An Evaluation of the Pseudo RGB-D SLAM framework based on AdaBins and ORB-SLAM2

Yang-En Lu[*], Tai-Chun Shih[*], Chin-Yang Lin[*], Ching-Ting Chia[*], Ting-Jun Wu[*]
[*] NCKU, Department of Geomatics., Tainan City, Taiwan
luyangen124@gmail.com
kevin89428@gmail.com
linjohn0903@gmail.com
ting20314@gmail.com
easy3123@gmail.com

***Abstract*—One of the most popular research areas is cost-effective navigation and positioning systems for autonomous cars. Determining the position of a vehicle within a lane is crucial to achieve a high level of automation. Vehicle navigation and positioning in open-sky scenarios relied heavily on the Global Navigation Satellite System (GNSS) service. Nonetheless, GNSS signals were slightly degraded due to various environmental situations such as urban canyons caused by multipath effects and non-line-of-sight (NLOS) issues. To provide robust performance in complex scenarios, sensor fusion is the most common solution. The recently emerging deep learning approach to monocular depth prediction plays an important role to estimate the absolute or relative depth of the environment without an additional depth sensor. The following paper presents a pseudo-RGB-D-SLAM framework to improve the lack of scaling factors for monocular cameras. In particular, using a deep learning approach to generate a pseudo-depth sensor can not only reduce hardware costs, but also improve traditional monocular SLAM disadvantages. The results show that the proposed framework can be used to estimate general 3D motion in an indoor environment and correct the unknown scale factor of Monocular Visual SLAM in a real-world setting.**

I. Introduction

According to the Boston Consulting Group (BCG) forecast, the global self-driving vehicle market will reach $42 billion in 2025, and autonomous vehicles will account for 12.4% of the total vehicle market. Therefore, it can be seen that mapping and navigation technology have a certain market. At present, GNSS has been highly relied on to obtain positioning and navigation services outdoors, but the positioning system based on GNSS has caused positioning errors due to the multipath effect caused by urban canyon, and signal cycle slips. To improve the robustness of positioning, the rapid development of multi-sensor integrated positioning systems not only improves positioning accuracy but also gradually reduces the cost of mapping.

To overcome the limitations of complex scenarios, multi-sensor platforms such as Inertial Navigation System (INS), Global Navigation Satellite System (GNSS), cameras, Light Detection and Ranging (LiDAR). Among them, camera sensors were inexpensive and widely studied in the fields of robotics and navigation. Camera sensors can provide vehicles relative pose change with Visual Odometry (VO) and Simultaneous Localization and Mapping (SLAM) techniques. In the simplest Visual Simultaneous Localization and Mapping (V-SLAM) systems, a monocular camera is used to determine the ego-motion and build a map without true scale. For example, ORB-SLAM [1] is a feature-based monocular SLAM system that operates in small and large, indoor and outdoor environments. The system allows wide baseline loop closing and relocalization and includes full automatic initialization which performs a real-time and robust trajectory estimation. However, several drawbacks were noticed in different VO or VSLAM algorithms. Such as Monocular SLAM lacks the scale factor. This issue causes so-called scale drift in both camera trajectory and 3D scene depth, thus reducing the robustness and accuracy of traditional monocular SLAM.

Traditional monocular depth estimation methods rely on depth cue for depth prediction with strict requirements. In recent years, the field of deep learning has developed rapidly. A large number of deep learning methods [2] have been proposed and shown promise in dealing with the traditionally ill-posed problem.

The global smartphone penetration rate is estimated to have reached over 78 percent in 2020. There are even mobile devices on the market that are equipped with LiDAR sensors. However, not at an affordable price. Therefore, the monocular camera is currently still the most widely used visual sensor. As mentioned above, the lack of baseline length makes it impossible to obtain real-scale 3D trajectory. To solve this problem, this article proposes a framework called pseudo RGB-D SLAM, which uses deep learning depth estimation from a monocular camera and a robust V-SLAM algorithm. Within this framework, the depth estimation architecture adopts Adabins and the V-SLAM algorithm chooses ORB-SLAM2.

II. Related Work

A. Visual SLAM

V-SLAM uses the camera as the main external sensor and creates a map of the environment while locating itself without prior knowledge of the environment [3].

