects to be masked as follows:

- Cars: moving objects that are not suitable for feature points.



(a) Original (b) GT (c) Result

Figure 4: Example output results of DeepLab v2[1]. Original image(4a), CARLA[3]'s GT(4b), DeepLab result(4c)

- Sky: this object is too far from the camera view, meaning it is difficult to estimate an accurate 3D position for this object.

Areas where the feature points should be detected from an image are excluded in the mask. We used DeepLab v2 [1], which can perform high-precision semantic segmentation, for this study. DeepLab v2 utilizes a network with the following structure:

1. For an input image, deep convolutional neural networks output an object existence probability heat map.

2. Bi-linear interpolation is performed on the probability heat map.

3. The final region segmentation results are outputted based on boundary refinement using a conditional random field (CRF) [11].

DeepLab v2 implements an atrous convolution, which can reduce computation cost and perform convolution operations without reducing resolution. By inserting zeros into the gaps between the pixels of the convolution filter and enlarging it, the output result maintains a high resolution. Additionally, in order to obtain information regarding objects of various sizes in the image, performance can be improved by combining spatial pyramid pooling [12, 8], which performs convolution in parallel on multiple scales, with atrous convolution (atrous spatial pyramid pooling (ASPP)). In Deeplab v2, by incorporating ASPP into an existing network (VGGNet or ResNet), the creators were able to derive a highly accurate probability heat map and refine the boundaries using a CRF [11].

In our experiment, we trained a DeepLab v2 based on ResNet-101 from scratch using 30,000 images (800×600 pixels) created by the CARLA driving simulator CARLA [3] (see Section 4.1). We used a single Tesla K80 GPU for training the network. We constructed the training data based on the experimental settings in Section 4. The dataset is comprised of images captured from a moving car and the

Figure 5: An example of the visualization of a sequence trajectory in our datasets (Town01, WetSunset)

semantic segmentation ground truth (GT) results (13 types of labels). The semantic segmentation GT data originally had 12 types of labels ("None," "Buildings," "Fences," "Other," "Pedestrians," "Pole," "Roadlines," "Roads," "Sidewalks," "Vehicles," "Walls," and "Trafficlights"), but we added a "Sky" label. We created the "Sky" label as follows. We changed the labels for "None" to "Sky" for the pixels labeled "None" in the GT of semantic segmentation with the deepest depth values ($1,000[m]$). The results from DeepLab v2 are presented in Fig. 4.

"Cars" (moving objects) and "Sky" (distant area) are objects that we empirically found to disturb the operation of vSLAM. We outputted these two label areas as a mask and implemented a process where feature points would not be detected from these masked areas.

4. Experiments

4.1. Datasets

In order to demonstrate the effectiveness of the proposed method, we used the CARLA driving simulator [3], which can simulate various environments. Existing benchmarks, such as KITTI, consider only limited circumstances where RANSAC is unlikely to fail. However, because various environments appear in the real world, it is not sufficient to evaluate a model using only the KITTI benchmarks. In this experiment, we simulated various weather conditions and environments with moving objects using CARLA to determine if the proposed method is effective under various conditions. CARLA can change weather conditions flexibly and can move cars automatically. CARLA is a driving simulator for automatic operation developed by Desovitskiy et al. We acquired images from a moving car in two towns (Town01, Town02) under 15 types of weather conditions

Table 1: Distribution of weather conditions in our datasets. We randomly selected 11 weather conditions and excluded four weather conditions that are difficult for vSLAM to handle (MidRainyNoon, HardRainNoon, MidRainSunset, HardRainSunset).

Weather	Town01	Town02
Default	3	2
ClearNoon	3	5
CloudyNoon	4	4
WetNoon	8	6
WetCloudyNoon	1	9
MidRainyNoon	0	0
HardRainNoon	0	0
SoftRainNoon	3	2
ClearSunset	6	3
CloudySunset	4	2
WetSunset	7	7
WetCloudySunset	3	4
MidRainSunset	0	0
HardRainSunset	0	0
SoftRainSunset	8	6
Sum	50	50

Table 2: Experimental results of two evaluation metrics for our datasets. We found that both metrics were improved by our method.

