University of Technology, Sydney

Doctoral Thesis

Motion Segmentation Based Robust RGB-D SLAM

Author:
Youbing Wang

Supervisor:
A/Prof. Shoudong Huang

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the

Centre for Autonomous Systems
School of Elec, Mech and Mechatronic Systems
Faculty of Engineering and Information Technology

May 2015

Declaration of Authorship

I, Youbing WANG, declare that this thesis titled, 'Motion Segmentation Based Robust RGB-D SLAM' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UNIVERSITY OF TECHNOLOGY, SYDNEY

Abstract

Centre for Autonomous Systems
School of Elec, Mech and Mechatronic Systems
Faculty of Engineering and Information Technology

Doctor of Philosophy

Motion Segmentation Based Robust RGB-D SLAM

by Youbing WANG

While research on simultaneous localisation and mapping (SLAM) in static environments can be regarded as a significant success due to intensive work during the last several decades, conducting SLAM, especially vision-based SLAM, in dynamic scenarios is still at its early stage. Although it seems like just one step further, the dynamic elements have brought in many unanticipated challenges, including motion detection, segmentation, tracking and $3D$ reconstruction of both the static environments and the moving objects, in addition to the handling of motion blur.

Solely based on RGB-D data with no prior knowledge available, this work centres upon proposing new practical solution frameworks for conducting SLAM in dynamic environments with efficient and robust motion segmentation methods serving as the basis. After a detailed review of the related achievements for SLAM in static environments as well as dynamic ones, and an analysis of the unaddressed challenges, four different motion segmentation methods, which include two 2-view sparse feature based motion segmentation algorithms, a 2-view semi-dense motion segmentation algorithm and an extended n-view dense moving object segmentation algorithm, are firstly proposed and their advantages, disadvantages and feasibility for different practical SLAM application scenarios are evaluated.

Based on the proposed motion segmentation methods, two kinds of solution frameworks for performing SLAM in dynamic scenarios are then put forward: the first one is formulated by integrating our sparse feature based motion segmentation techniques with the available pose-graph SLAM framework; and the other one is built upon dense moving object segmentation and tailored for dense SLAM. Related simulation and experimental results have demonstrated the effectiveness of our approaches.

Acknowledgements

First of all, I would like to offer my sincere gratitude to my supervisor, A/Prof. Shoudong Huang, who has supported me during my PhD study continuously with his patience, motivation, enthusiasm and immense knowledge while allowing me the room to work in my own way. Without his encouragement, guidance, and efforts, this work would not have been possible.

Besides my supervisor, I would like to thank Prof. Dikai Liu and Prof. Gamini Dissanayake for their invaluable support for my study at the Centre for Autonomous System.

Sincere thanks also go to the other staff members of CAS, Dr. Jack Jianguo Wang, A/Prof. Sarath Kodagoda, A/Prof. Guang Hong, A/Prof. Jaime Valls Miro, Dr. Gabriel Aguirre-Ollinger, Dr. Liang Zhao and Dr. Lei Shi. They have given me support for my study and life in various ways.

In my daily work, I have been blessed with a friendly and cheerful group of fellow students: Gibson Hu, Kasra Khosoussi, Lakshitha Dantanarayana, Yue Wang, a visiting student from Zhejiang University, and many other people, for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last several years.

Last but not the least, I would like to thank my family and all my friends, for their unconditional support to me spiritually throughout my life.

Contents

Declaration of Authorship i

Abstract iii

Acknowledgements iv

Contents v

List of Figures ix

List of Tables xii

Abbreviations xiii

1 Introduction 1
- 1.1 Simultaneous Localisation and Mapping (SLAM) . . . 1
- 1.2 SLAM in Dynamic Scenarios . . . 3
  - 1.2.1 Various Dynamic Scenarios to Handle . . . 3
  - 1.2.2 Conducting SLAM in Dynamic Environments . . . 3
  - 1.2.3 Another Chicken-and-Egg Problem? . . . 4
  - 1.2.4 Motivation . . . 4
- 1.3 The Scope and Limitations of This Thesis . . . 5
  - 1.3.1 The Targeted Problem . . . 5
  - 1.3.2 Research Scope . . . 5
  - 1.3.3 Limitations . . . 6
- 1.4 Main Contributions . . . 7
- 1.5 Publication List . . . 7
- 1.6 The Structure of This Thesis . . . 7

2 SLAM: From Static to Dynamic 9
- 2.1 SLAM in Static Environments . . . 10
  - 2.1.1 SfM and SLAM Problem . . . 10
  - 2.1.2 Two Kinds of Formulations . . . 10
  - 2.1.3 Two Categories of Typical Representations . . . 12
  - 2.1.4 Visual Odometry . . . 15

2.2 Available Methodologies for SLAM in Targeted Dynamic Scenarios 17
2.2.1 SLAMMOT, SMSaM and Multibody SLAM/MSaM 18
2.2.2 Motion Segmentation based Multibody SLAM/MSaM 22
2.2.3 Summary of Different Methodologies for Conducting SLAM in Dynamic Scenarios 22
2.3 Motion Segmentation 23
2.3.1 Definition 23
2.3.2 Different Implementations 23
2.3.3 Related Work 24
2.3.4 Evaluation Methods 25
2.3.5 Difficulties 25
2.3.6 Usability Analysis of Different Methods 26
2.4 Usability of Available SLAM Solution Frameworks for Conducting SLAM in Dynamic Scenarios 27
2.4.1 Pose-graph SLAM Versus Pose-feature-graph SLAM 27
2.4.2 Dense SLAM Versus Sparse Feature-based SLAM 28
2.5 Benchmark Datasets for RGB-D SLAM 28
2.6 Towards Motion Segmentation based Robust SLAM 29
2.6.1 Missing Elements 29
2.6.2 Our Basic Objectives 29

3 Sparse Feature Distance-based Motion Segmentation for Multibody RGB-D SLAM 31
3.1 Motivation 31
3.2 An Efficient RGB-D Motion Segmentation Method 32
3.2.1 Problem Restated 32
3.2.2 Theory 33
3.2.3 Algorithm 35
3.2.4 Discussion 36
3.2.5 Segmentation Results Using Simulated Data 37
3.2.6 Efficiency 37
3.3 Evaluation with A Simulated Multibody RGB-D SLAM Problem 39
3.3.1 The Simulated Multibody RGB-D SLAM Scenario 39
3.3.2 Motion Segmentation Results Using Simulated Noisy Data 39
3.3.3 Multibody SLAM Results Using Simulated Noisy Data 41
3.4 Evaluation Based on Our Own Dataset 42
3.4.1 Collecting Our Own Dataset 42
3.4.2 Motion Segmentation Results From Experimental Data With Outliers 43
3.4.3 Initial Values Based on Motion Segmentation and Visual Odometry 44
3.5 Summary 44

4 Motion Model based Segmentation for Robust RGB-D SLAM: From Sparse to Semi-dense 46
4.1 Related Work 47
4.1.1 RANSAC and Its Application in SLAM 47
4.1.2 Motion Segmentation Methods for SLAM in Dynamic Scenarios . 48
4.1.3 Different Solutions for Different Scenarios 49

4.2 Our Sparse Feature Motion Model based Motion Segmentation Algorithm 50
4.2.1 Sparse Feature Motion Model Based Motion Segmentation Algorithm . . . 51
4.2.2 Solution for Robust SLAM and Multibody SLAM . . . 51
4.3 Simulation and Experimental Results . . . 52
4.3.1 Results Using Simulated Dynamic Scenarios . . . 53
4.3.2 Experimental Results Using Static and Dynamic Scenarios . . . 54
4.4 Motivations to Semi-dense Feature Motion Model Based Motion Segmentation . . . 60
4.4.1 The Challenging Sequences and Its Impacts on Available Methods 61
4.4.2 Sparse Feature-based SLAM Cannot Deal with Image Blur and Textureless Areas . . . 63
4.4.3 Dense SLAM is Vulnerable to Moving Objects . . . 63
4.4.4 RANSAC and Robust Kernels Cannot Endure Too Many Outliers 64
4.5 Our Semi-dense Feature Motion Model based Motion Segmentation Method 64
4.5.1 SIFT Flow For Dense-Feature Detection And Matching . . . 64
4.5.2 Semi-dense Feature-based Motion Segmentation . . . 65
4.5.3 Pose-graph SLAM . . . 65
4.6 Experimental Evaluation Using Challenging Sequences . . . 66
4.6.1 The fr3_walking_static Sequence . . . 67
4.6.2 The fr3_walking_xyz Sequence . . . 69
4.6.3 The fr3_walking_rpy Sequence . . . 69
4.6.4 The fr3_walking_halfsphere Sequence . . . 72
4.6.5 Discussion . . . 73
4.7 Summary . . . 74

5 Towards Dense Moving Object Segmentation based Robust Dense RGB-D SLAM in Dynamic Scenarios 75
5.1 Introduction . . . 76
5.2 Related Work . . . 77
5.2.1 Two-view/N-view Based Motion Segmentation Versus Extended N-view Based Moving Object Segmentation [50] . . . 77
5.2.2 RGB-D SLAM: Sparse Versus Dense . . . 78
5.3 Moving Object Segmentation Based Robust Dense SLAM . . . 79
5.3.1 Proposed Robust Moving Object Segmentation Method . . . 79
5.3.2 Dense Visual SLAM . . . 82
5.4 SLAM Results for The Benchmark Dataset . . . 84
5.4.1 RGB-D SLAM Benchmark and Methods Involved . . . 84
5.4.2 Challenging Sequences . . . 86
5.4.3 Comparison of the Experimental Results . . . 86
5.4.4 Discussion . . . 87
5.5 Summary . . . 87

6 Conclusions and Future Work 88
6.1 What We Have Done: Motion Segmentation based Robust RGB-D SLAM 88
6.2 Future Work: Beyond Motion Segmentation . . . 89

A Analysis of The Computational Cost of Algorithm 3 **91**

Bibliography **93**

List of Figures

1.1 SLAM in relation with the other research topics in mobile robotics. 1
1.2 The flowchart of modern SLAM. 2

2.1 A sketch of the relative positions of the major topics to be covered in this chapter, where items in red represent the work has been proposed for conducting SLAM in static environments, those in green belong to visual SLAM, and those in blue are closely related to performing SLAM in dynamic scenarios. 9
2.2 Representing SLAM problems with DBN, in which $\boldsymbol{x}_{1..T}$ are invisible system states, $\boldsymbol{m}$ the unknown static environment, $\boldsymbol{u}_{1..T}$ input control and $\boldsymbol{z}_{1..T}$ the observations. 13
2.3 A graph representing SLAM problems in which $\boldsymbol{z}_{ij}$ represents the observation/odometry of $\boldsymbol{x}_j$ at the position of $\boldsymbol{x}_i$ and $\boldsymbol{Q}_{ij}$ is the corresponding covariance matrix. 14

3.1 Constrain a new point P using three non-collinear points P1, P2 and P3 . 35
3.2 Motion segmentation result for the simulated 3 planes scenario. Here different colours are only used to differentiate different motion groups, so not necessarily the same as the ground-truth marked. Best viewed in colour. 38
3.3 Motion segmentation result for the simulated 1 plane and 2 balls scenario. Here different colours are only used to differentiate different motion groups, so not necessarily the same as the ground-truth marked. Best viewed in colour. 38
3.4 Motion segmentation result for the simulated 3 balls scenario. Here different colours are only used to differentiate different motion groups, so not necessarily the same as the ground-truth marked. Best viewed in colour. . 39
3.5 Simulated multibody RGB-D SLAM scenario: blue and red points represent static features, and red ones are currently seen by the camera; green ones stand for camera's positions; yellow ones correspond to the paths of the two moving objects 40
3.6 One segmentation result from the simulated data: blue, red and green points represent the three groups the algorithm has found, corresponding exactly to the ground-truth groups 40
3.7 Angular, translational and position errors of the multibody RGB-D SLAM results using the simulated data. 41
3.8 Experimental setup: one Kinect, two boxes (one in blue, one in black) are located at the further side of the image, and have been marked out . . 42
3.9 One sample feature matching result from the experimental data 43
3.10 One sample segmentation result from the experimental data 43

3.11 The angular and translational errors of camera poses' initial values as compared with the ground truth. Best viewed in color. 44

4.1 A sample motion segmentation result of two frames from the fr3_long_office sequence (best viewed in colour): blue points respresent detected inliers, while red points represent outliers caused by inaccurate depth data or feature matching. 56
4.2 Our SLAM results for the fr3_long_office sequence compared with the ground truth (best viewed in colour): gray lines represent ground truth, blue lines represent the SLAM results, and red lines represent the differences between them at various corresponding timestamps. 56
4.3 A sample motion segmentation result of two frames from the fr3_sitting_xyz sequence (best viewed in colour): blue points respresent detected inliers, while red points represent outliers caused by inaccurate depth data or feature matching. 57
4.4 Our SLAM results for the fr3_sitting_xyz sequence compared with the ground truth (best viewed in colour): gray lines represent ground truth, blue lines represent the SLAM results, and red lines represent the differences between them at various corresponding timestamps. 57
4.5 A sample motion segmentation result of two frames from the desk_with_person sequence (best viewed in colour): blue points respresent detected inliers, while points in other colours represent outliers caused by inaccurate depth data, feature matching, or moving objects. 58
4.6 Our SLAM results for the fr2_desk_with_person sequence compared with the ground truth (best viewed in colour): grey lines represent ground truth, blue lines represent the SLAM results, and red lines represent the differences between them at various corresponding timestamps. 59
4.7 Two snapshots of the fourth sequence: fr3_walking_static. 60
4.8 Samples of *fr3_walking_static* 61
4.9 Samples of *fr3_walking_xyz* 62
4.10 Samples of *fr3_walking_rpy* 62
4.11 Samples of *fr3_walking_halfsphere* 63
4.12 An sample SIFT flow and motion segmentation result (Best viewed in colour). The first column are the original images, and the second column is the SIFT flow output. In the third column, different colours in the upper image represent different motion groups we've found, and the blue points in the lower image represent the static group. 67
4.13 The RMSE of RPE for the Results of dense SLAM with the TUM *walking_static* sequence 68
4.14 The RMSE of RPE for the Results of Visual odometry of our method with the TUM *walking_static* sequence 68
4.15 The RMSE of RPE for the Pose-graph SLAM Results of our method with the TUM *walking_static* sequence 69
4.16 The RMSE of RPE for the Results of dense SLAM with the TUM *walking_xyz* sequence 70
4.17 The RMSE of RPE for the Visual odometry of our method with the TUM *walking_xyz* sequence 70
4.18 The RMSE of RPE for the Pose-graph SLAM results of our method with the TUM *walking_xyz* sequence 71

4.19 The RMSE of RPE for the Results of dense SLAM with the TUM *walking_rpy* sequence . . . 71
4.20 The RMSE of RPE for the Visual odometry of our method with the TUM *walking_rpy* sequence . . . 72
4.21 The RMSE of RPE for the Pose-graph SLAM results of our method with the TUM *walking_rpy* sequence . . . 72

5.1 In this example, the biggest common part having depth information of the two non-consecutive images does not belong to the static group, making loop-closure detection harder to handle. . . . 78
5.2 Flowchart of the whole process for robust SLAM . . . 80
5.3 Flowchart of the original moving object segmentation method [50] with images showing the results at different stages. . . . 80
5.4 The left column represents the result before the merging using fundamental matrix, while the right one represents that after merging. The lower images represent the segmented group masks in different color intensities, and the upper images are the combination of the original image and segmentation results. Best viewed in colour. . . . 81
5.5 The left column represents the result before merging the connected neighbouring regions, while the right one represents that after merging. The lower images represent the segmented group masks in different color intensities, and the upper images are the combination of the original image and segmentation results. Best viewed in colour. . . . 81
5.6 The original image and segmentation results (from the *walking_static* sequence). . . . 83
5.7 Comparison of dense SLAM results before and after moving object segmentation using the four *walking* sequences. . . . 85

List of Tables

3.1 Computation time in relation with the number of feature points and groups 39

4.1 Comparison of Average Computation Time: Scenarios Composed of Two Motion Groups 54
4.2 Comparison of Average Computation Time: Scenarios Composed of Five Motion Groups 54
4.3 Comparison of RMSE of RPE (m/s) and ATE (m) for SMS and DVSLAM 60
4.4 Comparison of Visual Odometry and SLAM Results in Terms of RMSE of RPE (m/s) 73
4.5 Comparison of Visual Odometry and SLAM Results in Terms of RMSE of ATE (m) 73

5.1 Comparison of the RPE (m/s) & ATE (m) Results of Dense SLAM Versus Those of Algorithm 6 (MS_DSLAM) Using the Four Sequences 87

Abbreviations

SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
KF	**K**alman **F**ilter
EKF	**E**xtended **K**alman **F**ilter
EIF	**E**xtended **I**nformation **F**ilter
SfM	**S**tructure **f**rom **M**otion
MSaM	**M**ultiple **S**tructure **a**nd **M**otion
RANSAC	**RAN**dom **SA**mple **C**onsensus
GPS	**G**lobal **P**ositioning **S**ystem
IMU	**I**nertial **M**easurement **U**nit
RMSE	**R**oot-**M**ean-**S**quare **E**rror
RPE	**R**elative **P**ose **E**rror
ATE	**A**bsolute **T**ranslational **E**rror
SMSaM	**S**imultaneous **M**ultibody **S**tructure **a**nd **M**otion
MDL	**M**inimum **D**escription **L**anguage
KLT	**K**anade-**L**ucas-**T**omasi

Dedicated to my family...

Chapter 1

Introduction

1.1 Simultaneous Localisation and Mapping (SLAM)

As shown in Fig. 1.1, mobile robotics is composed of three closely related branches of research topics, i.e., mapping which concerns about building a map of the environment, localisation that deals with keeping track of the robot's location in the environment, and motion control and path planning. Accordingly, in practice, they are regarded as three basic capabilities of a robot and are usually treated in separate disciplines [1–3]. However, if a robot is deployed in a formerly unknown environment, mapping and localisation would become two highly interdependent problems: each of them relies on the results of the other's. It was under such an understanding that some researchers propose to deal with them together, thus SLAM as an independent research area came into beings [4]. Since SLAM can endow mobile robots with the ability of exploring in unknown environments autonomously, much attention has been attracted towards this topic. And within the last several decades, we have witnessed significant progress in this area, focusing on conducting SLAM in static environments. As the result, many researchers share the feeling that this problem has been solved [5].

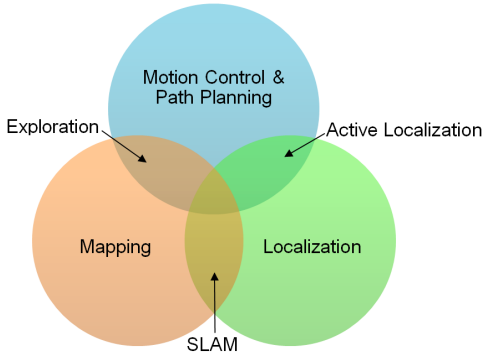


FIGURE 1.1: SLAM in relation with the other research topics in mobile robotics.

With the development of SLAM research, various kinds of sensors have been employed to collect data, among which are laser scanners, cameras (including stereo rig and single camera), RGB-D sensors which can provide both RGB and depth information, sonar sensors, inertial measurement units (IMUs), quite recently tactile sensors, radars, wifi signals, or combinations of them. Each of them calls for a specific method to extract odometry and loop-closures from the raw sensor data. Accordingly, SLAM can be further divided into many sub-areas, such as laser SLAM, stereoSLAM, monoSLAM (the corresponding research area in computer vision is known as structure from motion (SfM)), RGB-D SLAM and sensor-fusion based SLAM which includes more than one sensor, either the same kind or different kinds.

On the other hand, in terms of methodology, at its early stage, a variety of filter-based methods, such as Kalman filter (KF) [6], extended Kalman filter (EKF), extended information filter (EIF) and particle filters, constituted the predominant ways to tackle the problem. However, gradually people come to realise that those filter-based methods suffer from inconsistency problem [7], so optimisation-based methods boom quickly. Nowadays, the common practice in modern SLAM is the separation of the sensor-independent part of work, i.e., graph optimisation, from the sensor-dependent part which mainly concerns about extracting odometry, observations and loop closures from the raw sensor data. As shown in Fig. 1.2, the sensor-dependent part is usually called the *front-end* [8], and the sensor-independent part is widely known as the *back-end* [9, 10]. This division frees researchers from considering the special characteristics of different sensors, thus facilitating the development of general back-ends. As the result, many mature theories and algorithms have been established in this regard. However, as the front-end part is highly sensor-specific, there is no general or automatic way readily available.

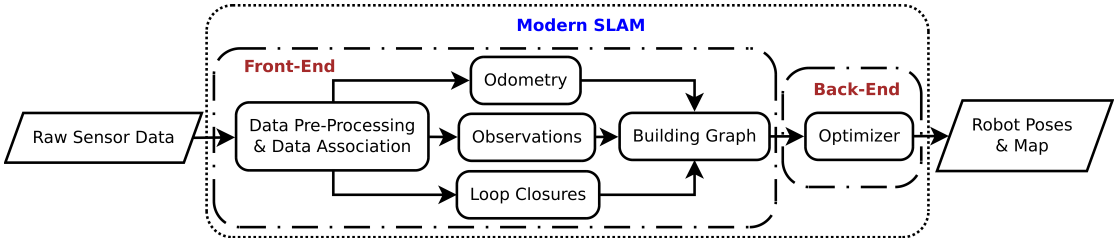


FIGURE 1.2: The flowchart of modern SLAM.

Under this situation, for any specific SLAM application, one needs to firstly choose an appropriate front-end method according to the sensor type, and then employ one of the general back-ends for SLAM.

1.2 SLAM in Dynamic Scenarios

1.2.1 Various Dynamic Scenarios to Handle

Undoubtedly, compared with dynamic scenarios, static environments are much simpler and we can easily reach consensus on the characteristics of them. However, as for dynamic scenarios and their corresponding features, it is very difficult to describe them using just one unified formulation. Furthermore, a multitude of researchers are dealing with them for divergent purposes. For example, as we know, we are living in a changing world whose appearance can be quite inconsistent due to variable light/weather conditions or in different seasons. To deal with this kind of problems, the research group led by Paul Newman proposes the concept of life long exploration [11] in outdoor environments, emphasising accumulating the changing maps: performing visual odometry (VO) and localisation at the same time, and if localisation fails, VO will be utilised to accumulate a new experience of the appearance of the same environment. Moreover, some other researchers are interested in another type of indoor dynamic scenarios where the environment will remain static during each round of SLAM, but parts of it may be changed due to man-made modifications [12]. In this case, they propose to integrate and update the corresponding pose graphs between different rounds. Nevertheless, this two kinds of scenarios share one key property: within each round, the environment remain static or the changes within it are extremely small, therefore the formulations for static environments can be directly utilised.

However, there still exists another kind of dynamic scenarios that are ubiquitous in our daily life: a considerable part of our home, offices, or streets, are dominated by moving people, activated vehicles or other dynamic objects. While the background remains the same, the dynamic objects constitute a large portion of the images we are taking, in which case RANdom SAmple Consensus (RANSAC) [13] -based VO will become highly inefficient and may fail in some situations. In this thesis, we are interested in this kind of dynamic scenarios. And without causing confusion, we will simply call our problem as *conducting SLAM in dynamic environments* thereafter.

1.2.2 Conducting SLAM in Dynamic Environments

Although in SLAM research, some effective and mature solution frameworks have already been established, most of them assume that the whole environment remains static while the robot as the only dynamic part moves through it performing SLAM tasks. As a matter of fact, when it comes to conducting SLAM in dynamic scenarios, most of them will fail. To tackle the new problem, different methodologies have been developed

for different purposes. Some focus on employing robust estimation methods, such as RANSAC or robust kernels, to filter out dynamic objects to make the odometry estimation and thus the SLAM results with respect to the static environment more robust [14]. And there are some others proposing to track the moving objects and perform SLAM at the same time [15–20]. The latter is closely related to what is now widely known as multibody SLAM, or multibody structure-and-motion (MSaM) in computer vision. Multibody SLAM/MSaM, as an extension to SLAM/SfM in static environments, aims to map both the dynamic objects and the static environment in addition to performing localisation at the same time, although due to many difficulties most of the available work is still restricted to tracking moving objects plus conducting static SLAM only.

