SceneSLAM: A SLAM Framework Combined with Scene Detection

Conference Paper · December 2017

CITATIONS
0

READS
129

3 authors, including:

Zhehang Tong
National University of Defense Technology
3 PUBLICATIONS **1** CITATION
SEE PROFILE

Shaowu Yang
National University of Defense Technology
23 PUBLICATIONS **161** CITATIONS
SEE PROFILE

SceneSLAM: A SLAM Framework Combined with Scene Detection

Zhehang Tong[1], Dianxi Shi[1], Shaowu Yang[2]

Abstract— **Different sensors have different capacities and may even fail in different scenes. This will influence accuracy and reliability of SLAM systems utilizing those sensors. To enhance the robustness of SLAM systems, we present SceneSLAM in this paper, which is a novel extensible SLAM framework by combing different SLAM systems facilitated by a scene detection method. SceneSLAM can activate different SLAM modules automatically based on the result of scene detection and perform map fusion to achieve globally consistent localization and mapping. To verify the extendibility and effectiveness of SceneSLAM, we build a prototype system based on SceneSLAM to enhance the reliability of existing SLAM systems when the light condition changes dramatically. The prototype system runs on a TurtleBot robot equipped with a Kinect sensor. The experimental results show that SceneSLAM provides an effective solution to enhance the robustness of existing SLAM systems in dynamic environments by activating corresponding SLAM module based on the scene detection results. SceneSLAM provides efficient support to developing and implementation of a SLAM system with multiple SLAM modules and a scene detection module. Furthermore, we release the source code of our framework with our prototype system, so that it can be easily extended by other researchers.**

I. INTRODUCTION

The simultaneous localization and mapping (SLAM) has been seen as a core problem in mobile robotics. SLAM simultaneously estimates the location of a robot equipped with on-board sensors and construct a map of the environment that the sensors are perceiving. It has been one of the most important qualifications for mobile robots to realize autonomy [1]. There are two basic solutions to the SLAM problem based on the on-board sensors in robotics, i.e., laser-based SLAM and vision-based SLAM. Laser scanners provide the range measurements of environments, while cameras can provide images which contain rich information of environments that allows for robust and accurate place recognition, and even scene understanding. As a result, vision-based SLAM has attracted much focuses in recent years and been supposed to have wide prospects of applications [2]. The fundamental properties of SLAM, including observability, convergence and consistency, have been intensively studied and understood in the last few years [3]. However, the current SLAM systems might be fragile because of the dynamic changes of environments. For instance, vision-based SLAM may fail in drastic illumination changes between day and night. In other words, specific sensor and the corresponding SLAM system may only work well in a suitable scenario. However, SLAM is a foundation of autonomous navigation for mobile robots when robots are applied in unknown environments, where dynamic changes, e.g. illumination changes, may be common and inevitable. How to improve the robustness and tackle the dynamic changes of environments is still a challenge to SLAM [3].

To address the above problem of SLAM, we propose a solution that enables SLAM systems to detect scene changes and selects suitable sensors for SLAM to prevent sensor failures. For example, if the SLAM system detects the illumination is sufficient, it can select vision sensor to perceive the environment and achieve 3D perception of the robot pose and the environment map. Once the illumination gets too weak, it can select laser sensor to perceive the environment since vision is likely to fail in dark. Another example is that a RGB-D camera can directly get depth information in indoor scenes but sometimes fails in outdoor scenes because of too strong illumination and too large scene scale. With this motivation in mind, in this paper, we propose SceneSLAM, a robust and extensible SLAM framework combining with a scene detection method. SceneSLAM can activate different SLAM modules automatically based on the result of the scene detection module and perform map fusion by coordinate transformation to get globally consistent localization and mapping. The main contributions of our paper are in four folds:

- We propose a robust and extensible SLAM framework to enhance the self-adaptive performance of current SLAM systems.
- By modular structure design, it is easy to modify the scene detection module to support complex application scenarios and utilize open-source SLAM systems to be SLAM modules of SceneSLAM.
- Based on SceneSLAM, we implement a prototype system running in TurtleBot robot equipped with a Kinect sensor to overcome drastic illumination changes.
- We make the SceneSLAM open-source to promote researches on robust and reliable SLAM systems.

To the best of our knowledge, such a robust and extensible SLAM framework combined with scene detection has not been proposed before. In the following we discuss related work in the field in Section II and introducing our framework in Section III. Section IV presents the experimental setup and results. Finally we draw conclusions and discuss future work in Section V.