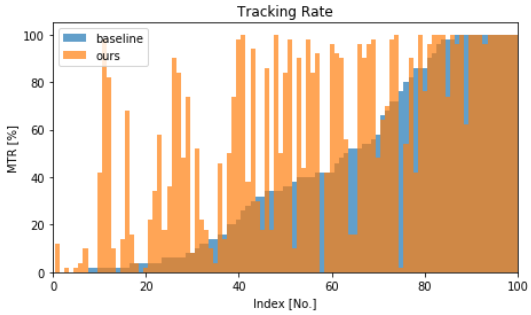
	MTR [%]	MTE [m]
Baseline[15]	42.4	14.9
Ours	**58.2**	**13.7**

(CloudySunset, SoftRain, Clearnoon, etc). Additionally, we acquired GT sensor data, such as GPS, LiDAR, semantic and segmentation, which are not available in the real world.

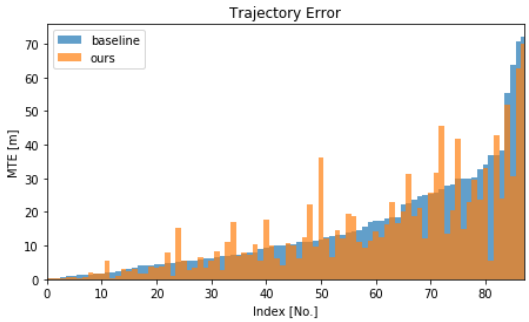
For this experiment, we created datasets from 50 runs in each of the two towns (total of 100 runs). Each run contains 1,000 images (800 $\times$ 600 pixels) captured at 15 fps. We refer to the images (video) obtained in a run as a sequence. The car was automatically driven using the autopilot mode with a travel distance of 100-500 m for each sequence. We also randomly chose weather conditions from 11 options. We excluded four weather conditions under which ORB-SLAM [15] cannot operate. The data distribution is listed in Table 1. Additionally, an example of the visualization of a sequence trajectory is presented in Fig. 5.

4.2. Evaluation metrics

In this section, we describe our evaluation metrics. We estimated a trajectory 50 times for each baseline method [15] and for our method for each sequence. We defined two evaluation metrics for computing average values for each data type.



(a) Tracking rate



(b) Trajectory error

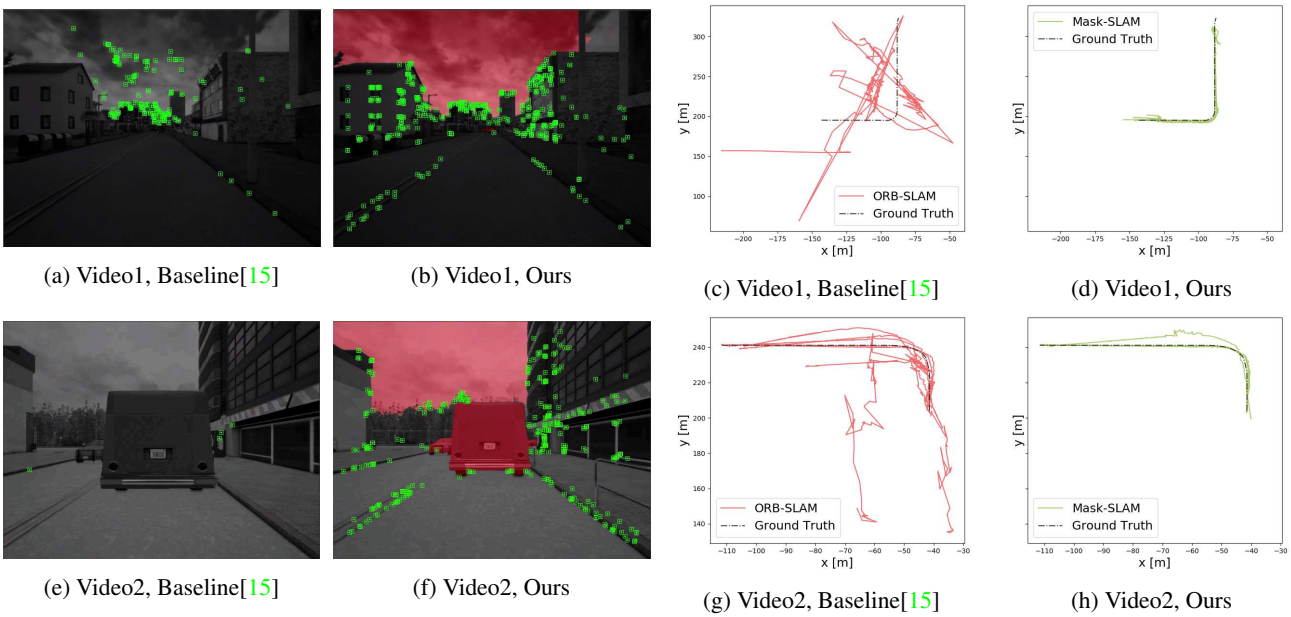
Figure 6: Results of each evaluation metric for our datasets. We sorted the data in ascending order of values relative to the baseline method for visibility. Superior results were obtained by our method.