Compared with traditional lidar data, visual data has the characteristics of low cost and a large amount of information. There are three main types of visual sensors: monocular camera, stereo camera and RGB-D camera. In addition to image preprocessing, these three sensors may be equipped with sensors such as IMU, so time synchronization processing is also required. The monocular camera cannot determine the true size of images. To solve the problem of restoring the metric scale, stereo and RGB-D are introduced. A stereo camera estimates the depth of each pixel with the disadvantage of a complex configuration and large computational consumption. The RGB-D camera obtains depth information by the infrared structure light or time-of-flight principle and has the limitation of narrow measurement range, small field of view, and interference with sunlight, which is a challenge in outdoor applications. V-SLAM mainly consists of two parts, the front-end for inter-frame estimation and the back-end for optimization. The front-end is known as visual odometry (VO). Visual odometry is used to estimate camera motion and the position of local feature points between adjacent photos. Due to the errors of sensors, it is inevitable that the estimated trajectory will show drift. Therefore, the back-end solves the problem of optimizing the vehicle's historical trajectory to cope with the error propagation.

Visual odometry can be divided into direct method and feature-based method. The feature-based SLAM extracts the feature points of the image and associates data. Data association compares the feature points of two images to determine if they correspond to the same object in the environment. Feature points are used to provide the basis to identify the environment. Feature point is composed of key points and descriptors. Keypoint refers to the position of the feature point in the image, and descriptor is the information describing the pixels around the key point. There are many methods for extracting feature points, such as ORB [4], SIFT [5], and SURF [6]. Algorithms such as parallel tracking and mapping (PTAM) [7] and ORB-SLAM [1] belong to feature-based keyframe SLAM.

B. *Monocular Depth Prediction*

From the viewpoint of computational models, estimating the depth of a scene from a single image is difficult for its low resource requirements. Monocular Depth Estimation [12] calculates depth from a single RGB image. Estimating depth from a single image is used in e-scene understanding, 3D modeling, robotics, autonomous driving, etc. Traditionally, we usually use stereo images, optical flow, and point clouds to evaluate the depth and resort with the same statistically significant, such as perspective and texture information, object sizes, object localization, and occlusions. With the development of depth estimation, monocular depth estimation does show the feasibility of depth estimating.

Monocular Depth Estimation is the task of estimating the depth value (distance relative to the camera) of each pixel given a single (monocular) RGB image [2]. It is an ill-posed problem and inherently ambiguous. However, humans can well perceive depth from a single image, given that sufficient samples (e.g. the appearances of nearby/distant objects) have been learned over lifetimes. With the success of deep learning techniques and available training data, the performance of monocular depth estimation has been greatly improved.

Nowadays, monocular depth prediction is a key prerequisite for determining scene understanding for applications such as 3D scene reconstruction, autonomous driving, and AR. State-of-the-art methods usually fall into one of two categories: designing a complex network that is powerful enough to directly regress the depth map, or splitting the input into bins or windows to reduce computational complexity.

There are many attempts to train different models for monocular depth estimation, including BinsFormer, EPCDepth, AdaBins, GCNDepth, etc. AdaBins [12] is to show a decisive improvement over the state-of-the-art on several popular depth datasets across all metrics. The traditional convolutional layers, only process global information once the tensors reach a very low spatial resolution at or near the bottleneck. Therefore, with the development of AdaBins, global processing is much more powerful when done at high resolution. The idea is to perform a global statistical analysis of the output of traditional encoder-decoder architecture and refine the output with a learned post-processing building block operating at the highest resolution.

III. METHODOLOGY

In this section, we will introduce the pseudo RGB-D SLAM framework. This framework integrates the monocular depth estimation by Adabins and V-SLAM by ORB-SLAM2. This study can be divided into the following phases:

1) *Data collection*
2) *Data preprocessing*
3) *Training monocular depth network (Adabins)*
4) *Pseudo RGB-D SLAM*
5) *Evaluate the performance of the framework*

A. *Data collection*

An important part of the study is data collection. In order to get better results in our experimental environments, collecting our own training dataset is crucial. The dataset consists of sequences from a variety of indoor scenes captured by both the RGB and Depth cameras from the Intel Realsense D455 (Fig 1). In order to align the depth sensor with the RGB sensor. Using Robot Operating System (ROS) is an easy way to align multiple sensors and record sensor raw data.