[1] Z. Tong, Prof. Dr. D. Shi are with National Key Laboratory of Parallel and Distributed Processing(PDL), National University of Defense Technology, Changsha, China, {tongzhehang15, dxshi}@nudt.edu.cn

[2]Dr. S. Yang is with the State Key Laboratory of High Performance Computing (HPCL), National University of Defense Technology, Changsha, China shaowu.yang@nudt.edu.cn

II. Related work

In this section we discuss related work on scene detection and SLAM. Besides discussing the related SLAM systems, we give a brief survey of the robustness issues of SLAM systems and related work in recent years.

A. Scene detection

The topic of vision-based scene detection has been explored both in computer vision and robotics communities. In computer vision field, scene detection was regarded as a classification problem, and it is often referred to as scene classification [4] [5]. A number of benchmark papers [5] [6] has driven this field of research forward. Traditional scene classification systems universally rely on a fixed set of hand crafted features that are extracted from the images and then used for classification. For example, [7] relies on SURF and CRFH, while [8] and [9] uses dense SIFT and CENTRIST [10], respectively. However, a recent trend in computer vision is using deep convolutional neural network [11] to extract learned features. [12] used ConvNet and won the annual ImageNet Large Scale Visual Recognition Challenge [13] on the task of semantic place categorization. In robotics field, scene classification combined with creating a map is often referred to as semantic mapping [7] [14]. Semantic mapping helps to enhance the autonomy and robustness of robots, and facilitate more complex tasks. [7] builds a semantic map with different rooms labeled. [15] recognizes several known objects in the map. [16] builds semantic map by integrating an extensible classification system.

B. Open-source SLAM systems

At the early stages of SLAM development, laser-based approaches based on nonlinear filters and maximum likelihood estimation are the state-of-art solutions [1] [17]. Gmapping [18] is one of the representative works and widely used in ROS [19] development community. Recently, vision-based SLAM based on graph optimization has been strongly developed as vision sensors are cheaper in general and vision-based SLAM has wider prospects for application. ORB-SLAM2 [20] is a vision feature-based SLAM system for monocular, stereo and RGB-D cameras and is able to detect loops and relocalize the camera in real-time. ORB-SLAM2 achieves impressive localization accuracy in public datasets. LSD-SLAM [21] is a novel direct SLAM technique. Instead of using image feature extraction as ORB-SLAM2 does, LSD-SLAM directly operates on image intensities both for tracking and mapping. Other open-source SLAM systems with good performance can be found in [22], [23], [24], [25].

C. Robustness issues of SLAM

Current SLAM systems fairly common assume that the world is static which is not authentic in practice. As a result, the current SLAM system might be fragile because of the dynamic change of the environment. We investigate three aspects causing the fragility in dynamic environment: algorithmic mistake, improper parameter and sensor failure.

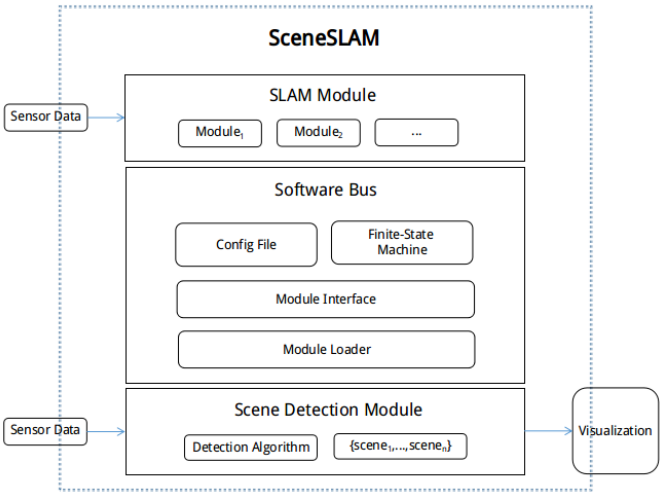


Fig. 1. SceneSLAM consists of three main components: Software bus, Scene detection module and SLAM module. Software bus is the core component of the framework while the scene detection module and the SLAM module are extendable and modifiable. Scene detection module and SLAM module both rely on sensor data. Moreover, the SLAM module presents the result through the visualization software.

Algorithmic mistake mainly caused by wrong data association in dynamic environment. Data association is crucial when tracking and detecting loop. Bag-of-words [26] have shown reliable performance on the task of loop closure detection and relocalization. However, it can not handle extreme illumination variations. There are some new methods to handle this, e.g. matching appearance-based sequences [27], [28].

In order to work correctly in a given scene, current SLAM systems rely on parameter tuning, such as camera calibration parameters, feature extraction and matching parameters. To enhance the robustness of SLAM system, parameters automatic tuning is useful. [29] provides a framework for online simultaneous localization, mapping and self-calibration which can detect and handle significant changes in the calibration parameters.

