NANYANG TECHNOLOGICAL UNIVERSITY

# Place Recognition and Localization for Multi-Robot SLAM

**Liu Xiangyu**

**SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING**

**MASTER OF SCIENCE IN COMPUTER CONTROL AND AUTOMATION**

**2019**

# Contents

# Abstract

Abstract goes here.

Here.

**Keywords:** Dissertation, keywords.

# Acknowledgement

Here.

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

SLAM(Simultaneous localization and mapping) is a key component in mobile autonomous systems. It describes the ability of a vehicle, once placed in an unknown environment, to explore and map that environment, while at the same time estimating its own position in the environment, using only its onboard sensing capabilities.

SLAM systems can be accomplished by both single or multiple robots. Multiple-robot SLAM or MRSLAM, offer several advantages compared to there single robot counterpart, for example:

- Robustness to single robot failure,
- Quicker exploration of environments in time critical SaR(Search and Rescue) mission.

However, Adapting SLAM technology to multiple-robot scenario brings some new changes as identified by Saeedi et al [2]:

- Relative Poses of Robots. In multiple-robot SLAM, the map provided by each robot in its own reference coordinates is called the local map. It is difficult task to integrate all of the local maps provided by the other robots to generate a global map of the environment, because the required alignments or transformation matrices, which relate these maps to each other, are in general unknown.

- Closing Loops. Loop closure, is defined as identifying a place observed previously but not very recent. Solving this problem for a team of multiple robots requires using all resources of information from individual robots.in Multi-robot SLAM, various events can trigger loop closure, such as direct encounter of the robots or rendezvous and indirect encounter, when the robots see the same area of features in the world.

- Communications. Availability of a medium for data sharing among robots is an important requirement in multiple-robot SLAM. Information between robots can be exchanged via communication channels. The quality of the communication channels is dependent on the environment. For instance, communication issues are a challenging problem for a team of robots in underwater environments, where the environment imposes limitations on the bandwidth and data rate.

Because of the limitation of the difficulties mentioned above, the development of multiple-robot SLAM is much slower than single-robot SLAM. Finding a solution to these problem will push the adaption of SLAM technology to a new level.

## 1.2 Motivation and Objectives

Recently, some solutions for multi-robot SLAM systems have been proposed. One of them is CORB-SLAM proposed in [1]. But in its initial work, the accuracy of its map fusion results are not evaluate in a quantitative evaluation results, while only a rough demonstration of its mapping results given. Therefore, quantitative evaluation of CORB-SLAM with mapping results of single-robot ORB-SLAM system and CORB-SLAM clients given for comparison can help understand the performance of this multi-robot slam algorithm and assess the feasibility of its potential applications.

Another critical requirement of applications of multi-robot slam algorithms is its ability to fuse sub maps under different illumination conditions and season, which is named as life-long application circumstances. Illumination variance method proposed in [37] with advantages of easy implement and low computational cost, may be able

to help CORB-SLAM to deal with illumination and season changes. Therefore related experiments are undertaken in this work in order to find a effective way to combine CORB-SLAM with illumination variance.

## 1.3 Major contribution of the Dissertation

1. Evaluation of CORB-SLAM on several datasets with quantitative trajectory evaluation results provided.

2. Experiment of combination of CORB-SLAM with illumination variance algorithm to test whether illumination variance method is suitable to enhance ability of CORB-SLAM to deal with illumination changes.

## 1.4 Organisation of the Dissertation

This dissertation is organised into several chapters:

1. Chapter 2 briefly outlines the development of visual SLAM technique. Firstly, the classic structure of visual SLAM system is introduced, and the critical algorithms involved are elaborated, so as the classifications of visual SLAM systems. Then some existing solutions of each category are demonstrated. This chapter also explores prior work in shade dealing algorithms required to implement life-long SLAM.

2. Chapter 3 explains the methodology used in this dissertation to evaluate map fusion performance of CORB-SLAM, and how to combine illumination variance method with CORB-SLAM system to test if illumination variance can be utilized to enhance the ability of CORB-SLAM to deal with illumination changes.

3. Chapter 4 shows the results of

   (a) the evaluation of CORB-SLAM on selected 2 datasets including (i) KITTI Visual Odometry Dataset [4], (ii) NTU Dataset [5],

   (b) the evaluation of CORB-SLAM combined with illumination variance on Oxford

RobotCar Dataset [6].

4. Chapter 5 analysis the results demonstrated in chapter 4 in detail, discussing the improvement and the disadvantages.

5. Chapter 6 concludes the work done in this dissertation, and comments on some limitation and drawbacks of algorithms used in this work, which future work need focus on.

# Chapter 2

# Literature Review

## 2.1 Visual SLAM

### 2.1.1 Introduction

Simultaneous Localizaiton and Mapping (SLAM) is a technique to obtain 3D structure of an unknown environment and sensor motion in the environment. After years of development, SLAM-based application have become widely broadened such as computer vision based 3D modeling, augmented reality(AR)-based visualization and self-driving cars.

In early SLAM algorithms, there exit many different modalities of sensors integrated in SLAM systems, such as rotary encoders, light detection and ranging radar (LiDAR), inertial sensors, GPS and cameras. In recent years, SLAM using cameras only, specifically referred to as visual SLAM (vSLAM), has been actively discussed because the sensor configuration is simple, low-cost, and contains abundant information. But meanwhile this technique also brings more difficulties than others using integrated sensors [7].

vSLAM algorithms have proposed widely in the field of computer vision, robotics and AR. The low requirement on the modalities of sensors, requiring cameras only, is the major advantage of vSLAM technique, so that it is very suitable for low-cost unmanned vehicles, robots with limited load capacity and power supply like drones, or mobile devices such as camera-mounted tablets or smart phones.

However, the difficulties brought by vSLAM can not be ignored. Instead of obtaining depth and location information directly from LiDAR, GPS or depth camera in integrated SLAM systems, vSLAM technique needs to compute all these information from color or gray images, which reduces stability and accuracy for several estimation steps involved in this process. Also obviously the computational cost are significantly higher. Therefore, the problem of how to improve the performance and reduce computational cost of vSLAM has always been widely concerned.

### 2.1.2 Framework

The framework of visual SLAM is mainly composed of five modules as follows:

1. Sensor Data Collection
2. Visual Odometry
3. Global Map Optimization
4. Loop Detection
5. Mapping

This framework is illustrated in Figure 2.1.

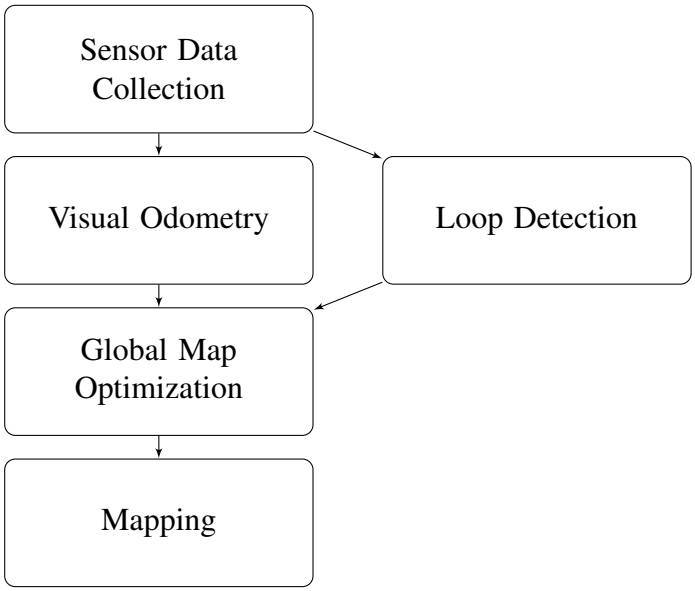



Figure 2.1: Classic structure of Visual SLAM

Sensor data collection module in visual SLAM systems, is responsible to read and preprocess the image information collected from cameras.

In the module of visual odometry (VO), the reconstrcuted map is tracked in the image to estimate the camera pose of the image with respect to the map. In order to do this, feature tracking or feature matching is executed to obtain 2D-3D correspondences between the image and the map. Then, the camera pose is computed by solving the Perspective-n-point (PnP) problem from the correspondences [8, 9].

The other module is loop detection, or loop closing, which is a technique to acquire the reference information. In this module, loop closure is detected by matching a current image with previously acquired images. if a closed loop is detected, it means one of the previously observed place is revisited. In this case, the accumulative error can be estimated. The closed loops and the estimated accumulative error will be sent to the next module of global map optimization.

The next module is global map optimization. The reconstructed map includes accumulative estimation error according to the movement distance of the camera. To suppress the accumulative error, the global map optimization is usually performed. In this module, the map is refined according to the consistency of the entire map. When a place is revisited and a closed loop is detected, reference information that represents the accumulative error can be computed. Then global map optimizer can suppress the accumulative error using loop closure from the reference information as a constraint.

Mapping is the last module. In this module, the map is constructed and expanded by computing the 3D structure of the environment according to the information collected and computed in the prior modules.

### 2.1.3 Related Techniques

#### ORB Features

[10]

#### Bag-of-Words Fast Place Recognition

[11]

**Bundle Adjustment**

[12] [13]

### 2.1.4 Algorithms

According to the different types of the information used in VO, the existing vSLAM algorithms can be categorized into feature-based, direct, and RGB-D camera-based approaches: (i) Feature-based approaches attract and track feature points, (ii) Direct approaches track a whole image without detecting feature points, (iii) RGB-D approaches use both monocular RGB images and its depth. Some popular vSLAM algorithms proposed in recent years are listed as follows:

**Feature-based Approach**

1. MonoSLAM

   MonoSLAM is the first monocular vSLAM, developed in 2003, by Davison et al. [14,15], which is considered as the representative method in filter-based vSLAM algorithms. In MonoSLAM, An extended Kalman filter (EKF) is used to simultaneously estimate the camera motion and 3D structure of an unknown environment. 3D positions of feature points and 6 degree of freedom (DoF) camera motion are represented as a state vector in EKF. The EKF in MonoSLAM assume uniform motion as a prediction model, and a result of feature point tracking as the observation. New feature points are added to the state vector depending on camera movement. The initial map is created by observing a known object where a global coordinate system is defined. In conclusion, there are two components in MonoSLAM system:

   (a) Map initialization, done by using a known object.

   (b) 3D positions of feature points and camera motion, estimated using EKF.

   the limitation of this algorithm is its computational cost increases proportionally with the size of an environment. In large environments, the number of feature points increase causing the size of the state vector to be large. Therefore, real-time performance is difficult to achieve in large environments.

2. PTAM

In order to solve the problem of computational cost in MonoSLAM, Parallel Tracking and Mapping (PTAM) proposed in [16] split the tracking and the mapping tasks into different threads. These two thread are running in parallel, then the computational cost in the mapping thread have no effect on tracking. As a result, bundle adjustment (BA) which requires extra computational cost in optimization can be added in the mapping. This means the tracking thread can estimate camera motion in real-time, and meanwhile the mapping thread can estimate accurate 3D positions of the features points with higher computational cost, with causing no effect on the real-time performance of the tracking thread.

A significant of PTAM is to firstly introduce the concept of keyframe. PTAM is the first vSLAM algorithm to use keyframe-based mapping. In the mapping, 3D positions of new feature points are computed using triangulation at certain frames called keyframes. To achieve accurate triangulation, an input frame is selected as a new keyframe, when a large disparity is measured between the input frame and the previous keyframes. Also, a new relocalization algorithm [17] in tracking is employed in the newer version of PTAM, which uses a randomized tree-based feature classifier to match input frames with keyframes.

In summary, PTAM has the following four components:

(a) Map initialization, done by the five-point algorithm [18].

(b) Camera pose estimation, from matched feature points between the input image and map points.

(c) 3D positions of feature points estimation, by triangulation computation, and optimized by BA.

(d) Tracking process, recovered by a randomized tree-based searching.

3. ORB-SLAM

There have been proposed many SLAM algorithms based on PTAM. ORB-SLAM, one of the most popular vSLAM algorithm proposed in [19, 20], also based on PTAM. The most important improvement of ORB-SLAM compared to PTAM, is

ORB-SLAM employs a new loop detection thread. The details of ORB-SLAM are reviewed in Chapter 2.1.5.

**Direct Approach**

1. DTAM

   Dense Tracking and Mapping (DTAM) was proposed by Newcombe et al. in [21], is fully direct method. In DTAM, the input image is compared with synthetic view images generated from the reconstructed map. This is equivalent to registration of an image on the 3D model of a map, which can be efficiently implemented on GPU. The initial depth map is created using stereo measurement like PTAM. In summary, there are three main components in DTAM:

   (a) Map initialization, done by stereo measurement.

   (b) Camera motion estimation, by synthetic view generation from the reconstructed map.

   (c) Pixel depth information estimation, using multi-baseline stereo, then optimized by considering space continuity.

   DTAM is optimized to achieving real-time processing on mobile phones in [22].

2. LSD-SLAM

   Large-scale direct monocular SLAM (LSD-SLAM), proposed in [23], is another leading direct method, which follows the idea from semi-dense VO [24]. Compared to DTAM which reconstructs full areas, the reconstruction targets are limited to areas only which have intensity gradient in LSD-SLAM. Therefore, in LSD-SLAM, textureless areas are ignored where it is difficult to estimate depth information. In the mapping, initial depth values for each pixel is set to random values, and then optimized considering photometric consistency. In conclusion, four components of LSD-SLAM are as follows:

   (a) Initial depth value, set as random values.

(b) Camera motion, estimated by synthetic view generated from the reconstructed map.

(c) Area reconstructed, limited to high-intensity gradient areas.

(d) 7 DoF pose-graph optimization, employed to obtain geometrically consistent map.

In [25], they optimized the LSD-SLAM algorithms to be able to run on mobile phones with real-time performance, and also evaluate the accuracy for low-resolution images. In [26, 27], LSD-SLAM is extended to stereo camera and omni-directional cameras.

**RGB-D Approach**

Recently, with structure light-based RGB-D cameras such as Microsoft-Kinect getting cheaper and smaller, RGB-D SLAM algorithms with RGB-D camera become more popular and affordable.