Figure 1. Intel Realsense D455 RGB-D camera

The dataset was collected in two types of environments shown in Fig 2. Respectively, a large environment: Jin Way Lecture theater and a small environment: Reception room at NCKU geomatics first floor. The sampling rate of the sensor is set to 30 frames per second which is a sufficient amount of images to perform SLAM and network training.

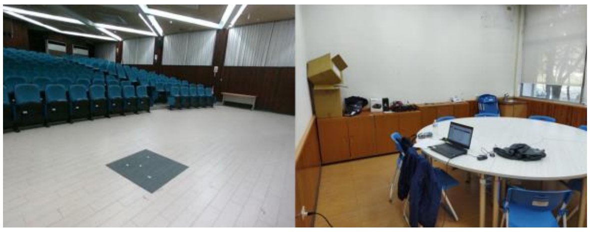

Figure 2. Data collection environments. Jin Way Lecture theater (left) and Reception room (right)

B. Data preprocessing

In order to achieve better results and efficiency when training the network, the data must be pre-processed. In this study, the data preprocessing workflow is sorted out using Fig 3.

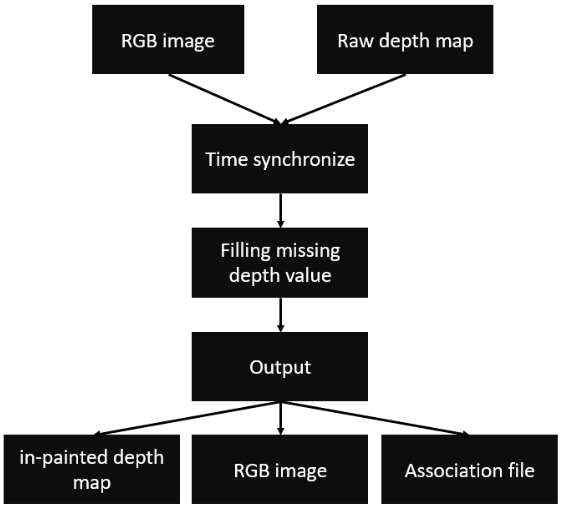

Figure 3. The data preprocessing workflow

1) Time synchronize

To prevent the sampling rate of RGB and depth sensor is not the same. The main purpose of time synchronization is to distinguish an RGB and depth map pair from each other. In other words, each RGB image is mapped to a depth image according to the timestamp.

2) Filling missing depth value

The raw projected depth image contains many missing values. Missing values in the depth image are the result of (a) shadows caused by the imbalance between infrared emitter and camera, or (b) random missing or incorrect values caused by specular or low albedo surfaces. Fig 4 shows the output from the RGB camera and depth camera of one frame. As already mentioned, several values are missing in the depth image, which is shown in dark blue color.

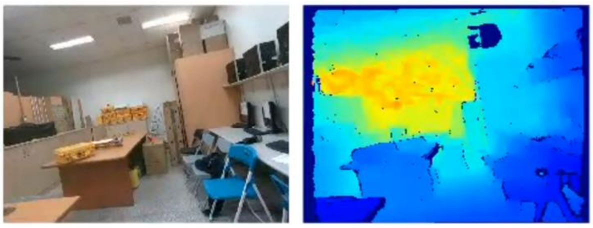

Figure 4. Output from the RGB camera (left) and depth camera (right).

The missing values may cause the training model to predict the depth value of the scene with a bias. In this study, we use the Levin et al's Colorization method to fill in the missing value of the ground truth depth. Colorization with optimization [8] requires neither precise image segmentation nor accurate region tracking. This method is based on a simple premise: neighboring pixels in space-time that have similar intensities should have similar colors. The source code of the method is written in Matlab and provided by the author. Due to the different usage scenarios, the code that fills in the missing depth value is a slight adaptation of the original code. The result of the colorization method is illustrated in Fig 5. On average, a 640×480 image takes about 3 seconds to process on an intel i7 laptop.