Sensor failure might appear when on-board sensors aging or sensors can not work in some scene, e.g. camera can not get color images in dark. An ideal SLAM system should be aware of sensor failure and establish mechanism to handle it. SceneSLAM presented in this paper aims to provide such capability through scene detection.

III. SceneSLAM

SceneSLAM is a framework of SLAM combined with scene detection built on Robot Operating System(ROS) [19]. ROS provides abundant libraries and tools to help software developers quickly create robot applications. The reason we build our framework based on ROS is to ensure compatibility and extendibility. A general overview of SceneSLAM is shown in Fig. 1. In particular, SceneSLAM consists of three main components:

- Software Bus that is responsible for calling the scene

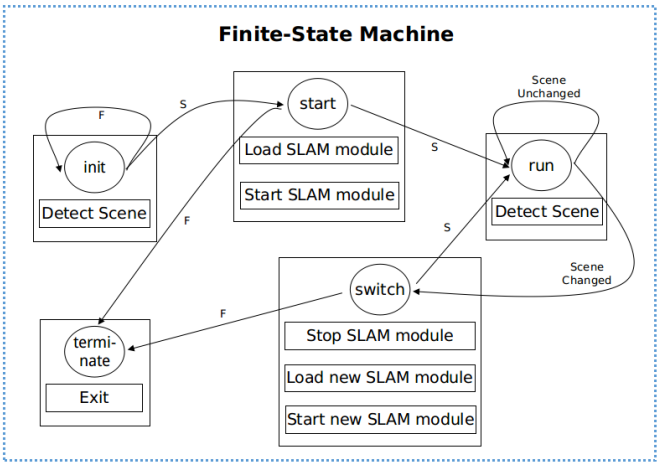


Fig. 2. There are five states in The Finite-State Machine: *init, start, run, switch, terminate*. S indicates the actions in the state are performed successfully and F indicates failed. Depends on Finite-State Machine, SceneSLAM can call the scene detection and the corresponding SLAM module properly and logically.

detection and the corresponding SLAM module properly and logically.

- Scene detection module that is responsible for detecting the scene around the robot.
- SLAM module that estimates the state of the robot and build a map in corresponding scene.

To achieve the extendibility, the whole framework is designed with the same interface for each module using modular structure. There are mainly two kinds of modules in our framework: the scene detection module and the SLAM module. In particular, our framework depends on the pluginlib [30] which is a C++ library for loading and unloading plugins from within a ROS package. With the powerful support of pluginlib library, modules can be instantiated by specific string name and method can be called by defined interfaces. The scene detection module and the SLAM module are both built as plugins based on the specific interface. By such modular structure design, it is very easy to modify and replace the scene detection module to support complex application scenarios and make other SLAM system to be a SLAM module of SceneSLAM. Benefited from our framework, once the scene detection module and the SLAM module are implemented, SceneSLAM will call the suitable SLAM module automatically based on the result of the scene detection module. We discuss more details of SceneSLAM in the following sections.

A. Software bus

The software bus is responsible for calling the scene detection module and the corresponding SLAM module based on the result of scene detection. There are three main parts in the software bus, namely (1) Finite-State Machine, (2) Module Interface and Loader, and (3) Config File.

To detect the scene changes in real-time and switch the SLAM module once the scene changes, we design a Finite-State Machine running in a separate thread. The Finite-State Machine is a widely used in software development. The most important in Finite-State Machine is that in every state some relevant actions are executed and the state may transform from one state to another state based on action results. Our Finite-State Machine has five different states: initial state, start state, running state, switch state and terminated state, represented by a state set $\{$*init, start, run, switch, terminate*$\}$. Only one state is set when the software bus is running. Let state x_s firstly set to *init* at the beginning of the software bus and in the *init* state *detect scene action* is executed. Software bus operates *detect scene action* by calling scene detection module. If the scene changes, the x_s transforms to *switch* state. In *switch* state, the current SLAM module is shutdown and a suitable SLAM module is activated. With the state is transformed and actions are executed in perspective, the SceneSLAM detects scene changes in real-time and then call the corresponding SLAM module. The full transformation between the five states is shown in Fig. 2.

To achieve the extendibility, the whole framework is designed using modular structure with two kinds of modules: scene detection module and SLAM module. We define interfaces for the scene detection module and SLAM module. A scene detection module and a SLAM module can be loaded and called by the module loader properly only when they implement the interface. The module loader is C++ pointer of interface type. Scene detection module interface includes two basic methods: *activate* and *shutdown*, to control the start and stop of the module. In addition, scene detection module interface includes *getScene* method to return the scene detection result. Let $x_{scene} \in \{scene_1, ..., scene_n\}$ be the scene detection result and the scene set $\{scene_1, ..., scene_n\}$ is the collection of all the type the module can distinguish. For a scene detection module, the key is to design algorithms to complete scene detection and implement the *GetScene* method to return the scene detection result. An example implementation of scene detection module is described in the section B.