1. KinectFusion

   KinectFusion was proposed by Newcombe et al. in [28]. In KinectFusion, a voxel space is used for representing the 3D structure of the environment. The 3D structure of the environment is reconstructed by combining obtained depth maps in the voxel space, and camera motion is estimated by the ICP algorithm using an estimated 3D structure and the input depth map, which is depth-based vSLAM.and it is optimized with GPU to achieve real-time performance.

   In [29], KinectFusion is optimized to run on mobile devices in real time. To

2. SLAM++

   SLAM++ was proposed as an object level RGB-D vSLAM algorithm by Salas-Moreno et al in [30]. In SLAM++, several 3D objects are registered into the database in advance, and these objects are recognized in an online process. The estimated map is refined by recognizing 3D objects, and 3D points are replaced by 3D objects to reduce the amount of data.

As a similar algorithm, another real-time segmentation method for RGB-D SLAM in [31] by Tateno et al. Segmented objects are labeled and then used as recognition targets.

### 2.1.5 ORB-SLAM

One of the state-of-the-art vSLAM solutions for single-robot systems is ORB-SLAM, initially proposed in [19], and upgraded to a second version in [20].

ORB-SLAM is a feature-based monocular SLAM system that operates in real time, in small and large, indoor and outdoor environments. In the proposed work in [19], ORB-SLAM is built on the main ideas of PTAM, the place recognition work of Gálvez-López and Tardós [11], the scale-aware loop closing of Strasdat et. al [32] and the use of covisibility information for large scale operation [33], [34]. As a novel monocular SLAM system, the main contributions of ORB-SLAM are:

1. !!!The same features are used in all tasks: tracking, mapping, relocalization and loop closing. Using same features makes the system more efficient, reliable and simple. And using ORB features allows real-time performance without GPUs, with good invariance to changes in viewpoint and illumination.

2. Real time performance in large environments. The tracking and mapping modules focus in a local covisible area, independent of the global map, thanks to the use of a covisibility graph.

3. Real time loop closing. The optimization of a pose graph called the Essential Graph is adapted to realize real time loop closing performance. The Essential Graph is built from loop closures links, strong edges from the covisibility graph and a spanning tree maintained by the system.

4. Real time camera relocalization with significant invariance to viewpoint and illumination. This allows recovery from tracking failure and also enhances map reuse.

5. A new automatic and robust initialization procedure based on model selection that permits to create an initial map of planar and non-planar scenes.

6. A survival of the fittest approach to map point and keyframe selection that is generous in the spawning but very restrictive in the culling. This policy improves tracking robustness, and enhances life-long operation because redundant keyframes are discarded.

ORB-SLAM system, see on Figure 2.2, incorporates three threads that run in parallel: tracking, local mapping and loop closing.

The tracking thread is in charge of localizing the camera with every frame and deciding when to insert a new keyframe. The module firstly match current frames with previous frames, and optimize the pose using motion-only bundle adjustment. If the

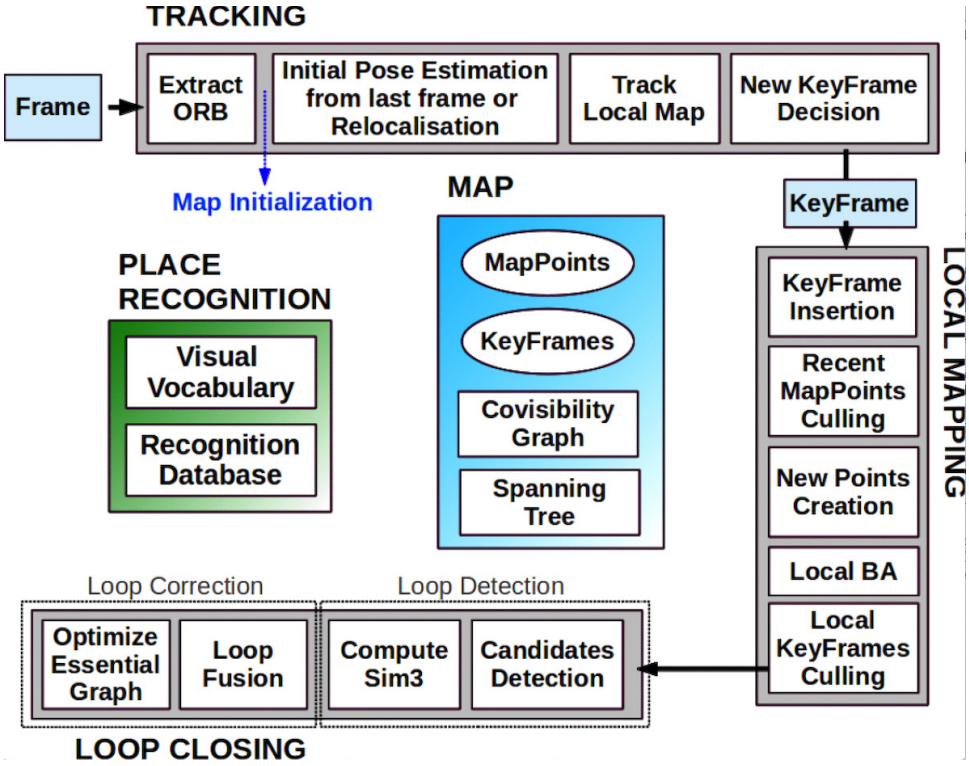



Figure 2.2: ORB-SLAM system overview.

## 2.2 Multi-Robot Algorithms

### 2.2.1 Multi Ground Robot System

### 2.2.2 Multi Hybrid Robot System

### 2.2.3 CORB-SLAM

Proposed by F.Li et al. in [1], CORB-SLAM is a vSLAM algorithm focusing on multi-robot systems. As presented in Figure 2.3, the system of CORB-SLAM consists of robot-end clients and a server.

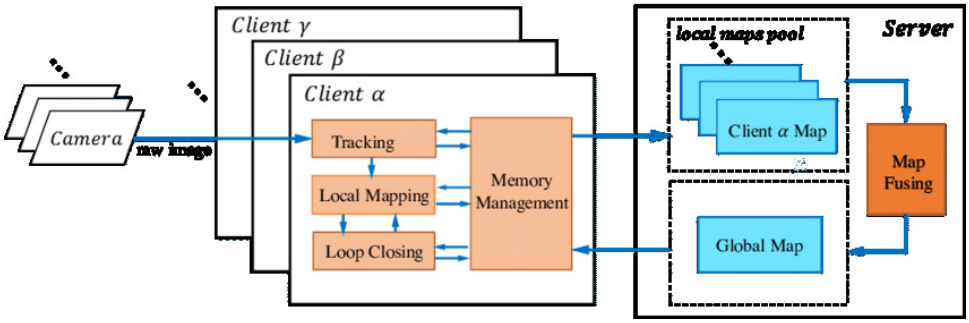



Figure 2.3: The framework of CORB-SLAM system.

#### Robot-end SLAM Client

The robot-end client of CORB-SLAM is an ORB-SLAM client extended to have the functionality to communicate with the server, transmitting the keyframe information, with all functions and modules in original ORB-SLAM as listed in Chapter 2.1.5 reserved.

#### Map Fusing in the Server

In the server, the map fusing module receives and fuses the local maps from the clients, achieving an optimized global map. The map fusing algorithm is shown in Figure 2.4, including two main parts: map overlap detection and local-map fusion.

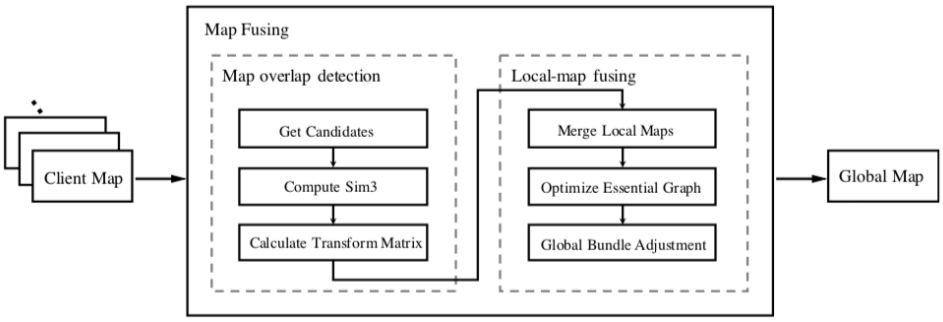



Figure 2.4: The flowchart of Map Fusing module.

1. Initializing global map

   Initially,

2. Map overlap detection

   To

3. Local map fusing

   To

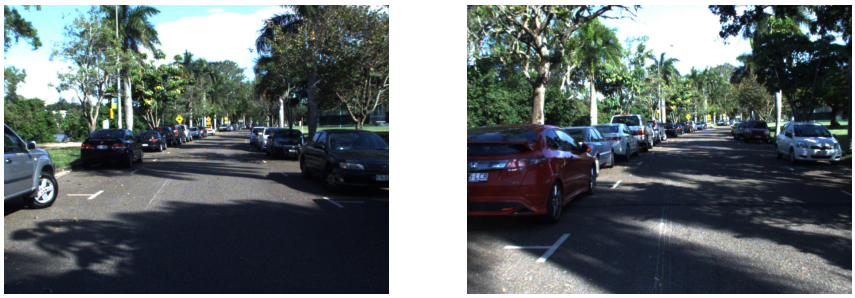
## 2.3 Illumination Variance

### 2.3.1 Appearance Change From Illumination

For vision systems concerned with localizing in known environments, dealing with appearance changes is an ongoing challenge. Appearance changes can result from several sources, such as (i) different lighting conditions, (ii) varying weather conditions, and (iii) dynamic objects like pedestrians or vehicles.

In previous work of Colin McManus et al. , they demonstrated how to leverage knowledge of prior 3D structure to suppress distracting objects for improved pose estimation in busy urban environments [35], and how to cope with long-term appearance variation caused by changing weather conditions [36]. In [37], they proposed a new approach to address problem (i named as Illumination Variance Approach.

Appearance change caused by different lighting conditions in (i) is illustrated in Figure 2.5 with pictures selected from St Lucia dataset [38]. Compared to approaches



(a) pic1. (b) pic2.

Figure 2.5: Appearance changes caused by different lighting conditions. Pictures are selected from St Lucia dataset corresponding to the car rides recorded on 10/09/2009 at 8:45 am and at 2:10 pm

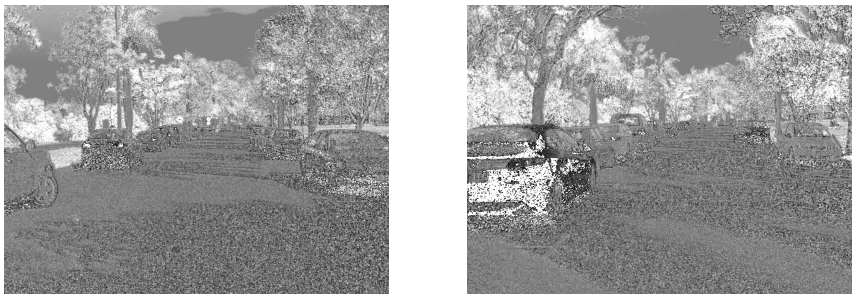
proposed in [35] and [36], illumination variance approach is not model-based, requiring less computational cost. And in most of applications of vSLAM, appearance changes caused by (i) are a more common problem than (ii)(iii). Therefore, how to combine illumination variance approach with multi-robot SLAM algorithms, to improve the performance of place recognition in changing illumination conditions, is the major objective focused on in this work.

### 2.3.2 Illumination Variance

Illumination variance approach proposed in [37], is a simple method based on only one equation computing illumination variant images. The basic idea of this approach is to map color images to an illumination invariant color space, where illumination change caused by different lighting condition like shade can be suppressed. The mapping equation is presented in Equation 2.1.

$$I = \log(G) - \alpha \log(B) - (1-\alpha)\log(R) \tag{2.1}$$

where, $R, G, B$ are the color channels of the input image, and $I$ is the resultant illumination invariant image. As shown in 2.2, $\alpha$ is a parameter which depends on the peak spectral responses of each color channel $(\lambda_R, \lambda_G, \lambda_B)$, which are usually available in camera specifications.



(a) pic1. (b) pic2.

Figure 2.6: An example of illumination invariance application in St Lucia dataset. It shows how this approach suppress the effects caused by sun

$$\frac{1}{\lambda_G} = \frac{\alpha}{\lambda_B} + \frac{1-\alpha}{\lambda_R} \tag{2.2}$$

Therefore, considering the peak spectral responses, $\alpha$ can be easily calculated as exposed in Equation 2.3.

$$\alpha = \frac{(\frac{\lambda_B}{\lambda_G} - \frac{\lambda_B}{\lambda_R})}{(1 - \frac{\lambda_B}{\lambda_R})} \tag{2.3}$$

The influence of applying the illumination invariant transformation is showed in Figure 2.6.

### 2.3.3 ??? Life-Long SLAM

An open source toolbox named as OpenABLE, for life-long visual localization is implemented in [39]. The proposed implementation in [39] employs the philosophy of the topological place recognition approach named ABLE introduced in [40–42] which uses illumination variant images for relocalization.

A graphic representation about how the methodology proposed by OpenABLE is showed in Figure 2.7.