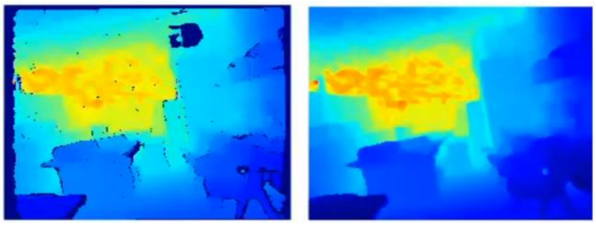

Figure 5. The result of filling missing depth value algorithm. Raw depth map (left) and colorization method applied depth map (right).

3) Training and Testing dataset output

After the depth missing value is filled, the output file contains RGB image, in-painted depth map and the association file. The dataset is divided into different folders which correspond to each ’scene’ being filmed. In the folder, files that begin with the prefix rgb are the RGB image. Files that begin with ‘sync_depth’ are the frames from the depth sensor. The RGB file is saved in JPG format and the depth map is saved in 16-bit PNG format.

The data type of the preprocessing depth maps is float. To save it in image format, the depth map is first multiplied by a scale of 1000 and then saved as a uint16 data type. In other words, a pixel value of 1000 equals 1 meter. The maximum distance is limited to 65,535 meters, but it's still a big gap between the Realsense hardware specification limit (about 20 meters).

The association file stores the information that an RGB image with respect to its synchronization depth image. The file format is saved as text file format.

C. Training monocular depth network (Adabins)

In this study, we train the monocular deep network, using the pre-trained model of NYU dataset for transfer learning.

This study fine tune three types of models with different combinations of our NCKU Geomatics Depth dataset scenarios. Respectively, Jin Way Lecture theater, Reception room, and the mixture of both scenarios.

We use the same parameters for better completion results of different combinations of training and testing.

The parameters are shown in TABLE 1.

TABLE 1. The parameters of the training network

Key	Value
Batch Size	2
Degree	2.5
Distributed	False
DIV Factor	25
Do Kb Crop	False
Do Random Rotate	True
Eigen Crop	True
Epoch	10
Final Div Factor	100
Grag Crop	False
Input Height	416
Input Width	544
Lr	0.000357
Max Depth	20
Max Depth Eval	20
Min Depth	0.001
Min Depth Eval	0.001
N Bins	256
Norm	Linear
Validate Every	100

D. *Pseudo RGB-D SLAM*

In this study, we postulate using an Adabins depth estimation model as a pseudo depth sensor, so we can use only monocular cameras to implement pseudo-RGB-D-SLAM and still get significant improvements in robustness and accuracy compared to RGB-SLAM. ORB-SLAM2 [1] is a complete SLAM system for monocular, stereo, and RGB-D cameras. It has four parts of inputs to execute RGB-D SLAM (1) RGB images (2) Each image predicted results (3) Synchronized information (4) Camera calibration and distortion parameters. The system workflow is shown in Fig 6 and Fig 7

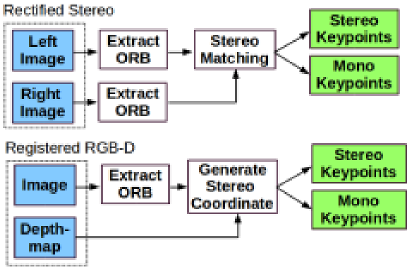

Figure 6. The workflow of registered RGB-D and rectified stereo

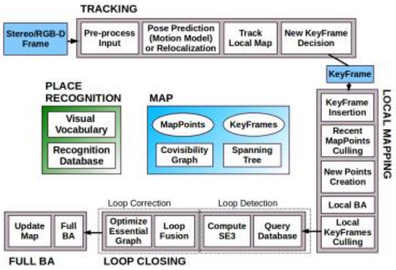

Figure 7. The overall workflow of the ORB-SLAM2 system

E. *Evaluate the performance of the framework*

We conduct experiments to evaluate depth prediction and pose of our framework. Using Adabins and V-SLAM based pose estimation respectively.

1) *Metrics for depth evaluation*

For quantitative depth evaluation, we use the standard metrics, including the Absolute Relative (Abs Rel) error, Squared Relative (Sq Rel) error, RMSE, RMSE log, $\delta < 1.25$ (namely a1), $\delta < 1.25^2$ (namely a2), and $\delta < 1.25^3$ (namely a3) as defined in [10].