SLAM module interface also includes *activate* and *shutdown* methods, similar to the scene detection module interface. Beyond that, SLAM module interface includes *SetPose* and *GetPose* methods. Every SLAM module has its own coordinate system and the origin of coordinate system is the pose of robot when the module is activated. We use T_{local} to represent the pose of the robot in SLAM module coordinate system which is a transformation matrix consisting of a rotation matrix R_{local} and a translation vector t_{local}. To get a global and consistent location and map after switching between SLAM modules, SceneSLAM performs conversion between global map coordinate system and local SLAM module coordinate system which we use $T_{conversion}$ to represent. $T_{conversion}$ also consists of a rotation matrix and a translation vector. We update the $T_{conversion}$ as

$$T_{conversion} = T_{conversion} \times T_{local}, \tag{1}$$

whenever scene changes and SLAM module switching is operated successfully. Based on the $T_{conversion}$ and t_{local}, we can get the coordinate of robot in global map coordinate

(a) Turtlebot robot (b) Experimental environment

Fig. 3. We run our prototype system in a TurtleBot robot equipped with a Kinect sensor. The experiments are carried out in our lab. The current scene is *Bright* and at the right upper corner is the door to *Dark* scene.

system in the form of equation as

$$c_{global} = R_{conversion} \times t_{local} + t_{conversion} \tag{2}$$

which we use c_{global} to represent the global coordinate of the robot in global map coordinate system. All the pose information is set and got through *SetPose* and *GetPose* methods

Config File is simple but important. SceneSLAM can detect scene and then call suitable SLAM module applied to localization and mapping. The config file establishes the relationship between scene and SLAM module using key-value pairs. Key is the x_{scene} and the value is the name of the corresponding SLAM module. Therefore, in general, the size of scene set is equal to the number of SLAM modules. Some implementation details of the SLAM modules is described in the section C.

B. Scene detection module

Scene detection module provides the concrete implementation of scene detection interface. It is responsible for detecting the scene around the robot. All the scene types can be distinguished by the detection algorithm denoted as a scene set $\{scene_1, ..., scene_n\}$. And every $x_{scene} \in \{scene_1, ..., scene_n\}$ is a key of key-value pairs in config file, as we describe in section A. Detecting result can be accessed through *GetScene* method.

Vision-based scene detection has been explored both in computer vision and robotics community, but there is no consensus on one particular scene detection technique that can solve the scene detection problem perfectly, especially in robotics images are spatio-temporal correlation and consecutive. As scene detection is still an open problem, we build a simple scene detection module that can detect the dark scene and bright scene to verify the effectiveness of our framework. However, other scene detection approaches can be used to replace and upgrade the scene detection module, owing to the extendibility of the SceneSLAM framework.

The goal of our example detection module is to prevent vision sensor failure in dark scene through detecting dark scene and bright scene. In other words, in our detection module, we have $x_{scene} \in \{$*Dark, Bright*$\}$. We design an algorithm to detect x_{scene} based on color images. x_{scene} is equal to *Bright* when the color images fulfill the following condition: The mean of gray-scale value of image pixels is greater than a threshold. We have found that a robust threshold is set to 100. Otherwise x_{scene} is equal to *Dark*. The gray-scale value S of a pixel is calculated by the formula as

$$S = R * 0.299 + G * 0.587 + B * 0.114, \tag{3}$$

where R,G and B are the color value in the RBG space. To enhance stability, we set up a two seconds time window to confirm a scene change. Only all the images in the time window fulfill the condition, we confirm the scene has changed. The explain of scene detection condition and threshold set is explained in Section IV.

C. SLAM module

SLAM module provides the concrete implementation of SLAM interface to locate the position of a robot and build map based on specific on-board sensors. In recent years, SLAM system is in rapid development and many SLAM systems are publicly available to promote the research. SceneSLAM support all the C++ based open-source SLAM systems to work as a SLAM module after slight modifications.

To verify the compatibility, we modify ORB-SLAM2 [20] and Cartographer [25] to work as two SLAM modules in SceneSLAM. ORB-SLAM2 is a popular open-source vision-based SLAM system which provides monocular, stereo and RGB-D interface to locate a robot and to build a environment map. We encapsulate the RGB-D interface of ORB-SLAM2 and implement all the SLAM module interface as we describe in section A. We refer it as the RGBD SLAM module. Cartographer is a laser-based SLAM system. We modify and encapsulate Cartographer which is the same as ORB-SLAM2. We refer it as the Laser SLAM module.