- The evaluation value for a "sequence" is defined as "the average value obtained by repeatedly deriving an estimate of 50 trajectories for a sequence."
- The evaluation value for the "entire sequence" is defined as "the average value of the evaluation values for each sequence."

When examining the performance of vSLAM, it is common to measure the distance errors between the trajectory estimated by vSLAM and the GT. However, in this experiment, we focus on the overall improvement of vSLAM. We define improvement as the proposed vSLAM operating without problems when the original vSLAM loses its position and halts its operation. We observed several cases in which original vSLAM halted before completion because of errors caused by cars or the sky.

Taking these ideas into consideration, we used the following two kinds of evaluation metrics:

- **Mean Tracking Rate (MTR)**: This metric indicates whether or not vSLAM tracks without losing its position or halting operation on a per-sequence basis. If tracking is successful, vSLAM outputs the estimation



(a) Video1, Baseline[15] (b) Video1, Ours (c) Video1, Baseline[15] (d) Video1, Ours

(e) Video2, Baseline[15] (f) Video2, Ours (g) Video2, Baseline[15] (h) Video2, Ours

Figure 7: Example results for the datasets. Video1 is an example result that was improved by excluding the sky area. Video2 is an example result that was improved by excluding moving objects. There are only a few feature points to be referred to by existing SLAM (e). Immediately after this, existing SLAM stops its operation.

	video 1		video 2	
	MTR[%]	MTE[m]	MTR[%]	MTE[m]
Baseline[15]	20.0	5.0	4.0	36.8
Ours	**74.0**	**1.0**	**22.0**	**5.5**

Table 3: Comparison of average values of results for Video 1 and Video 2

result for the camera motion in each frame. We defined a "Failed Tracking" as one that failed to obtain an estimation result for more than 80 [%] of the 1,000 frames in a sequence. We output the success rate over 50 trials for a single sequence as the "Tracking Rate [%]." Conversely, we defined a "Successful Tracking" as one that succeeded in tracking more than 80 [%] of the 1,000 frames. If improvement in an evaluation value is observed, this indicates that the performance of vSLAM is improved. Specifically, we formulated the "Mean Tracking Rate [%]" for all trials in a sequence as follows:

$$MTR = \frac{1}{m}\sum_{i=1}^{m=50}(TrackingRate_i) \qquad (1)$$

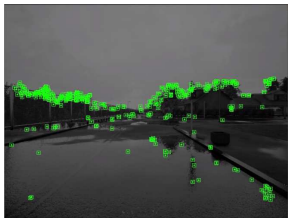
For each sequence, $m(=50)$ results ("Tracking Rate [%]") from $i = 1, ..., m = 50$ are obtained and their average value is the "Mean Tracking Rate (MTR)."