1.2.3 Another Chicken-and-Egg Problem?

Historically, mapping and localisation form a typical chicken-and-egg problem: without a prior map, localisation can be elusive; without localisation, mapping is impossible, although with the introduction of SLAM, after decades of intensive research, we come to realise that tackling these two problems together is possible.

To some degree, conducting SLAM in dynamic scenarios will burden us with another chicken-and-egg problem, especially for those systems only equipped with allothetic sensors such as laser scanners and cameras: without getting rid of the moving parts, odometry and SLAM results are subject to errors; without prior knowledge of the static environment and camera poses, motion segmentation would be complicated and limited by sensor noises and errors.

1.2.4 Motivation

Now many researchers share the opinion that conducting SLAM in static environments has become a relatively solved problem due to intensive research work in the last several decades. However, when it comes to performing SLAM in dynamic scenarios, it is a different story. Although the latter has also gained much attention for a long time and some achievements have been made, it is still far from being solved. Many reasons can be attributed to this situation, among which lacking robust tools to deal with the serious challenges imposed by general dynamic scenarios which include motion detection, segmentation, tracking and $3D$ reconstruction of the both static environments and moving objects, constitutes the basic one.

Nevertheless, the importance of conducting SLAM in dynamic environments could not be overstated. Firstly, we are living in a highly dynamic world, robots need to be able to

localise themselves, map and navigate through this changing world before human-robot coexistence becomes reality. Secondly, on the other hand, if we can model the dynamic process of the changing world including human activities, it will lay a solid foundation for many other applications such as semantic SLAM and human robot interaction.

It is under such a general situation that this work is devoted to looking for robust methods that can enable us to perform SLAM in dynamic scenarios only using a RGB-D camera. Although dynamic scenarios have brought in many new challenges, at least parts of them are static or rigid, and for them we can still employ the formulations for performing SLAM in static environments. Therefore, in our methodology, what we have learned from conducting SLAM in static environments constitutes the starting point for executing SLAM in dynamic scenarios.

However, when it comes to performing SLAM in general dynamic scenarios, if we want to make full use of what we have learned in performing SLAM in static environments, effective measures capable of dividing the originally complex scenarios into correct groups, i.e., motion segmentation methods, amount to be the key bridging step.

1.3 The Scope and Limitations of This Thesis

1.3.1 The Targeted Problem

While conducting visual SLAM in a static environment, only the camera is moving and all of the other objects are static (some moving objects may appear from time to time, but they are usually limited to just few shots and occupy very small parts of the images); if some moving objects constitute considerable parts of the scenario, and remain in front of the camera performing some activities for a continuous length of time, then it becomes a typical dynamic scenario that we aim to address in this thesis. Although, compared with SLAM in static environments, conducting SLAM in dynamic scenarios seems only need to handle the moving objects in addition to the static environment and camera poses, the persistent moving objects could impose serious impacts on the traditional SLAM process and results. Therefore, we need to find new robust methods to cope with the problem.

1.3.2 Research Scope

In this thesis, we are interested in proposing practical measures that enable us to perform visual SLAM in real dynamic scenarios that include both slow and fast continuous non-rigid movements. The RGB-D data provided by Microsoft Kinect is the only source of

information we are using, and we have no prior restriction and knowledge of the motion model of the camera nor any object model. In other words, our analysis is solely based on the RGB-D video itself.

And the two major emphases targeted in this thesis are:

Effective Motion Segmentation Methods for Challenging Real Scenarios. With both sparse feature-based and dense segmentation based methods in consideration, we pay special attention to their capability of discerning slow and continuous non-rigid movements close to the camera, which means that moving objects constitute a large part of the images.

Robust SLAM Solution Frameworks for RGB-D SLAM in Dynamic Scenarios. By incorporating the proposed motion segmentation methods to form robust front-ends and employing pose-graph based SLAM back-end, we are especially concerned about practical solution frameworks capable of handling motion blur and featureless regions.

1.3.3 Limitations

Since essentially we are conducting visual SLAM without any other sensor, one corresponding limitation is that we need enough visual information in each image that can enable us to obtain acceptable visual odometry for SLAM. Otherwise, visual SLAM is impossible at the current stage.

In addition, in this thesis, we concentrate on obtaining camera poses instead of mapping after motion segmentation. Motion segmentation constitutes a very important topic in our work. After motion segmentation, through pose-graph SLAM, we can obtain an estimation of the camera poses. Based on the results, for rigid moving objects, we can achieve satisfactory mapping results by utilising the readily available surface mapping applications such as Kinectfusion [21]. However, when it comes to mapping non-rigid moving objects, there are still many challenges involved. Mapping of dynamic objects will be a part of our future work.

Bearing these two major limitations in mind, this thesis is oriented to make full use of what we have achieved in SLAM, computer vision and many other closely-related areas to handle dynamic scenarios in RGB-D SLAM area.

1.4 Main Contributions

Centring around providing practical solutions for challenging real scenarios, different methods based on different assumptions and information have been put forward. More specifically, the main contributions of this thesis are four folded: by combining the ideas of available $2D$ and $3D$ motion segmentation methods, a more efficient and robust method than traditional robust methods such as RANSAC and robust kernels is proposed; from sparse to dense, robust two-view based motion segmentation methods capable of handling challenging dynamic scenarios are proposed; from two-view to n-view, a practical pipeline that can segment densely moving object out is proposed; based on these proposed motion segmentation methods, practical solution frameworks for robust SLAM in dynamic scenarios are proposed.

1.5 Publication List

- **Y. Wang** and S. Huang, "Towards Dense Moving Object Segmentation based Robust Dense RGB-D SLAM in Dynamic Scenarios", in *International Conference on Control, Automation, Robotics and Vision (ICARCV), 10th-12th, December, 2014*
- **Y. Wang** and S. Huang, "Motion Segmentation based Robust RGB-D SLAM", *in World Congress on Intelligent Control and Automation (WCICA), 29th,June-4th, July, 2014*
- **Y. Wang** and S. Huang, "An Efficient Motion Segmentation Algorithm for Multi-body RGB-D SLAM", in *the Australasian Conference on Robotics and Automation (ACRA), Sydney, 2nd-4th, December, 2013*

1.6 The Structure of This Thesis

In Chapter 2, we give a detailed illustration and review of the related problems involved in conducting SLAM in static and dynamic scenarios and the corresponding methodologies having been proposed so far in robotics and computer vision communities. More importantly, we focus on what is still missing while dealing with the challenges. Based on in-depth analysis, we present our ideas to improve the existing techniques in various aspects.

In Chapter 3, we propose a sparse feature distance-based motion segmentation method for RGB-D SLAM and together with the pose-graph back-end we could obtain satisfactory simulation and experimental results.

In Chapter 4, another sparse feature motion-model-based motion segmentation method is presented. To cope with highly dynamic scenarios, a semi-dense and adaptive version of it is also put forward. Furthermore, for each version of the method, we demonstrate how to integrate it with the pose-graph SLAM framework to obtain satisfactory results.

In Chapter 5, we focus on dense moving object segmentation methods so that dense SLAM can be employed to deal with highly dynamic scenarios.

Chapter 6 summarises our work and discusses future directions.

Chapter 2

SLAM: From Static to Dynamic

In this chapter, as shown in Fig. 2.1, based on a summary of the major achievements of conducting SLAM in static scenarios in terms of problem formulations and solutions, we give a detailed review of the available methods proposed so far for performing SLAM in dynamic scenarios, most of which are closely related to the accumulated contributions of SLAM in static environments, focusing on what is still missing, then make it clear what we need to do, and how we can do it to push the research forward.

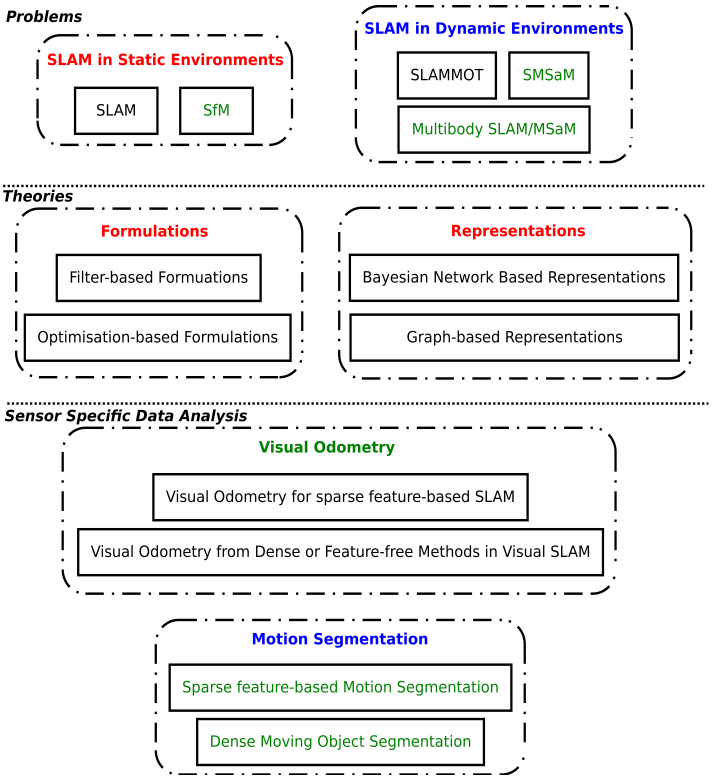


FIGURE 2.1: A sketch of the relative positions of the major topics to be covered in this chapter, where items in red represent the work has been proposed for conducting SLAM in static environments, those in green belong to visual SLAM, and those in blue are closely related to performing SLAM in dynamic scenarios.

More specifically, in Section 2.1, we present some of the most influential theoretical formulations and representations for SLAM in static scenarios. In Section 2.2, we then review those methodologies having been proposed to address the SLAM problem in dynamic scenarios. Following that, in Section 2.3, we pay special attention to the concept, implementations of a variety of motion segmentation methods. Then we evaluate the related solution frameworks (in Section 2.4) and benchmark datasets (in Section 2.5) in terms of their appropriateness for conducting SLAM in dynamic environments. And finally we present our basic research objectives in Section 2.6.

2.1 SLAM in Static Environments

Thanks to the intensive work of the last several decades, most of the theoretical problems of SLAM in static environments have become clear and numerous enlightening formulations have been proposed and verified in practice. In this part, we will review some of the most classical ones.

2.1.1 SfM and SLAM Problem

Historically, there were two groups of researchers working on a theoretically similar problem, i.e., estimating sensor poses while mapping an unknown static environment, but with different sensors, emphases and terminologies. Originated from photogrammetry, the former group is widely known as SfM in computer vision, emphasising 3D reconstruction from a small bunch of camera images based on projective geometry and optimisation, while the latter is called SLAM in mobile robotics, highlighting estimating a robot's poses in real time while it moves through an unknown environment with various kinds of sensors onboard, therefore filter-based methods have played a crucial role in this field, although optimisation-based methods have also gained popularity. In the following section, we discuss the related theoretical explorations and achievements in detail.

2.1.2 Two Kinds of Formulations

Although theoretically we still have not fully understood the structure of SLAM problems, and are still not sure of the number of local minima and how to attain the global minimum in general [22], numerous practical solution frameworks capable of producing satisfactory results are readily available for us to use, most of which fall into one of the two categories: filter-based formulations and optimisation-based ones.

Filter-based Formulations

Ever since the early days of SLAM as an independent research topic, various filters have been proposed to handle the problem. KF/EKF and particle filters are the most famous ones.

Kalman Filter, which is named after one of the primary contributors of the theory, Rudolf (Rudy) E. Kálmán [23], can provide statistically optimal estimation of the underlying system state based on a series of noisy observation data. However, it could only handle linear system models while most practical systems are nonlinear. Therefore, EKF was proposed to handle the more prevalent nonlinear situations, the core of which is the calculation of matrix partial derivatives, i.e., Jacobian, at predicted states based on multivariate Taylor series expansions. As a result, while the state transition and observation models do not need to be linear, they need to be differentiable. And their widely adopted discrete-time forms could be illustrated as follows:

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \boldsymbol{w}_{k-1} \tag{2.1}$$

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{2.2}$$

Here, $\boldsymbol{x}_k$ represents the combination of the inner states and feature positions, $\boldsymbol{u}_k$ is the control vector, while $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ correspond to the process and observation noises that are usually assumed to be normally distributed with zero mean and $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$ as their respective covariance matrices, with $\boldsymbol{z}_k$ being the observation vector.

And the corresponding prediction equations are:

$$\hat{\boldsymbol{x}}_{k|k-1} = f(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_{k-1}) \tag{2.3a}$$

$$\boldsymbol{F}_{k-1} = \frac{\partial f}{\partial \boldsymbol{x}}|_{\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_{k-1}} \tag{2.3b}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_{k-1}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}_{k-1}^T + \boldsymbol{Q}_{k-1} \tag{2.3c}$$

And the update step is composed of:

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1}) \tag{2.4a}$$

$$\boldsymbol{H}_k = \frac{\partial h}{\partial \boldsymbol{x}}|_{\hat{\boldsymbol{x}}_{k|k-1}} \tag{2.4b}$$

$$\boldsymbol{S}_k = \boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k \tag{2.4c}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T\boldsymbol{S}_k^{-1} \tag{2.4d}$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\tilde{\boldsymbol{y}}_k \tag{2.4e}$$

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k|k-1} \tag{2.4f}$$

As we can see, although f and h could be directly utilised to predict the new state and observation, their Jacobians are needed to update the covariance.

Nonetheless, EKF suffers from several disadvantages as well. Firstly, for a nonlinear system, the estimation result is no longer optimal. Secondly, if the estimated initial value of the system state is too far away from the true value, or the system model is incorrect, EKF could diverge quickly. Last but not least, the estimated covariance matrix is usually optimistic when compared with the ground truth and therefore inconsistency is highly possible [7].

When the system model is partially unknown or not accurate, Monte Carlo methods such as particle filters are usually employed [24, 25]. Nonetheless, they are computationally expensive, especially when it comes to deal with high dimensional problems.

Since filter-based methods are easy to implement and capable of producing real time estimation results, they are still being widely employed in various applications.

Optimisation-based Formulations

In SfM, bundle adjustment [26] has been the predominant method. While in SLAM, a global optimisation-based method and its corresponding graph formulation was firstly put forward in [27]. However, it took several years for the method to become popular due to its complexity. With increased understanding of the structure of SLAM problems and development of sparse linear algebra, efficient solutions to this formulation become available, leading to the boom of such kind of formulations. Now, they amount to be the state of the art methods for conducting SLAM in static environments.

In addition to the mathematical modelling of the filter-based methods, graph-based methods constitute another kind of intuitive representation of the SLAM problem, which is different from the widely employed Bayesian framework.

2.1.3 Two Categories of Typical Representations

Bayesian Network Based Representations

Since SLAM is dealing with noisy sensor data, we can summarise the objective of SLAM as estimating the posterior

$$\arg\max_{\boldsymbol{x}_{1:T},\boldsymbol{m}} p(\boldsymbol{x}_{1:T}, \boldsymbol{m} | \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}) \tag{2.5}$$

where T is the last time step, $\boldsymbol{x}_{1:T}$ represent the robot poses, $\boldsymbol{m}$ corresponds to the map, $\boldsymbol{z}_{1:T}$ represent the observations and $\boldsymbol{u}_{1:T}$ are the control inputs. However, without taking the special structure of SLAM problems, i.e., the static environment and Markov assumptions, into account, this high dimensional problem is intractable. Therefore, being capable of effectively illustrating the structure, Bayesian networks are usually employed. And a dynamic Bayesian network (DBN) that highlights the temporal structure and therefore corresponds to the filtering process is shown in Fig. 2.2.

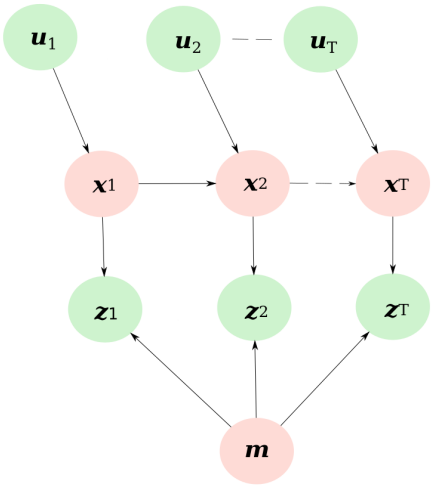


FIGURE 2.2: Representing SLAM problems with DBN, in which $\boldsymbol{x}_{1..T}$ are invisible system states, $\boldsymbol{m}$ the unknown static environment, $\boldsymbol{u}_{1..T}$ input control and $\boldsymbol{z}_{1..T}$ the observations.

Graph-based Representations

Compared with Bayesian networks, graph-based representations [28] emphasise the spatial structure and are very suitable for illustrating the idea of optimisation-based methods.

As shown in Fig. 2.3, for a typical graph in SLAM, its nodes represent robot/camera poses and/or features; its edges correspond to the observation or odometry/loop closure information: if an edge connects a pose and a feature, the edge represents the observation of the feature from the pose; if the edge connects two poses, it corresponds to the odometry or loop closure information of the pose the arrow pointed to in the frame of the other. And if the nodes are all poses, it will be called a pose graph, or if both poses and features are involved, it would be called a pose-feature graph. Moreover, by employing some simple techniques, a pose-feature graph could be converted into a

pose graph easily [29]. In this way, the dimension of the original problem could be decreased therefore simplifying the problem to some degree. As a result, this kind of formulation has drawn much attention and numerous efficient optimisation techniques and applications have been proposed based on it [30–32].

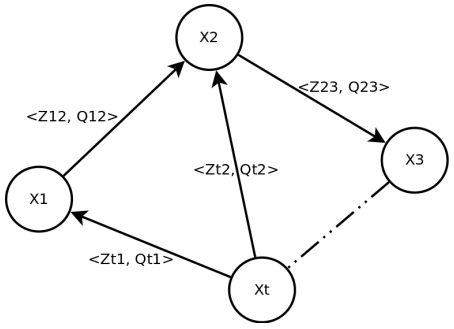


FIGURE 2.3: A graph representing SLAM problems in which $\boldsymbol{z}_{ij}$ represents the observation/odometry of $\boldsymbol{x}_j$ at the position of $\boldsymbol{x}_i$ and $\boldsymbol{Q}_{ij}$ is the corresponding covariance matrix.

Let $\boldsymbol{x} = (x_1, x_2, ...x_n)^T$ represents the states of the nodes in the graph, $\boldsymbol{z}_{ij}$ and $\boldsymbol{\Omega}_{ij}$ denote the actual observation and information matrix of the observation of node j from node i, and $f_{ij}(\boldsymbol{x})$ corresponds to the predicted observation. Then, the residual $\boldsymbol{r}_{ij}$ could be calculated as:

$$\boldsymbol{r}_{ij}(\boldsymbol{x}) = \boldsymbol{z}_{ij} - f_{ij}(\boldsymbol{x}) \tag{2.6}$$

And the corresponding weighted error for this edge is:

$$e_{ij}^2(\boldsymbol{x}) = \boldsymbol{r}_{ij}(\boldsymbol{x})^T \boldsymbol{\Omega}_{ij}(\boldsymbol{x}) \boldsymbol{r}_{ij}(\boldsymbol{x}) \tag{2.7}$$

And the accumulated error of all edges is:

$$E^2(\boldsymbol{x}) = \sum_{(i,j)\in\mathcal{G}} e_{ij}^2(\boldsymbol{x}) = \sum_{(i,j)\in\mathcal{G}} \boldsymbol{r}_{ij}(\boldsymbol{x})^T \boldsymbol{\Omega}_{ij}(\boldsymbol{x}) \boldsymbol{r}_{ij}(\boldsymbol{x}) \tag{2.8}$$

where $\mathcal{G}$ represents the union composed of the indices of the end points of all existing edges. And the solution to the graph-based SLAM is:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \sum_{(i,j)\in\mathcal{G}} \boldsymbol{r}_{ij}(\boldsymbol{x})^T \boldsymbol{\Omega}_{ij}(\boldsymbol{x}) \boldsymbol{r}_{ij}(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} \sum_{(i,j)\in\mathcal{G}} ||\boldsymbol{r}_{ij}(\boldsymbol{x})||^2_{\boldsymbol{\Omega}_{ij}} \tag{2.9}$$

which also corresponds to the goal of optimisation-based SLAM.

Robust Graph-based SLAM

According to modern graph-based SLAM, the front-end is in charge of constructing a graph based on the raw sensor data, and the correctness of the abstracted graph structure could directly affect the optimisation results of the back-end. Therefore, various measures, including robust kernels and switch variables, have been proposed to handle any possible outliers, i.e., misplaced false constraint edges, during the optimisation process [10, 33–35].

2.1.4 Visual Odometry

For most SLAM research and applications in static environments, no matter what kind of technique is to be applied to solve the problem, combining the raw sensor data in temporal sequence, i.e., data association, constitutes the common important step of initialisation. And for graph-based SLAM, initial motion estimation after data association, or obtaining odometry, is also a prerequisite. In modern SLAM, data association and motion estimation belongs to the front end. Some navigation systems make use of global positioning system (GPS), inertial measurement units (IMUs) or rotary encoders, to provide odometry information. However, many other systems are equipped with sensors that could not provide such straight forward odometry data. Therefore, special methods need to be employed to abstract odometry information from such kind of raw sensor data. Through laser scan matching related iterative closest point (ICP) methods, laser odometry can be obtained; or, if the sensors are cameras, then the process of estimating ego-motion of the robot using camera data is called visual odometry (VO) [36, 37].

Depending on the kinds of sensors employed, SLAM as an independent research area could be further divided into several sub-areas. For those research that performs SLAM solely based on cameras has a specific name, i.e., visual SLAM. Within this sub-area, those systems with single cameras are called monoSLAM, those with stereo rigs are called stereoSLAM, and those with RGB-D cameras are called RGB-D SLAM.

Since this work mainly concerns RGB-D SLAM, here we will focus on summarising the achievements in VO to date.

Visual Odometry for Sparse Feature-based Visual SLAM

For sparse feature-based visual SLAM, given a set of calibrated images, we could abstract useful features through a feature detection and matching process. Historically, many point-feature detectors such as corner detectors and blob detectors have been presented,

and nowadays SIFT [38] and SURF [39] are among the most popular methods due to their robustness and effectiveness.

Usually, to maximise the quality of VO, the composition of the images is crucial: we firstly need to ensure that sufficient illumination is supplied while taking the pictures. Secondly, we need to point the camera to a scene with sufficient texture. Furthermore, sufficient scene overlap between two consecutive frames is also of great importance.

Depending on the dimensions of the matched features, three kinds of VO estimation methods could be employed: 2D-to-2D, 3D-to-3D and 3D-to-2D.

2D-to-2D: if the features in two specified images are specified in 2D image coordinates as monoSLAM does, estimating the corresponding essential matrix or fundamental matrix [40] is usually performed firstly, from which the rotation matrix $\boldsymbol{R}$ and translation $\boldsymbol{t}$ (up to a scale) could then be extracted.

3D-to-3D: if there is depth information contained in the images such as RGB-D sensors, or we can derive the depth data from stereo cameras, we can easily obtain the $3D$ coordinates of detected features, deriving the rotation matrix $\boldsymbol{R}$ and translation $\boldsymbol{t}$ then amounts to be minimising the $\boldsymbol{L}_2$ distance between two sets of $3D$ points:

$$\arg\min_{\boldsymbol{T}_k} \sum_i ||\tilde{\boldsymbol{X}}_k^i - \boldsymbol{T}_k \tilde{\boldsymbol{X}}_{k-1}^i||^2 \tag{2.10}$$

where i denotes the ith feature, k represents the kth frame, and $\tilde{\boldsymbol{X}}_k^i$, $\tilde{\boldsymbol{X}}_{k-1}^i$ correspond to $3D$ points in homogeneous form, i.e., $\tilde{\boldsymbol{X}} = [x, y, z, 1]^T$, while $\boldsymbol{T}_k = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix}$.