Combing with our example of scene detection module and the associated configuration file, the system can handle the drastic illumination changes automatically. RGBD SLAM module will be called when the room is bright enough. RGBD SLAM module processes color images and depth images to produce 3D feature points map. However, when get into a dark room, RGBD SLAM module will be shutdown and the Laser SLAM module will be activated. Laser module processes Laser data simulated by depth images and produce 2D occupancy grid map. More experimental details are explained in the next Section.

IV. Experiments

To verify the effectiveness of SceneSLAM, we implement a prototype system based on SceneSLAM to handle the drastic illumination change problem. The design goal of our system is to locate the robot and build map based on vision sensors in good light conditions, with rich image information. While vision sensors fail in dark places, we change to locate the robot and build a map based on a laser sensor. With the support of SceneSLAM, it is convenient to achieve our design goal by implementing a scene detection module and encapsulating open-source SLAM systems as SLAM modules, as we explain in Section III.

(a) *Bright* scene

(b) *Dark* scene

(c) Change to *Dark*

(d) Change to *Bright*

(e) TurtleBot's position

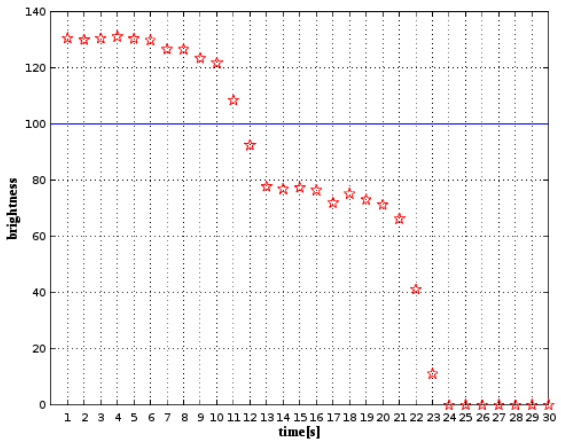
(f) TurtleBot's position

Fig. 4. Fig(a) was taken in the bright room, Fig(b) was taken in the dark room, Fig(c) was taken when the robot was about to get in the dark room and scene detection module confirmed scene change to *Dark*, the TurtleBot's position is shown in Fig(e) at this moment. Fig(d) was taken when the robot was just about to get out of the dark room and scene detection module confirmed scene change to *Bright*, the TurtleBot's position is shown in Fig(f) at this moment.

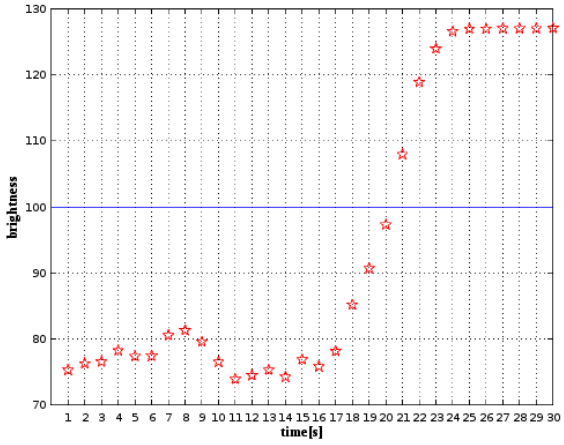
We have run our prototype system in a TurtleBot robot equipped with a Kinect sensor and an Intel Core i7-4710MQ laptop with 8Gb RAM running ROS Indigo. The robot is shown in Figure 3(a). We have conducted several experiments to prove that SceneSLAM can call the scene detection module to get the detection result as expected and our example detection module can detect *Bright* and *Dark* scene accurately and efficiently. Then, we demonstrated that the system can call the corresponding SLAM module as we configured in config file. Moreover, we also demonstrated that our system can perform SLAM module switching correctly when the scene changes. Finally, we demonstrated that the coordinate transformation is correct when the system performs SLAM module switching. We conducted the experiments in our lab within two rooms. One room is *Bright* scene as the light in the room is turned on while the other room is *Dark* scene as the light is turned off, as shown in Fig. 3(b).

A. Scene detection module

We design an example of scene detection module to detect *Bright* scene and *Dark* scene based on the color images to verify the effectiveness of SceneSLAM. Our experiments show that it can accurately and efficiently detect the scene.