- **Mean Trajectory Error (MTE)**: The goal of general vSLAM is to "estimate of the locus of the camera close to the GT." As an evaluation of proximity, we output the distance error for each time step and the average value of a sequence as "Trajectory Error $[m]$". Because we are using monocular vSLAM in our implementation, we calculate the error after adjusting the scale to the GT according to Horn's method [9]. However, it should be noted that we calculate the error only for "Success Tracking" ("Tracking Rate" exceeds 80 [%]). This is because if the "Tracking Rate" is low, one can set the error to zero when adjusting the scale based on Horn's method. However, this is not a proper evaluation. We formulated the "Mean Trajectory Error (MTE) $[m]$" for all trials in a sequence as follows:

$$MTE = \frac{1}{m}\sum_{i=1}^{m}(\frac{1}{n_i}\sum_{t=1}^{n_i}\|X_t - Y_{it}\|_2) \qquad (2)$$

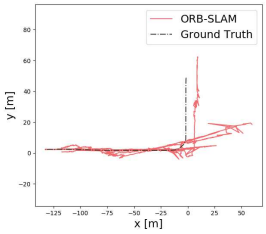
For each sequence, $m(\leq 50)$ results ("Tracking Error $[m]$") of $i = 1, ..., m = 50$ are obtained ($m(\leq 50)$ is the number of "Successful Tracking" trials). It is necessary to compare the trajectories at each time step t. The "Tracking Error" for a trial (i) is calculated by averaging the error between the GT 3D position X_t and 3D position Y_{it} in the estimated trajectory (scaled according to Horn's method [9]) over the entire time series $(t = 1, ..., n_i)$.

In Section 4.3, we compare the results of baseline vSLAM

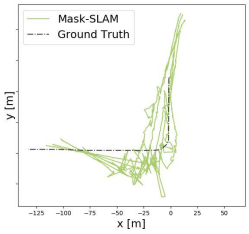


(a) Video3, Baseline[15]

(b) Video3, Ours



(c) Video3, Baseline[15]



(d) Video3, Ours

Figure 8: A failure example for our method (Video 3). Restricting detection areas in our method makes it easier for feature points to be found in puddle areas, which reduces accuracy.

[15] and our method for these two evaluation metrics.

4.3. Results

In this section, we present the results of our experiment. The results of evaluation ("Mean Tracking Rate" and "Mean Trajectory Error") for all datasets are listed in Table 2. One can see that improvement was observed in both metrics when using our method. Furthermore, in order to visualize the results in detail, we illustrated the results for each sequence in Fig. 6.

We determine that vSLAM's performance has improved when the "Tracking Rate" value increases or the "Trajectory Error" value decreases. As one can see in Fig. 6, we found that both evaluation metrics improved for all sequences when using the proposed method.

The degree of improvement in "Mean Trajectory Error (MTE)" may appear small in Table 2. However, as described in Section 4.2, we only used data from "Successful Tracking" trials (whose "Tracking Rate" is 80[%] or more) for the calculation of "Mean Trajectory Error (MTE)." Because the baseline "Tracking Rate" is low, we calculated the "Trajectory Error" using a small number of data samples (trials). In contrast, because our method's "Tracking Rate" is high, we calculated our method's "Trajectory Error" using a large number of trials. The error differences between the baseline and our method are small because the trials were thresholded and only successful trials were considered.

In order to analyze the details of how our method contributes to improvement, we refer to certain examples in detail. We focus on two results out of 100 sequences comprising datasets. We refer to these results as Video 1 and Video 2. The results are presented in Fig. 7 and Table 3. Fig. 7 presents feature extraction for one frame by each method and the estimated trajectories of each sequence. Regarding the estimated trajectories, the top five tracks based on "Tracking Rate" are drawn overlapping.

	video 3	
	MTR [%]	MTE [m]
Baseline[15]	**14.0**	**7.1**
Ours	10.0	16.8

Table 4: Comparison of average values of results for Video 3

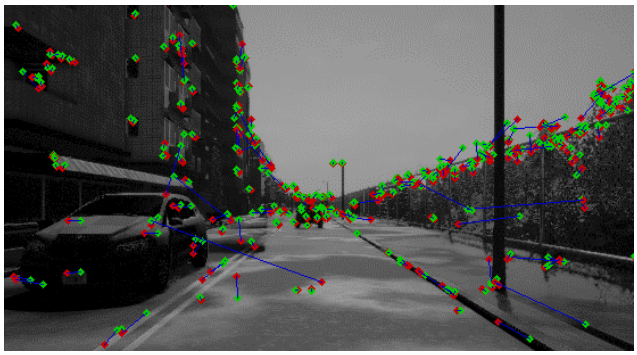


Figure 9: The limitations of ORB-SLAM [15]. Red points are the extracted feature points in this frame. Green points are the re-projected points matching the red points from the last frame. The blue lines connect pairs of points. If a red point and green point overlap (close), it can be said that correct correspondence was obtained between those two points. However, such correspondences were very rare.