According to [41], a possible solution to the above mentioned formula could be obtained through SVD. The translation is given by

$$\boldsymbol{t}_k = \overline{\boldsymbol{X}}_k - \boldsymbol{R}\overline{\boldsymbol{X}}_{k-1} \tag{2.11}$$

where $\overline{\boldsymbol{X}}_k$ and $\overline{\boldsymbol{X}}_{k-1}$ are the corresponding arithmetic mean values.

The rotation matrix could be efficiently calculated as

$$\boldsymbol{R}_k = \boldsymbol{V}\boldsymbol{U}_T \tag{2.12}$$

where $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T = svd((\boldsymbol{X}_k - \overline{\boldsymbol{X}}_k)(\boldsymbol{X}_{k-1} - \overline{\boldsymbol{X}}_{k-1})^T)$, and $\boldsymbol{X}_k, \boldsymbol{X}_{k-1}$ are the corresponding sets of $3D$ points.

3D-to-2D: in this case, although the depth information is also available, while deriving the rotation matrix $\boldsymbol{R}$ and translation $\boldsymbol{t}$, we are trying to minimize the $\boldsymbol{L}_2$ distance between the predicted projections of 3D features and their real positions in images:

$$\arg\min_{\boldsymbol{T}_k} \sum_i ||\boldsymbol{p}_k^i - \tilde{\boldsymbol{p}}_{k-1}^i||^2 \tag{2.13}$$

where $\tilde{\boldsymbol{p}}_{k-1}^i$ represents the reprojection of $3D$ point $\tilde{\boldsymbol{X}}_{k-1}^i$ into the kth frame according to $\boldsymbol{T}_k$.

Although some researchers have found that the 3D-to-2D method could produce more accurate results than 3D-to-3D does [42, 43], but the latter is being widely adopted due to its efficiency, simplicity and ease of use.

Since feature matching results usually contain outliers, RANSAC is usually employed before motion estimation. After motion estimation, some researchers may also utilise a local optimisation process over more than two frames to further improve the estimation results.

Visual Odometry from Dense or Feature-free Methods in Visual SLAM

Some researchers have also employed dense methods such as optical flow [44] or featureless methods [45] to obtain visual odometry. Although they are usually computationally expensive and less accurate compared with spare feature-based methods, they are especially useful when it comes to handling low-texture regions.

2.2 Available Methodologies for SLAM in Targeted Dynamic Scenarios

In a typical dynamic scenario that we are interested in this thesis, some moving objects constitute considerable parts of the scenario, and remain in front of the camera performing some activities throughout most parts of the process.

As a matter of fact, based on different assumptions and originated from different view points, divergent ways have been proposed to deal with such kind of scenarios, and one basic difference between them is: to deal with SLAM and moving objects simultaneously or separately (either sequentially or parallel)? Or, in other words, to solve the new problem within an extended version of the traditional SLAM framework (either at

the back-end or using filters) or quest for a more powerful front-end to solve it? According to this criterion, the available methods can be roughly divided into three major categories. The first one can be represented by the formulation called simultaneous localisation, mapping, and moving object tracking (SLAMMOT) [16] and the second is a dense estimation method proposed by [20] for simultaneous multibody structure and motion (called SMSaM thereafter for simplicity), while the last one can be called motion segmentation based multibody SLAM or multibody structure-and-motion (MSaM) in computer vision.

2.2.1 SLAMMOT, SMSaM and Multibody SLAM/MSaM

SLAMMOT and SMSaM constitute two typical solutions of SLAM community for SLAM in dynamic scenarios. The former can be regarded as an extension to EKF, while the latter is built upon the modern SLAM back-end.

SLAMMOT. This methodology is firstly proposed in [16], which aims to conduct $2D$ SLAM and moving object tracking in outdoor environments with laser data. To some degree, it can be regarded as an extension to the traditional filter-based approach.

In this method, each scan is firstly divided into small segments solely based on the distances of adjacent points. At the heart of this method is its special moving object detection techniques which include a consistency-based approach and a moving object map based method. The consistency-based approach firstly assumes that two sequential scans could be registered with ICP methods. Then, based on the registration result, the previous scan is transformed to the current scan position and compared with the latter in a polar coordinate system. Those corresponding pairs of points whose range differences go beyond a certain threshold are then identified as moving points, and those segments that are composed of over one half of moving points will then be identified as moving objects. As for the moving object map based method, it mainly assumes that if a segment is located in an area that used to be occupied by a moving object, then this object could be identified as a potential moving object.

In [15], the moving object detection is based on a binary Bayes method and inverse depth check ([46] implements a similar idea, although local optimisation based method is employed there). In the former method, for each new feature that could be static or dynamic, two corresponding hypotheses are made and accordingly two EKF processes are initialised. The difference between the two hypotheses is defined as:

$$d_m = (\boldsymbol{x}_m^c - \boldsymbol{x}_r^c)^T \Sigma_{\boldsymbol{x}_r^c}^{-1} (\boldsymbol{x}_m^c - \boldsymbol{x}_r^c) \tag{2.14}$$

where $\boldsymbol{x}_r^c$, $\boldsymbol{x}_m^c$ are camera poses for two hypotheses without and with adding the new feature, and $\Sigma_{\boldsymbol{x}_r^c}$ is the covariance matrix of $\boldsymbol{x}_r^c$.

Then a binary Bayes filter is employed to integrate the difference over a fixed number of steps as follows:

$$l_t(\mathcal{H}_m) = \log\frac{p(\mathcal{H}_m|d_m)}{1-p(\mathcal{H}_m|d_m)} - \log\frac{p(\mathcal{H}_m)}{1-p(\mathcal{H}_m)} + l_{t-1}(\mathcal{H}_m) \tag{2.15}$$

If the accumulated log odds value $l_t(\mathcal{H}_m)$ is bigger than a threshold, then the new feature will be classified as static, otherwise it will be identified as moving.

Essentially, in addition to employing EKF for SLAM and moving object tracking, this method also relies on EKFs to check whether each newly emerged feature is static or not. Therefore, no other motion segmentation method is required. And it constitutes the foremost advantage of this method.

However, such kind of moving object detection methods have some limitations. For the laser data based method, the initial object segmentation method based on distances is hard to be generalised; with moving objects in sight, the ICP results tend to be erroneous. For the method used in stereoSLAM, it assumes a correct initial step that no dynamic feature involved; in the current EKF framework, each moving object is usually represented as several feature points tracked by Kanade-Lucas-Tomasi (KLT) tracker, therefore direct extension to attain dense details of the moving objects would be hard. Moreover, these two kinds of methods also assume that dynamic object could be identified within the initial few steps. However, it does not always hold. On the other hand, as we know from SLAM in static scenarios, EKF itself suffers from inconsistency problem [47].

SMSaM. More recently, by extending upon the modern SLAM framework [48] to incorporate motion group labels as another kind of variables into the optimisation framework, Anastasios et al. [20] propose to deal with the elements of the multibody structure and motion problem in a simultaneous way. Different from the filter-based approach, it amounts to be a formal formulation of the problem in terms of a modern optimisation based energy minimisation process:

$$E(d, L, \mathcal{J}) = \lambda E_{data} + \alpha E_{reg} + \beta E_{potts} + \gamma MDL \tag{2.16}$$

where E_{data} represents the sum of photometric errors over all of the frames, E_{reg} is a spatial regularisation term for the depth map, E_{potts} corresponds to discrete pairwise costs encouraging adjacent pixels to share same models, and MDL (Minimum

Description Language) is in favour of solutions composed a smaller number of motion groups.

However, this batch method is limited to $2.5D$ mapping of a small volume corresponding to a reference frame, applicable to short videos composed of rigid objects only, and relying on GPU to shoulder the computational cost. Moreover, although the paper has proposed a hill-climbing solution approach for the problem, there is no theoretical justification that for such a complicated multivariate formulation we could always find the global minimum in this way. And handling non-rigid moving objects in this way would also be hard. Therefore, much work is still in need in this direction before it could find applications in general dynamic scenarios.

Multibody SLAM/MSaM. Contrary to SLAMMOT and SMSaM, multibody SLAM/MSaM firstly focuses on employing separate motion segmentation methods to decouple the complex dynamic scenarios into several motion groups, and then for each group making use of available optimisation-based back-ends to obtain SLAM results.

In computer vision, especially in recent years, more and more researchers have studied MSaM using a single camera. Konrad et al. [49] propose a generic recover-and-select procedure based on model selection which is a branch of statistics and information theory for MSaM outlined as in Algorithm 1:

Algorithm 1: Outline of n-view multibody structure-and-motion method [49]

1. ***Tracking***: track feature points through the sequence;
2. ***Generating candidates***: for each pair of consecutive frames $(j, j+1)$;
(a) Sample a set of two-view motions $\{\mathcal{Q}_i^j\}$;
(b) For each $\mathcal{Q}_i^j$, estimate inlier set and standard deviation;
(c) Cluster $\{\mathcal{Q}_i^j\}$ and re-estimate representatives $\{\overline{\mathcal{Q}}_i^j\}$ for each cluster j;
3. ***Motion linking***: recursively link $\{\overline{\mathcal{Q}}_i^j\}$ through frames to obtain candidate motions $\{\mathcal{D}_k\}$;
4. ***Model selection***: pick the best subset from $\{\mathcal{D}_k\}$;
(a) build the codelength/likelihood function $\mathcal{D}(\mathbf{b})$ for the candidate motions;
(b) maximize $\mathcal{D}(\mathbf{b})$ over the index vector $\mathbf{b}$ to determine the best subset;
5. ***Postprocessing***: enforce temporal consistency to clean up segmentation;
6. (optional) ***Triangulation***: triangulate $3D$ coordinates of feature points

In [18], the authors further explore the challenges a practical algorithm for MSaM using a single camera is facing and thus the requirements it needs to meet, advocating online estimation of the number of moving objects including its changes, feature tracking, segmentation and $3D$ reconstruction simultaneously. Then, they propose a probabilistic solution (as shown in Algorithm 2) based on [49] to address some of the requirements

including motion splitting and merging, as well as mirror symmetry ambiguity, and demonstrate promising results using several short datasets.

Algorithm 2: Dynamic structure and motion pipeline [18]

1) Instantiate new features, and track all features;

2) **if** *not enough parallax (e.g. average feature displacement* < 50 *pixels)* **then**
| goto step 1;

else if *sufficient parallax and no SfM yet* **then**
| do initial segmentation and 3D structure computation;
| create symmetric models for small objects;
| goto step 1;

else
| continue;

end

3) Try to estimate new 3D motion for active models;

if *too many outliers (e.g.* $> 25\%$*) for a motion model* **then**
| try to split;

end

Instantiate new models if necessary;

4) **if** *waiting period is over and the number of unexplained tracks surpasses a threshold (e.g.* 10%*)* **then**
| try to detect new motion models;
| create symmetric models for small objects;

end

5) **if** *waiting period is over* **then**
| try to fuse active motion models greedily;

end

6) Remove symmetric models, where enough evidence available (e.g. $\mathcal{M}_{mirrored}/\mathcal{M}_{correct} > 1.2, \mathcal{M} =$ model score);

7) goto step 1;

In [19], the authors propose another realtime incremental multibody SLAM algorithm emphasising on combining multiple cues to deal with degenerate situations.

Please note that, despite of the differences, all of the above mentioned categories of methods claiming to be capable of dealing with dynamic scenarios are restricted to handling rigid moving objects.

2.2.2 Motion Segmentation based Multibody SLAM/MSaM

In multibody SLAM/MSaM, the basic idea can be summarised in two steps: firstly allocate the available sparse features into different groups according to their motion types, widely known as motion segmentation that we will cover in more details in Section 2.3, and then for each of the group, if they are rigid, we can directly apply the classic SLAM framework developed for static environments to solve it. As we have seen, till now, most of the available work concerns dynamic scenarios with rigid moving objects only. Quite recently, dense segmentation methods capable of handling non-rigid moving objects are also emerging [50, 51]. However, $3D$ reconstruction of their structures is still an open problem.

Therefore, multibody visual SLAM/MSaM calls for a robust motion segmentation method to handle various challenging dynamic scenarios which may include occlusion, motion blur and featureless regions.

2.2.3 Summary of Different Methodologies for Conducting SLAM in Dynamic Scenarios

SLAMMOT. As a classical method widely employed in SLAM, EKF-based approaches are also capable of detecting moving objects. Nonetheless, they may suffer from the inconsistency problem.

SMSaM. The formulation of the problem in terms of energy minimisation is theoretically attractive. However, putting all variables in a mega-problem make it computationally intensive and liable to be trapped within local minima. Furthermore, it is strictly limited to short videos of rigid objects in small regions.

Motion Segmentation based Multibody SLAM/MSaM. The *divide-and-conquer* methodology enable us to focus on addressing different sub-problem at each step, thus facilitating the development of powerful specialized methods for each of them. For example, at the current stage, we can see that non-rigid moving object segmentation methods are emerging [50, 51]; on the other hand, after motion segmentation, the mature algorithms for SLAM in static environments can be directly utilized to solve the second sub-problem, and the ongoing intensive research on non-rigid object reconstruction in computer vision [52] will also be directly beneficial to the mapping task of non-rigid objects in SLAM as well.

Through the comparison above, we think the idea of the third methodology is quite promising for handling practical challenging data. However, to ensure that we can push

the boundary along this track, firstly we need to be aware of what has been achieved in this direction so far, and what is still missing. Since the achievements in SLAM have already been summarised during previous sections of this chapter, in the following section, we will focus on evaluating the available methods for motion segmentation and SLAM, corresponding to those two steps involved in the process.

2.3 Motion Segmentation

2.3.1 Definition

Motion segmentation originates from computer vision for image understanding and video coding. Initially it mainly concerns segmenting an image into coherent regions solely based on $2D$ $(\delta u, \delta v)$ motion model, but now $3D$ motion model based segmentation methods prevail. Although motion segmentation seems like so intuitive a notation that we can hardly find a formal definition for it in computer vision, according to our understanding, motion segmentation algorithms aim to separate a dynamic scenario composed of moving elements and static environment apart from each other according to their different motion types. In other words, motion segmentation means dividing a target image into different motion groups. Generally, there are some basic questions that motion segmentation algorithms need to answer: how many moving groups are there in the scenarios? What are they? Can we detect them as soon as possible? In addition, the number of motions can change from time to time, so ideally motion segmentation algorithms should be sensitive and quick enough to detect the changes in real time.

However, due to lack of clear specifications of the problem in details, implementations of the existing motion segmentation methods are quite different from each other, either in terms of the input, output, or the process.

2.3.2 Different Implementations

When it comes to practice, the required input to motion segmentation methods can be sparse features or the whole image, and the data can be just two frames or a sequence of images (called 2-view based or n-view based motion segmentation methods respectively). Accordingly, the output are groups of sparse features or densely separated regions. When it comes to the segmentation process, most available methods in computer vision are n-view based [53], relying a factorization process to cluster the point trajectories into different groups. For 2-view based methods using only RGB data, fundamental matrix of multi-view geometry [40] is widely employed as the criteria for segmentation; as to

2-view based methods using RGB-D data, simple distance-based algorithms have been proposed [54].

However, there may exist some misunderstanding of the differences between n-view and 2-view based segmentation methods. N-view based segmentation methods call for the point trajectories across several frames, therefore they assume that data association (also called motion linking) is performed beforehand. On the other hand, for 2-view based motion segmentation methods, they only require that the same point appears in the current two frames, and its segmentation is solely based on and for the current two frames only. As a result, data association information beyond this two frames is absent from both its input and output. However, it is not due to the inability of 2-view based motion segmentation methods themselves; instead, it is just because they do not require data association beyond two frames to perform segmentation, while n-view based methods calls for robust data association across several frames as their prerequisite. In other word, both of the two available kinds of motion segmentation methods can only perform the job of segmentation, therefore to conduct SLAM in dynamic scenarios, we are still in need of some other methods to execute the data association task.

On the other hand, the available methods are all based on the raw sensor data, therefore their capability of distinguishing small movements is limited by the accuracy of the sensor data.

2.3.3 Related Work

In computer vision, ever since 1990's, motion segmentation has been regarded as a very important step for MSaM and many other applications. From $2D$ to $3D$, numerous algebraic and statistical motion segmentation techniques based on different camera models have been put forward, and [53] has given a detailed review of them. Nevertheless, motion segmentation using only RGB data turns out to be a non-trivial task, especially when the number of motion groups is unknown, or when the motion groups are changing from time to time. Most of the available methods assume that the number of motion groups occurring in the scenes is known beforehand and fixed during the process, then through a clustering analysis, they can allocate each point to one of the groups. Furthermore, most of the available algorithms are n-view based batch process, which means that their segmentation results are always delayed.

Another branch of work, which can be seen as an extension to the method in [15, 16], determines the number of motions through a model selection process [55].

In [19], the authors propose an incremental motion segmentation method by combining optical flow and two-view geometry for monocular data. It relies on imposing geometric and flow vector bound constraints to conduct segmentation and deal with degenerate situations.

On the other hand, $6D$ vision [56] combines stereo information to detect moving cars or pedestrians, but it assumes that the ego-motion is known with the help of another sensor. In contrast, in this thesis, we are conducting segmentation with RGB-D data as the only source of information.

Motilal et al. [17] propose a RANSAC-based realtime algorithm for moving object detection in stereoSLAM, and demonstrates its effectiveness in scenarios that contains a moving person. By assuming that the largest motion group of features corresponds to the static environment and through image warping, it could detect the moving person in some simple cases. It is the first of its kind. However, as noted by some other researchers [53], the efficiency of RANSAC alone will be compromised significantly when there are more moving objects in the scenario.

2.3.4 Evaluation Methods

In computer vision, to evaluate motion segmentation algorithms, it is usually assumed that the benchmark datasets are free from outliers, then checking the accuracy of segmentation results in terms of true positive, false positive [53] turns out to be a common practice. However, for real data, such kind of clean input is rare.

Nonetheless, when it comes to SLAM, especially RGB-D SLAM, there is no such kind of benchmark datasets available yet. Furthermore, the aforementioned accuracy quotation for motion segmentation methods usually could not represent their ultimate performance in SLAM applications. For example, it is often the case that a motion segmentation method could achieve high accuracy, but its corresponding SLAM estimation results can still be far from satisfactory, the underlying possible reason of which is that false positive points, although small in quantity, could still spoil the estimated camera poses.

Therefore, in this thesis, we emphasize on comparing the visual odometry and final SLAM results to appraise the performance of the related algorithms.

2.3.5 Difficulties

Till now, most of the available motion segmentation methods are limited to rigid objects and large movements, because such scenarios are much easier to deal with. When the

differences between two motion groups become closer, choosing appropriate threshold turns out to be a very tricky problem. And for articulated/non-rigid objects, how to deal with the intersection parts between different groups is largely untouched, let alone describing their motions. In this thesis, we pay special attention to such kind of difficulties involved in the segmentation process, i.e., continuous movements and non-rigid objects.

2.3.6 Usability Analysis of Different Methods

Compared with dense methods, sparse feature-based motion segmentation methods are generally quicker, with both 2-view and n-view versions available. However, they suffer from some limitations. Firstly, sparse features cannot catch all the details of moving objects in the scenarios. Secondly, they are generally sensitive to sensor noises and outliers. While dense methods can be more robust in these scenarios, their ability to cope with motion blur in general is still not satisfactory.

N-view Sparse Feature-based Motion Segmentation

N-view sparse feature-based motion segmentation methods require that the same sets of feature points be visible in a series of frames. It turns out to be a very good condition to enable the related methods to focus on motion segmentation itself. Firstly, it stipulates that the input feature trajectories are pre-filtered against outliers; secondly, it requires that data association be achieved beforehand, therefore motion segmentation methods in this category do not need to concern about data association at all.

However, fulfilling this condition amounts to be a very tough task for practical applications. If an object goes through a long distance, or rotates beyond a certain angle, the previously tracked feature points can easily get lost. As a result, those currently available methods are still limited to very short videos composed of small range of translations and rotations.

2-view Sparse Feature-based Motion Segmentation

The input to the 2-view sparse feature-based motion segmentation methods [17, 57] can be two consecutive or non-consecutive frames, and they may come from a single camera, stereo rig, or RGB-D camera. Theoretically, compared with those n-view based methods, 2-view based algorithms can detect possible motions as soon as possible. Nevertheless, their segmentation results are liable to noises, and they need a motion linking step to combine segmentation results at each time step to form a general idea of the correct membership of each point, which amounts to be a accumulative filtering process that no mature solution is available yet.

Dense Moving Object Segmentation

The current available dense moving object segmentation methods are also based on long-term trajectory analysis, but employ optical flow instead of feature detection and matching results as the starting point. Theoretically, dense moving object segmentation aims to assign each pixel to a motion group, therefore it can produce very intuitive results. However, this methodology is still not robust enough to deal with general dynamic scenarios. We will also make some endeavours in this direction.

Nevertheless, we need to note that moving object segmentation and motion segmentation are essentially two different concepts that only overlap for scenarios composed of rigid objects. Otherwise, for articulated moving objects contain many internal motion groups for their different parts; for general deformable non-rigid objects, it is even harder to simply use motion groups to describe their activities.

2.4 Usability of Available SLAM Solution Frameworks for Conducting SLAM in Dynamic Scenarios

After motion segmentation, theoretically, for each group, we can directly employ the available SLAM back-ends to estimate their respective motions and structures. However, in terms of usability, which SLAM solution framework is the best choice? We will give our answer after a brief comparison of them.

2.4.1 Pose-graph SLAM Versus Pose-feature-graph SLAM

In addition to suffering from the inconsistency problem [7], filters have also been shown in [58] to produce less accurate results than keyframe-based BA. On the other hand, graphs amount to be an effective kind of formulations of the SLAM problem to highlight the spatial structure. Therefore, in this work, we prefer to employ graph-based SLAM frameworks of which pose-graph SLAM could be regarded as the state-of-the-art SLAM back-end formulation in terms of simplicity, speed and accuracy. Compared with pose-feature graphs, pose graphs enable us to firstly concentrate on solving camera poses that have lower dimensions thus mitigating the complexity of the original problem and improving the efficiency. Therefore, it is especially suitable for robust SLAM in dynamic scenarios.

In this thesis, we have chosen pose graph as our back-end representation method, and employed G2O [33] to perform the optimisation process.

2.4.2 Dense SLAM Versus Sparse Feature-based SLAM

Dense SLAM is generally regarded as a slower but more robust method when compared with sparse feature-based SLAM. While conducting SLAM in dynamic scenarios, the images are shared by different motion groups, which means that for each group the total number of features will be much fewer than usual. Therefore, dense SLAM is especially useful for this situation. Furthermore, the latest version of dense SLAM [59] in RGB-D SLAM can run in real time, which further facilitate its usage in our applications.

In this thesis, we have chosen dense visual SLAM as our reference in Chapter 3 and 4, and endeavoured to combine it with our latest dense moving object segmentation method to form a robust solution framework for SLAM in general dynamic scenarios in Chapter 5.

2.5 Benchmark Datasets for RGB-D SLAM

Recently, low cost lightweight commodity RGB-D sensors, such as Microsoft Kinect and Asus Xtion Pro Live sensor, have caught much attention. Since they can supply depth information in addition to RGB information, yet are simpler to use than the traditional stereo cameras, and cheaper than laser scanners, more and more researchers have adopted them as a new kind of sensor to solve SLAM problems. Also thanks to the other closely related mature areas, such as monoSLAM, stereoSLAM, laser SLAM and many others in computer vision, RGB-D SLAM has a solid foundation to move forward. It is under such a general background that RGB-D SLAM as an important new branch has developed very quickly right after its emergence. Now, numerous efficient and robust RGB-D SLAM algorithms have been proposed and many of their codes are available for practical applications. Some of these achievements should be attributed to the availability of good benchmark datasets, which have been provided during the initial forming stage of this area.