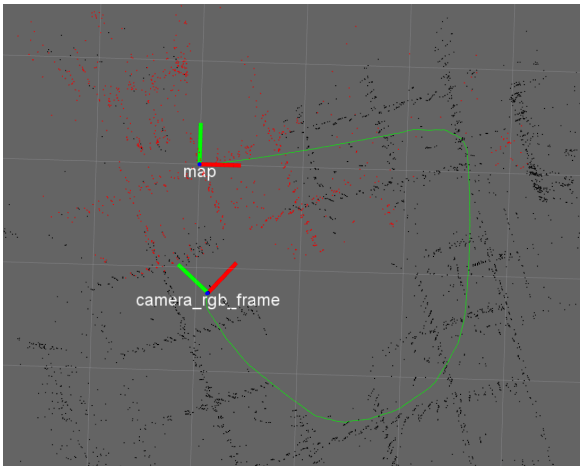
(a) Bright room to dark room



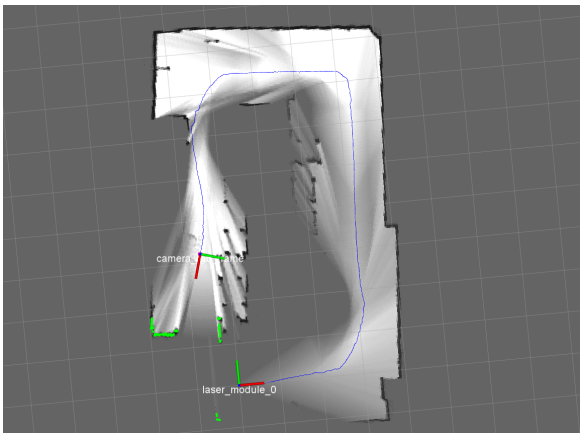
(b) Dark room to bright room

Fig. 5. Fig(a) shows the mean of gray-scale value changes over time when the TurtleBot travels from the bright room to the dark room. In the 12nd second, the TurtleBot travels into the dark room. Fig(b) shows the mean of gray-scale value changes over time when the TurtleBot travels from the dark room to bright room. In the 21th second, the TurtleBot travels out of the dark room.

Figure 4 shows some typical images of our experimental environment in good light, dark and when scene changes. The mean of gray-scale value changed sharply over time when TurtleBot traveled from the bright room to dark room or from the dark room to bright room which are shown in Figure 5. Figure 5 is plotted by Matlab. Through multiple experiments, we found that when set threshold value of gray-scale to 100, it can distinguish the *Bright* and *Dark* effectively. As shown in Figure 4(e), detection module confirmed a scene change to *Dark* when the robot was about to get into the dark room. And detection module confirmed a scene change to *Bright* when the robot was about to get into the bright room as shown in 4(f). As our scene detection module just relies on the mean of gray-scale value which are extremely fast to compute, it is very efficient and takes few resource. We find that compute every image's mean of gray-scale value only costs less than 15ms in our experiment platform. As robots are generally resource-constrained platforms, efficiency of



(a) Good light scene



(b) Dark scene

Fig. 6. We have run our system in the bright room and dark room respectively. Fig(a) was built in the bright room and Fig(b) was built in the dark room.

detection module is very important. The detection module returns the detection result by implementing *getScene* interface so that SceneSLAM can call corresponding SLAM module and perform SLAM module switching based on the detection result.

B. SLAM module

As we configure *Bright* scene corresponding to RGBD SLAM module while *Dark* scene corresponding to Laser SLAM module in the config file, we have run our system in the bright room and dark room respectively to verify whether SceneSLAM can call the corresponding SLAM module correctly. When start our system in the bright room, RGBD SLAM module will be called. RBGD SLAM module builds a 3D feature point map and the green line is the TurtleBot's moving trajectory as shown in Figure 6(a). When start up our system in the dark room, Laser SLAM module will be called. Laser SLAM module builds a 2D occupancy grid map and we use blue line to represent TurtleBot's moving trajectory generated by Laser SLAM module as shown in Figure 6(b). The experiment results demonstrate that SceneSLAM is compatible to existing open-source SLAM systems. In addition, with the support of SceneSLAM and proper configuration, corresponding SLAM module will be called properly based on the scene detection result.