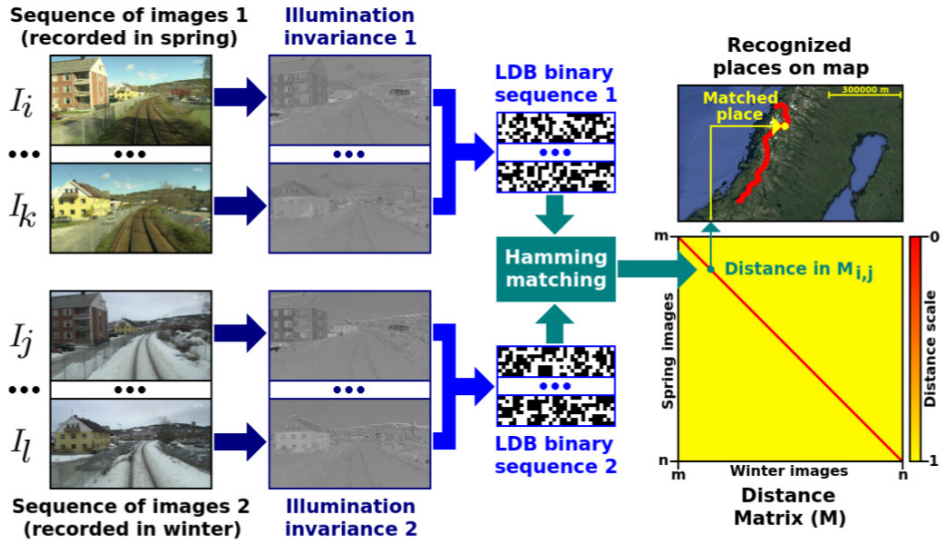



Figure 2.7: A graphic representation about how the methodology proposed by ABLE works.

The limitation of illumination variance approach is the transformation process produces resultant images with low resolution because all pixel values are turned into log values. These low-resolution resultant images still can be used as the input images of visual topological localization where high resolution images are actually not needed. But in the mapping task, illumination variant images are too blurry to estimate camera motion and then reconstruct the map. Therefore, to improve the mapping performance in changing illumination conditions, rgb images and illumination variant images are both needed to perform relocalization and mapping, as the block-flow proposed in [3] presented in 2.8.

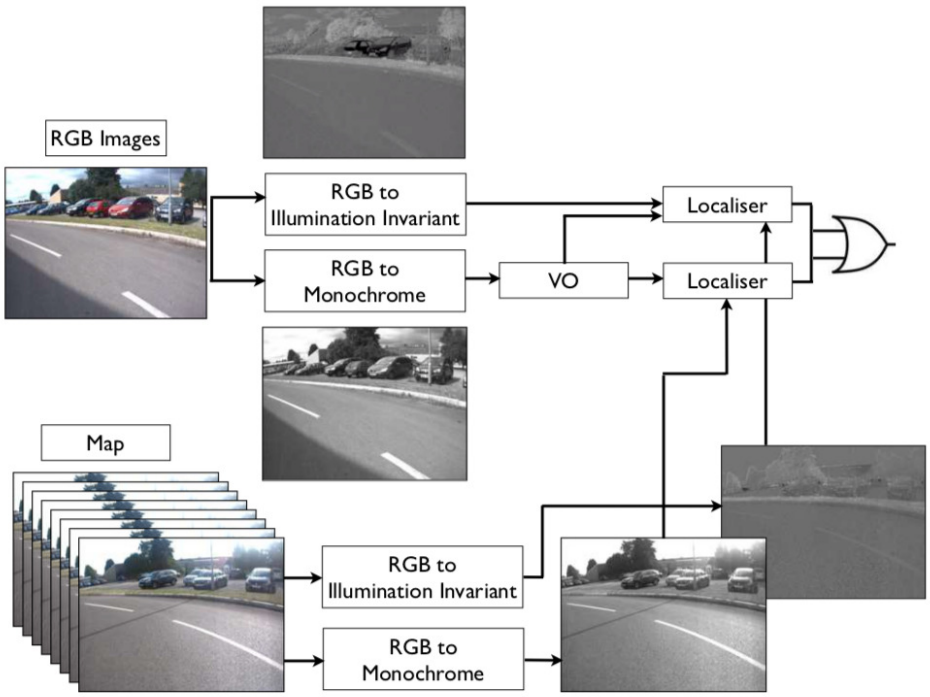



Figure 2.8: Block-flow diagram of the combined stereo localisation approach.

In the framework presented in 2.6, there is a second localizer making use of illumination invariant images in parallel with the main localization system. In [37], although the metric relative poses calculated from illumination variant images tends wo be more noisy, the integrated localizer are less likely to fail if the scene appearance change is due to sunlight intensity direction or spectrum variation.

# Chapter 3

# Approach

## 3.1 Quantitative Trajectory Evaluation Method

To evaluate the mapping performance, the proposed method in [43] is modified to multi robot case, and employed in this work.

The previous work of CORB-SLAM in [1] only provides a rough overview of the mapping result of the multi robot system, as seen in Figure 3.1.

In [37] and [39], the results of the illumination variance localization are also only briefly introduced, with no quantitative results given.

In this paper, mapping results of CORB-SLAM and CORB-SLAM integrated with illumination variance are evaluated following the quantitative trajectory evaluation method proposed in [43]. Quantitative evaluation results of each datasets are demonstrated in several figures and tables including contents as follows, and see Section 4.2.1 as an example:

1. Ground truth trajectories of each partial sequence and the complete dataset for reference, e.g. Figure 4.8.

2. Mapping results of each client and fused map in server end, compared with ground truth trajectories, e.g. Figure 4.13.

3. Four charts of quantitative results, e.g. Figure 4.14, including

    1). Chart of relative translation error in meter, e.g. Figure 4.14(a).

2). Chart of relative translation error in percent, e.g. Figure 4.14(b).

3). Chart of relative yaw error in degree, e.g. Figure 4.14(c).

4). Chart of scale error in percent, e.g. Figure **??**.

4. A table presenting numeric results of charts in 3, e.g. Table 4.6.

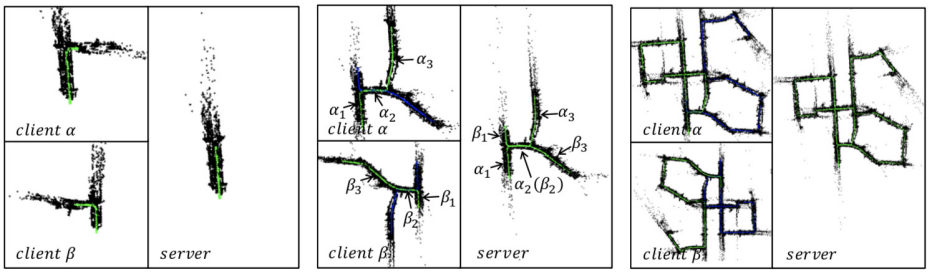
5. Charts of quantitative results of mapping each partial sequences in each client, in the same format of 3, e.g. Figure 4.10.

6. A table presenting numeric results of charts in 5, e.g. Table 4.4.

7. (if applicable) The mapping results of CORB-SLAM mapping the entire sequence without partial sequences for reference, e.g. Figure 4.12.

8. (if applicable) Charts of quantitative results of mapping the entire sequence for reference, in the same format of 3, e.g. Figure 4.9.

9. (if applicable) A table presenting numeric results of charts in 8, e.g. Table 4.3.




(a) 10s (b) 80s (c) 400s

Figure 3.1: Mapping results of CORB-SLAM in [1].

The main task of map fusion module of multi robot SLAM server is to find relative relations including rotation and transformation matrices between client maps, based on which client maps are fused into a consist global map. Therefore, inaccurate rotation and transformation result in an inaccurate fused global map where all client maps but the one set to be the initial global map are stitched dramatically offsetting the ground

truth trajectories, which therefore causes serious translation and yaw error. Therefore in charts and tables of quantitative results, three types of relative errors are selected:

1. Relative Translation Error.

2. Relative Translation Error in percent.

3. Relative Yaw Error.

Computation of relative error follows the basic idea stated in [43], that the estimation quality can be measured by computing relative relations between states in different times, since no global reference including global position and yaw, is provided in VO systems.

Formally, given the ground truth trajectory $\boldsymbol{X}$ and the estimated trajectory $\hat{\boldsymbol{X}}$:

$$\boldsymbol{X} = \{\boldsymbol{x}_i\} = \{R_i, p_i, v_i\}, \quad i = 1, ..., t. \tag{3.1}$$

where, $p_i \in \Re^3$ is the position, $R_i \in SO(3)$ is the rotation matrix and $v_i \in \Re^3$ is the velocity of the system.

the following steps should be taken to explain and calculate relative errors:

Step 1. Given the ground truth positions$p_i$ and estimated positions $\hat{p}_i$, the estimated trajectory can be aligned to the aligned estimated trajectory $\hat{\boldsymbol{X}}'$ by finding a similarity transformation $S' = \{s', R', t'\}$ that minimize:

$$S' = \underset{S=\{s,R,t\}}{\arg\min} \sum_{i=0}^{N-1} \|p_i - sR\hat{p}_i - t\|^2. \tag{3.2}$$

Then,

$$\hat{R}'_i = R'\hat{R}'_i, \quad \hat{p}'_i = s'R'\hat{p}'_i + t', \quad \hat{v}'_i = s'R'\hat{v}_i. \tag{3.3}$$

Step 2. A set of K pairs of states, each of which defines a sub trajectory, need to be selected from $\hat{\boldsymbol{X}}$ by a criteria e.g. distance or duration traveled:

$$\boldsymbol{\xi} = \{\boldsymbol{d}_k\}_{k=0}^{K-1}, \quad \boldsymbol{d}_k = \{\hat{\boldsymbol{x}}_s, \hat{\boldsymbol{x}}_e\}, \quad e > s \tag{3.4}$$

Step 3. Then for each $\boldsymbol{d}_k$, a relative error $\delta\boldsymbol{d}_k$ is calculated by:

$$
\begin{aligned}
\delta \boldsymbol{d}_k &= \{\delta\phi, \delta p_k, \delta v_k\}, \\
\delta\phi_k &= \angle \delta R = \angle R_e (\hat{R}'_e)^\top, \\
\delta p_k &= \| p_e - \delta R_k \hat{p}'_e \|_2, \\
\delta v_k &= \| v_e - \delta R_k \hat{v}'_e \|_2.
\end{aligned}
\tag{3.5}
$$

Step 4. Collecting the relative errors in Equation 3.5 for all pairs in $\boldsymbol{\xi}$ gives:

$$
\begin{aligned}
RE_{rot} &= \{\delta\phi_k\}_{k=1}^{K-1}, \\
RE_{pos} &= \{\delta p_k\}_{k=0}^{K-1}, \\
RE_{vel} &= \{\delta v_k\}_{k=0}^{K-1},
\end{aligned}
\tag{3.6}
$$

which is illustrated in Figure 3.2. relative translation errors in meter and percent can be extracted from $RE_{pos}$, and relative yaw errors from $RE_{rot}$.

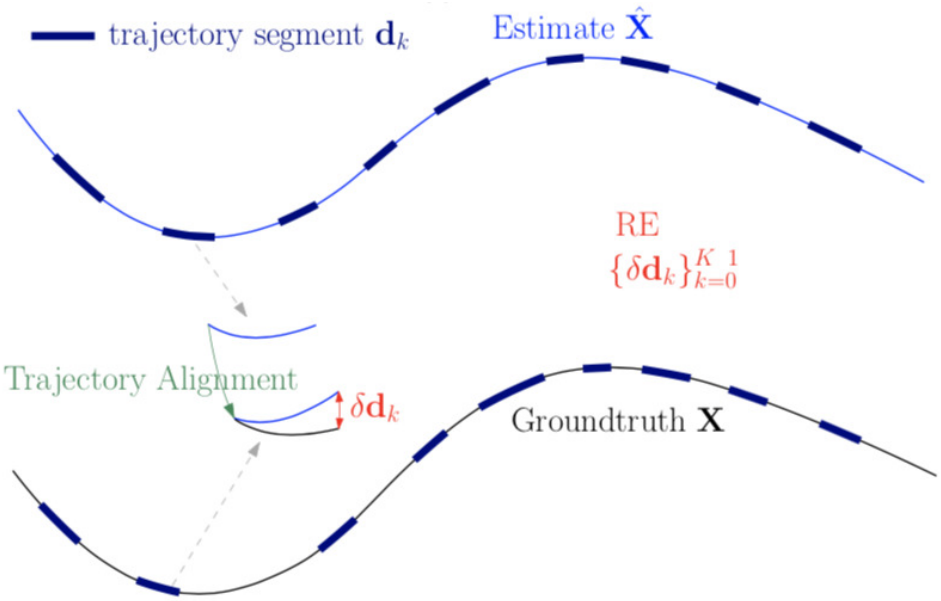
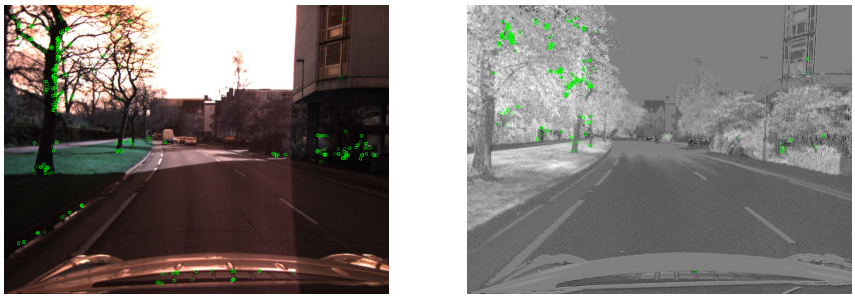



Figure 3.2: Illustration of relative error.

## 3.2 CORBSLAM with Illumination Variance

To experiment whether illumination variance method is possible to be utilized to enhance the ability of CORB-SLAM to map in different illumination conditions and seasons, it is combined into the map fusion modules of CORB-SLAM server in this work.

The block diagram of the integrated system is illustrated in Figure 3.3.



(a) Keypoints extracted from an rgb image. (b) Keypoints extracted from an illumination variance image.

Figure 3.4: Keypoints extracted from rgb images and illumination variance images.