2) *Metrics for pose evaluation*

For quantitative pose evaluation, we compute the Root Mean Square Error (RMSE) of the predicted camera trajectory. We align the camera trajectory estimated by each method with the ground truth trajectory using the EVO toolbox [11]. The reference trajectory of this study will be obtained by running the ground truth depth version of ORB-SLAM.

IV. Data Description

In this study, we select the NYU depth dataset which contains a rich amount of indoor scenes. In this study, the NYU depth dataset is used to train as a pre-trained model. As mentioned in section A of this chapter, this study also collects our own data set to get better results in our experimental setting. The data collected in this study will be used to fine-tune the depth network. Below we briefly describe the NYU Depth Dataset V2 and the NCKU Geomatics Depth Dataset.

A. *NYU Depth Dataset V2*

A dataset that provides images and depth maps for different indoor scenes captured at a pixel resolution of 640 × 480 [9]. The dataset contains 120K training samples and 654 testing samples [10]. We train our network on a 50K subset. The depth maps have an upper bound of 10 meters.

B. *NCKU Geomatics Depth Dataset V1*

A dataset that provides images and depth maps for different indoor scenes captured at a pixel resolution of 640 × 480. In recent, there are two indoor scenes. The dataset contains 12585 samples. TABLE 2 shows the dataset size and how the dataset splits into train and test.

TABLE 2.
NCKU geomatics train and test dataset split

Scene	Train	Test
Jin Way Lecture theater	4093	3144
Reception room	4705	643

V. Experimental Evaluation

A. *Depth network evaluation*

In the following, we evaluate the performance of our depth estimation on the NCKU depth dataset sequences. Three models are trained using different types of combinations of the scenario data. The result is shown in TABLE 3, TABLE 4, and TABLE 5.

TABLE 3.
Depth evaluation of model 1

Training	**Reception room (model 1)**		
Testing	Reception room	Jin Way Lecture theater	Mixed
Abs Rel	**0.107**	0.523	0.313
Sq Rel	**0.186**	2.085	1.126
RMSE	**0.771**	4.061	2.399
RMSE log	**0.179**	1.126	0.647
a1	**0.919**	0.119	0.524
a2	**0.974**	0.303	0.642
a3	**0.984**	0.480	0.735

TABLE 4
Depth evaluation of model 2

Training	**Jin Way Lecture theater (model 2)**		
Testing	Reception room	Jin Way Lecture theater	Mixed
Abs Rel	0.285	**0.167**	0.227
Sq Rel	0.427	**0.203**	0.316
RMSE	1.357	**1.133**	1.246
RMSE log	0.389	**0.210**	0.301
a1	0.489	**0.746**	0.616
a2	0.782	**0.946**	0.863
a3	0.928	**0.990**	0.958

TABLE 5
Depth evaluation of model 3

Training	**Mixed (model 3)**		
Testing	Reception room	Jin Way Lecture theater	Mixed
Abs Rel	**0.135**	0.137	0.136
Sq Rel	0.291	**0.189**	0.242
RMSE	0.838	**0.771**	0.806
RMSE log	0.192	**0.187**	0.190
a1	**0.887**	0.862	0.874
a2	**0.970**	0.968	0.969
a3	0.983	**0.988**	0.986

As result, the depth network is case sensitive. In order to get better results in depth estimation, it is important that the training data set is environment diverse. The mixed model (Model 3) trained in both scenarios proved satisfactory in this study. Model 3 qualitative of result is shown in Fig 8.

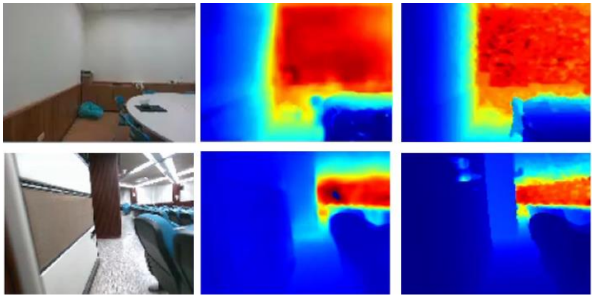

Figure 8. Qualitative depth evaluation results on NCKU depth test set. RGB (left), predict depth (middle) and ground truth depth (right)

B. Pseudo RGB-D SLAM pose evaluation

In this section, we evaluate the pose estimation using testing image sequence from the NCKU Geomatics Depth Dataset V1. Comparison of the trajectory of Monocular SLAM, RGB-D SLAM and Pseudo RGB-D SLAM with three different models. Then evaluate the performance of each result reference by true RGB-D SLAM.