Although many benchmark datasets are available in laser and camera based SLAM, they are still relatively rare in RGB-D SLAM. Two famous benchmark datasets in RGB-D SLAM are the MIT Stata Center dataset [60] and the one provided by TUM [61]. The first one is a vast scale dataset collected over a long time in a 10-storey building, and it aims for a wide range of research areas in addition to visual SLAM, but only part of it includes RGB-D data.

While TUM benchmark includes RGB-D data along with ground-truth values from an external motion capture system, and provides some readily available tools for automatic

evaluation. It contains 39 sequences of various kinds of textures and structures, from single objects to some typical office or industrial hall scenarios; during the sequences, the camera is put on a robot or hand-held, therefore it sometimes move slowly with constraints, at other times move very fast; and the environment is totally static in some sequences, while in some others moderately dynamic (one or two people with small movements) or in the rest highly dynamic (two people move rapidly in front of the camera). In other word, it is deliberately composed of systematic real sequences with various difficulties, therefore it is very convenient for fellow researchers to develop, polish and then test new algorithms step by step with this dataset. Since it aims to benchmark RGB-D SLAM in static environments as well as robust SLAM in dynamic scenarios, there is only ground truth for camera poses.

Similar to what Middlebury benchmark datasets [62–64] have achieved in computer vision, those few RGB-D SLAM benchmark datasets have stimulated the birth of new methods both as a director and also common touchstone. As a result, it has become a common practice for a new algorithm to report its performance on benchmark datasets and show its superiority than the other related ones.

In this thesis, we have mainly employed the TUM dataset as a benchmark to compare the performance of our algorithms against that of related methods.

2.6 Towards Motion Segmentation based Robust SLAM

2.6.1 Missing Elements

Based on the comparative analysis of the related work, we discover that there is still much work needed to be done: the available motion segmentation methods are generally limited to short videos of rigid moving objects; the development of robust data association methods for dynamic scenarios has been largely relying on RANSAC; more research on suitable evaluation methods for real data are needed; practical solution frameworks for robust SLAM in fast dynamic environments are still missing.

2.6.2 Our Basic Objectives

Following the track of motion segmentation based multibody SLAM/MSaM, in this thesis, we firstly focus on proposing effective motion segmentation methods that can handle challenging real data. Then, by combining our motion segmentation methods and pose-graph SLAM framework, we aims to propose practical motion segmentation

based robust SLAM frameworks that can handle both static and dynamic scenarios in a unified manner. Furthermore, based on the available challenging benchmark dataset, we will discuss some useful ways for evaluating our motion segmentation algorithms and SLAM frameworks.

Chapter 3

Sparse Feature Distance-based Motion Segmentation for Multibody RGB-D SLAM

To perform SLAM in dynamic scenarios, multibody SLAM, i.e., conducting SLAM in a static environment with some rigid objects moving in it, is a comparatively simpler case to handle than the others. Therefore, in this chapter, we have chosen multibody RGB-D SLAM as the starting point, aiming to propose a sparse feature distance-based motion segmentation algorithm and a corresponding back end for it and then verify its efficiency and robustness using both simulated and real RGB-D data.

In the following sections, we firstly review related work in motion segmentation for multibody RGB-D SLAM, then we present our motion segmentation theory, algorithm, implementation, simulation and experimental results. In the end, we discuss the applicability of our algorithm for multibody RGB-D SLAM and other dynamic scenarios as well, highlighting the relevance of our work in the following chapters.

3.1 Motivation

In Chapter 2, we have given a detailed review of the related motion segmentation methods that have been proposed in multibody SLAM/MSaM. While using a single camera, many 2-view based motion segmentation methods are readily available and most of them are built upon the fundamental/essential matrix of multi-view geometry [57]. However, this criterion is not so strict. Therefore, the corresponding segmentation results usually contain some false positive elements. Since RGB-D cameras can provide us with depth

information along with RGB data, in this chapter, we will explore the possibility of presenting a more stringent algorithm for motion segmentation.

As we know, for both static and rigid moving objects, the distance between any two points within each motion group will remain the same through out the dynamic process. It is based on this observation that [54] proposes a simple and efficient method to segment motions using RGB-D data. In RGB-D SLAM, FOVIS [65] has also made use of this property to filter out moving objects. But as the authors of [54] have pointed out, their algorithm is only using a necessary instead of sufficient condition to segment motions, which means that even after the segmentation, one still can not safely conclude that any two points within the same motion group really have no relative movements.

However, since relative motion represents one of the essential differences between being relatively static and moving, as we will see, they can be applied to various situations, independent of the types and number of motions. If we can deduce the conditions that are both necessary and sufficient, we can propose a more reliable algorithm based on it to segment the motions existing in scenarios that multibody RGB-D SLAM aims at, and then we can utilise the framework of the currently available pose-graph solution for SLAM in static environments to solve multibody problem. This constitutes the basic idea of this chapter.

3.2 An Efficient RGB-D Motion Segmentation Method

Firstly, we try to find out the necessary and sufficient conditions that we need to check before we can safely conclude that one point has no relative motion to the other points during the process and thus they belong to the same motion group. Then, we propose an efficient motion segmentation algorithm based on these conditions.

3.2.1 Problem Restated

For two overlapping but not necessarily sequential RGB-D frames, through feature detection and matching, we can extract two corresponding sets of feature points residing either on moving objects or in the static environment. Since we have depth information, we can easily obtain their $3D$ positions relative to their camera centres. Then, the problem becomes: for these two sets of $3D$ points, how can we divide them into different motion groups efficiently?

More specifically, given two RGB-D frames F_i and F_j (where i, j correspond to the frame identifiers), we define that *a pair of shadow points* means two local $3D$ points $\mathbf{p}_{ik}$ and

$\mathbf{p}_{jk}$ (where k represents the point identifier and $\mathbf{p}_{mn} = (x_{mn}, y_{mn}, z_{mn})^T$) corresponding to two matched $2D$ features $\mathbf{f}_{ik}$ and $\mathbf{f}_{jk}$ (where $\mathbf{f}_{mn} = (u_{mn}, v_{mn})^T$, u_{mn} and v_{mn} are image coordinates, and $[\mathbf{f}_{mn}; 1] = M[\mathbf{p}_{mn}; 1]$ for a pinhole camera with a 3×4 projection matrix M). If the matching is correct, this pair of shadow points should correspond to the same $3D$ point $\mathbf{P}_k = (x_k, y_k, z_k)^T$ in the physical world which can be called a *physical point*. Similar to ordinary shadows, those two shadow points are just the $3D$ snapshots of this physical point in two local coordinates at two different time steps i and j respectively. Furthermore, we stipulate that *the distance between shadow points* is a Euclidean distance that can only be defined and calculated between two shadow points from the same image frame, thus in the same local coordinates. Now, suppose we have two pairs of shadow points $(\mathbf{p}_{ik}, \mathbf{p}_{il})$ and $(\mathbf{p}_{jk}, \mathbf{p}_{jl})$ (corresponding to two $3D$ physical points $\mathbf{P}_k$ and $\mathbf{P}_l$) from frames F_i and F_j respectively, we can obtain easily the distances between them, i.e., d_{ikl} and d_{jkl} (where d_{lmn} represents the Euclidean distance between shadow points $\mathbf{p}_{lm}$ and $\mathbf{p}_{ln}$ in frame l, i.e., $d_{lmn} = \sqrt{(x_{lm} - x_{ln})^2 + (y_{lm} - y_{ln})^2 + (z_{lm} - z_{ln})^2}$). From now on, we will focus on comparing these two distance values. Ideally, without considering noises, *if their corresponding* $3D$ *physical points come from the same motion group, these two values should remain the same; otherwise, we can conclude that these two pairs of shadow points belong to different motion groups.* Similarly, for a third pair of shadow points, if their distances to the initial two pairs of shadow points remain the same across the two frames, we can also decide that their corresponding physical points come from the same group. These are the simplest situations, however. Now, if we have another or more pairs of shadow points, is it enough for us to just compare their distances to the first three pairs of points? Or, do we need to check the changes of the distances between every possible pairs of points before we can safely draw a conclusion? The following section tries to answer this question.

3.2.2 Theory

Based on geometry, we can deduce the necessary and sufficient conditions for grouping relatively static $3D$ points.

Proposition: a $3D$ physical point's position can be uniquely specified by three predefined non-collinear $3D$ points through three distances and a side discrimination.

As we know, in $3D$ space, the potential points having a specified distance to a predefined point will constitute a sphere. Then, as shown in Fig. 3.1, the potential points that have predefined distances $d1$ and $d2$ to any two specified points $\mathbf{P}1$ and $\mathbf{P}2$ (where $\mathbf{P}i = (x_i, y_i, z_i)^T$) respectively ($d1 + d2 > d12$ where $d12$ denotes the distance between $\mathbf{P}1$ and $\mathbf{P}2$), will constitute a circle which is the intersection of the two spheres defined

by the following two equations:

$$\begin{cases} (x-x_1)^2+(y-y_1)^2+(z-z_1)^2 = d1^2 & (3.1) \\ (x-x_2)^2+(y-y_2)^2+(z-z_2)^2 = d2^2 & (3.2) \end{cases}$$

If we choose a third point $\mathbf{P}3$ that does not lie on the line defined by $\mathbf{P}1$, $\mathbf{P}2$, and specify that the potential points need to have a preset distance $d3$ to the third point, then the potential points meeting the $d1$, $d2$ and $d3$ conditions simultaneously will lie on the intersection of the circle and the third sphere, which has at most two possible points, located on both sides of the plane determined by the three pre-specified points and symmetrical to each other with respect to the plane, which also correspond to the solutions to the following three equations:

$$\begin{cases} (x-x_1)^2+(y-y_1)^2+(z-z_1)^2 = d1^2 & (3.3) \\ (x-x_2)^2+(y-y_2)^2+(z-z_2)^2 = d2^2 & (3.4) \\ (x-x_3)^2+(y-y_3)^2+(z-z_3)^2 = d3^2 & (3.5) \end{cases}$$

Therefore, if we further specify that the potential points should remain on the same side of the plane which can be identified using the norm of the plane defined by $\mathbf{P}_{12} \times \mathbf{P}_{13}$ where $\mathbf{P}_{ij} = \mathbf{P}j - \mathbf{P}i$, then we can reach the conclusion that there is only one possible point that can meet the requirements.

Although this proposition just mentions $3D$ physical points, for their corresponding shadow points which can be regarded as the same points captured by the camera at different times, we can estimate their relative motion by comparing their distances in different frames. Therefore, similarly, for any pair of shadow points, during a motion process, only stipulating that its distances to three pairs of non-collinear shadow points remain the same is not enough, we need to make sure that it remains on the same side of the plane defined by these three specified pairs of points. Only when all of these conditions are met, can we conclude that this pair of shadow points belong to the motion group defined by the three specified pairs of shadow points.

More generally, we can also see that, for any further coming pair of shadow points, checking their distances and side with respect to the three pairs of specified non-collinear shadow points is enough to judge whether the additional points come from the same motion group or not. In other word, we do not need to check the distances between every two possible pairs of points. Because of this, the the proposed algorithm is very efficient.

In summary, this proposition makes it possible for us to conduct motion grouping in an efficient way.

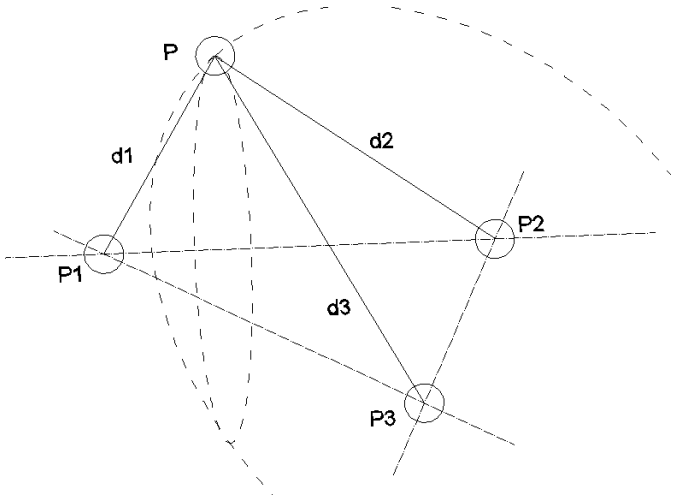


FIGURE 3.1: Constrain a new point P using three non-collinear points P1, P2 and P3

3.2.3 Algorithm

The inputs to our algorithm are two RGB-D frames, from which we can extract two sets of shadow points. And the main function of our algorithm is to divide the pairs of shadow points into different groups. Before coming to the details, we firstly introduce two definitions.

Definition 1: *A suitable group* for a pair of points means that by comparing that pair with the points from the group in the way mention above, their distance changes are within specified thresholds which are to be discussed below.

Definition 2: If the number of pairs in any group reaches three, and the corresponding pairs of points are non-collinear, then they become the *basic components* of this group, and the group becomes a *mature* group, from which we can derive the normal vector of the corresponding plane.

And our RGB-D motion segmentation algorithm is illustrated as follows:

Algorithm 3: Sparse feature distance based RGB-D motion segmentation algorithm

Inputs: two frames of RGB-D data;

Output: different groups of matched $3D$ feature pairs;

1. ***Obtain a set of*** $3D$ ***matched point pairs***: given two RGB-D frames, conduct feature detection and matching, and only those pairs of shadow points with depth are kept in an initial set $\mathcal{P}$;

2. ***Go through all point pairs***: for each pair of points left in $\mathcal{P}$, perform tasks *2.1-2.2*;

2.1 ***Find a corresponding group***: firstly compare them with the existing groups one by one till finding one suitable group for them;

(a) ***Deal with immature groups***: if the current group is not mature, compare the distances of the chosen pair of points to the available one or two pairs of shadow points in the group. If the distances do not change across the two frames, then put the pair of points into the current group; otherwise compare with the next group;

(b) ***Deal with mature groups***: if the current group is mature, calculate the distances of the chosen pair of points to the three basic components of the group, and its relative position to the plane formed by the basic components (which side it is located). If these four results remain the same, then the chosen pair of points belongs to the current group. Otherwise, compare with the next group;

(c) ***Allocate a new group if needed***: finally, if it turns out that the chosen pair of points belongs to none of the existing group, allocate a new group for it;

2.2. ***Remove the current pair from*** $\mathcal{P}$;

In this way, just after going through all of the pairs of points sequentially in one round (no iterations), we can correctly group them according to their motions.

3.2.4 Discussion

The characteristics of this algorithm include:

Efficiency: theoretically, the computation cost is $O(N_p)$ (in which N_p denotes the total number of the pairs of points). Detailed analysis can be found in Appendix A.

Dependent on sparse features with depth information: since it is based on the essential property of being relatively static as opposed to be relatively dynamic, if we have depth information, we can use this algorithm to segment rigid body motion groups. In addition, it just needs a set of sparse features as input.

Reliability and robustness: it checks both the necessary and sufficient conditions for rigid body motion segmentation, so theoretically its result should correspond to the ground truth motion groups. However, in practice, inaccurate sensor data and outliers will inevitably bring in some additional small groups. Nevertheless, if we just filter out such very small groups, we can still obtain results of the number of motion groups and their respective members consistent with the ground truth.

The thresholds are only affected by the precision of sensor data: it means that only when we apply this algorithm to $3D$ point data coming from a different type of sensors, do we need to adapt its threshold values. For Kinect, its depth noise is about $2mm$ at $1.0m$, and $2.5cm$ at $3.0m$ [66], varying across the distance. In this chapter, since we are concerned about features within $1.0 - 3.0m$, we simply choose $2.5cm$ as the threshold.

The choice of base points: its results will be affected by the quality of the three base points of each group. We may firstly choose those points lying closer to the camera whose depth data is relatively more accurate; and ensuring that the distances between them to be more than ten times of the range of sensor noises could also help to boost the accuracy of the calculated distances and normal vectors.

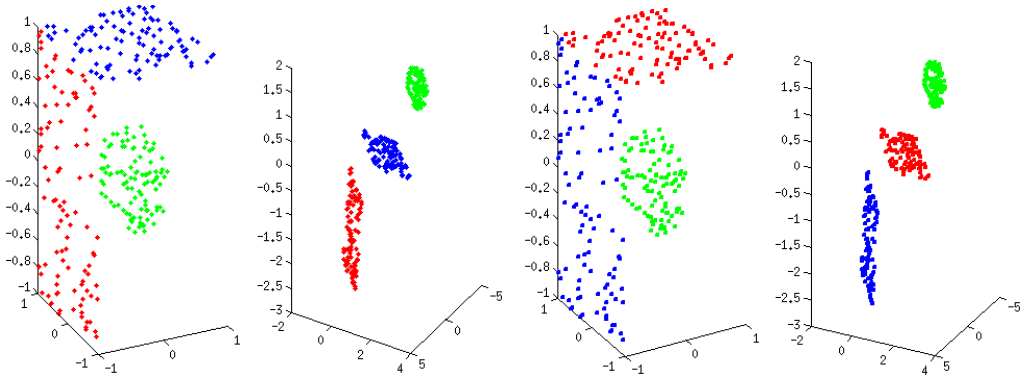
3.2.5 Segmentation Results Using Simulated Data

We firstly run the algorithm using some simulated data with the combination of different shapes such as planes and balls, and motion parameters, for which we assume that there is no noise in positioning and the feature matching is perfect. Some typical data and results are shown in Fig. 3.2a-3.4b. Among them, Fig. 3.2a, 3.3a and 3.4a represent three kinds of simulated scenarios, in which different colours denote different motion groups; Fig. 3.2b, 3.3b and 3.4b show the motion segmentation results, in which different colours also represent detected motion groups. Please note that here different colours are just utilised to distinguish different motion groups, and it does not means that our results need to employ the same colour as the ground-truth has done to represent any specific group. As we can see, the segmentation results are 100% accurate.

3.2.6 Efficiency

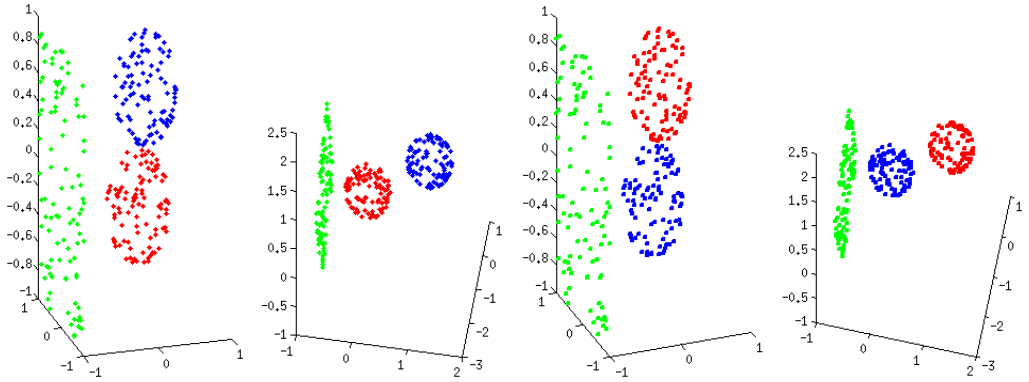
Using a CPU of Intel(R) Xeon@3.33GHz, the efficiency of our algorithm is shown in Table 3.1. In this table, the second column N_p denotes the number of feature points to be grouped, the third column N_g represents the number of motion groups, which is

unknown before our segmentation step, and the last column T_{mean} symbolises the computation time for segmentation. The first three rows correspond to simulation results, and the last row comes from our experimental data which will be discussed next. With the currently un-optimised sequential MATLAB code, we can obtain the segmentation results at the approximate rate of 7.5 points/ms, independent of motion group number, and the required time is proportional to the number of feature points. All of these have confirmed our theoretical analysis of its efficiency.



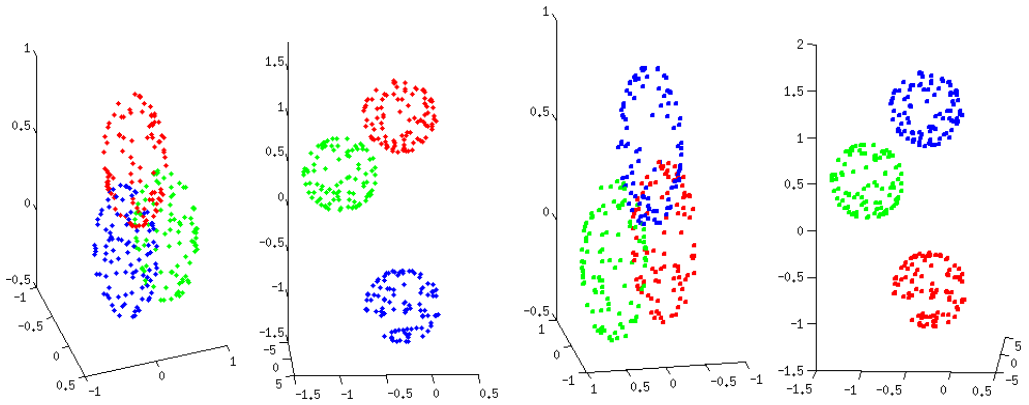
(A) Simulated motions. (B) Motion segmentation result.

FIGURE 3.2: Motion segmentation result for the simulated 3 planes scenario. Here different colours are only used to differentiate different motion groups, so not necessarily the same as the ground-truth marked. Best viewed in colour.



(A) Simulated motion. (B) Motion segmentation result.

FIGURE 3.3: Motion segmentation result for the simulated 1 plane and 2 balls scenario. Here different colours are only used to differentiate different motion groups, so not necessarily the same as the ground-truth marked. Best viewed in colour.



(A) Simulated motion. (B) Motion segmentation result.

FIGURE 3.4: Motion segmentation result for the simulated 3 balls scenario. Here different colours are only used to differentiate different motion groups, so not necessarily the same as the ground-truth marked. Best viewed in colour.

TABLE 3.1: Computation time in relation with the number of feature points and groups

Dataset NO.	N_p	N_g	T_{mean}(ms)
Simulation1	450	3	55
Simulation2	300	2	38
Simulation3	150	3	19
Experiment1	152	3	21

3.3 Evaluation with A Simulated Multibody RGB-D SLAM Problem

3.3.1 The Simulated Multibody RGB-D SLAM Scenario

We simulate a typical multibody RGB-D SLAM scenario as shown in Fig. 3.5, which contains a depth camera and two moving objects in addition to some static features. Throughout the process, the two objects are moving independently along different paths, and we keep recording the ground truth images at discrete steps around a loop.

3.3.2 Motion Segmentation Results Using Simulated Noisy Data

In order to test the robustness of Algorithm 3, we add some independent Gaussian noises to the u, v (standard deviation within 0.5 pixel) and d (standard deviation within 1% of the ground truth depth value) values of the recorded images respectively, and then

take the results as the input to our algorithm. Theoretically, any value within the range determined by the maximal noises and minimal motion parameters could be selected as the threshold. At this stage, we roughly choose a threshold of $0.1m$ with which 100% accurate segmentation results have been achieved as shown in Fig. 3.6.