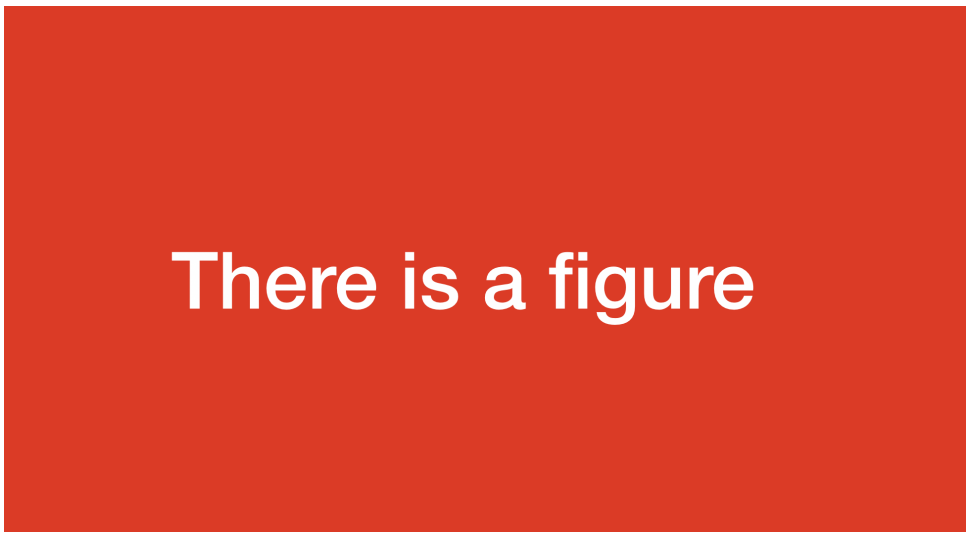



Figure 3.3: The block diagram of the modified system of CORB-SLAM integrated with illumination variance.

In the client end, the following modifications are made:

1. A new thread running in parallel is added to process the input frame to transform into illumination variance images, and extract ORB keypoints in produced images (named as *II keypoints* in this paper), as illustrated in Figure 3.4.

2. II keypoint data is added into each frame as new member variables. And then following the CORB-SLAM methodology, only the integrated keyframes are transmitted to the server, which are serialized and packed by boost serialization library, and transmitted through ROS service.

Besides the above changes, the following modification are made in the server end:

1. A new keydataset containing II keypoint information.

2. A new illumination variance localizer running in parallel with the rgb localizer. When processing the input keyframe, a new localizer thread is started if the rgb localizer returns no result. The results from rgb and illumination variance localizer are added together in this work.

3. Since extracted II keypoint positions are not accurate as rgb keypoints, transform matrix is still trying to calculate from rgb keypoints where overlapping is detected.

# Chapter 4

# Test and Experiments

## 4.1 Datasets

Evaluation are performed in several datasets including KITTI Visual Odometry 2012 dataset [4], Oxford RobotCar dataset [6] and NTU dataset collected in NTU. The listing of used datasets is shown in Table 4.1

### 4.1.1 KITTI Visual Odometry Dataset

KITTI Visual Odometry 2012 is a part of KITTI Vision Benchmarck Suite, presented in [4,44]. KITTI datasets are captured by driving around a mid-size city, in rural areas and on highways. The recording platform is equipped with two high resolution stereo camera systems, capturing color and gray images, a Velodyne HDL-64E LIDAR, and an OXTS RT 3003 localization system which combines GPS, GLONASS, an IMU and RTK correction signals.

KITTI Visual Odometry Evaluation 2012 provides 11 sequences with ground truth trajectories for training, and another 11 sequences without ground truth for evaluation.

Table 4.1: Information of datasets used

| Datasets | Settings | Approx Scale | Diversity |
|---|---|---|---|
| KITTI | rural area | $<$ 1 hour | one city, one weather condition, daytime |
| Oxford | city | 214 hours | one city, multiple weather conditions, daytime |
| NTU | campus | $<$ 1 hour | one campus (NTU), one weather condition |

Example images are shown in Figure 4.1.


Figure 4.1: Example image in KITTI Visual Odometry 2012 dataset.

### 4.1.2 Oxford RobotCar Dataset

Oxford RobotCar Dataset is presented by Will Maddern et al. in [6], as a challenging dataset for autonomous driving. Collected over the period of May 2014 to December 2015, this datasets recorded images from 6 cameras mounted Nissan LEAF, along with LIDAR, GPS and INS ground truth. Images were recorded under different weather and illumination condition from 9:00 to 16:00 on average, from May to December. Example images is shown in Figure 4.2.


Figure 4.2: Example image in robotcar dataset.

The RobotCar platform is a Nissan LEAF equipped with sensors as following [6]:

1. Stereo Camera: Bumblebee XB3 trinocular stereo camera $\times$ 1, 1/3" Sony ICX445 CCD, 1280$\times$960$\times$3, 16Hz, 3.88mm lens, 66° HFoV, 12/24cm baseline, global shutter.

2. Monocular Camera: Grasshopper2 $\times$ 3, 2/3" ICX285 CCD, 1024$\times$1024, 11.1Hz, 2.67mm fisheye lens, 180°, global shutter.

3. 2D LIDARL SICK LMS-151 2D LIDAR $\times$ 2, 270° FoV, 50Hz, 50m range, 0.5° resolution.

4. 3D LIDAR: SICK LD-MRS 3D LIDAR $\times$ 1, 85° HFoC, 3.2° VFoV, 4 panes, 12.5Hz, 50m range, 0.125° resolution.

5. GPS/INS Module: NovAtel SPAN-CPT ALIGN inertial and GPS navigation system $\times$ 1, 6 axis, 50Hz, GPS/GLONASS, dual antenna.

The RobotCar platform and the sensor locations are demonstrated in Figure 4.3.

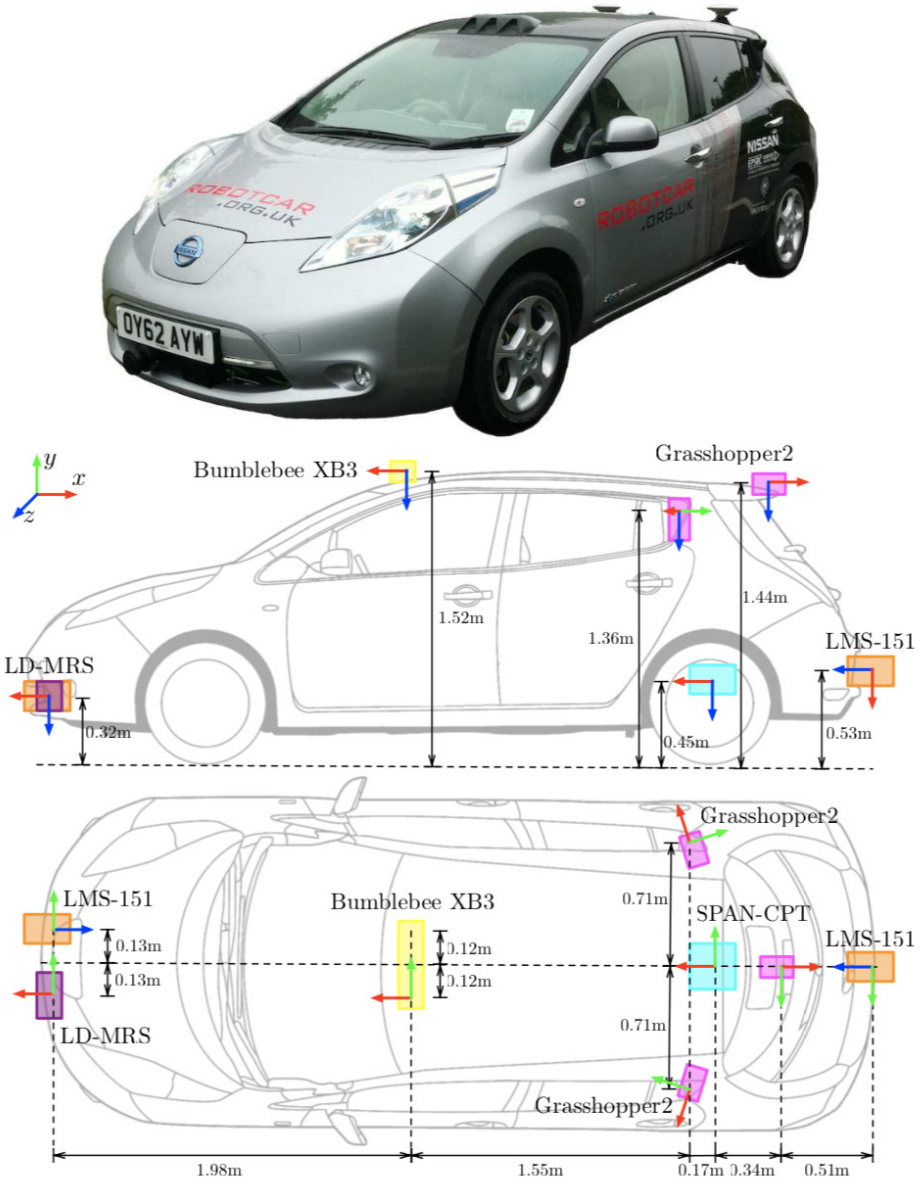
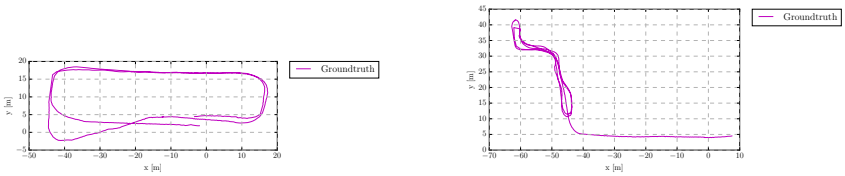



Figure 4.3: The robotcar platform and sensor location diagram.

RobotCar dataset is especially suitable to evaluate life-long SLAM systems, since it contains images taken in different hours of daytime under different illumination conditions, and in different seasons. The comparison between image in different illumination conditions and different seasons is shown in Figure 4.4.

(a) Image captured in 14:49 07/14/2014. (b) Image captured in 12:32 02/24/2015.

Figure 4.4: Comparison of images captured in the same location in different seasons in RobotCar dataset.




(a) Ground truth trajectory of Bag.0. (b) Ground truth trajectory of Bag.1.

Figure 4.5: Ground truth information of each rosbags in NTU Dataset.

### 4.1.3 NTU Dataset

Our NTU dataset [5] is collected by multi ground robots consisting of two husky UGV platforms, recording driving around the carpark in front of School of EEE building.

Our UGV platform is a HUSKY Clearpath robot, equipped with a ZED stereo camera $\times$ 1, 672$\times$376, 87° HFoV, 56° VFoV. The picture of the platform and example images are shown in Figure 4.6 and 4.7.

The dataset provides 4 rosbag files. 3 of them are recorded by UGV, while the other one is recorded by UAV. The basic information of 4 rosbags are listed in Table 4.2. And the ground truth trajectories are shown in Figure 4.5.

Table 4.2: Main characteristics of the rosbags in NTU Dataset used in the experiment.

| Bag No. | Data(M/D/Y) | Platform | Height(m) | Dep. Angle |
|---|---|---|---|---|
| 0 | 10/27/2018 | UGV | $\approx 0.7m$ | 0° |
| 1 | 10/27/2018 | UGV | $\approx 0.7m$ | 0° |

Figure 4.6: Overview picture of NTU Husky platform.

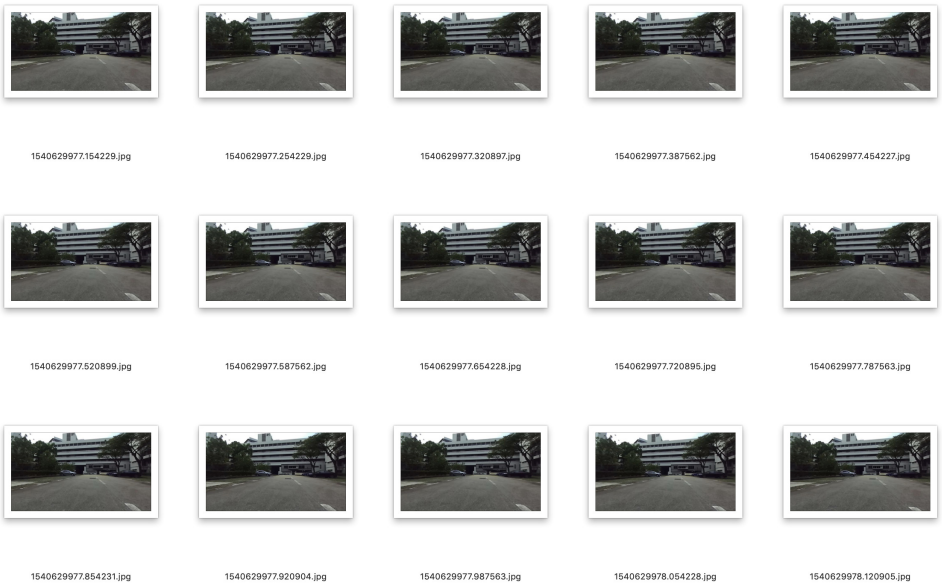



Figure 4.7: Example images of NTU dataset.

Table 4.3: Quantitative results of mapping unseparated Sequence 00.

| Distance(m)[1] | Rel. Trans.(m)[2] | Rel. Trans.(%)[3] | Rel. Yaw(deg)[4] |
|---|---|---|---|
| 371 | 229.69 | 61.91 | 0.37 |
| 742 | 260.10 | 35.05 | 0.37 |
| 1113 | 260.20 | 23.38 | 0.31 |
| 1485 | 240.93 | 16.22 | 0.40 |
| 1856 | 255.16 | 13.74 | 0.32 |

[1] Distance in meter traveled before each time of statistics.
[2] Mean relative translation error in meter.
[3] Mean relative translation error in percent.
[4] Mean relative yaw error in degree.

## 4.2 Evaluation of CORBSLAM

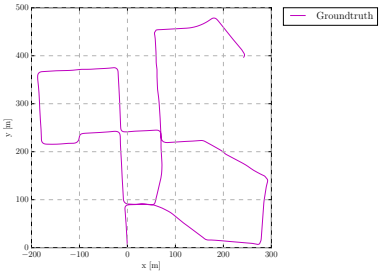
### 4.2.1 KITTI Datasets

In order to evaluate CORB-SLAM system, sequence 00 is utilized and separated into two sub sequences with proper length of overlap. The following separating method is employed: We assume the time period of a KITTI sequence if $Seq.0[0,t]$. Then the sequence is separated into two sub sequences $Seq.01[0,\frac{t}{2}+\delta t]$ and $Seq.02[\frac{t}{2},t]$ as the assumed input of two client robots.