TABLE 6
SLAM overall result in reception room scenario.

Input	**Poses**	**Length(m)**	**duration(s)**
mono	78	7.494	60.833
Model 1	643	24.086	64.23
Model 2	643	21.138	64.235
Model 3	643	25.319	64.235
rgbd	643	25.247	64.235

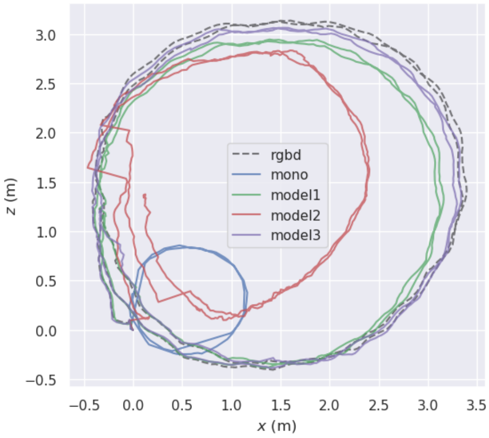

Figure 9. V-SLAM trajectory through different input data in the Reception room.

TABLE 7.
SLAM overall result in jin-way lecture theater scenario.

Input	**Poses**	**Length(m)**	**duration(s)**
mono	(failed)	(failed)	(failed)
model1	3144	41.948	104.806
model2	3144	90.151	104.806
model3	3144	93.615	104.806
rgbd	3144	92.200	104.806

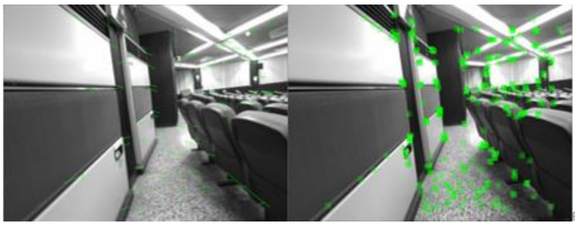

Figure 10. SLAM initialize with mono (left) and mono + depth(right).

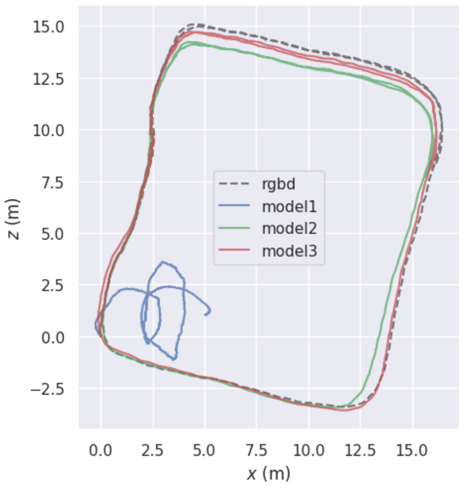

Figure 11. V-SLAM trajectory through different input data in the Jin-Way lecture theater.

TABLE 8.
Pose evaluation for each method reference to true RGB-D SLAM.

method	Reception room			Jin Way Lecture theater		
	rmse	mean	std	rmse	mean	std
mono	1.256	1.253	0.093	x	x	x
model1	0.124	0.122	0.023	7.876	7.713	1.593
model2	0.544	0.523	0.147	0.393	0.369	0.133
model3	**0.049**	**0.044**	**0.021**	**0.232**	**0.209**	**0.102**

According to the above results, TABLE 6. shows that Monocular SLAM has a very large scale factor error (length), which we can see also in Fig 8. In V-SLAM initialization, adding depth images can make it easier to complete (Fig 10). As a result, monocular SLAM failed but others succeed in the Jin Way Lecture theater scenario.