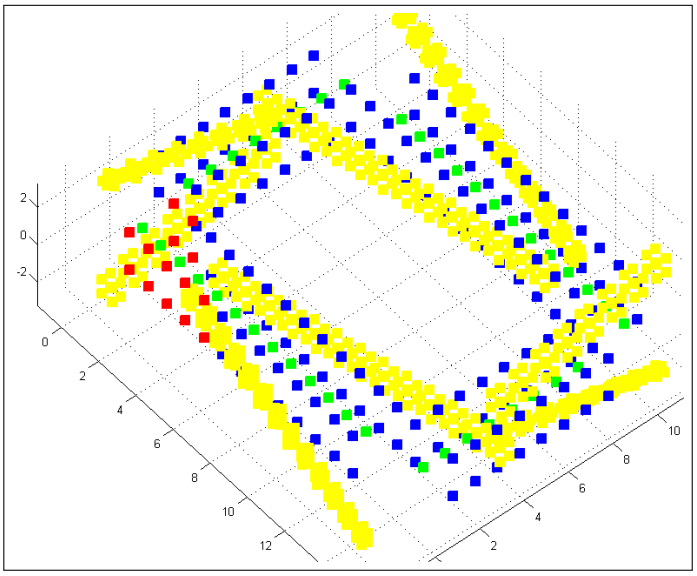


FIGURE 3.5: Simulated multibody RGB-D SLAM scenario: blue and red points represent static features, and red ones are currently seen by the camera; green ones stand for camera's positions; yellow ones correspond to the paths of the two moving objects

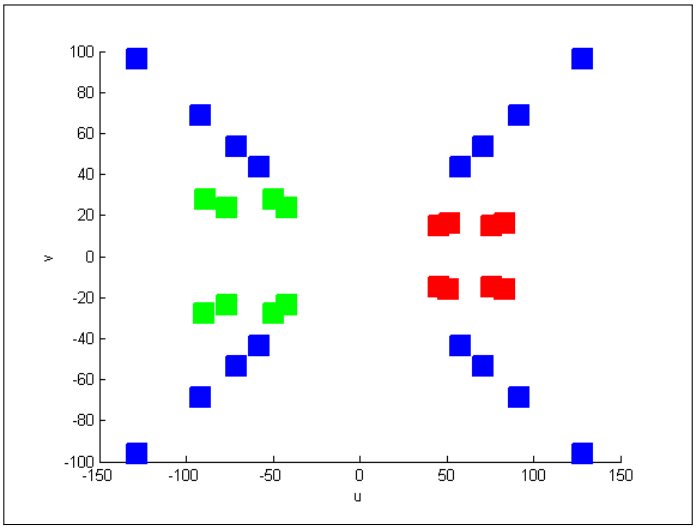
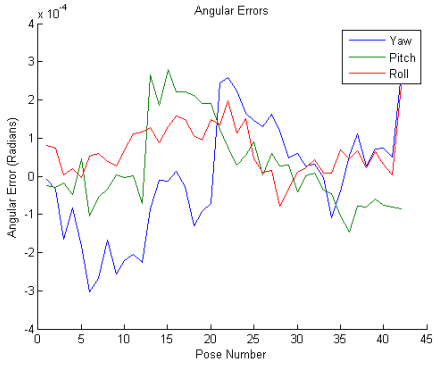


FIGURE 3.6: One segmentation result from the simulated data: blue, red and green points represent the three groups the algorithm has found, corresponding exactly to the ground-truth groups

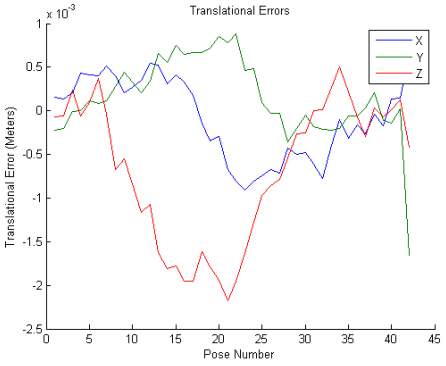
3.3.3 Multibody SLAM Results Using Simulated Noisy Data

Based on the segmentation results, according to [41], we can estimate pair-wise visual odometry and loop closures for each motion group, from which we can calculate the initial values of the features' positions and camera's poses. Then, we choose the static features and camera as a group to perform a RGB-D SLAM employing Gauss-Newton method for optimisation. Once we have attained the camera poses, we utilise them to calculate the positions of the dynamic features at each time step of the process.

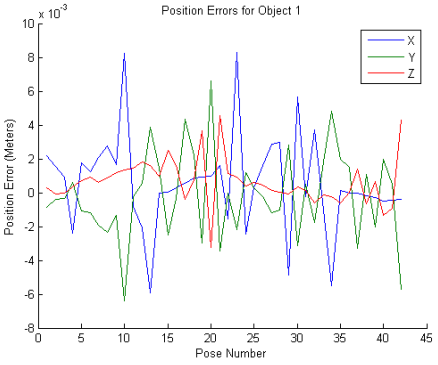
Fig. 3.7a, 3.7b, 3.7c and 3.7d show our SLAM results as compared with the ground truth values.



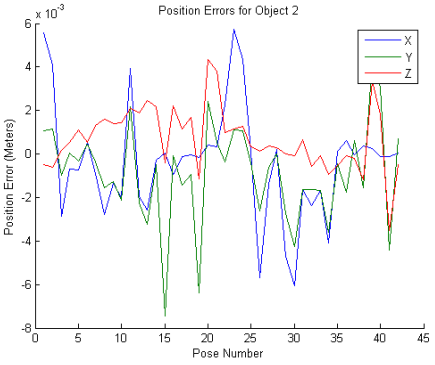
(A) Angular errors of camera poses using simulated data.



(B) Translational errors of camera poses using simulated data.



(C) Average position errors of the first moving object from the simulated data.



(D) Average position errors of the second moving object from the simulated data

FIGURE 3.7: Angular, translational and position errors of the multibody RGB-D SLAM results using the simulated data.

3.4 Evaluation Based on Our Own Dataset

3.4.1 Collecting Our Own Dataset

Given the difficulty in finding a suitable RGB-D dataset containing only rigid moving objects to test Algorithm 3, we have instead collected a small dataset with ground truth values for evaluating both our motion segmentation algorithm and visual odometry results.

As shown in Fig. 3.8, we choose a special space in our lab with the dimension of $4m \times 2m$. The background has also been decorated with some feature rich objects. In the scenario, there are two moving boxes and a Kinect sensor. From one step to the next, the two boxes, in black and blue respectively, and Kinect will translate or rotate independently, and any of the moving objects can remain static at sometimes, which means that the number of motion groups is changing throughout the process. At each discrete step, we measure and record the position and orientation of each object manually. At the same time, we keep recording the synchronised RGB-D data using Kinect and ROS.

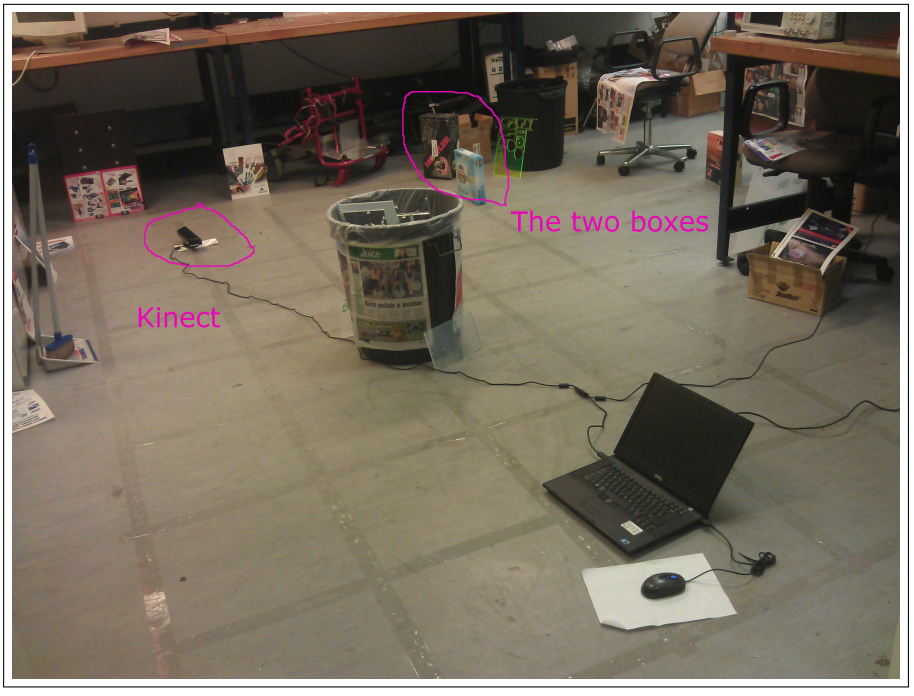


FIGURE 3.8: Experimental setup: one Kinect, two boxes (one in blue, one in black) are located at the further side of the image, and have been marked out

3.4.2 Motion Segmentation Results From Experimental Data With Outliers

After obtaining the data, we firstly perform feature detection and matching using SIFT [38], and filter out those that do not have depth values. Then we apply Algorithm 3 to the matched sets of points.

The sample feature matching and segmentation results are shown in Fig. 3.9 and 3.10. In Fig. 3.10, different colour represents different motion groups, consistent with the ground truth. As we can see that outliers have been separated into small independent groups which are usually composed of less than 3 feature points, and we can just discard such small groups. As a result, although the feature matching results unavoidably contain outliers, our motion segmentation algorithm can still handle them.

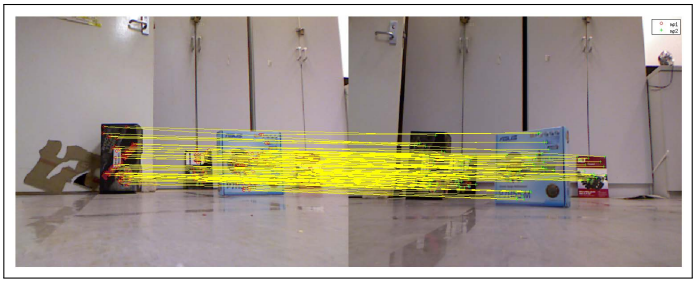


FIGURE 3.9: One sample feature matching result from the experimental data

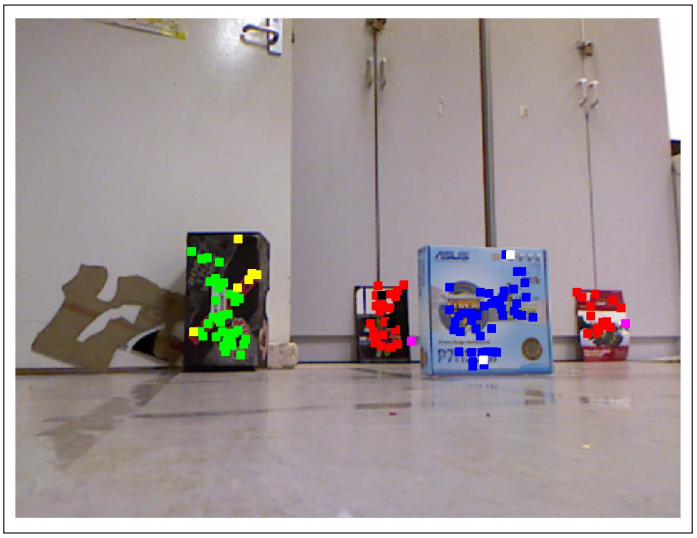
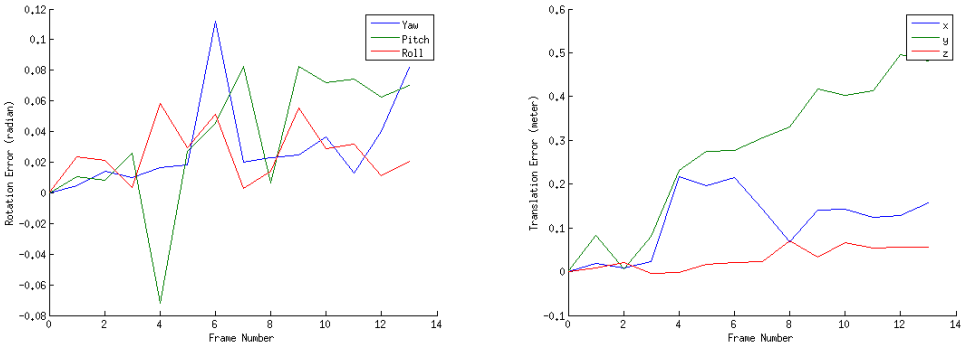


FIGURE 3.10: One sample segmentation result from the experimental data

3.4.3 Initial Values Based on Motion Segmentation and Visual Odometry

From the segmentation results, we choose the $3D$ feature points corresponding to the static group to calculate visual odometry, by accumulating which we can then derive the initial values of the camera poses for pose-graph optimisation. The angular (Yaw, Pitch and Roll) and translational (x, y and z) errors of camera poses as compared with the ground truth are shown in Fig. 3.11a and 3.11b respectively. In Fig. 3.11a, the x-axis represents frame number, and the y-axis represents the angular errors in radians; in Fig. 3.11b, the x-axis represents frame number, and the y-axis represents the translational errors in meters.

As we can see, although the segmentation results are good, the visual odometry and thus the initial values of camera poses turn out to be not as sastifactory as we have expected. This may be due to several reasons. Firstly, after the division, the number of feature points for each motion group is much smaller compared with ordinary SLAM scenarios, and therefore the least squares results will inevitably degrade to some degree. Secondly, as we know, the noises contained in the depth information of RGB-D cameras do not fit into Gaussian distributions; instead, they are distance dependent [66]. Last but not least, we have not fully preserved the accuracy of the depth information during the image format transformation and saving process.



(A) The angular errors. (B) The translational errors.

FIGURE 3.11: The angular and translational errors of camera poses' initial values as compared with the ground truth. Best viewed in color.

3.5 Summary

In this chapter, we have deduced the necessary and sufficient conditions for grouping $3D$ points into different motion sets, upon which a distance-based $3D$ motion segmentation algorithm using RGB-D data is proposed. Results from both simulated and real data

show that our algorithm is efficient and robust. Furthermore, we have applied it to solve a simulated multibody SLAM problem, and the results are very promising. Finally, we have also shown that the satisfactory segmentation results using real dataset enable us to attain visual odometry for solving practical SLAM problems.

Although simulation and experimental results prove that the proposed motion segmentation algorithm is applicable to dynamic scenarios composed of rigid objects with large motions, when it comes to images with continuous video of non-rigid moving objects, it would be difficult; or if motion blur or featurless regions occur, the number and quality of available feature points will drop dramatically, jeopardising the following motion analysis. In the next chapter, we will firstly explore the possibilities of choosing appropriate $2D/3D$ base points and thresholds for each motion group, based on which two other frameworks capable of dealing with scenarios including quick non-rigid moving objects and featureless zones are then proposed.

Chapter 4

Motion Model based Segmentation for Robust RGB-D SLAM: From Sparse to Semi-dense

Solely based on detecting the changes of the relative $3D$ position between feature points, the sparse feature-based motion segmentation method proposed in last chapter is especially suitable for segmenting large movements. As an extension in this direction, in this chapter, we firstly propose another sparse feature-based motion segmentation method according to motion models. Compared with the former method, this motion model based segmentation algorithm is based on feature flow analysis (suitable for both $2D$ and $3D$), therefore it is capable of capturing motion details in continuous videos. Furthermore, we demonstrate that this motion segmentation method along with the pose-graph SLAM framework constitutes a unified way for robust SLAM in both static and dynamic scenarios.

However, when it comes to dealing with some of the challenging TUM [61] benchmark sequences, we discover that motion blur and featureless regions are pervasive, in which case sparse features become insufficient in quantity and unreliable in quality. To overcome these difficulties, we then propose another semi-dense feature version of our motion segmentation method based on optical flow. Using challenging real data, our method could produce much better results than the state-of-the-art method.

The structure of this chapter is as follows. Firstly, we present related work in Section 4.1. Then we propose our motion segmentation algorithm in Section 4.2, in which no prior

motion model or object model is assumed. After that, we choose standard RANSAC as a reference, and make detailed comparisons between them using simulated data; based on that, we further test our algorithm using TUM benchmark dataset and compare our results with those of the state-of-the-art in RGB-D SLAM, i.e., dense visual SLAM [59] in Section 4.3. Based on analysis in Section 4.4 of the challenging *walking* sequences of the TUM benchmark, we proposed a semi-dense motion segmentation method in Section 4.5 and verify it using the *walking* sequences in Section 4.6. Finally, we summarise the chapter and talk about possible future work in this direction.

4.1 Related Work

In this section, we will mainly discuss RANSAC and give a brief review of related work that has made use of RANSAC to deal with outliers and moving objects in SLAM. Moreover, following the detailed survey of the available motion segmentation algorithms in multibody SLAM in Chapter 2, we will focus on summarising the related work from a different perspective.

4.1.1 RANSAC and Its Application in SLAM

As an important robust estimator, RANSAC has become a standard and indispensable tool to deal with outliers in sparse feature-based SLAM [43, 67] as well as many other areas. It was firstly proposed in [13], and the standard RANSAC is composed of iterations of three simple steps: generate random hypothetical minimal inliers, fit a model, and obtain the consensus set. It will keep iterating till the possibility of obtaining a good model with the corresponding consensus set is above a pre-set value or the maximum iteration times have been reached.

Theoretically, with appropriate threshold values, given enough time, RANSAC can always find the most accurate model, but when the number of iterations is limited, the obtained solution is not guaranteed to be optimal. As we all know, in practice, time is a precious resource, therefore the balance between accuracy and time consumption amounts to be a constant topic of RANSAC. Furthermore, for one of the common cases where we are only concerned about the predominant model in the data, only when outliers just constitutes a small part of the whole data, standard RANSAC turns out to be a simple and efficient solution; as the number and portion of moving parts and other outliers increase, RANSAC will become inefficient very quickly. To improve the efficiency, accuracy and robustness of the standard RANSAC, many extension versions have been proposed. The basic conclusion is that RANSAC can be improved in one regard but at

the expense of the others, and interested reader can refer to [68] for a detailed survey and performance evaluation of them.

On the other hand, the standard RANSAC can not be directly used to estimate multi-model in a scenario, although [17] has made use of it to detect one moving object which can be regarded as a simpler case of complex dynamic scenarios in SLAM. As a result, some researchers propose to conduct sequential RANSAC [69]: apply RANSAC to obtain a model and its inliers, remove the found inliers from the data and then for the remaining data apply RANSAC again. The iterations will keep going till no model can be recovered from the data. However, during the process, inaccurate inliers detection will affect the estimation of the following models. To tackle this problem, multiRANSAC [70] is proposed, which performs the estimation in a parallel manner. Nevertheless, it requires the user to specify the number of model instances, which is usually unknown in most cases.

Generally speaking, although many variants of RANSAC have been proposed to partially address the shortcomings of the standard RANSAC, the problem has not been fully solved yet. And as far as we know, up till now, none of them has been applied in multibody SLAM yet.

In this chapter, we are mainly concerned about robust estimation of camera/robot poses, or ego-motion, in various situations where the static features usually amount to be the predominant part of the data, therefore we choose to compare our algorithm with the standard RANSAC here.

4.1.2 Motion Segmentation Methods for SLAM in Dynamic Scenarios

In multibody SLAM, motion segmentation algorithms have been regarded as an important step to decompose the dynamic scenario into different motion groups. And almost all of the available motion segmentation algorithms conduct segmentation without discrimination, which means that while performing segmentation, they do not care about which group belongs to the static environment.

In terms of targeted applications, most of the available motion segmentation algorithms are for RGB data only, quite recently some work on RGB-D data is also emerging [71]. In Chapter 2, an efficient algorithm for motion segmentation in RGB-D SLAM based on the necessary and sufficient conditions of being relatively static is proposed. However, since the segmentation algorithm solely relies on the distance values whose accuracy is limited by the capability of the sensor, i.e., Kinect, while it is good at detecting large

movements, it may fail to detect small movements within the precision range of Kinect. Furthermore, the approach is restricted to scenarios composed of rigid moving objects.

As to the output of motion segmentation methods, the segmentation results can be dense or sparse. In the former case, every pixel in an image will be assigned a label associated with a motion group; while the latter only provide identifications for some sparse pixels in the image. To achieve dense results, some algorithms start from dense optical flow, which is usually obtained by employing available mature algorithms in computer vision. And therefore their accuracy depends on that of optical flow results. On the other hand, some others firstly conduct a long trajectory analysis to obtain a sparse segmentation result, then through a label diffusion process to obtain dense results [50]. Accordingly, the resulting accuracy mainly depends on that of its initial stage.

4.1.3 Different Solutions for Different Scenarios

As we know, in SLAM, there are three long standing problems that are pervasive and till now they are regarded as different problems and being treated in different ways, either from the front end or back end.

Firstly, in traditional filter, pose-feature-graph or pose-graph based SLAM, while performing data association during the initial process, there are unavoidable outliers coming from feature matching and sensor noises. How to get rid of them constitutes a classical problem. Nowadays, RANSAC [13] is widely utilised to find the motion model for the camera/robot that can encompass most available feature points as inliers. Once such a model is found, the points that are inconsistent with the model are regarded as outliers and discarded. Since the proportions of this kind of outliers are usually very low, RANSAC's performance proves to be quite satisfactory in most cases [72, 73], although it is an iterative and non-deterministic method.

Secondly, dynamic scenarios can bring in another kind of outliers. As we know, the traditional solution framework of SLAM assumes that everything is static except the camera/robot. And when it comes to dynamic scenarios, the effects of moving objects can be very serious. In this case, we do not care about what the moving objects are and how they are moving; instead, our sole purpose is just to filter out these extra outliers (moving objects) together with those mentioned above. RANSAC method can still work in some special cases, but as we will see, its performance will degrade rapidly as the fraction of outliers increases. Dense visual SLAM [59], which represents the latest achievement in RGB-D SLAM, makes use of the colour and depth information at every pixel while conducting SLAM and employ t-distribution to enhance its robustness; in [14], however, weighting function is utilised to handle moving objects. On the other

hand, instead of relying solely on the data association stage to get rid of all of the outliers, robust kernels such as Huber function [74], and recently some other robust approaches [10, 75, 76], have also been proposed and adopted at the back ends to cope with unwanted outlier edges brought from the front ends. The basic idea of them is to minimise the effects of outliers including dynamic elements while performing the least squares optimisation instead of explicitly identifying and separating them away beforehand.

Thirdly, as mentioned before, in multibody SLAM which concerns about depicting all of the parts, either static or moving ones, in the scenarios, we usually firstly resort to motion segmentation algorithms to decompose the dynamic scenarios into different motion groups and then apply the current available solution frameworks for SLAM static environments to handle each of them. As a matter of fact, in multibody SLAM, moving objects become one of our focuses instead of unwanted, useless pseudo noises; at the same time, we still need to deal with outliers caused by feature matching and sensor noises. Therefore, to some degree, multibody SLAM is a more general task, and those situations we have mentioned before constitute some of the special cases of it. It means that motion segmentation algorithms need to be able to cope with such kind of situations as well. In other word, motion segmentation algorithms can be a general solution to all these three kind of problems.

Although the sparse feature-based motion segmentation proposed here in this chapter is originally targeted for RGB-D SLAM in dynamic scenarios, we argue that a good motion segmentation algorithm can act as a unified solution to the three different problems mentioned above, and we will see that it is efficient and effective in handling those three kinds of situations in the same framework. We have tested it using simulated data and several benchmark RGB-D sequences, some of which are difficult ones, and our results are quite promising. As a result, we are confident that as a supplement for the traditional solution frameworks of SLAM in static environments, this algorithm will enable us to conduct robust SLAM in both static and dynamic scenarios.

4.2 Our Sparse Feature Motion Model based Motion Segmentation Algorithm

The inputs to our motion segmentation algorithm are two frames of RGB-D data, from which we can obtain sparse SIFT feature pairs through feature detection and matching, and the outputs are separated motion groups of feature pairs.

As we know, for each dynamic scenario composed of some moving objects, its corresponding flow field is made up of several coherent regions, therefore each pair of inlier points is not singular. Instead, each of them can find some close neighbours that have similar flow vectors. On the other hand, for those who can not find any similar neighbour, it is highly possible that they are outliers. It is based on this observation that we firstly select two similar neighbouring pair of points and then together with the given pair to calculate the rotation matrix $\mathbf{R}$ and translation $\mathbf{T}$ across the two frames during the segmentation process.

More specifically, when judging whether two vectors are similar or not, we are using the following formula as in [77]:

$$s = \frac{|\mathbf{V}_i - \mathbf{V}_0|}{0.5(|\mathbf{V}_i| + |\mathbf{V}_0|) + \varepsilon} \tag{4.1}$$

In formula 4.1, s represents the similarity score, $\mathbf{V}_0 = (\Delta x_0, \Delta y_0, \Delta z_0)^T$ denotes the $3D$ flow vector of the current chosen point, $\mathbf{V}_i$ corresponds to the $3D$ flow vector of one of the neighbours and ε is a constant whose value is set according to the noise level of the data. After some primitive verification using some sampled points, we discover that if ε is set to 1 and $s < 0.05$, the corresponding neighbour could be regarded as similar to the current point. And these two values are fixed in this chapter except for the last sequence (Section 4.6.4) where adaptive thresholds are adopted.