C. SLAM module switching

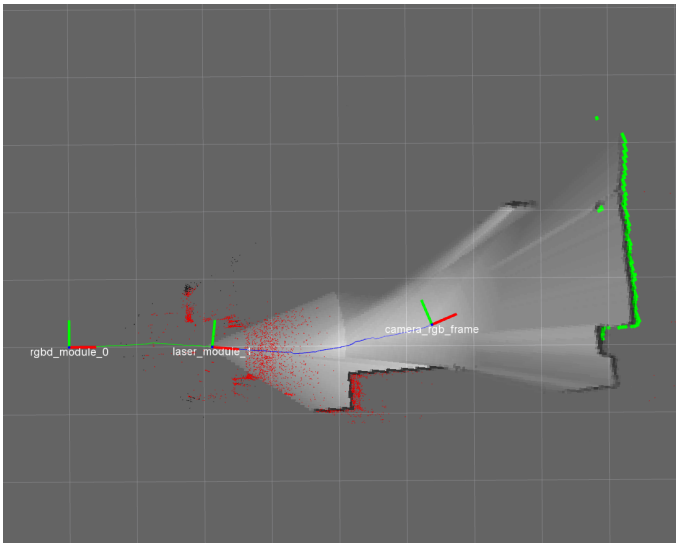
To prove SceneSLAM can perform SLAM module switching when scene changes, we controlled the TurtleBot moving from the bright room to the dark room. As shown in Figure 7, our prototype system built a hybrid map and generated a global consistent trajectory of TurtleBot. At first, TurtleBot was in the bright room and with scene detection module detected *Bright*, our system called the RGBD SLAM module. The red points were the 3D feature points and the green line was the Turtlebot's moving trajectory which were both built by RGBD SLAM module. As the RBGD SLAM module built the initial map, global coordinate system was same to the local RBGD map coordinate system. When TurtleBot was about to get into the dark room, scene detection module confirmed the scene change to *Dark*. RGBD SLAM module was shutdown and Laser module was activated. According to the TurtleBot's current pose, transformation between global map coordinate system and Laser SLAM module coordinate system was performed. As a result, a consistent grid map and blue path was generated by Laser SLAM module. In addition, the laser data was simulated by depth images. All the module switching and coordinate transformation are performed automatically with the support of SceneSLAM.

D. Coordinate transformation

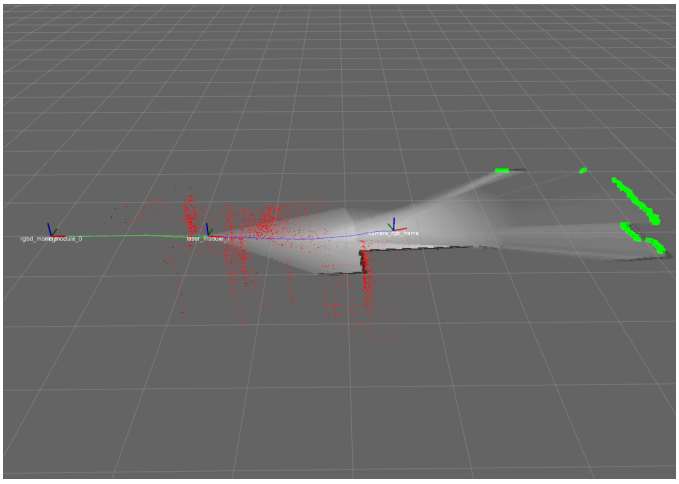
To verify our coordination transformation between SLAM modules is effective and correct when module switching performed, we perform module switching and compared the TurtleBot's moving trajectory with the one without module switching, as shown in Figure 8. The black line is the TurtleBot's moving trajectory without module switching. The green line is the trajectory after we perform module switching. Before module switching, the two moving trajectories are totally coincident. While after we simulated module switching, there was a little difference in the two moving trajectories when the TurtleBot took a sharp turn. But the differences between the two trajectories are less than 10cm. We consider it is caused by the performance of different SLAM systems, rather than our coordination transformation between SLAM modules. We have conducted other several experiments and gotten the similar results.

E. Stability issue of SLAM module switching

The SLAM modules normally need some time to initiate, especially RGBD module as it has to load a bag-of-words file. To prove that SceneSLAM imposes a little cost, we test some results when switching SLAM modules. ORB-SLAM2 SLAM system takes nearly 8 seconds to start up in our experiment platform, while it takes nearly the same time when switching from Laser module to RGBD module. Cartographer can start up more quickly taking less 1 second which is nearly the same time as it takes in the situation when switching from RGBD module to Laser module.



(a) Top down view



(b) Third person view

Fig. 7. TurtleBot started from bright room and RGBD SLAM module was called. As it entered the dark room, scene change was confirmed and Laser module was called. Transformation between global map coordinate system and SLAM module coordinate system was performed when switching SLAM module. Fig(a) was in top down view and Fig(b) was in third person view

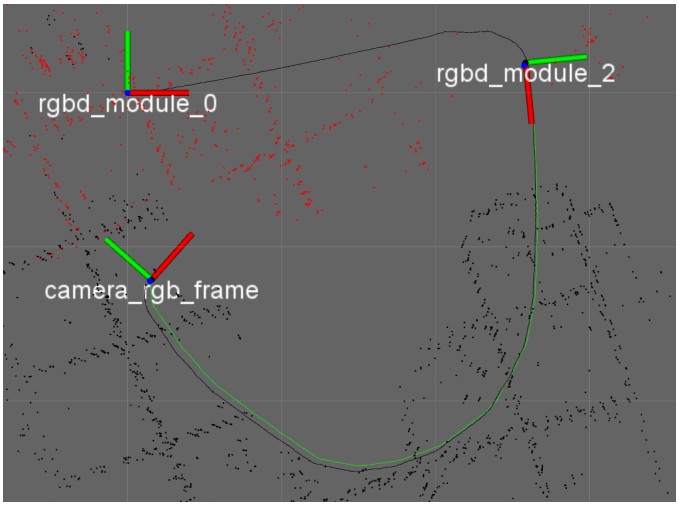


Fig. 8. The black line is the TurtleBot's moving trajectory without module switching. The green line is the trajectory when we simulated module switching.