In our framework, we use three models to estimate depth images. According to the trajectory result, model 3 has highly overlapping results with true RGB-D SLAM, as shown in (Fig 9 and Fig 11). Likewise, pose evaluation statistics on TABLE 8 shows that model 3 has the best accuracy performance in V-SLAM.

As a result, within our framework.

1) Adabins depth estimation model can help the monocular camera improve the accuracy and robustness of SLAM.

2) A diverse dataset can make our pseudo RGB-D SLAM more adaptable to different environments.

VI. Conclusion and Future Work

This study mainly uses the properties of deep learning to perform a pseudo-depth sensor. The V-SLAM is greatly improved without the addition of an active depth sensor. In the present framework, the relative translation error can reach 0.2% which is a convincing result that the pseudo depth sensor can not only resolve the scale-factor of monocular SLAM, but also provide good visual odometry results. In other words, a 100-meter trajectory has an RMSE of about 20 centimeters. The depth evaluation shows that the mixed model (train with both scenario data) has the best depth prediction among other models in both test scenarios. Also, the hybrid model performs the most accurate pose estimation in V-SLAM among other models in both test scenarios. The result can be explained by the fact that the better the pseudo depth sensor, the better the V-SLAM result. This study also notes that framework performance is largely dependent on the case sensitivity of the test scenario. In other words, without prior information about the environment, the depth estimate and the V-SLAM are not robust in such a scenario.

Currently, our framework only works in an off-line mode, so developing an online real-time system remains one of our future works. Another avenue for our future works is to move towards more challenging settings, e.g., outdoor or uncalibrated cameras. As mentioned, this framework works well in a known environment and device. We're still trying to generalize the framework. Camera calibration is an important issue for various devices. One solution is online calibration. We can use the previous moving frames to estimate the camera instinct parameters. For the unknown scenes, we can refer to the self-improving framework by Tiwari et al. [13].

References

[1] R.Mur-Artal, J. M. M.Montiel, and J. D.Tardos, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System,” IEEE Trans. Robot., vol. 31, no. 5, pp. 1147–1163, 2015, doi: 10.1109/TRO.2015.2463671.

[2] Y.Ming, X.Meng, C.Fan, and H.Yu, “Deep learning for monocular depth estimation: A review,” Neurocomputing, vol. 438, pp. 14–33, 2021, doi: 10.1016/j.neucom.2020.12.089.

[3] Y.Chen and Y.Zhou, “A Review of V-SLAM＊ ,” no. August, pp. 603–608, 2018.

[4] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544.

[5] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” International journal of computer vision, 2004, 60(2):91-110.

[6] H. Bay, A. Ess, T. Tuytelaars, et al. “Speeded-up robust features (SURF),” Computer vision & image understanding, 2008, 110(3):346- 359.

[7] G Klein, D. Murray, “Parallel tracking and mapping for small AR workspaces,” Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. Nara, Japan, 2007, pp.225-234.

[8] A.Levin, D.Lischinski, and Y.Weiss, “Colorization using optimization,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 689–694, 2004, doi: 10.1145/1015706.1015780.

[9] N.Silberman, D.Hoiem, P.Kohli, andR.Fergus, “Indoor segmentation and support inference from RGBD images,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 7576 LNCS, no. PART 5, pp. 746–760, 2012, doi: 10.1007/978-3-642-33715-4_54.

[10] D.Eigen, C.Puhrsch, and R.Fergus, “Depth map prediction from a single image using a multi-scale deep network,” Adv. Neural Inf. Process. Syst., vol. 3, no. January, pp. 2366–2374, 2014.

[11] Grupp, M.: evo: Python package for the evaluation of odometry and slam. https://github.com/MichaelGrupp/evo (2017)

[12] Shariq Farooq Bhat, Ibraheem Alhashim, Peter Wonka, “AdaBins: Depth Estimation using Adaptive Bins”, arXiv:2011.14141v1, 2021

[13] Lokender Tiwari, Pan Ji, Quoc-Huy Tran, Bingbing Zhuang, Saket Anand, and Manmohan Chandraker, “Pseudo RGB-D for Self-Improving Monocular SLAM and Depth Prediction” arXiv:2004.10681v3, 7 Aug 2020