After obtaining the initial $\mathbf{R}$ and $\mathbf{T}$ with similar neighbours according to [41], we will iterate between updating $\mathbf{R}/\mathbf{T}$ and inliers till no more inliers can be included. This process is similar to the second part of the locally optimal RANSAC [78].

4.2.1 Sparse Feature Motion Model Based Motion Segmentation Algorithm

The detailed steps of our algorithm are described in Algorithm 4.

4.2.2 Solution for Robust SLAM and Multibody SLAM

The chosen group and its corresponding $\mathbf{R}$ and $\mathbf{T}$ will be put into a pose-graph optimisation process to estimate the camera poses. For pose-graph based robust SLAM, the result is thus obtained. And for multibody SLAM, it is also a good start. For example, once ego-motion is known, it can help us identify various moving objects (including both

rigid and non-rigid objects) and their motions types, even when there are not sufficient features on some moving objects in some frames.

Algorithm 4: Sparse feature motion model based motion segmentation algorithm

Inputs: two frames of RGB-D data $\mathcal{D}_1$ and $\mathcal{D}_2$;

Output: the static group $\mathcal{S}$ and its corresponding $\mathbf{R}/\mathbf{T}$;

Assumption: the largest group corresponds to the static group;

1. ***Obtain the set of matched feature pairs***: for $\mathcal{D}_1$ and $\mathcal{D}_2$, perform feature detection and matching , filtering out those feature points with no depth information, and then we can obtain a set of matched $3D$ feature pairs $\mathcal{P}$;

2. ***obtain corresponding*** $3D$ ***feature flow***: convert obtained features into (x, y, z) form, and then obtain their corresponding $3D$ flow $(\Delta x, \Delta y, \Delta z)$;

3. ***Loop***: iterate through *3.1-3.3* till $\mathcal{P}$ is empty or left with points pairs that could not find similar pairs;

3.1 ***Choose similar points as a seed***: for one pair of matched features, choose its neighbours that have similar $3D$ flow as a cluster to obtain an initial guess of $\mathbf{R}$ and $\mathbf{T}$, based on which corresponding inliers are obtained. If for one pair, we cannot find a minimum number (3 is used in this chapter) of similar neighbours, it is highly possible that it is an outlier and we will skip it;

3.2 ***Iterative update***: keep iterating between updating $\mathbf{R}/\mathbf{T}$ and inliers until no more inliers can be included;

3.3 ***Remove found group***: remove the found inlier group from $\mathcal{P}$ and remember its corresponding $\mathbf{R}/\mathbf{T}$.

4. ***Group Selection***: after all of the features have been classified, choose the largest group as the static group $\mathcal{S}$.

4.3 Simulation and Experimental Results

Firstly, to show the efficiency of Algorithm 4, we compare it with the standard RANSAC (without causing confusion, we will just call it RANSAC thereafter) using simulated data. As we will see, as the fraction of the targeted group decreases, the performance of RANSAC degrades quickly while ours remains almost the same. Please note that for RANSAC, the computation time include just one round, which can only tell us the inliers corresponding to the biggest group, while Algorithm 4 will segment all the points into different groups at one go.

Then, we test Algorithm 4 using several benchmark RGB-D sequences, which include both static and dynamic scenarios. The segmentation results are verified by the accurate visual odometry achieved, based on which pose-graph SLAM results can be obtained.

4.3.1 Results Using Simulated Dynamic Scenarios

Simulated Dynamic Situations

We randomly generate one thousand of $3D$ points (x, y, z), separate them into different groups, and then translate and rotate them using different sets of motion parameters (**R**s and **T**s), all of which constitute the ground-truth values.

From the $3D$ points, we can obtain their images at their different positions through a simulated RGB-D camera (its intrinsic matrix is the same as the Kinect used in the real dataset), then we add Gaussian noises to the u, v and d values respectively (whose standard deviations are within 0.5 pixel for the u and v values, 1% for the d values).

Simulation Results

Given thus obtained two sets of u, v and d values, we apply Algorithm 4 (represented as MS3D in this section) and RANSAC to perform segmentation respectively. According to the noise level of current data, the roughly chosen threshold for both MS3D and RANSAC here is the same value $0.08m$.

As to the iteration times k of RANSAC, assuming that the probability of choosing an inlier component is w, if we want to ensure that we can obtain a subset of n-component coming from the inlier set with the probability of p, the traditional way to calculate k as follows [13]:

$$k = \frac{\log(1-p)}{\log(1-w^n)} \tag{4.2}$$

While directly applying this formula to our data, we find that in most cases, RANSAC cannot find a solution at all. According to our understanding, this formula just tell us that within k times, we can find one subset of inliers; on the other hand, as we know, for noisy data, we cannot expect that one randomly chosen inlier subset could give us a good estimation. Therefore, in our comparisons, we have used 30 times of the calculated k as the limit. Even at this larger limit, there still exist some occasions that RANSAC could not find a coherent inlier group to estimate **R** and **T**.

If we change the portion of the predominant group, the run time for RANSAC will increase dramatically while that of MS3D remains almost the same, as shown in Table 4.1 and Table 4.2. More specifically, in Table 4.1 where there are two motion groups, when the percentage of the targeted group decreases from 100% to 51%, to obtain

the same accuracy, the consumed time for RANSAC increases from 0.01 second to 2.7 seconds (discarding those situations in which it fails to find the results) while the running time for MS3D remains to be around 0.03 second. Similar trend can be seen in Table 4.2 which represents the five-motion-group scenario.

TABLE 4.1: Comparison of Average Computation Time: Scenarios Composed of Two Motion Groups

Percentage for The Targeted Group (%)	**Average Run Time (Seconds)**	
	RANSAC	**MS3D**
100	0.01	0.02
90	0.3	0.03
80	0.5	0.03
70	0.9	0.03
60	1.6	0.03
51	2.7	0.03

TABLE 4.2: Comparison of Average Computation Time: Scenarios Composed of Five Motion Groups

Percentage for The Targeted Group (%)	**Average Run Time (Seconds)**	
	RANSAC	**MS3D**
80	0.5	0.3
70	0.9	0.2
60	1.6	0.2
50	2.9	0.2
40	5.8	0.2
30	14.2	0.2

4.3.2 Experimental Results Using Static and Dynamic Scenarios

We then test Algorithm 4 using several typical benchmark RGB-D sequences provided by TUM [61], among which are static (fr3_long_office), dynamic (fr3_sitting_xyz, fr2_desk_with_person) scenarios, and even a very challenging sequence that no visual odometry results have been reported before (fr3_walking_static).

For each sequence, we are not using all of the data; instead, we firstly select some keyframes according to two criteria: either every two consecutive frames at least have a minimum number of common features (we choose 100 here), or have a minimum distance (we specify that the sum of the norm of the relative translation (in meters) and absolute rotation angles (in radians) should be bigger than 0.03) from each other. Apart from these conditions, there is no further manual pruning involved. Then the relative translation and rotation, i.e., visual odometry, can be obtained after sparse feature (we are using SIFT) detection, matching and segmentation.

Based on the visual odometry results from every two consecutive frames, we can obtain the initial values for the camera poses; furthermore, we also check every frame with its closest neighbours within a certain distance (we use $0.2m$ as the limit). Those frames that have at least 100 common features with the current frame amount to be satisfactory loop closures.

With all of these **R** and **T** values, we then employ G2O [33] to perform a pose-graph based SLAM. The respective visual odometry and SLAM results for each sequence are shown below (the last one only has visual odometry results because its loop closures are impaired due to shortage of common features).

Our visual odometry comes right after motion segmentation, and its accuracy can directly reflect the quality of motion segmentation results. According to [61] and [14, 59], the RMSE of relative pose error (RPE) in meters per second is a good evaluation standard for visual odometry; while the RMSE of absolute trajectory error (ATE) is especially suitable for evaluating the SLAM results. In this thesis, we follow these rules and make use of the final results of dense visual SLAM [59] (called DVSLAM thereafter) as a reference.

fr3_long_office

In this sequence, the camera is moving around a large table, therefore the outliers mainly come from feature detection, matching and/or sensor errors.

a) Motion Segmentation Result

As shown in Fig. 4.1, Algorithm 4 can detect outliers effectively.

b) Visual Odometry Result

The RMSE of RPE for our visual odometry is $0.024m/s$; while the RMSE of RPE for DVSLAM is $0.054m/s$.

c) After Pose-graph SLAM

Our RMSE of ATE is $0.050m$, and the obtained camera poses as compared with the ground truth values are shown in Fig. 4.2; while the RMSE of ATE for DVSLAM is $0.084m$.

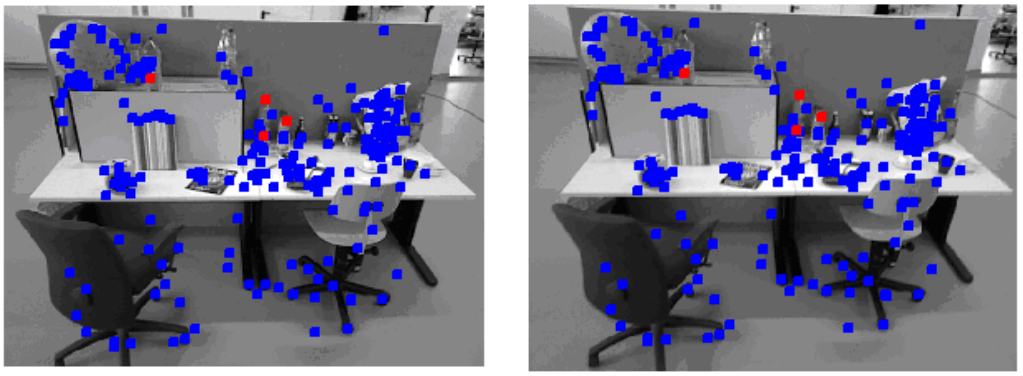


FIGURE 4.1: A sample motion segmentation result of two frames from the fr3_long_office sequence (best viewed in colour): blue points respresent detected inliers, while red points represent outliers caused by inaccurate depth data or feature matching.

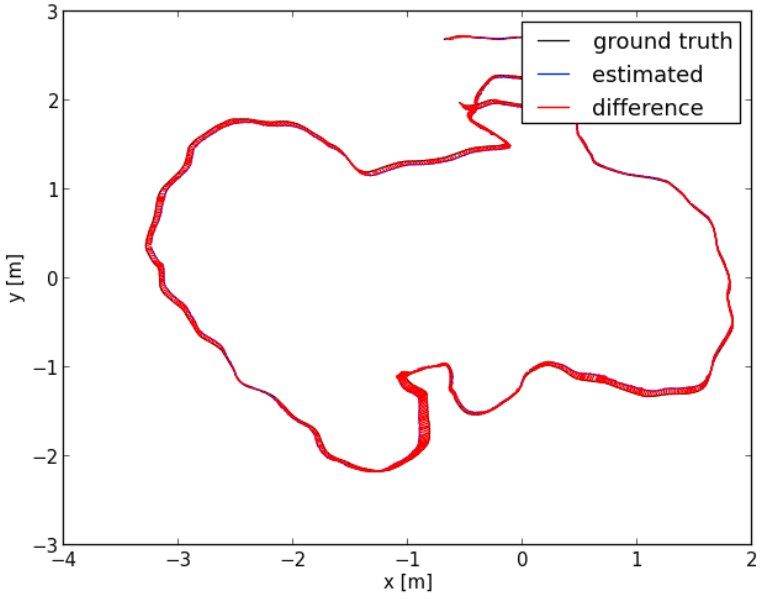


FIGURE 4.2: Our SLAM results for the fr3_long_office sequence compared with the ground truth (best viewed in colour): gray lines represent ground truth, blue lines represent the SLAM results, and red lines represent the differences between them at various corresponding timestamps.

fr3_sitting_xyz

In this sequence, two people are sitting in front of a table before the camera, chatting with small body movements.

a) Motion Segmentation Result

As shown in Fig. 4.3, Algorithm 4 can detect moving points and other outliers effectively.

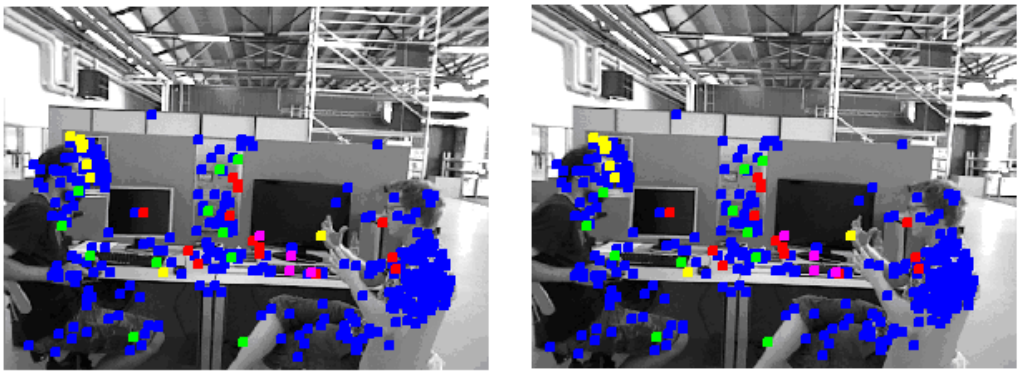


FIGURE 4.3: A sample motion segmentation result of two frames from the fr3_sitting_xyz sequence (best viewed in colour): blue points respresent detected inliers, while red points represent outliers caused by inaccurate depth data or feature matching.

b) Visual Odometry Result

The RMSE of RPE for our visual odometry is $0.025m/s$; while the RMSE of RPE for DVSLAM is $0.049m/s$.

c) After Pose-graph SLAM

Our RMSE of ATE is $0.018m$, and our camera poses as compared with the ground truth values are shown in Fig. 4.4; while the RMSE of ATE for DVSLAM is $0.060m$.

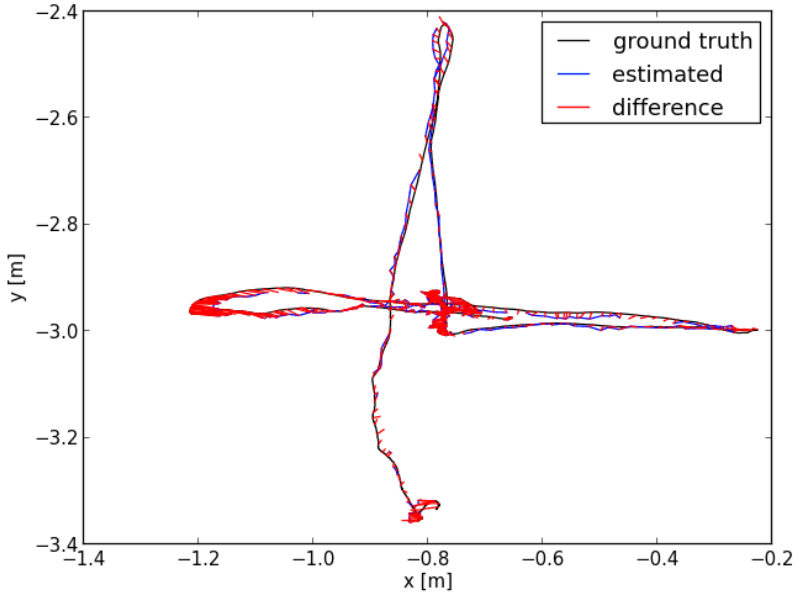


FIGURE 4.4: Our SLAM results for the fr3_sitting_xyz sequence compared with the ground truth (best viewed in colour): gray lines represent ground truth, blue lines represent the SLAM results, and red lines represent the differences between them at various corresponding timestamps.

fr2_desk_with_person

In this sequence, one people comes to sit before the desk, simulating working there with some big body movements and making changes to parts of the environment.

a) Motion Segmentation Result

As shown in Fig. 4.5, Algorithm 4 can separate the moving parts of the human body and other outliers from the static group effectively.

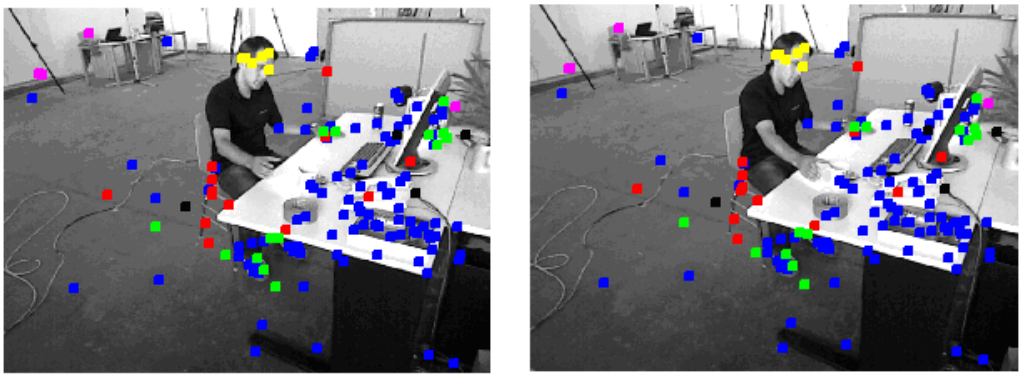


FIGURE 4.5: A sample motion segmentation result of two frames from the desk_with_person sequence (best viewed in colour): blue points respresent detected inliers, while points in other colours represent outliers caused by inaccurate depth data, feature matching, or moving objects.

b) Visual Odometry Result

The RMSE of RPE for our visual odometry is $0.018m/s$; while the RMSE of RPE for DVSLAM is $0.018m/s$.

c) After Pose-graph SLAM

Our RMSE of ATE is $0.055m$, and the camera poses as compared with the ground truth values are shown in Fig. 4.6; while the RMSE of ATE for DVSLAM is $0.073m$.

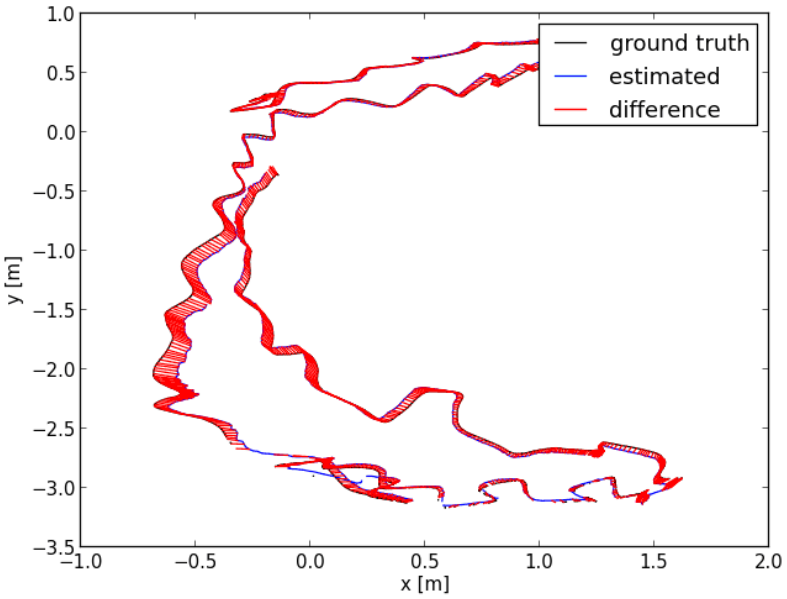


FIGURE 4.6: Our SLAM results for the fr2_desk_with_person sequence compared with the ground truth (best viewed in colour): grey lines represent ground truth, blue lines represent the SLAM results, and red lines represent the differences between them at various corresponding timestamps.

fr3_walking_static

As shown in Fig. 4.7, in this sequence, two people are moving in front of the camera, there are blur caused by quick camera movements and in some frames useful features are limited. Till now, no one else has publish reasonable results on this sequence yet.

Since there are wide ranges of dynamic changes in the scenarios, a fixed threshold value is no longer valid for segmenting all of the frames into different motion groups. Therefore, we have modified Algorithm 4 so that it can change the threshold value on the fly to adapt to different situations. However, similar to [79], it is based on a basic assumption that there are at least 20 common features in each pair of frames. Nevertheless, we can only obtain relatively better visual odometry results at this stage.

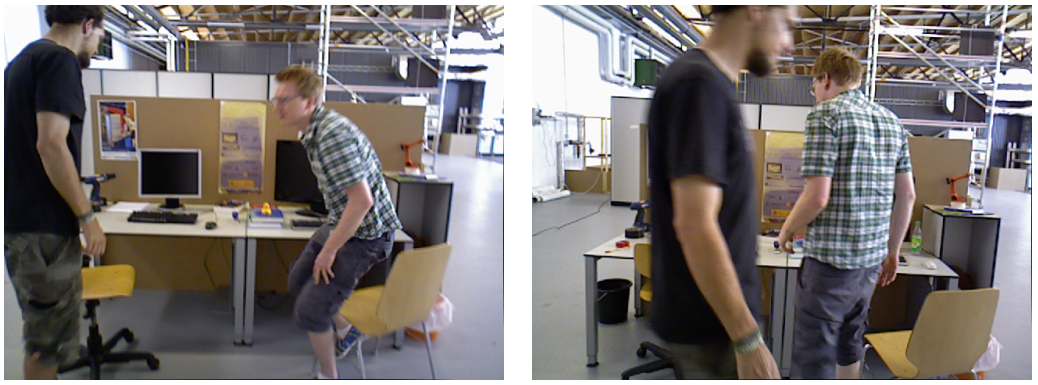


FIGURE 4.7: Two snapshots of the fourth sequence: fr3_walking_static.

For our visual odometry, the RMSE of ATE is $0.161m$, and the RMSE of RPE is $0.084m/s$; while for DVSLAM, the RMSE of ATE is $0.470m$, and RMSE of RPE is $0.309m/s$.

And Table 4.3 is a summary of the comparison of the results of our methods (represented as SMS) with those of DVSLAM.

TABLE 4.3: Comparison of RMSE of RPE (m/s) and ATE (m) for SMS and DVSLAM

Sequence No.	**RPE**		**ATE**	
	DVSLAM	**SMS-VO**	**DVSLAM**	**SMS**
1	0.054	0.024	0.084	0.050
2	0.049	0.025	0.060	0.018
3	0.018	0.018	0.073	0.055
4	0.309	0.084	0.470	0.161

4.4 Motivations to Semi-dense Feature Motion Model Based Motion Segmentation

As we know, based on TUM benchmark, many algorithms for RGB-D SLAM have been proposed ever since 2012 [80], among which are RGB-D SLAM system [67], muti-resolution mapping [81], and dense visual SLAM [59]. And, as we can see, in terms of accuracy, all of them can produce satisfactory results for most of the sequences of static scenarios. However, when it comes to robustness, none of the available algorithms has mentioned their performance on those sequences of dynamic scenarios. As far as we know, [14], which concentrates on robust visual odometry estimation, is the first one that has reported its estimation results on some moderately dynamic scenarios (the TUM *sitting* sequences); as to those highly dynamic scenarios (the TUM *walking* sequences), it says that no reasonable result can be obtained yet. A closer look will show that those dynamic scenarios really pose great challenges for us to overcome.

It is under this general situation that we have determined to further improve the robustness upon the current SLAM framework so that we can perform SLAM in more challenging scenarios in a less constrained manner, and this constitutes the motivation of the following work of this chapter. More specifically, in the following parts of this chapter, we firstly analyse the characteristics of the challenging sequences that we are to address. After summarising the pros and cons of the most popular methods that have been employed to achieve robustness, we propose a semi-dense version of motion model based motion segmentation algorithm that can act as the front end. Then, together with pose-graph SLAM, we show the improved performance of this new framework in four of the most challenging sequences. Finally, we give an evaluation of our framework and talk about possible future work.

4.4.1 The Challenging Sequences and Its Impacts on Available Methods

The TUM RGB-D *walking* sequences are composed of four challenging shots, whose samples have been shown in Fig. 4.8, 4.9, 4.10 and 4.11. As we can see, the camera is moving quickly, so blur is pervasive in these sequences; the moving people sometimes constitute a large portion of the image; the illumination is changing; and textureless areas emerge from time to time. All of these turn out to be great challenges to the related state-of-the-art methods.