V. CONCLUSIONS

In this paper, we present an extensible SLAM framework called SceneSLAM to enhance the self-adaptive performance of SLAM by combing SLAM system with scene detection. It can call the scene detection and the corresponding SLAM module based on the detection result. Moreover, when scene detection confirm scene changes, it performs SLAM module switching and coordination transformation to get globally consistent map and location of the robot. To verify the extensibility and effectiveness of SceneSLAM, we implement a full system based on SceneSLAM to enhance the robustness of SLAM system when it works in dynamic environment with drastic illumination changes. Our exemplary scene detection module can detect bright and dark scene accurately and efficiently based on the color images. RBGD SLAM module is implemented to work in bright scene and Laser SLAM module is implemented to work in dark scene. Several experiments prove that SceneSLAM is compatible to existing open-source SLAM systems. Once scene detection module and SLAM module implemented correctly, SceneSLAM can work well as expected. We have released the source code[1] of SceneSLAM, with our prototype system and instructions, so that it can be extended by other researchers.

As SLAM and scene detection are still open problems, there are some potential future extensions of SceneSLAM. Once the relationship between SLAM system and scene type explored in depth, fine-grained and complex scene detection for robot might be needed. To get more precise localization and mapping, pose graph optimization and loop closure between SLAM modules might be needed. Map saving and update strategy might be discovered for long-time running. In our understanding, detecting scene around the robot may be the key point to improve the self-adaptive performance and robustness of SLAM system.

REFERENCES

[1] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[2] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81, 2015.

[3] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[4] Andrzej Pronobis, O Martinez Mozos, Barbara Caputo, and Patric Jensfelt. Multi-modal semantic place classification. *The International Journal of Robotics Research*, 29(2-3):298–320, 2010.

[5] Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119(1):3–22, 2016.

[1] https://github.com/zhehangT/SceneSLAM

[6] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010.

[7] Andrzej Pronobis and Patric Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3515–3522. IEEE, 2012.

[8] Ananth Ranganathan. Pliss: Detecting and labeling places using online change-point detection. *Robotics: Science and Systems VI*, 2010.

[9] Jianxin Wu, Henrik I Christensen, and James M Rehg. Visual place categorization: Problem, dataset, and algorithm. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4763–4770. IEEE, 2009.

[10] Jianxin Wu and Jim M Rehg. Centrist: A visual descriptor for scene categorization. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1489–1501, 2011.

[11] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[12] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[14] Sachithra Hemachandra, Matthew R Walter, Stefanie Tellex, and Seth Teller. Learning spatial-semantic representations from natural language descriptions and scene classifications. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2623–2630. IEEE, 2014.

[15] Sudeep Pillai and John Leonard. Monocular slam supported object recognition. *arXiv preprint arXiv:1506.01732*, 2015.

[16] Niko Sünderhauf, Feras Dayoub, Sean McMahon, Ben Talbot, Ruth Schulz, Peter Corke, Gordon Wyeth, Ben Upcroft, and Michael Milford. Place categorization and semantic mapping on a mobile robot. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5729–5736. IEEE, 2016.

[17] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[18] Giorgio Grisettiyz, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437. IEEE, 2005.

[19] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009.

[20] Raul Mur-Artal and Juan D Tardos. Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras. *arXiv preprint arXiv:1610.06475*, 2016.

[21] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[22] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, page 0278364916669237, 2016.

[23] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *Experimental robotics*, pages 477–491. Springer, 2014.

[24] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *arXiv preprint arXiv:1607.02565*, 2016.

[25] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1271–1278. IEEE, 2016.

[26] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[27] Michael J Milford and Gordon F Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1643–1649. IEEE, 2012.

[28] Winston Churchill and Paul Newman. Experience-based navigation for long-term localisation. *The International Journal of Robotics Research*, 32(14):1645–1661, 2013.

[29] Nima Keivan and Gabe Sibley. Online slam with any-time self-calibration and automatic change detection. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5775–5782. IEEE, 2015.

[30] ros.org. pluginlib. `http://wiki.ros.org/pluginlib`. Accessed June 30, 2017.