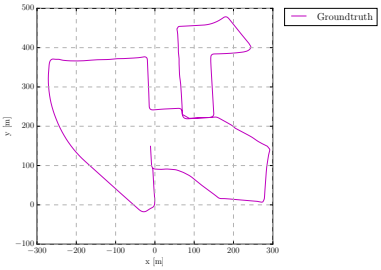
Therefore, in this case, Sequence 00 containing $f = 4541 frames$ and covering a total distance of $s = 1856m$ is separated into two partial sequences: Seq.0$[0,\frac{2}{3}f]$ and Seq.1$[\frac{1}{3}f,f]$, both containing $\frac{2}{3}f \approx 3027 frames$ and covering distances of $\frac{2}{3}s \approx 1237m$ (a rough estimate since distances between each pair of frames are not equal).

The ground truth information of Seq.0, Seq.1 and the complete ground truth trajectory of Sequence 00 are shown for reference in Figure 4.8. And Figure 4.13 demonstrates mapping results of each partial sequence and the map fusion results of the server. Four charts in Figure 4.14 contains quantitative evaluation results, with corresponding numeric results shown in Table 4.9.

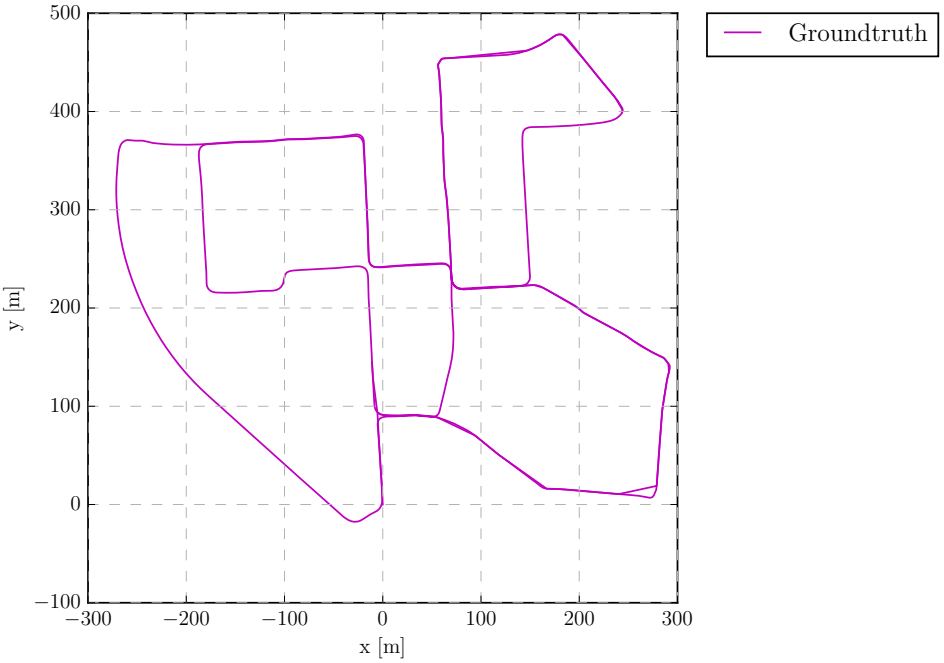
Clients' quantitative mapping results of each partial sequence are provided in Figure 4.10, 4.11 and Table 4.4, 4.5. And because the two partial sequences are extracted from Sequence 00, so the completed mapping results of CORB-SLAM client on Sequence 00 can be provided as a comparison by Figure 4.9, 4.12 and Table 4.3 in the same format as above. Results are further discussed in Section 5.1.




(a) Ground truth trajectory of Seq.0.

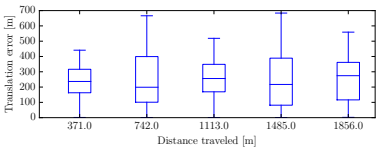



(b) Ground truth trajectory of Seq.1.

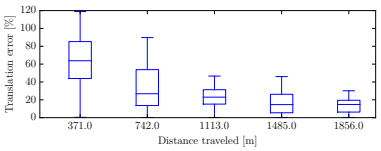



(c) Ground truth trajectory of complete Sequence 00.

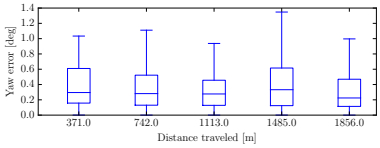
Figure 4.8: Ground truth trajectory of partial and complete sequences of KITTI Datasets.




(a) Relative translation error.

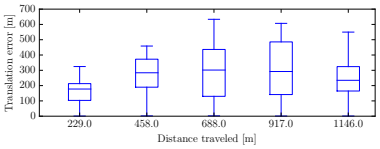



(b) Relative translation error by percent.

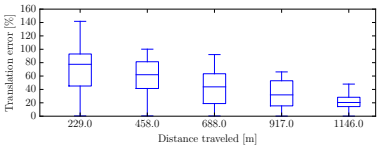



(c) Relative yaw error.

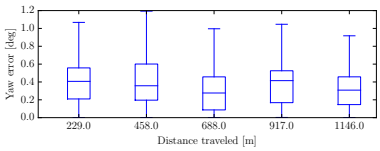
Figure 4.9: Quantitative evaluation results of CORB-SLAM client mapping the entire KITTI Sequence 00.




(a) Relative translation error.

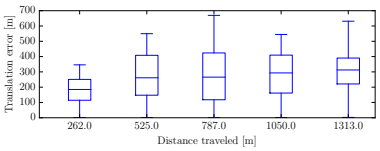



(b) Relative translation error by percent.

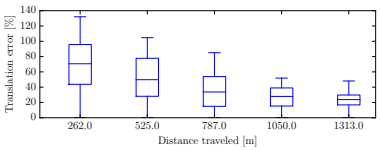



(c) Relative yaw error.

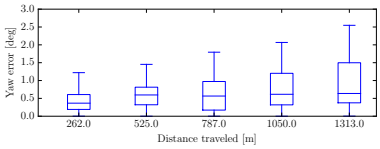
Figure 4.10: Quantitative evaluation results of CORB-SLAM client mapping KITTI partial Seq.0.




(a) Relative translation error.




(b) Relative translation error by percent.




(c) Relative yaw error.

Figure 4.11: Quantitative evaluation results of CORB-SLAM client mapping KITTI partial Seq.1.

Table 4.4: Quantitative results of mapping Seq.0.

| Distance(m)[1] | Rel. Trans.(m)[2] | Rel. Trans.(%)[3] | Rel. Yaw(deg)[4] |
|---|---|---|---|
| 229 | 161.01 | 70.31 | 0.40 |
| 458 | 267.43 | 58.38 | 0.42 |
| 688 | 299.07 | 43.47 | 0.31 |
| 917 | 306.37 | 33.41 | 0.39 |
| 1146 | 256.09 | 22.35 | 0.34 |

[1] Distance in meter traveled before each time of statistics.
[2] Mean relative translation error in meter.
[3] Mean relative translation error in percent.
[4] Mean relative yaw error in degree.

Table 4.5: Quantitative results of mapping Seq.1.

| Distance(m)[1] | Rel. Trans.(m)[2] | Rel. Trans.(%)[3] | Rel. Yaw(deg)[4] |
|---|---|---|---|
| 262 | 229.41 | 114.28 | 0.40 |
| 525 | 412.58 | 78.59 | 0.57 |
| 787 | 386.45 | 49.10 | 0.61 |
| 1050 | 390.23 | 37.16 | 0.77 |
| 1313 | 464.24 | 35.36 | 0.89 |

[1] Distance in meter traveled before each time of statistics.
[2] Mean relative translation error in meter.
[3] Mean relative translation error in percent.
[4] Mean relative yaw error in degree.

Table 4.6: Quantitative results of map fusion evaluation on KITTI partial sequences.

| Distance(m)[1] | Rel. Trans.(m)[2] | Rel. Trans.(%)[3] | Rel. Yaw(deg)[4] |
|---|---|---|---|
| 526 | 283.40 | 53.88 | 0.46 |
| 1053 | 276.16 | 26.23 | 0.51 |
| 1580 | 153.74 | 9.73 | 0.48 |
| 2106 | 284.95 | 13.53 | 0.57 |
| 2633 | 219.66 | 8.34 | 0.54 |

[1] Distance in meter traveled before each time of statistics.
[2] Mean relative translation error in meter.
[3] Mean relative translation error in percent.
[4] Mean relative yaw error in degree.

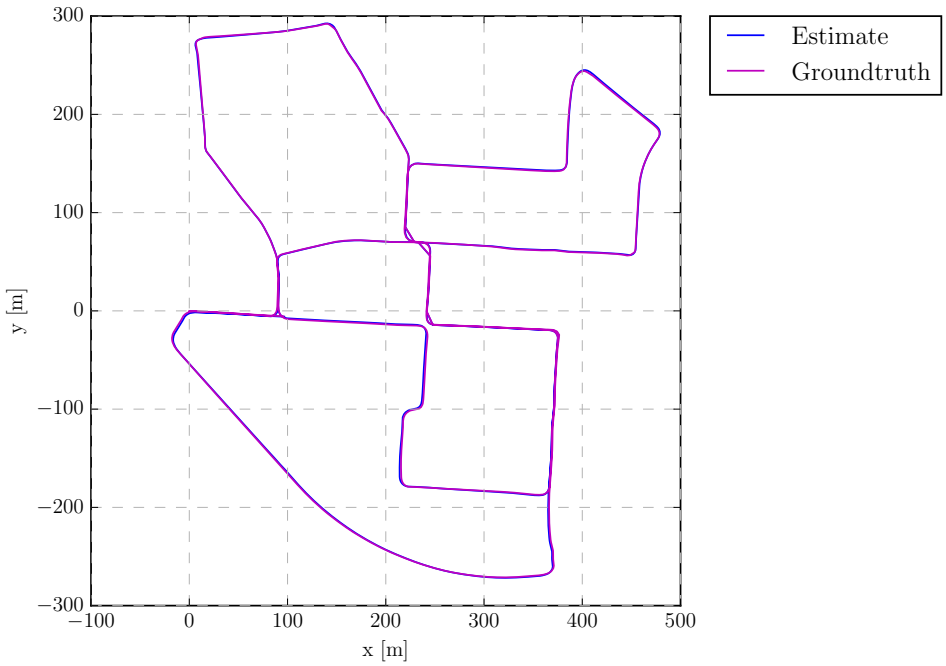
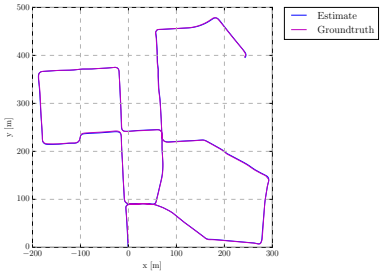
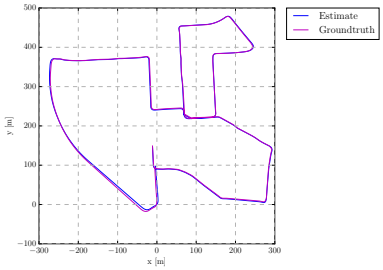


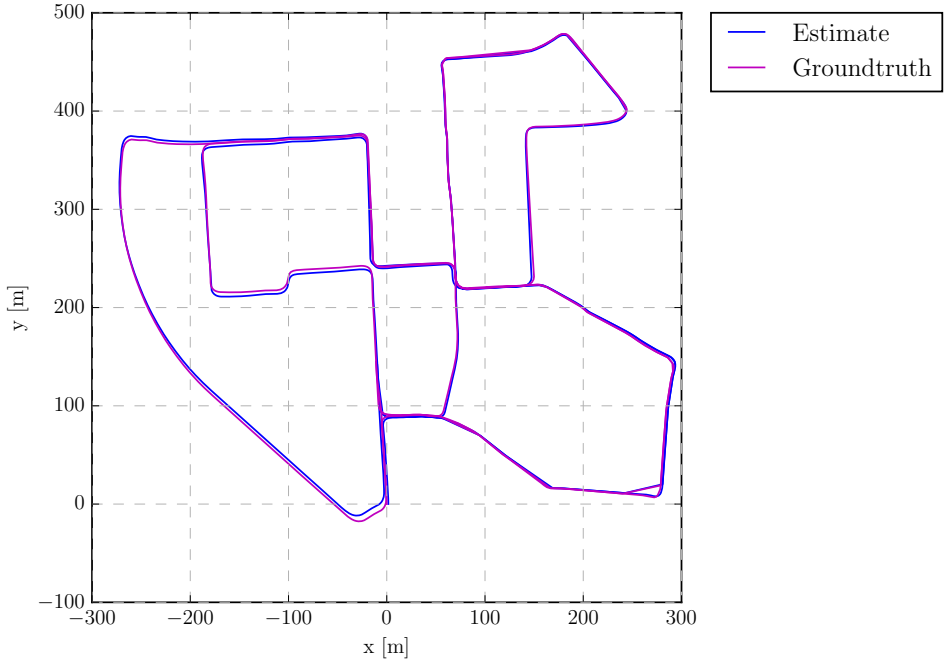

Figure 4.12: Mapping results of the entire sequence without partial sequence.




(a) Mapping result of Seq.0 compared with ground truth.

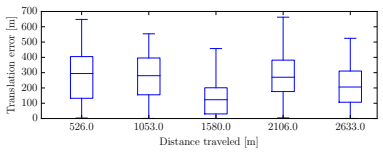



(b) Mapping result of Seq.1 compared with ground truth.

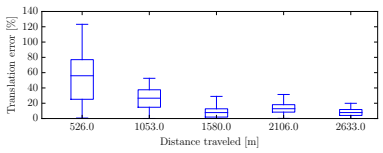



(c) Map Fusion results of Seq.0 and Seq.1 compared with ground truth.