FIGURE 4.8: Samples of *fr3_walking_static*

FIGURE 4.9: Samples of *fr3_walking_xyz*

FIGURE 4.10: Samples of *fr3_walking_rpy*

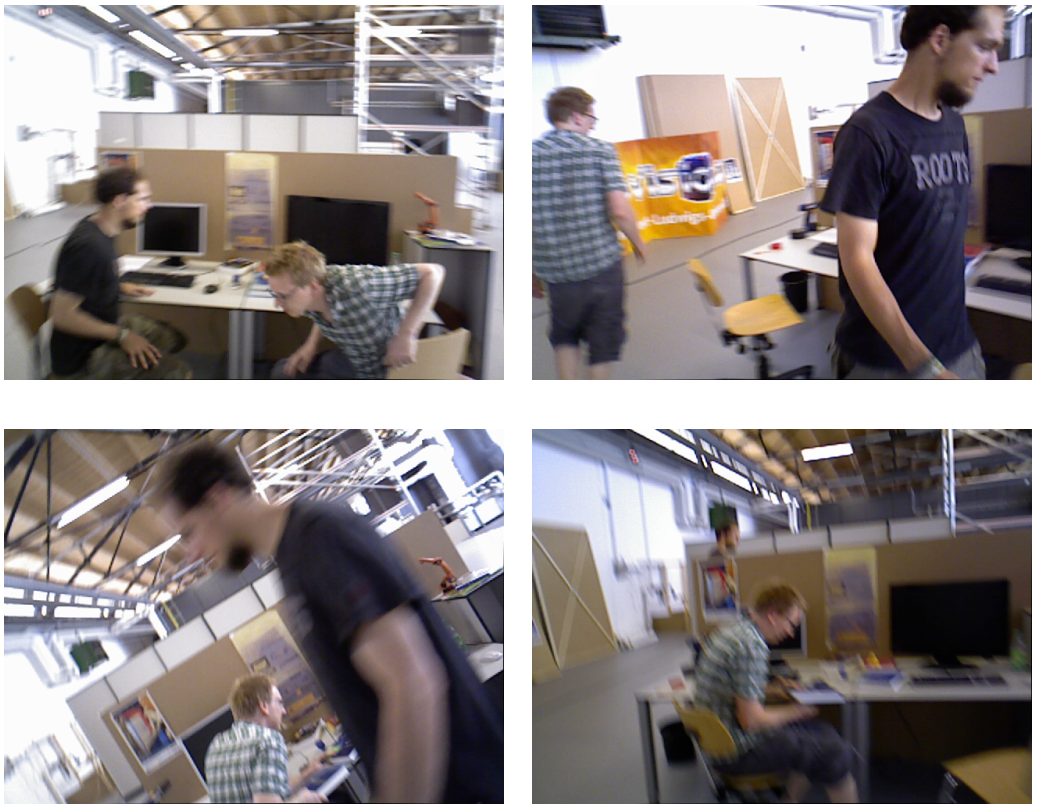


FIGURE 4.11: Samples of *fr3_walking_halfsphere*

4.4.2 Sparse Feature-based SLAM Cannot Deal with Image Blur and Textureless Areas

Sparse feature-based SLAM usually relies on SIFT/SURF or some other kinds of sparse features to calculate visual odometry, but when the illumination changes too much, textureless areas appears, or image blur occurs, the number and quality of such kinds of detectable features will decrease dramatically, all of which will directly affects the accuracy of the visual odometry results.

4.4.3 Dense SLAM is Vulnerable to Moving Objects

Dense SLAM can deal with some textureless scenarios, motion blur and illumination changes gracefully, but moving objects will have a great impact on its performance. While performing dense registration, a large portion of moving objects will surely impair the results.

4.4.4 RANSAC and Robust Kernels Cannot Endure Too Many Outliers

The widely adopted RANSAC and robust kernels have acceptable performance when outliers just constitute a minor portion of the features, but they cannot deal with situations where outliers constitute a significant part.

In summary, these challenging sequences contain mainly two kinds of challenges, i.e., tracking against blur/illumination changes/textureless areas and tolerating more than 50% of outliers. The state-of-the-art sparse feature-based and dense SLAM, RANSAC and robust kernels cannot accommodate all of these challenges at the same time. Therefore, we need to find new way to deal with them.

4.5 Our Semi-dense Feature Motion Model based Motion Segmentation Method

As we know, to cope with the challenges, firstly we need to ensure that we always have enough features for tracking. Dense tracking is ideal but is not practical at present due to its heavy computational cost. Therefore, we choose to conduct a semi-dense tracking instead. On the other hand, we need to find a robust method that can tolerate more than 50% of outliers, and our choice is motion segmentation. We propose to employ an efficient motion segmentation algorithm which is adaptive to deal with outliers in various scenarios. After these two steps, we can obtain visual odometry and loop closure constraints which constitute the inputs to standard pose-graph SLAM to estimate the camera/robot poses. Therefore, our method mainly concentrates on the front end.

4.5.1 SIFT Flow For Dense-Feature Detection And Matching

Since in the current context, the traditional SIFT/SURF feature detectors are no longer helpful, we firstly turn our attention to optical flow algorithms which can provide us with dense correspondences across two consecutive frames in the video. Optical flow as a classical problem in computer vision has attracted much attention, and numerous algorithms [64] have been proposed. Nevertheless, most of the available algorithms assume constant illumination and the displacement between the two frames cannot be too large, which has been verified by our preliminary test results. Finally, we find SIFT flow [82], which combines SIFT detector and the framework of optical flow together to produce a dense result and is robust against illumination and scale changes.

However, we are not using all of the output of SIFT flow, instead, we down-sample it with the factor of 16. It is a good balancing point that can retain the motion details while being lightweight for post-processing.

4.5.2 Semi-dense Feature-based Motion Segmentation

In this version of motion segmentation algorithm, in addition to the changes of features from sparse to semi-dense, we are also trying to make the thresholds adaptive to various dynamic scenarios. And the details of our algorithm are shown in Algorithm 5.

Through this process, we can find out the static group successfully in most cases. However, as we know, in the challenging sequences, there exist some shots that the feature points on the moving people turn out to be the biggest group. In such cases, we need to find some other way to identify the static group. It has not been full solved yet, and some large errors in our results using real data can be attributed to this problem.

Another practical problem is that, for different scenarios, the noise levels of the SIFT flow are different. Therefore, we need to adjust the related threshold values accordingly. However, automatically adapting thresholds to different scenarios is still an open problem, although some progress has been made [79] [83]. In this chapter, we simply stipulate that the biggest group should have at least 100 point members; otherwise, we will increase the thresholds step by step ($0.001m$ each time) till we can find enough points or maximum increments (5 in this chapter to limit the maximal thresholds) have been reached.

Fig. 4.12 is a sample output of SIFT flow and motion segmentation result using this algorithm.

4.5.3 Pose-graph SLAM

Based on the visual odometry, we choose some keyframes and loop closure relative pose constraints to perform a pose-graph SLAM by using standard optimisation tools such as G2O [33]. As to the standard of choosing keyframes, we stipulate that the relative pose between two keyframes cannot be too large (the sum of the norm of translation and the

norm of Euler angles is no more than 0.03), and the number of their common features should be no less than 150.

Algorithm 5: Semi-dense feature motion model based motion segmentation method

Inputs: two frames of RGB-D data $\mathcal{D}_1$ and $\mathcal{D}_2$;

Output: the static group $\mathcal{S}$ and its corresponding $\mathbf{R}/\mathbf{T}$;

Assumption: the largest group corresponds to the static group;

1. ***Obtain semi-dense set of feature pairs***: calculate SIFT flow with the two images, down-sample it to obtain the semi-dense result.

2. ***Obtain*** $3D$ ***flow***: with the semi-dense set of feature pairs $\mathcal{P}$ and related depth images, we calculate the corresponding $3D$ flow ($\mathbf{V(p)} = [\Delta x, \Delta y, \Delta z]$ for each pixel $\mathbf{p}$ in the first image) across the two frames;

3. ***Loop***: iterate through *3.1-3.3* till $\mathcal{P}$ is empty or left with points pairs that could not find similar pairs;

3.1. ***Choose a seed***: for a pair in $\mathcal{P}$, choose its two closest neighbours that have similar $3D$ flow together with itself to obtain a tentative visual odometry. More specifically, We firstly select one point and its $3D$ flow vector ($\mathbf{V}_0$) as the reference, compare the others' flow vectors ($\mathbf{V}_i$) with this one, and then choose the two closest similar neighbours (according to (4.1)) along with itself to calculate $\mathbf{T}$ and $\mathbf{R}$. If one point cannot find similar neighbours, it is highly possible that it is an outlier, so we just skip it;

3.2. ***Find inliers***: use the tentative $\mathbf{T}$ and $\mathbf{R}$ to obtain the corresponding inliers. For each point, we can predict its position in the second image. Then we compare these predictions with their real positions in the second image. All of those whose differences are within a certain limit (its initial value is set to be $0.003m$) constitute the inliers to this group of $\mathbf{T}$ and $\mathbf{R}$;

3.3. ***Local optimisation***: employ locally optimal RANSAC [78] to further expand the inlier set. Randomly choose 5 points from the initial inlier set, calculate $\mathbf{T}$ and $\mathbf{R}$ and obtain corresponding new inlier set as Step 3.2 has done. If the new inlier set is bigger than the initial one, it will replace the initial one. Repeat this process for a number of times (we have adopted 100 in this chapter), then we can obtain an updated inlier set that constitutes a new group. Remove members of this group from $\mathcal{P}$;

4. ***Find the static group***: finally choose the biggest group and take its corresponding $\mathbf{T}$ and $\mathbf{R}$ as the visual odometry.

4.6 Experimental Evaluation Using Challenging Sequences

We have applied our semi-dense feature based method to the four challenging TUM *walking* sequences. For comparison, we also employ the state-of-the-art, dense SLAM

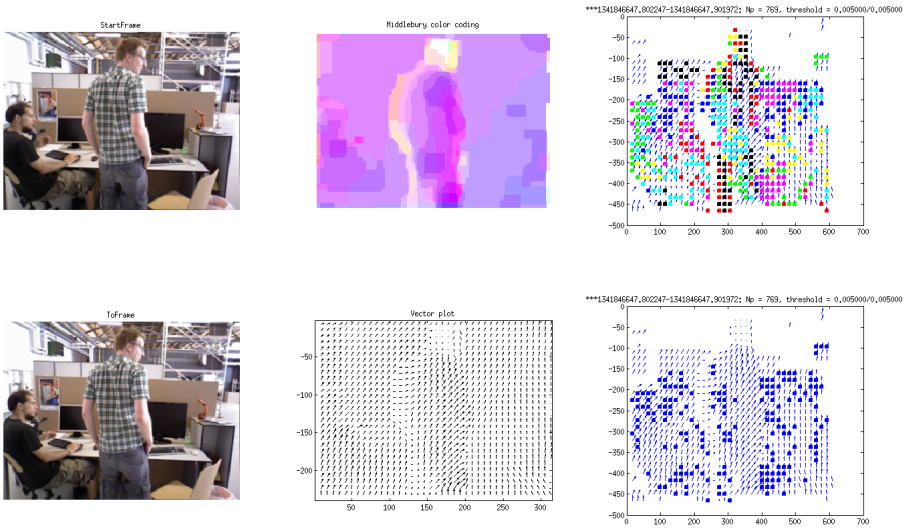


FIGURE 4.12: An sample SIFT flow and motion segmentation result (Best viewed in colour). The first column are the original images, and the second column is the SIFT flow output. In the third column, different colours in the upper image represent different motion groups we've found, and the blue points in the lower image represent the static group.

[59], whose code has also included the work of robust visual odometry [14], to the same sequences. Again, we adapt the RMSE of ATE and RPE as the metric for the SLAM results. Therefore, we have shown both of these two metric results for every sequence.

4.6.1 The fr3_walking_static Sequence

In this sequence, the camera's movements are relatively small, but the movements of two people are large, sometimes constituting a large portion of the images.

For dense SLAM, the RMSE of RPE is $0.309m/s$, and RMSE of ATE is $0.470m$, whose details are shown in Fig. 4.13.

Our visual odometry results are: RPE is $0.021m/s$, and ATE is $0.034m$; after pose-graph SLAM, RPE is $0.020m/s$, and ATE is $0.037m$, whose details are shown in Fig. 4.14 and 4.15 respectively.

As we can see, our method can obtain much better results. Even our visual odometry results are good enough that are comparable to what dense SLAM has achieved on some other static sequences. They show that our motion segmentation algorithm is good at separating dynamic parts out. But the improvement of our SLAM results is not so obvious, which indicates that loop closures have not brought in enough complementary constraints.

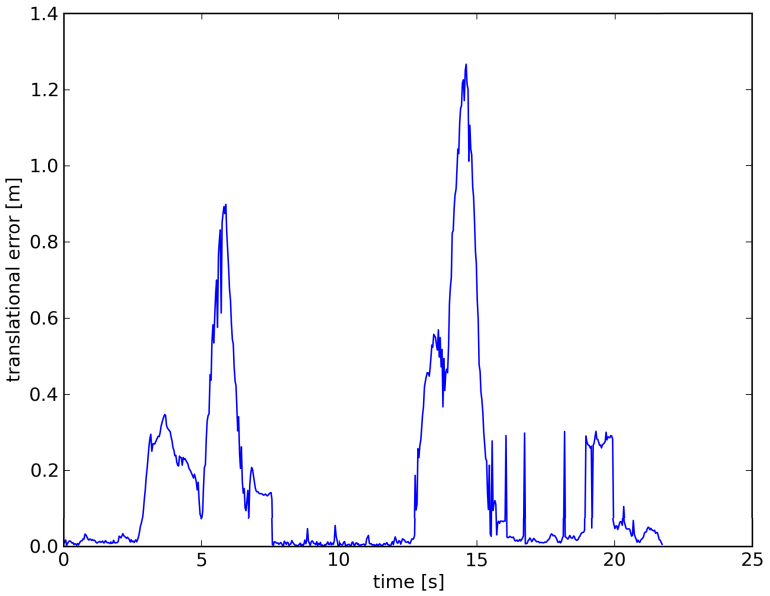


FIGURE 4.13: The RMSE of RPE for the Results of dense SLAM with the TUM *walking_static* sequence

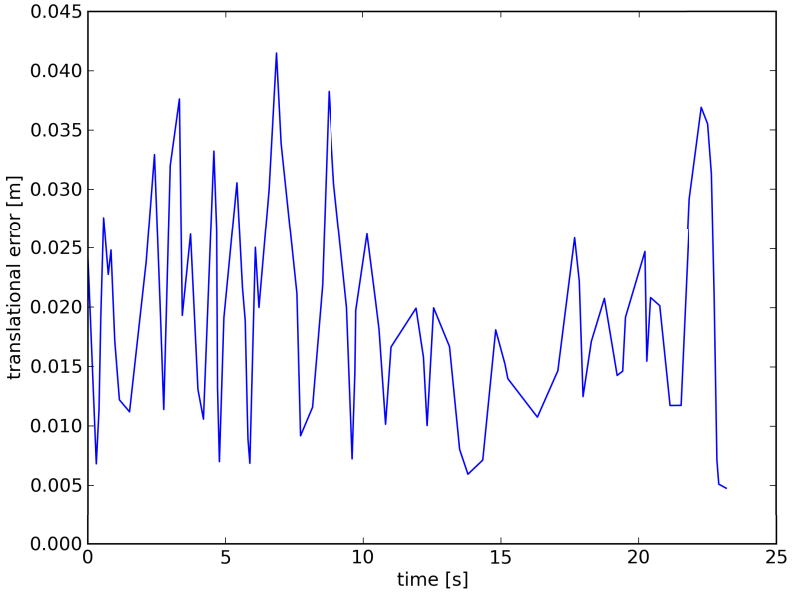


FIGURE 4.14: The RMSE of RPE for the Results of Visual odometry of our method with the TUM *walking_static* sequence

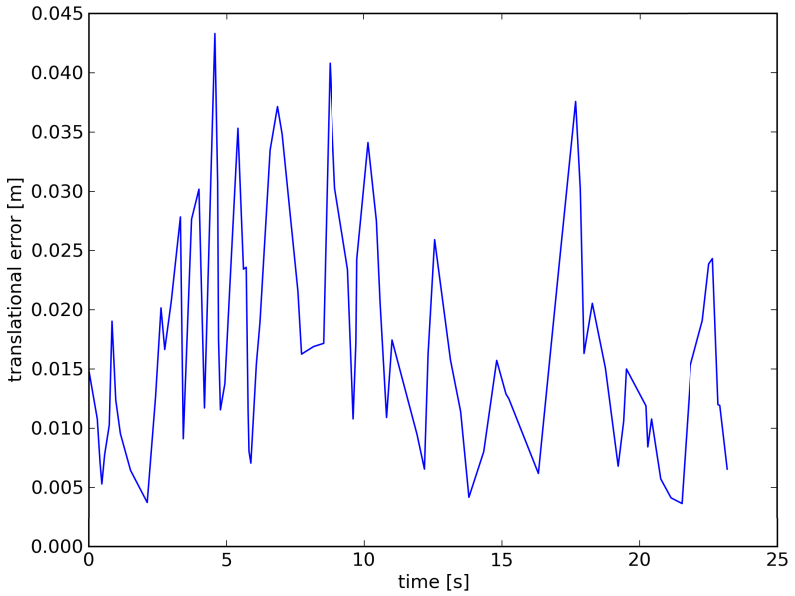


FIGURE 4.15: The RMSE of RPE for the Pose-graph SLAM Results of our method with the TUM *walking_static* sequence

4.6.2 The fr3_walking_xyz Sequence

In this sequence, the camera's movements become larger, and the movements of two people are large, sometimes covering most of the textured parts and constituting a very large portion of the images.

For dense SLAM, its RPE is $0.450m/s$, and ATE is $1.302m$, whose details are shown in Fig. 4.16.

Our visual odometry results are: RPE is $0.125m/s$, and ATE is $0.184m$; after pose-graph SLAM, RPE is $0.119m/s$, and ATE is $0.162m$, as shown in Fig. 4.17 and 4.18.

As we can see, dense SLAM cannot get reasonable results; on the other hand, our results are still much better, although not as good as with the first sequence.

4.6.3 The fr3_walking_rpy Sequence

For dense SLAM, its RPE is $0.454m/s$, and ATE is $0.789m$, whose details are shown in Fig. 4.19.

Our visual odometry results are: RPE is $0.321m/s$, and ATE is $0.496m$; after pose-graph SLAM, RPE is $0.286m/s$, and ATE is $0.433m$.

As shown in Fig. 4.20 and 4.21, our results have significant degradation, although they are still better than those of dense SLAM.

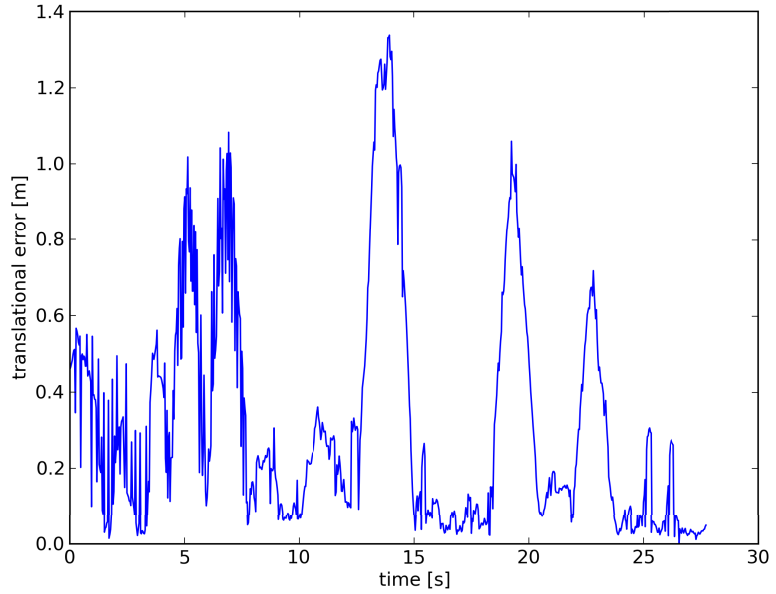


FIGURE 4.16: The RMSE of RPE for the Results of dense SLAM with the TUM *walking_xyz* sequence

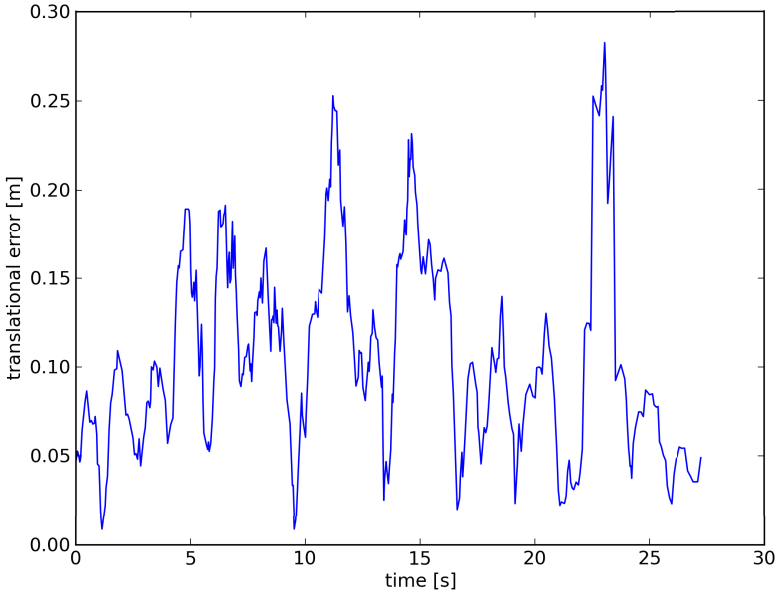


FIGURE 4.17: The RMSE of RPE for the Visual odometry of our method with the TUM *walking_xyz* sequence

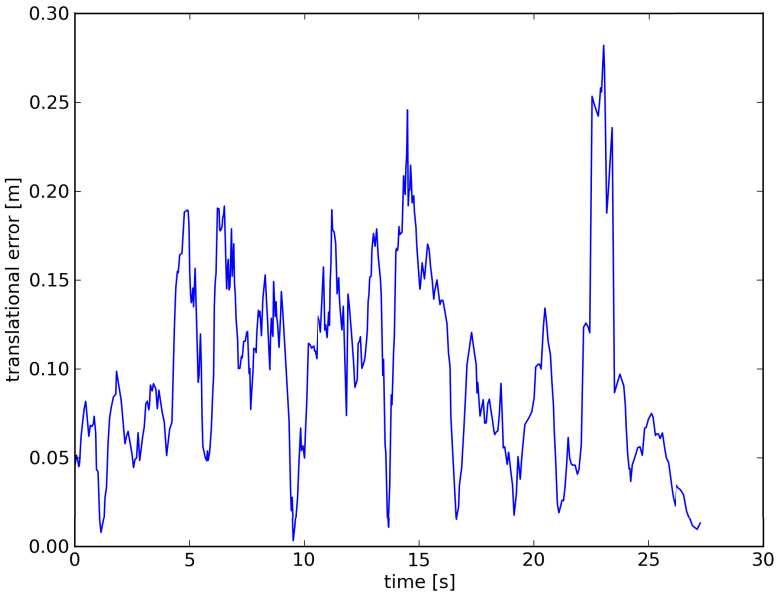


FIGURE 4.18: The RMSE of RPE for the Pose-graph SLAM results of our method with the TUM *walking_xyz* sequence