It is confirmed that the results are improved by our method compared to the baseline method. With respect to Video 1, the concentration on the sky region was suppressed by restricting the feature point detection area using the mask. As a result, feature points were extracted from the entire view, and the estimation result became stable. Regarding Video 2, baseline vSLAM halted and lost its position because car covered the entire view. However, in the proposed method, by successfully excluding the detection of feature points in car regions using the mask, tracking of feature points succeeded without halting the operation of our method. When comparing the results of trajectories in each video, the baseline method outputs trajectories that deviate from the GT, but the proposed method outputs relatively accurate trajectories overlapping the GT. As a result, both the "Tracking Rate" and "Trajectory Error" are improved.

The proposed method does not always produce good results when using all the data. Video 3 is a failure example.

This is clearly shown in Fig. 8 and Table 4. The weather in Video 3 is WetCloudy Sunset and there are puddles on the road. Tracking feature points of the contours of buildings mirrored on the surfaces of puddles caused degradation of the estimation accuracy of vSLAM. In our method, by excluding cars and the sky using the mask, feature points were more likely to be detected in the puddle area compared to the baseline method. As a result, both "Tracking Rate" and "Trajectory Error" were degraded.

We also analyzed the limited performance enhancement when the ORB-SLAM [15] runs in an ideal outdoor environment. For this, we used the GT semantic segmentation obtained from CARLA instead of the output of semantic segmentation process as the mask for Mask-SLAM. We label this method as "ours (oracle)." Our method (oracle) represents the performance limit of ORB-SLAM, but there are still many errors. The MTR was 58.6 and the MTE was 12.0. One can see from the result 9 of visualizing the correspondences between ORB feature points in ORB-SLAM that a considerable number of correspondences are incorrect. To further improve the performance, we must study features with better performance than ORB features [16].

5. Conclusion

We proposed a novel approach to combining feature-based vSLAM and semantic segmentation. Semantic segmentation outputted the results of classifying objects into semantic areas within an image and we generated a mask from the semantic areas that were inappropriate for vSLAM. Our vSLAM method only extracts feature points from regions excluded by the mask and can select only reliable feature points. This enables vSLAM to work accurately and stably without losing its position.

We clarified the effectiveness of our method using simulated data that is as close as possible to a real environment. Our method achieved superior results when compared to baseline vSLAM in terms of two evaluation metrics that evaluate the performance of vSLAM.

As a future research direction, it is necessary to implement an algorithm to automatically determine the mask area in order to cope with situations similar to the situation in Video 3. Additionally, the introduction of semantic segmentation is too time consuming at this stage, meaning our method cannot operate in real time. It is necessary to design a high-speed semantic segmentation algorithm for adoption into vSLAM.

References

[1] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *arXiv:1606.00915*, 2016. 4

[2] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007. 1

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1, 2, 4

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. 3

[5] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, June 1981. 1, 2, 3

[6] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016. 2

[7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2

[8] K. He, X. Zhang, S. Ren, and J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In *european conference on computer vision*, pages 346–361. Springer, 2014. 4

[9] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987. 6

[10] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007. 1

[11] P. Krähenbühl and V. Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *Advances in neural information processing systems*, pages 109–117, 2011. 4

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2169–2178. IEEE, 2006. 4

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 3

[15] Mur-Artal, Raúl, Montiel, J. M. M., Tardós, and J. D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 1, 3, 5, 6, 7, 8

[16] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011. 3, 8

[17] K. Tateno, F. Tombari, I. Laina, and N. Navab. CNN-SLAM: real-time dense monocular SLAM with learned depth prediction. *CoRR*, abs/1704.03489, 2017. 2

[18] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. SfM-Net: Learning of Structure and Motion from Video. *arXiv preprint arXiv:1704.07804*, 2017. 2

[19] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised Learning of Depth and Ego-Motion from Video. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2