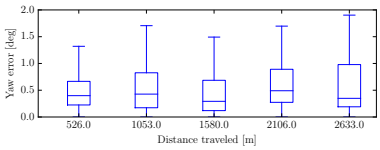
Figure 4.13: Mapping results of Seq.0 and Seq.1, and the map fusion results of KITTI Datasets.




(a) Relative translation error.




(b) Relative translation error by percent.




(c) Relative yaw error.

Figure 4.14: Quantitative evaluation results of fused map of KITTI Datasets.

Table 4.7: Quantitative results of mapping evaluation on Bag.0 NTU Datasets.

| Distance(m)[1] | Rel. Trans.(m)[2] | Rel. Trans.(%)[3] | Rel. Yaw(deg)[4] |
|---|---|---|---|
| 29 | 28.75 | 99.13 | 45.55 |
| 58 | 38.97 | 67.19 | 34.89 |
| 88 | 51.96 | 59.05 | 43.74 |
| 117 | 37.88 | 32.38 | 36.31 |
| 147 | 9.29 | 6.32 | 10.71 |

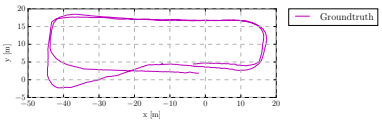
[1] Distance in meter traveled before each time of statistics.
[2] Mean relative translation error in meter.
[3] Mean relative translation error in percent.
[4] Mean relative yaw error in degree.

### 4.2.2 NTU Datasets

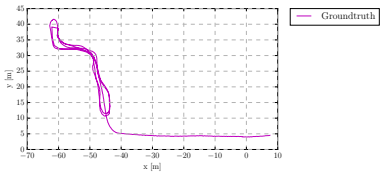
An obvious drawback of the evaluation on KITTI dataset is the images which are overlapped by two clients are exactly identical because they are extracted from the same sequence. Therefore, the results are expected to be much better than real-world applications in which case it is impossible the images recorded by different clients can be identical.

In order to get more reliable and convincing evaluation results of CORB-SLAM system, another evaluation on multi ground robots is performed utilizing NTU Datasets. Bag.0 and Bag.1 described in Section 4.1.3 are selected in this test. These two bags recorded by two UGVs, have different starting and ending location, with limited overlapping, which is much more similar to the case of real-world applications. Clients' mapping results and map fusion results in the server end compared to ground truth trajectories are demonstrated in Figure 4.16. And ground truth information is provided in Figure 4.15 for reference. Quantitative results of the fused global map are represented in Figure 4.19 and Table 4.9. Quantitative results of each client are provided in Figure 4.17, 4.18 and Table 4.7, 4.8 in the same format as above.

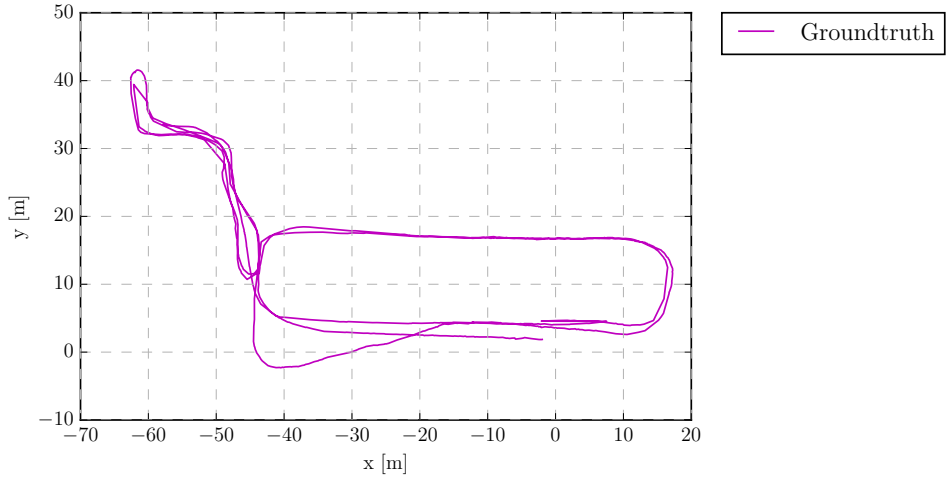
Results are further discussed in Section 5.1.




(a) Ground truth trajectory of Bag.0.




(b) Ground truth trajectory of Bag.1.



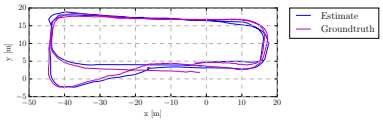

(c) Complete ground truth trajectory of Bag.0 and Bag.1.

Figure 4.15: Ground truth trajectory of partial and complete bags of NTU Datasets.

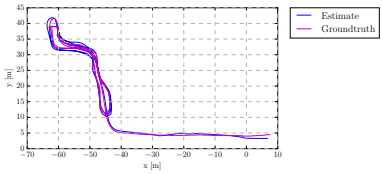
Table 4.8: Quantitative results of mapping evaluation on Bag.1 NTU Datasets.

| Distance(m)[1] | Rel. Trans.(m)[2] | Rel. Trans.(%)[3] | Rel. Yaw(deg)[4] |
|---|---|---|---|
| 25 | 24.79 | 99.17 | 71.03 |
| 51 | 37.20 | 72.94 | 88.44 |
| 76 | 29.90 | 39.34 | 49.64 |
| 102 | 34.40 | 33.73 | 67.14 |
| 127 | 38.99 | 30.70 | 91.89 |

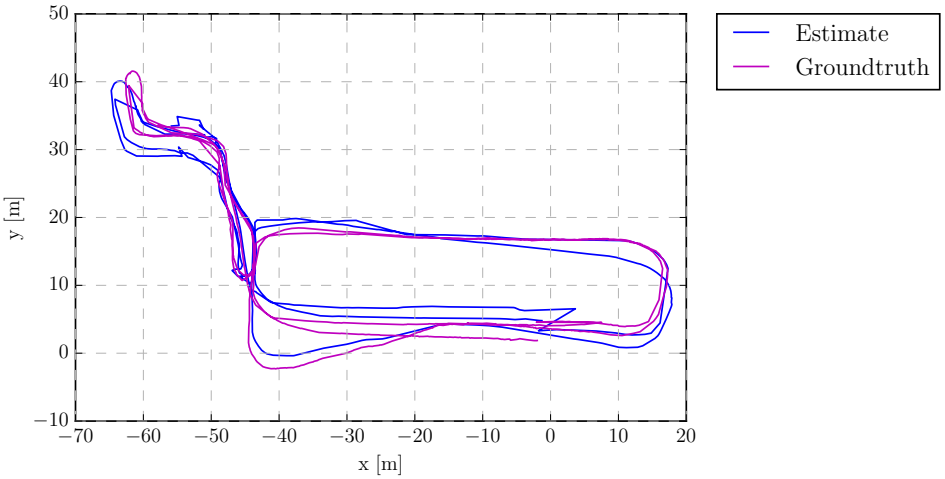
[1] Distance in meter traveled before each time of statistics.
[2] Mean relative translation error in meter.
[3] Mean relative translation error in percent.
[4] Mean relative yaw error in degree.




(a) Mapping result of Bag.0 compared with ground truth.

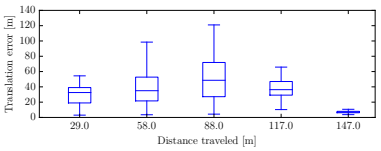



(b) Mapping result of Bag.1 compared with ground truth.

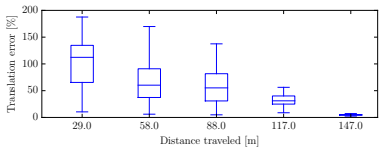



(c) Map Fusion results in server end of Bag.0 and Bag.1.

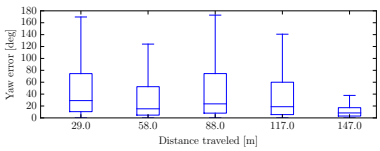
Figure 4.16: Mapping results of Bag.0 and Bag.1 and the map fusion result of server.




(a) Relative translation error.

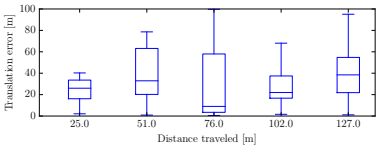



(b) Relative translation error by percent.

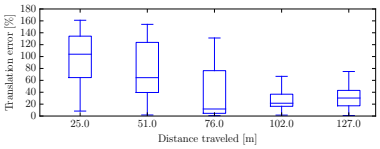



(c) Relative yaw error.

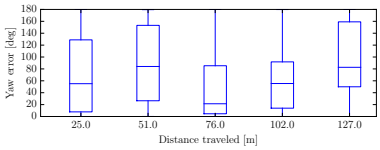
Figure 4.17: Quantitative evaluation results of mapping Bag.0 of NTU Datasets.




(a) Relative translation error.

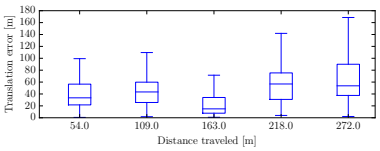



(b) Relative translation error by percent.

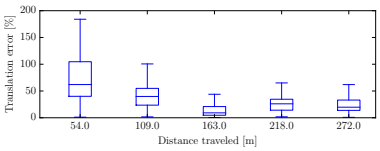



(c) Relative yaw error.

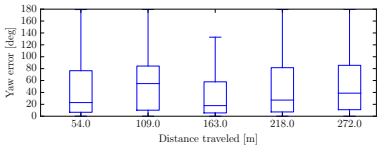
Figure 4.18: Quantitative evaluation results of mapping Bag.1 of NTU Datasets.




(a) Relative translation error.




(b) Relative translation error by percent.




(c) Relative yaw error.

Figure 4.19: Quantitative evaluation results of fused map of NTU Datasets.

Table 4.9: Quantitative results of map fusion evaluation on NTU Datasets.

| Distance(m)[1] | Rel. Trans.(m)[2] | Rel. Trans.(%)[3] | Rel. Yaw(deg)[4] |
|---|---|---|---|
| 54 | 39.38 | 72.93 | 44.39 |
| 109 | 44.88 | 41.18 | 57.01 |
| 163 | 26.09 | 16.00 | 32.12 |
| 218 | 54.99 | 25.23 | 46.11 |
| 272 | 64.66 | 23.77 | 54.65 |

[1] Distance in meter traveled before each time of statistics.
[2] Mean relative translation error in meter.
[3] Mean relative translation error in percent.
[4] Mean relative yaw error in degree.

## 4.3 Evaluation under different illumination

### Oxford RobotCar Datasets

CORB-SLAM system integrated with illumination variance is firstly evaluated on the selected sequences of Oxford RobotCar Datasets, and then the mapping results are compared with the ground truth trajectories, with quantitative evaluation results calculated.

Two partial sequences are selected according the following principles:

1. Exclude the overexposed photo, which will cause tracking lost in ORBSLAM system. Because the dataset was collected in real-world outdoor street environment, there are some frames with overexposure e.g. Figure 4.20, which cannot be process by vSLAM. Therefore, in this work, this dataset is intercepted into two sub sequences excluding overexposed images.


Figure 4.20: Image sequence with overexposed frames in RobotCar dataset.

2. Avoid partial sequences where traffic congestion occurred. Because RobotCar Datasets are recorded in different hours during daytime, there are congestion starting at approximately 15:00.

Table 4.10: Partial datasets selected in RobotCar dataset.

| Seq. No. | Data(M/D/Y) | Time | Weather | Timestamps |
|---|---|---|---|---|
| 0 | 07/14/2014 | 14:49 | summer overcast | 1405349847738682 to 1405350059147905 |
| 1 | 02/24/2014 | 12:32 | winter overcast | 1417794166325288 to 1417794407042717 |

3. Select partial sequence containing images with overlapping under different illumination conditions and in different season, as shown in Figure 4.4.

According to the above selection principles, the two sub sequences selected are listed in Table 4.10. The ground truth GPS/INS trajectories of two sub sequences and the combined overall trajectories are shown in Figure 4.21. The estimate trajectories and the fused map of two partial sequences are shown in Figure 4.22. Results of Oxford RobotCar Datasets are further discussed in Section 5.2.

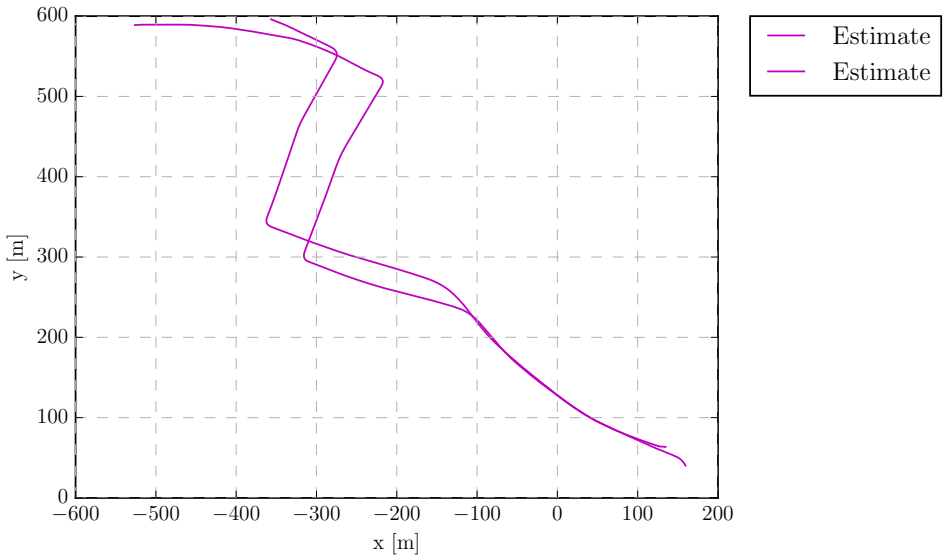
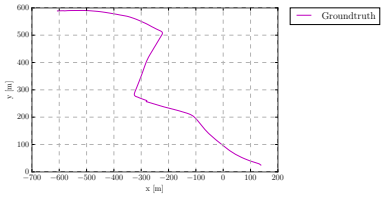
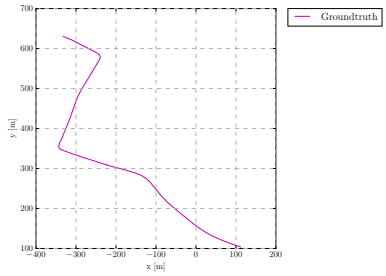


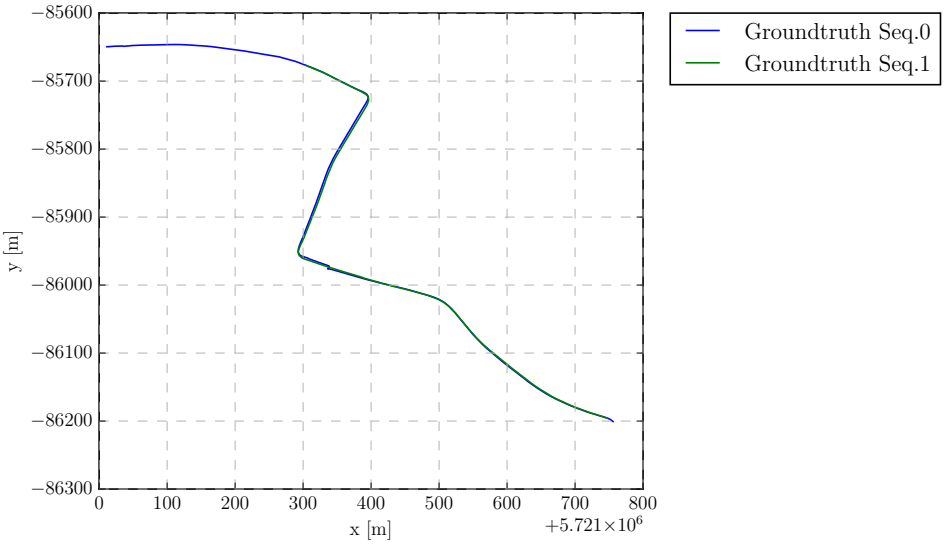

Figure 4.23: Map fusion results of Seq.0 and Seq.1 without ground truth in Oxford RobotCar Datasets.




(a) Ground truth trajectory of Seq.0.

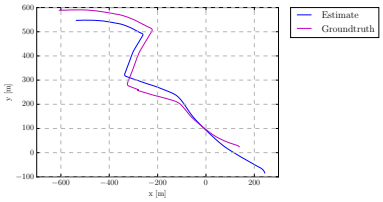



(b) Ground truth trajectory of Seq.0.

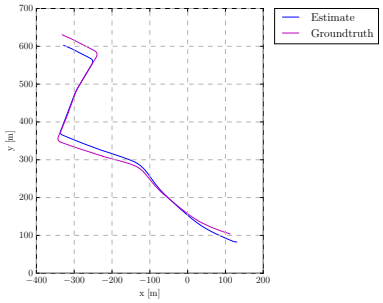



(c) Overall ground truth trajectory of Seq.0 and Seq.1.

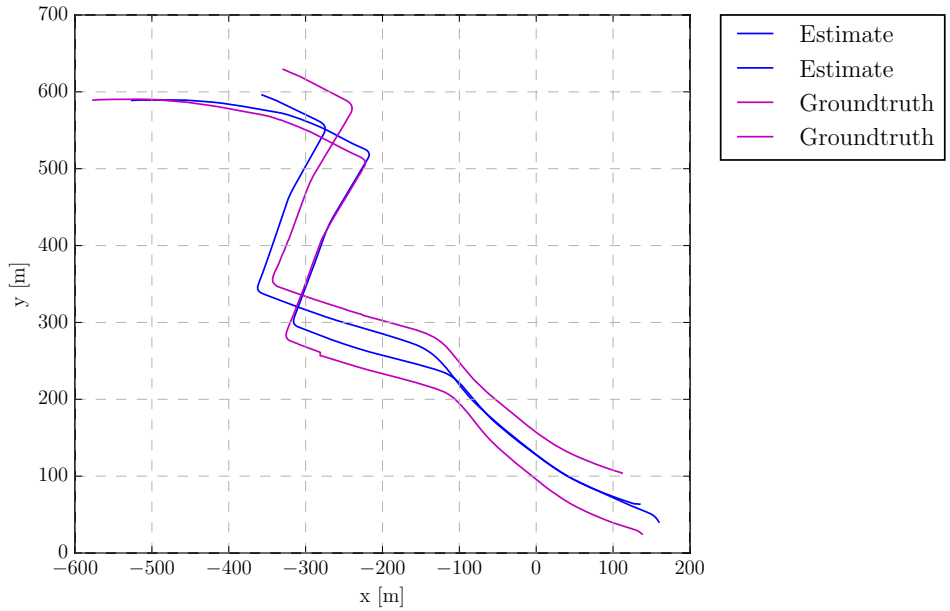
Figure 4.21: Ground truth individual and overall trajectories of Seq.0 and Seq.1 in Oxford RobotCar Datasets.




(a) Mapping result of Seq.0.




(b) Mapping result of Seq.1.




(c) Map Fusion results in server end of Seq.0 and Seq.1.

Figure 4.22: Mapping results of Seq.0 and Seq.1 and the map fusion result of server in Oxford RobotCar Datasets.

# Chapter 5

# Discussion

## 5.1 Results of multi ground robots cluster

Seen graphically according to Figure 4.13, the mapping results of CORB-SLAM clients is able to match the ground truth trajectory fairly well enough. And seeing the numeric analysis in Table 4.6, the completed map fused by the server has 8.34% mean relative translation error and 0.54° mean relative yaw error with all distance traveled.

Comparing the fused map with the mapping results on the original unseparated sequence, according to the differences between Table 4.6 and 4.3, map fusion in CORB-SLAM server slight reduces the accuracy on relative yaw and scaling. However from the table, the relative translation accuracy is numerically increased, of which a critical reason is the distance traveled by each client is inevitable shortened because the sequence is divided into two parts, which means translation errors on each client are accumulated during shorter distances. Therefore, it is not concluded so far that map fusion module in server can increase accuracy of relative translation.

Evaluation on KITTI Dataset is an ideal circumstance, with all overlapped images identical. To evaluate under a more general real-world circumstance to get more universal and convincing results, another evaluation is performed utilizing NTU Dataset.

According to the comparison of quantitative map fusion result ( in Figure 4.19 and Table 4.9 ) and mapping results of single bags ( in Figure 4.17, 4.18 and Table 4.7, 4.8 ), compared to 6.32% relative translation error of client of Bag.0, and 30.70% of Bag.1, the global fused map in server has an error of 23.77%, which can be considered

as an acceptable result, considering Bag.1 has a complicated trajectory consisting both parts of indoor and outdoor environment causing client map has a relatively higher translation error.

## 5.2 Failure Reasons and Drawbacks of Illumination Variance Method

According to the ground truth information of the selected partial sequences of Oxford RobotCar Datasets, the correct fused estimate trajectories of clients should be coincident with very little offset seen as Figure 4.21. However as shown in Figure 4.23, integrating CORB-SLAM system with illumination variance failed to enhance the ability to deal with illumination and season changes in Oxford RobotCar Datasets.

As seen in Figure 4.23, illumination variance method introduce many incorrect matches of keypoints while it is expected to allow ORB matcher to find more correct keypoint pairs that hard to match in raw images due to illumination changes.

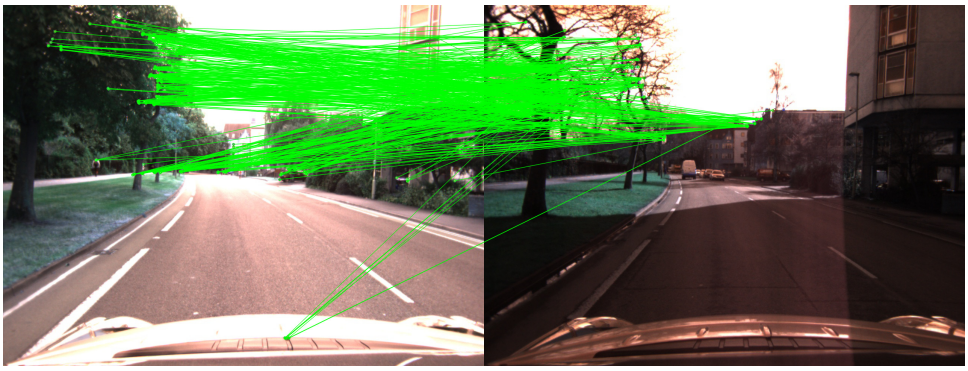
By analyzing the basic formula to compute illumination variance, Equation 2.1 which is repeated below as Equation 5.1 for reference, and the result images in Figure 5.1, the following reasons and drawbacks of illumination variance method can be concluded to explain the failure of it to help CORB-SLAM deal with images under different illumination conditions and seasons:

$$I = \log(G) - \alpha \log(B) - (1 - \alpha) \log(R) \tag{5.1}$$

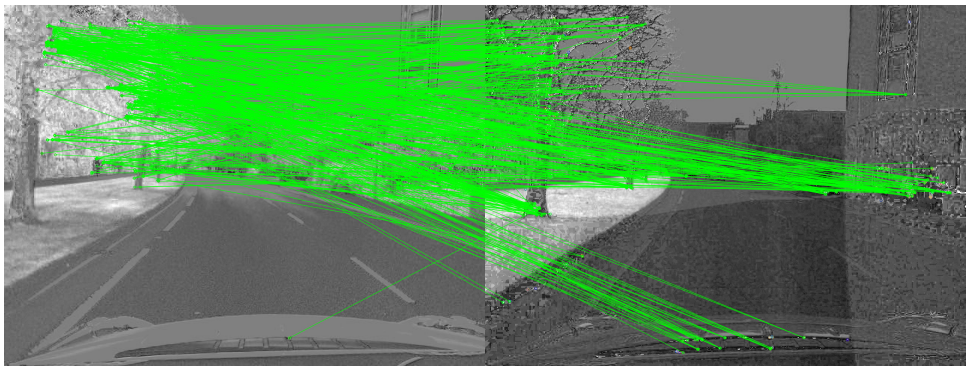
1. Loss of resolution. According to the computation shown in Equation 5.1, there are two steps in this method to calculate the value of illumination variance: for each pixel, 1) logarithm of r,g, b channel values, 2) take their weighted difference as the result illumination variance value. Both steps cause serious loss of resolution, with the result image very blurry as seen in Figure 5.1.

2. Requirement of high brightness and contrast of input color images. In Figure 5.1, compared to Figure 5.1(a), Figure 5.1(a) has a higher resolution. This is because with the loss of resolution during the illumination variance computation, images

with higher brightness and contrast can remain more details in the result images. But the background of application of this method is in life-long localization and mapping system, running under significant lighting and season conditions, which means it is an unreasonable request to demand the input images always bright and sharp.



(a) Corresponding matched keypoint pairs of raw images Figure 4.4(a) and 4.4(b).

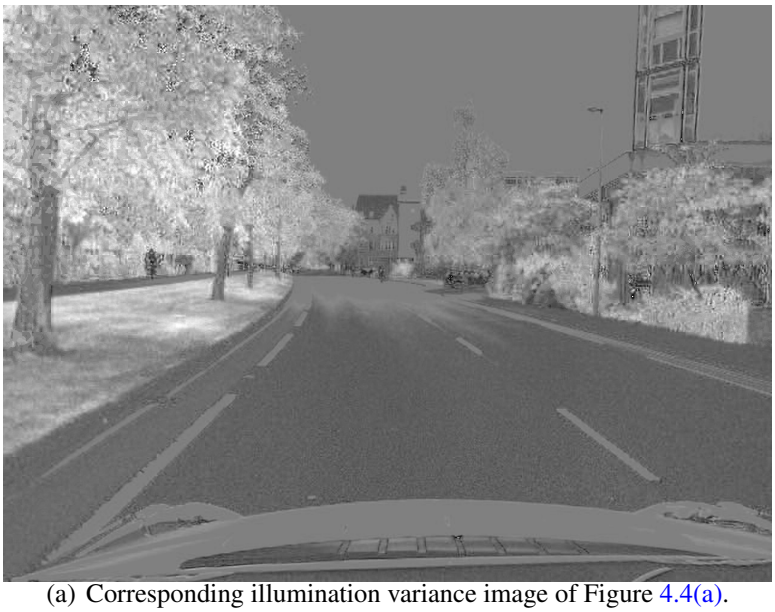


(b) Corresponding matched keypoint pairs of illumination variance images Figure 5.1(a) and 5.1(b).

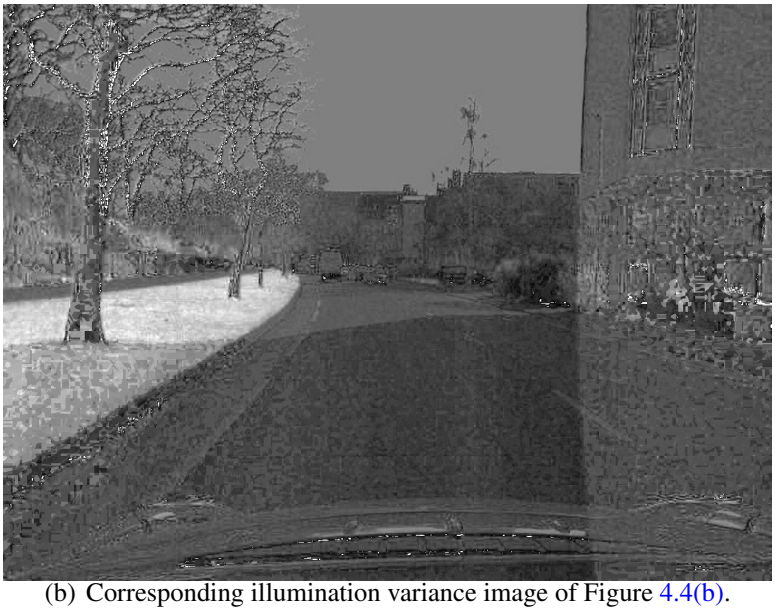
Figure 5.2: ORB Keypoint matching results of raw and illumination variance images in Oxford RobotCar Datasets.

However, in both the original work [37] and the related implement in [39–42], the application of this method only focus on localization, which means the main function is designed to detect loop closures in images taken in different illumination and seasons, but not suitable for mapping. In all above papers, evaluation results are given only by localization results.

A critical difference of localization and mapping tasks is high resolution images are not necessary in localization, while in mapping task they are. Cited by [40], [45] explains that performing localization using a sequence of images rather than single



(a) Corresponding illumination variance image of Figure 4.4(a).



(b) Corresponding illumination variance image of Figure 4.4(b).

Figure 5.1: Generated illumination variance images of example raw images in Oxford RobotCar Datasets.

image removes the requirement that the image matching scheme be able to reliably calculate a single global image match. However, without the functionality of calculating matches between single images, sequence-based image matching algorithms have significant drawbacks that cannot calculate the transformation between images.

And in the case of CORB-SLAM, the image matching algorithm combing ORB keypoint and Bag of Words, is able to calculate matching and transformation between individual images, but fit not well with illumination variance method.

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

By analyzing results of the experiments in this work, we can conclude:

1. On systems consisting of multi ground robots, CORB-SLAM provides promising map fusion results, without bringing in obvious relative translation and yaw error.

2. Illumination variance method proposed for dealing with illumination changes, does not cooperate with CORB-SLAM effectively, because of its characteristic of basic equations to generate low resolution images, while ORB Keypoint extraction, DBoW2 bag of words and keypoint-based mapping algorithms are all feature-based which highly rely on high resolution of input images.

## 6.2 Future Work

Despite of the outstanding map fusion performance of CORB-SLAM system, it still have some drawbacks that remain to improve in future work.

1. In server end, filtering conditions of (i) loop closure detection, (ii) keypoint inliers in iterations of the Perspective-n-Point solver (PnP Solver), are set significantly strictly in order to achieve high accuracy in computation of translation matrix between client maps, but as a result, the application of the system are also limited to

circumstances where images of clients should have high similarity, with only very slight illumination and season change allowed, so as the point of view difference, which means the application on multi hybrid robot systems is also seriously limited.

2. In the current version of CORB-SLAM server, the map fusion modules stops when a translation matrix accurate enough to satisfy the filtering condition. But in fact, the computation of translation matrix can be in an incremental way where current part of client maps are fused to the global map by the current translation matrix, after a new relation is calculated, the next part of client map is fused by the new matrix. By this way, the map fusion accuracy should be improved with simple modifications.

# Bibliography


[1] Fu Li, Shaowu Yang, Xiaodong Yi, and Xuejun Yang. Corb-slam: a collaborative visual slam system for multiple robots. In Eai International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2017.

[2] Sajad Saeedi, Michael Trentini, Mae Seto, and Howard Li. Multiple-robot simultaneous localization and mapping: A review. Journal of Field Robotics, 33(1):3–46, 2016.

[3] Colin McManus, Winston Churchill, Will Maddern, Alexander D Stewart, and Paul Newman. Shady dealings: Robust, long-term visual localisation using illumination invariance. In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 901–906. IEEE, 2014.

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[5] Jun Zhang, Prarinya Siritanawan, Yufeng Yue, Chule Yang, Mingxing Wen, and Danwei Wang. A two-step method for extrinsic calibration between a sparse 3d lidar and a thermal camera. In 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), pages 1039–1044. IEEE, 2018.

[6] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. The International Journal of Robotics Research, 36(1):3–15, 2017.

[7] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: A survey from 2010 to 2016. IPSJ Transactions on Computer Vision and Applications, 9(1):16, 2017.

[8] Reinhard Klette, A Koschan, and K Schluns. Three-dimensional data from images. Springer-Verlag Singapore Pte. Ltd., Singapore, 1998.

[9] David Nistér and Henrik Stewénius. A minimal solution to the generalised 3-point pose problem. Journal of Mathematical Imaging and Vision, 27(1):67–79, 2007.

[10] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. 2011.

[11] 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.

[12] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003.

[13] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In International workshop on vision algorithms, pages 298–372. Springer, 1999.

[14] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In null, page 1403. IEEE, 2003.

[15] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. IEEE Transactions on Pattern Analysis & Machine Intelligence, (6):1052–1067, 2007.

[16] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pages 1–10. IEEE Computer Society, 2007.

[17] Brian Williams, Georg Klein, and Ian Reid. Real-time slam relocalisation. 2007.

[18] David Nistér. An efficient solution to the five-point relative pose problem. IEEE transactions on pattern analysis and machine intelligence, 26(6):0756–777, 2004.

[19] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. IEEE Transactions on Robotics, 31(5):1147–1163, 2015.

[20] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Transactions on Robotics, 33(5):1255–1262, 2017.

[21] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In 2011 international conference on computer vision, pages 2320–2327. IEEE, 2011.

[22] Peter Ondrúška, Pushmeet Kohli, and Shahram Izadi. Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. IEEE transactions on visualization and computer graphics, 21(11):1251–1258, 2015.

[23] 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.

[24] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In Proceedings of the IEEE international conference on computer vision, pages 1449–1456, 2013.

[25] Thomas Schöps, Jakob Engel, and Daniel Cremers. Semi-dense visual odometry for ar on a smartphone. In 2014 IEEE international symposium on mixed and augmented reality (ISMAR), pages 145–150. IEEE, 2014.

[26] David Caruso, Jakob Engel, and Daniel Cremers. Large-scale direct slam for omnidirectional cameras. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 141–148. IEEE, 2015.

[27] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1935–1942. IEEE, 2015.

[28] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In 2011 IEEE International Symposium on Mixed and Augmented Reality, pages 127–136. IEEE, 2011.

[29] Olaf Kähler, Victor Adrian Prisacariu, Carl Yuheng Ren, Xin Sun, Philip Torr, and David Murray. Very high frame rate volumetric integration of depth images on mobile devices. IEEE transactions on visualization and computer graphics, 21(11):1241–1250, 2015.

[30] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1352–1359, 2013.

[31] Keisuke Tateno, Federico Tombari, and Nassir Navab. When 2.5 d is not enough: Simultaneous reconstruction, segmentation and recognition on dense slam. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 2295–2302. IEEE, 2016.

[32] Hauke Strasdat, J Montiel, and Andrew J Davison. Scale drift-aware large scale monocular slam. Robotics: Science and Systems VI, 2(3):7, 2010.

[33] Hauke Strasdat, Andrew J Davison, JM Martìnez Montiel, and Kurt Konolige. Double window optimisation for constant time visual slam. In 2011 International Conference on Computer Vision, pages 2352–2359. IEEE, 2011.

[34] Christopher Mei, Gabe Sibley, and Paul Newman. Closing loops without places. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3738–3744. IEEE, 2010.

[35] Colin McManus, Winston Churchill, Ashley Napier, Ben Davis, and Paul Newman. Distraction suppression for vision-based pose estimation at city scales. In 2013 IEEE International Conference on Robotics and Automation, pages 3762–3769. IEEE, 2013.

[36] Winston Churchill and Paul Newman. Practice makes perfect? managing and leveraging visual experiences for lifelong navigation. In 2012 IEEE International Conference on Robotics and Automation, pages 4525–4532. IEEE, 2012.

[37] Will Maddern, Alex Stewart, Colin McManus, Ben Upcroft, Winston Churchill, and Paul Newman. Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles. In Proceedings of the Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, volume 2, page 3, 2014.

[38] Arren J Glover, William P Maddern, Michael J Milford, and Gordon F Wyeth. Fab-map+ ratslam: Appearance-based slam for multiple times of day. In 2010 IEEE international conference on robotics and automation, pages 3507–3512. IEEE, 2010.

[39] Roberto Arroyo, Pablo F Alcantarilla, Luis M Bergasa, and Eduardo Romera. Openable: An open-source toolbox for application in life-long visual localization of autonomous vehicles. In Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on, pages 965–970. IEEE, 2016.

[40] Roberto Arroyo, Pablo F Alcantarilla, Luis M Bergasa, J Javier Yebes, and Sergio Gámez. Bidirectional loop closure detection on panoramas for visual navigation. In Intelligent Vehicles Symposium Proceedings, 2014 IEEE, pages 1378–1383. IEEE, 2014.

[41] Roberto Arroyo, Pablo F Alcantarilla, Luis M Bergasa, J Javier Yebes, and Sebastián Bronte. Fast and effective visual place recognition using binary codes and disparity information. In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pages 3089–3094. IEEE, 2014.

[42] Roberto Arroyo, Pablo F Alcantarilla, Luis M Bergasa, and Eduardo Romera. Towards life-long visual localization using an efficient matching of binary sequences from images. In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pages 6328–6335. IEEE, 2015.

[43] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7244–7251. IEEE, 2018.

[44] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[45] Michael Milford. Visual route recognition with a handful of bits. Proc. 2012 Robotics: Science and Systems VIII, pages 297–304, 2012.

# Appendix A: Example Code

### Map Fuse To Global Map

```
bool MapFusion::mapFuseToGlobalMap(ServerMap *sMap) {
  // if global map is null, this submap is inserted into the global
   map and does not do transform
  {
    std::unique_lock<mutex> lock( nullGlobalMapMutex );
    if( ifNullGlobalMap ) {

      cv::Mat Tnorm = cv::Mat::eye(4,4, CV_32F);
      std::unique_lock<mutex> lock(mSubMapUpdatedMutex);
      insertServerMapToGlobleMap(sMap, Tnorm);
      ifSubToGlobalMap[(*sMap).pCacher->pClientId] = true;
      subMapTransM[(*sMap).pCacher->pClientId] = Tnorm;
      pubToClient->transMs[ (*sMap).pCacher->pClientId ] = Tnorm;
      sMap->clear();
      ifNullGlobalMap = false;

      cout <<"Global Map is not null!\n";
      return true;
    }
  }

  bool flag = false;
  std::vector<KeyFrame*> allKeyFramesInMapy = sMap->pMap->
   GetAllKeyFrames();

  bool bOK = false;
  std::vector<KeyFrame *> candidateKFs;
```

```
26  KeyFrame * currentKF;
27
28  for( int mit = 0; mit < (int)allKeyFramesInMapy.size(); mit++ ) {
29
30    KeyFrame * tKF = allKeyFramesInMapy[mit];
31
32    cv::Mat oldTwc = tKF->GetPoseInverse();
33    cv::Mat oldTcw = tKF->GetPose();
34    cv::Mat newTcw = cv::Mat::eye(4,4, newTcw.type());
35
36    candidateKFs.clear();
37    currentKF = tKF;
38
39    bOK = detectKeyFrameInServerMap( globalMap, tKF, newTcw,
    candidateKFs);
40
41    if( bOK ) {
42
43      mpGBA->setCurentKeyFrame( currentKF );
44
45      mpGBA->setCandidates( candidateKFs );
46
47      if( mpGBA->ComputeSim3() ) {
48
49        ROS_INFO("Detect In serverMap[%d], from keyframe id[%d]",
50        (int)sMap->pCacher->pClientId, (int)tKF->mnId);
51        // if this keyframe is deteted in the serverMapx
52
53        cv::Mat To2n = oldTwc * newTcw;
54        cv::Mat Tnorm = cv::Mat::eye(4,4, newTcw.type());
55
56        flag = true;
57
58        subMapTransM[ (*sMap).pCacher->pClientId ] = To2n;
59        pubToClient->transMs[ (*sMap).pCacher->pClientId ] = To2n;
60        ifSubToGlobalMap[ (*sMap).pCacher->pClientId ] = true;
61        insertServerMapToGlobleMap( sMap, To2n );
```

```
62          sMap->clear();
63        }
64
65        mpGBA->CorrectLoop();
66
67        time_t end_t = clock();
68
69        cout << "mapfuse " << globalMap->pMap->KeyFramesInMap() << " "
      << allKeyFramesInMapy.size() << " " << (double)( end_t - start_t
      )/(double)CLOCKS_PER_SEC << endl;
70
71        break;
72
73        }
74
75      }
76
77    }
78
79    if( flag ){
80    resentGlobalMapToClient();
81    }
82    return flag;
83  }
```

### Illumination Variance Conversion

```
1  Mat illumination_conversion(Mat image){
2    vector<Mat> channels(3);
3    split(image, channels);
4
5    Mat imageB,imageG,imageR;
6    Mat imageI = Mat(Size(image.cols,image.rows),CV_32FC1);
7    Mat imageI8U = Mat(Size(image.cols,image.rows),CV_8UC1);
8
9    channels[0].convertTo(imageB, CV_32FC1, 1.0/255.0, 0);
10   channels[1].convertTo(imageG, CV_32FC1, 1.0/255.0, 0);
11   channels[2].convertTo(imageR, CV_32FC1, 1.0/255.0, 0);
```

```
12
13  float valueG, valueB, valueR;
14  for (int i = 0; i < imageI.rows; i++){
15  for (int j = 0; j < imageI.cols; j++){
16
17      if (imageG.at<float>(i,j) != 0)
18        valueG=log(imageG.at<float>(i,j));
19      else
20        valueG=0;
21
22      if (imageB.at<float>(i,j) != 0)
23        valueB=alpha*log(imageB.at<float>(i,j));
24      else
25        valueB=0;
26
27      if (imageR.at<float>(i,j) != 0)
28        valueR=(1-alpha)*log(imageR.at<float>(i,j));
29      else
30        valueR=0;
31
32      imageI.at<float>(i,j) = 0.5 + valueG - valueB - valueR;
33
34      if (imageI.at<float>(i,j)<0) imageI.at<float>(i,j) = 1;
35
36    }
37
38  }
39
40  imageI.convertTo(imageI8U, CV_8UC1, 255.0, 0);
41
42  return imageI8U;
43 }
```

# Appendix B: Detailed Table of Quantitative Trajectory Evaluation Results

(Code Here)