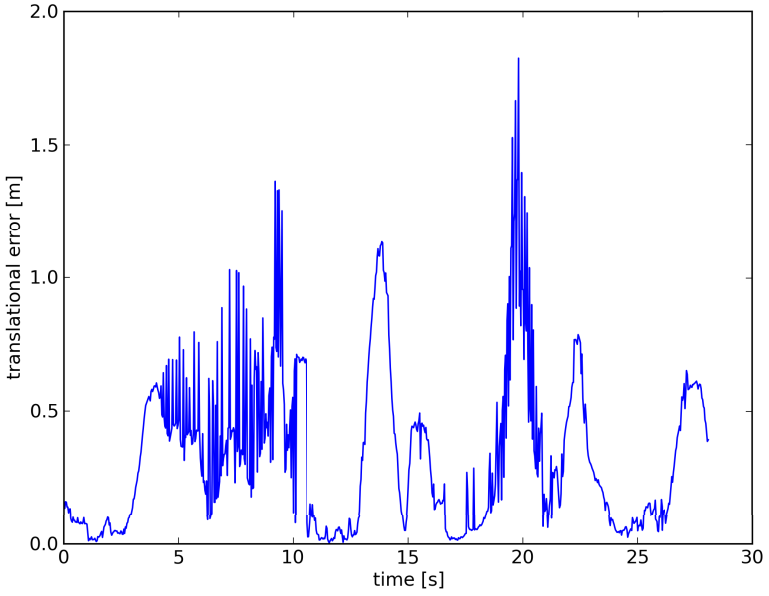


FIGURE 4.19: The RMSE of RPE for the Results of dense SLAM with the TUM *walking_rpy* sequence

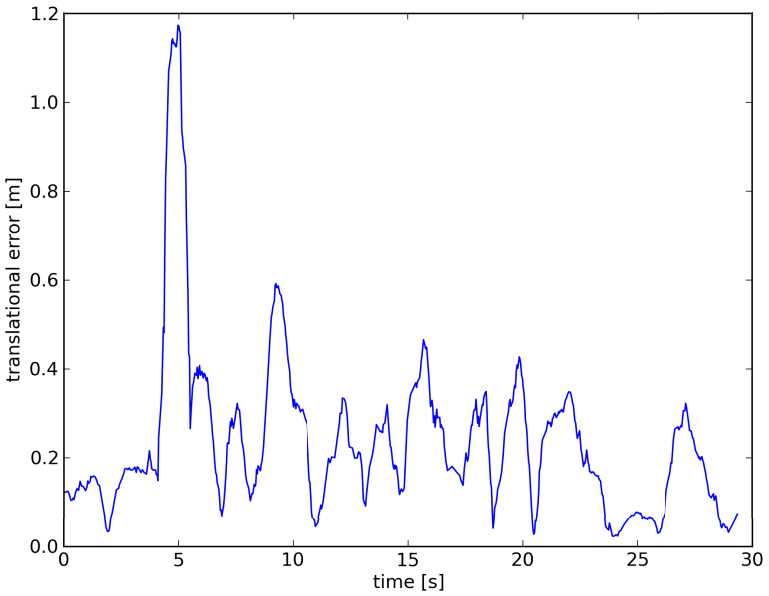


FIGURE 4.20: The RMSE of RPE for the Visual odometry of our method with the TUM *walking_rpy* sequence

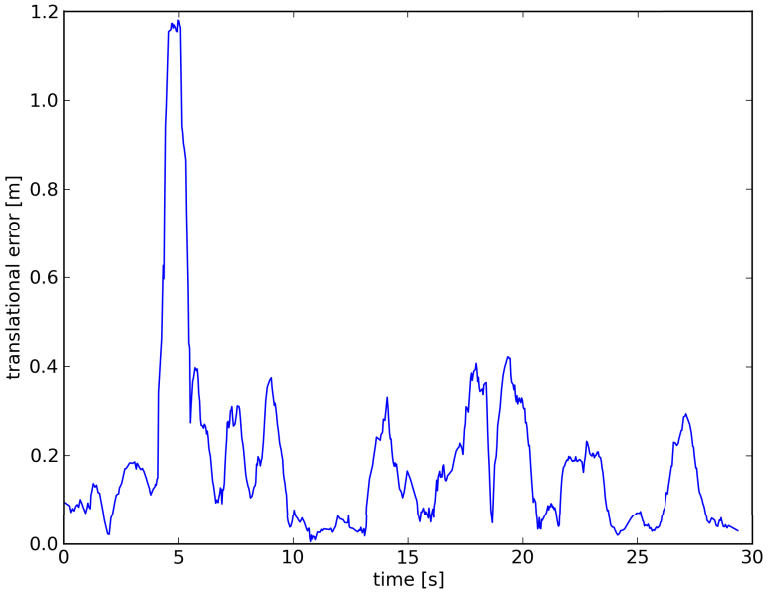


FIGURE 4.21: The RMSE of RPE for the Pose-graph SLAM results of our method with the TUM *walking_rpy* sequence

4.6.4 The fr3_walking_halfsphere Sequence

For dense SLAM, its RPE is $0.406m/s$, and ATE is $0.584m$.

Our visual odometry results are: RPE is $0.284m/s$, and ATE is $0.536m$; after pose-graph SLAM, RPE is $0.188m/s$, and ATE is $0.319m$. Our visual odometry has obvious degradation, although robust kernel based pose-graph SLAM has improved the results to some degree and the results are still better than that of dense SLAM.

In summary, the comparison of the results obtained by our method (denoted as SDMS) and dense visual SLAM (represented as DVSLAM) is shown in Table 4.4 and 4.5.

TABLE 4.4: Comparison of Visual Odometry and SLAM Results in Terms of RMSE of RPE (m/s)

Sequence No.	**DVSLAM**	**SDMS**	
		VO	**SLAM**
1	0.309	0.021	0.020
2	0.450	0.125	0.119
3	0.454	0.321	0.286
4	0.406	0.284	0.188

TABLE 4.5: Comparison of Visual Odometry and SLAM Results in Terms of RMSE of ATE (m)

Sequence No.	**DVSLAM**	**SDMS**	
		VO	**SLAM**
1	0.470	0.034	0.037
2	1.302	0.184	0.162
3	0.789	0.496	0.433
4	0.584	0.536	0.319

4.6.5 Discussion

As we can see, when compared with the state-of-the-art method, i.e., dense visual SLAM, for the four challenging sequences, our method can always produce much better results. On the other hand, when compared with the best SLAM results in static or less challenging scenarios, our method performs very well for the first sequence, and even its performance for the second sequence is still quite reasonable. But for the last two sequences, its performance is not as satisfactory as we have expected.

As a matter of fact, there are two possible reasons contributing to the degradation. Sometimes due to the covering of human bodies, acceptable features in the image are not enough in quantity, to meet the requirement of minimum 100 common feature points between two frames, we have to increase the thresholds. As a result, some human moving parts may be included in the static group. The other reason is that in some scenarios when human bodies come closer to the camera and many of the feature points on human bodies have coherent motion pattern, thus our current assumption that the biggest group corresponds to the static group will not hold. In the end, the wrong group may have

been picked up. Therefore, our current work is by no means the end of dealing with challenges caused by motion blur, large featureless zones and overwhelmingly dynamic parts. Instead, we hope our endeavours in this direction will draw attention to this problem and could inspire more fruitful work in the near future.

4.7 Summary

In this chapter, we firstly propose a general sparse feature-based motion model based segmentation algorithm for robust RGB-D SLAM. We have shown that together with the solution framework of static SLAM, it is able to handle outliers and dynamic objects in an efficient, effective and unified manner. Using real dataset, we have achieved promising results that are comparable or better than what has been achieved using dense method [14, 59].

However, in the *walking* sequences of TUM, motion blur and textureless areas are pervasive and therefore the number and quality of useful sparse features drops quickly, preventing us from obtaining better results from them. Nevertheless, those challenges represent what we need to address to enable robust SLAM in highly dynamic scenarios, and overcoming them will symbolise that the achieved SLAM method framework is directly applicable to much wider range of real problems. Therefore, it is very worthwhile for us to further investigate this issue. To face up to these challenges, we then propose another robust method which is mainly composed of a semi-dense version of motion segmentation method and pose-graph SLAM. Results using some of the most challenging sequences show that our method is more effective and robust than state-of-the-art to overcome the challenges. As far as we know, this is the first endeavour towards robust SLAM in highly challenging scenarios.

Nevertheless, to fully address all of the challenging sequences, on one hand, we need to find better way to automatically adjust the thresholds ([79] and [83] have made some beneficial endeavours, but more work is needed to fully solve this fundamental problem); on the other hand, we have to find more robust way to identify the static group, and in the next chapter we will seek another way to tackle this problem.

Chapter 5

Towards Dense Moving Object Segmentation based Robust Dense RGB-D SLAM in Dynamic Scenarios

Different from the previous chapter which focuses on proposing sparse or semi-dense feature-based motion segmentation methods and combining them with available back ends to perform SLAM in dynamic environments, the work in this chapter is targeted for dense moving object segmentation based dense SLAM by building upon the latest achievements in computer vision and RGB-D SLAM. Dense segmentation is more appealing to us, because not only its intuitive results are more in accordance with our expectation, but also it will make the following SLAM problem become much easier to handle and more robust results are possible.

The structure of this chapter is as follows. After introducing the specific challenges we are concerned and the background within which we are to formulate a feasible method in Section 5.1, we present different closely-related motion segmentation methods and RGB-D SLAM algorithms in Section 5.2, showing the advantages of combining n-view based moving object segmentation and dense SLAM to handle more general dynamic scenarios. After this, we propose the framework for robust SLAM in dynamic scenarios, including a practical way for dense moving object segmentation in Section 5.3. In Section 5.4, to demonstrate the effectiveness of our approach, we show some sampling moving object segmentation results and the improved SLAM results of our method compared with those of the original dense SLAM as well as spare feature-based SLAM when possible using some challenging real data. Section 5.5 concludes the chapter.

5.1 Introduction

As we have mentioned in Chapter 4, various measures have been taken to enhance the robustness of related SLAM algorithms. For example, RANSAC [13] and robust kenerls [74] are being widely adopted to handle outliers. While in multibody SLAM/MSaM, various motion segmentation methods including generalised principal component analysis [85] and local subspace affinity [86], can be employed to separate rigid objects from each other based on feature trajectory analysis. However, they usually require tough conditions in real scenarios. As a result, most of the applicable motion segmentation methods in multibody SLAM/MSaM are two-view based [57].

Nevertheless, as the latest achievement in computer vision along the line of n-view based motion segmentation, [50] tries to remove some unrealistic assumptions of the traditional n-view based motion segmentation methods and has proved to be capable of producing satisfactory dense segmentation results of moving objects for some real data.

Similar to previous chapters, in this chapter, we are still concerned about conducting SLAM with some continuous videos taken by a freely-moving RGB-D camera which is the only input and no other prior knowledge available. In addition to motion blur and feartureless regions, those videos contain non-rigid objects that may be static, partially moving, or totally moving from time to time. To cope with motion blur and feartureless regions, dense SLAM [59] proves to be a better choice than those sparse feature-based methods. However, special measures need to be taken to enhance its robustness in dynamic scenarios. Inspired by multibody SLAM, to conduct dense SLAM in such kinds of environment, we need to densely separate the moving objects out before executing SLAM. And the approach reported in [50] is quite promising for this goal. To substantiate our ideas, firstly, through theoretical analysis, we argue that *moving object segmentation* is a better choice to enable us to perform robust SLAM in dynamic scenarios as well as multibody SLAM when compared with other motion segmentation methods. Then, we propose a practical avenue to achieve this goal: based on the findings of [50], we propose practical measures to improve the dense segmentation results, then employ dense SLAM to estimate the camera poses. The main characteristics of our method are that it aims to separate different moving objects out before conducting SLAM, it can handle both rigid and non-rigid moving objects in a unified manner, and both the segmentation and SLAM are performed densely.

5.2 Related Work

In this section, we firstly give a brief review of different ways to perform motion segmentation. Motion segmentation constitutes a very important step for conducting SLAM in dynamic scenarios, but practical ways capable of handling general dynamic scenarios are still missing.

Furthermore, we give a short comparison of the available methods for visual odometry in RGB-D SLAM, showing their advantages and disadvantages.

5.2.1 Two-view/N-view Based Motion Segmentation Versus Extended N-view Based Moving Object Segmentation [50]

As we know, motion segmentation aims to separate the available scenarios into different motion groups without prior knowledge as of the moving objects or the camera's motion. Generally speaking, most of the available methods are only suitable for rigid or articulated objects, and they can be divided into two groups, i.e., two-view based and n-view based methods. And their output is the detected moving groups (corresponding to detectable moving parts beyond a threshold) of different objects.

From $2D$ to $3D$, numerous two-view based motion segmentation methods have been proposed [57, 71, 77, 87], and two-view based motion segmentation methods constitute a quick way for us to detect instant motion and obtain visual odometry in SLAM and SFM [57] [87], as our work in Chapter 3 and 4 have shown. However, intuitively, only when the two frames are discrete, the scenarios only contain rigid moving objects and the motion between them are big enough (but not too big to limit the number of matchable pairs of points), two-view based motion segmentation methods can separate different objects into different motion groups. Otherwise, two-view based motion segmentation methods can only tell us those moving parts that have gone beyond the threshold determined by the sensor noise level. Accordingly, we can see that the visual odometry and loop-closures constraints obtained by two-view based motion segmentation methods may be biased to some degree (one loop-closure example is shown in Fig. 5.1) no matter which threshold we choose, and a theoretically better way to avoid this problem is n-view based moving object segmentation.

At the same time, there are many n-view based $3D$ motion segmentation methods available in computer vision, and interested readers can refer to [53] for a detailed review. Nevertheless, some strong assumptions have prevented most of them from finding practical applications in SLAM. Firstly, most of them usually assume that each point need

to appear in every frame, which can only be met in controlled experiments. Secondly, most of them are sensitive to non-Gaussian noises and cannot tolerate errors brought

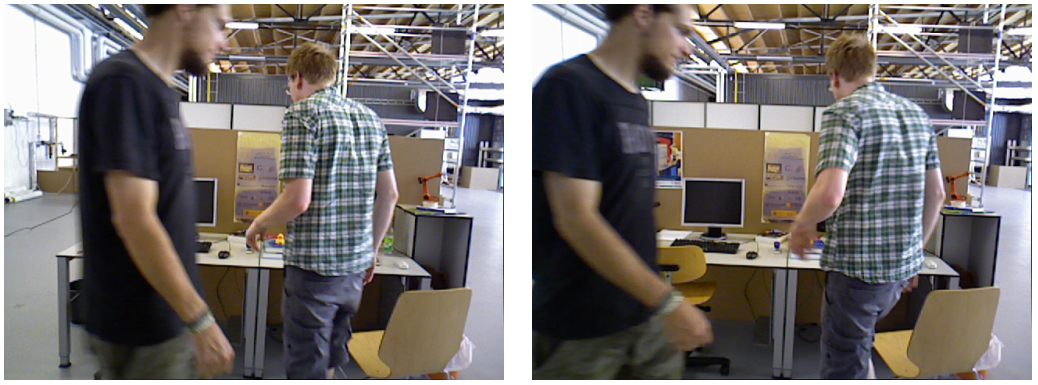


FIGURE 5.1: In this example, the biggest common part having depth information of the two non-consecutive images does not belong to the static group, making loop-closure detection harder to handle.

forward from feature detection, matching and tracking. Last but not least, most of them can only handle rigid or articulated body while practical SLAM need to handle rigid as well as various non-rigid moving objects including human beings. Therefore, in SLAM, we are expecting a robust motion segmentation method that is capable of handling noisy continuous data composed of non-rigid and rigid bodies in a unified manner.

Until quite recently, [50] proposes a robust method for moving object segmentation based on long term point trajectory analysis without prior model, making it applicable for some real scenarios. Firstly, it can handle point trajectories of arbitrary length. Secondly, by employing spectral clustering and a model selection process, it can handle noisy data with outliers. Furthermore, for some scenarios that contain more than one non-rigid moving objects, it also proves to be able to obtain acceptable segmentation results. It is based on this method that we propose a practical way to perform dense moving object segmentation for dense SLAM in dynamic scenarios.

5.2.2 RGB-D SLAM: Sparse Versus Dense

With the advent of affordable Microsoft Kinect, we are endowed with dense depth information along with RGB data. And with the help of pose-graph SLAM, localisation and mapping can be decoupled into two sub-steps. In RGB-D SLAM, maps can be dense because of the dense depth information provided by RGB-D cameras. However, in pose-graph based RGB-D SLAM, to estimate camera poses, we firstly can choose to employ sparse feature-based or dense methods to obtain visual odometry and loop-closure relative pose constraints [43]. Initially, sparse feature-based methods are very

popular because dense methods are usually much slower. Now with the emergence of dense SLAM [59], the situation is changing. Dense SLAM is making use of every pixel, both its colour and depth information, and it can obtain SLAM results in real time. To cope with motion blur and featureless regions, dense SLAM is a better choice.

Nevertheless, dense SLAM is vulnerable to the negative effects of dynamic elements in the images. Therefore, we need to densely separate the dynamic objects out before applying dense SLAM for dynamic scenarios. In this chapter, we aims to combine a practical extended n-view based dense moving object segmentation method with dense SLAM, enabling it to work in dynamic scenarios as well.

5.3 Moving Object Segmentation Based Robust Dense SLAM

A robust moving object segmentation method is proposed to enhance the robustness of dense SLAM in dynamic scenarios, and the overall process of our SLAM framework is illustrated in Fig. 5.2.

5.3.1 Proposed Robust Moving Object Segmentation Method

Moving object segmentation, as the pre-requisite of SLAM, need to be robust and adaptive enough to handle various dynamic scenarios. However, the state-of-the-art method may produce under or over-segmentation results, depending on the scenarios. Therefore, we propose further measures solely based on RGB information to overcome this kind of problems.

The Original Moving Object Detection and Segmentation Method and Its Results

As shown in Fig. 5.3, the original moving object segmentation method [50] can be divided into three steps: calculating optical flow, sparse point trajectory clustering and densification. Through this process, we can obtain satisfactory segmentation results for some videos including rigid and non-rigid moving objects. However, for many other videos, over-segmentation (a moving object has been segmented into several groups) or under-segmentation (several moving objects have been put into one group) can happen, and examples are shown in Fig. 5.4 and 5.5. Even tuning the related parameters (including the scale parameter λ which determines the normalisation fitness for trajectory

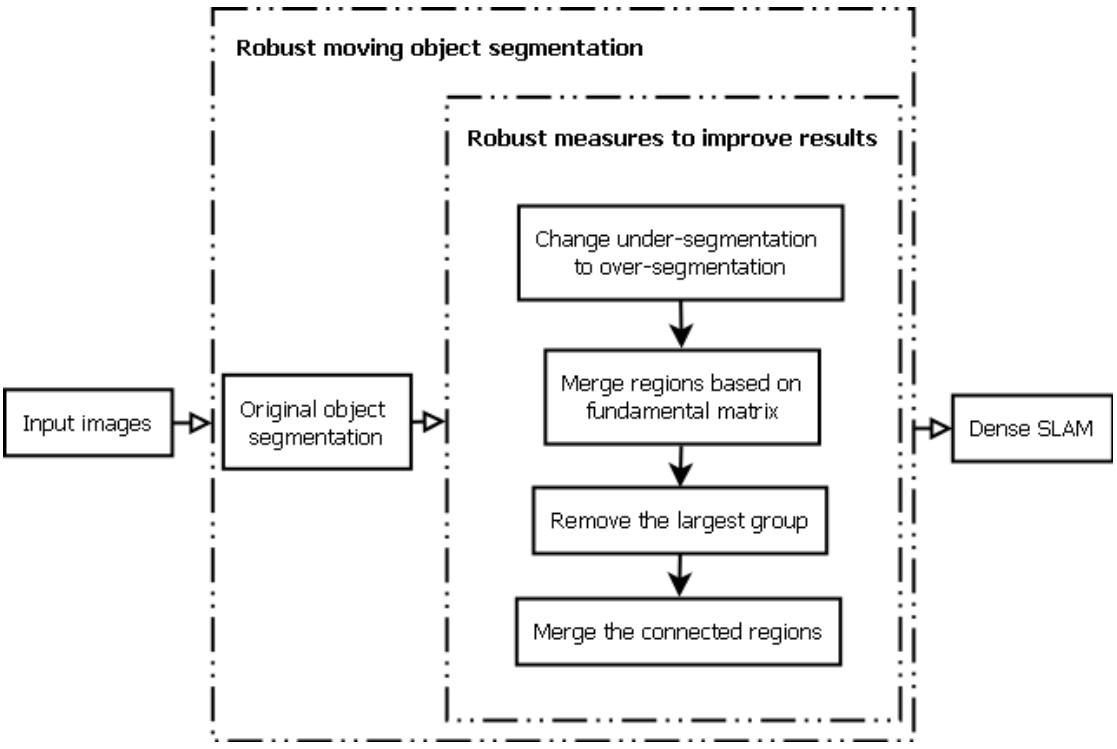


FIGURE 5.2: Flowchart of the whole process for robust SLAM

grouping, and weight parameter ν which keeps the balance between split and smoothness) cannot help solve the problem, although these two kinds of results can be converted into each other by tuning the parameters.

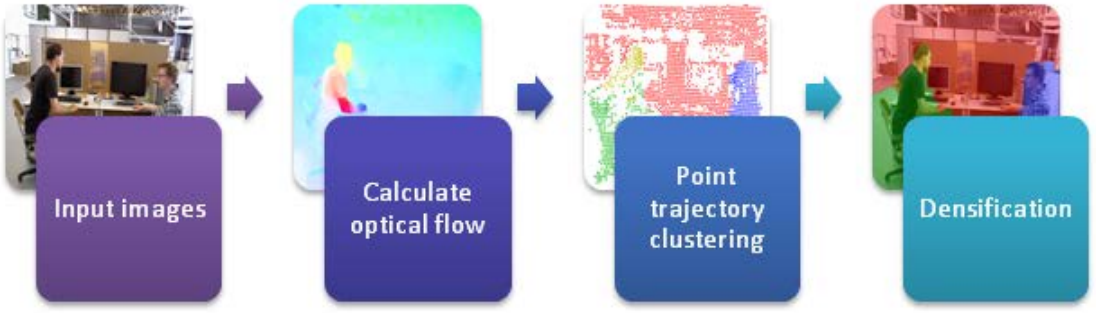


FIGURE 5.3: Flowchart of the original moving object segmentation method [50] with images showing the results at different stages.

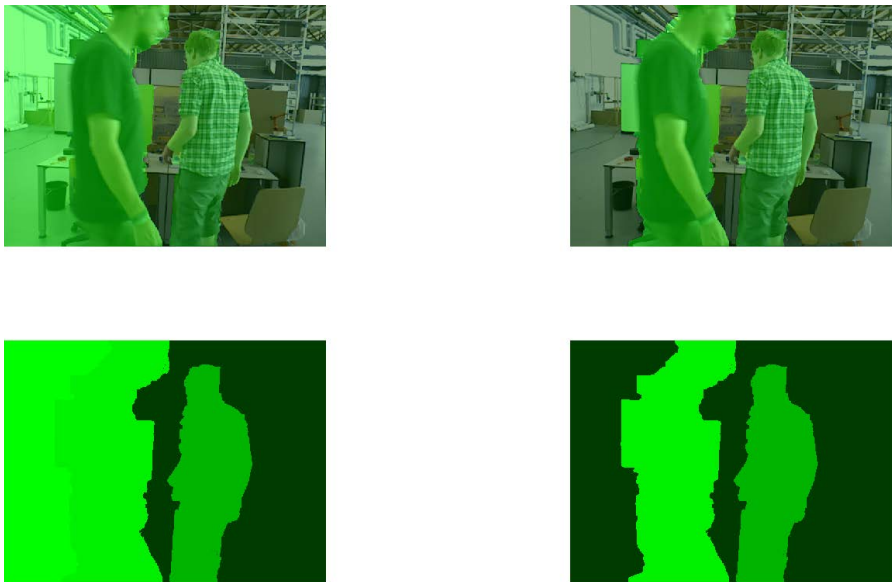


FIGURE 5.4: The left column represents the result before the merging using fundamental matrix, while the right one represents that after merging. The lower images represent the segmented group masks in different color intensities, and the upper images are the combination of the original image and segmentation results. Best viewed in colour.

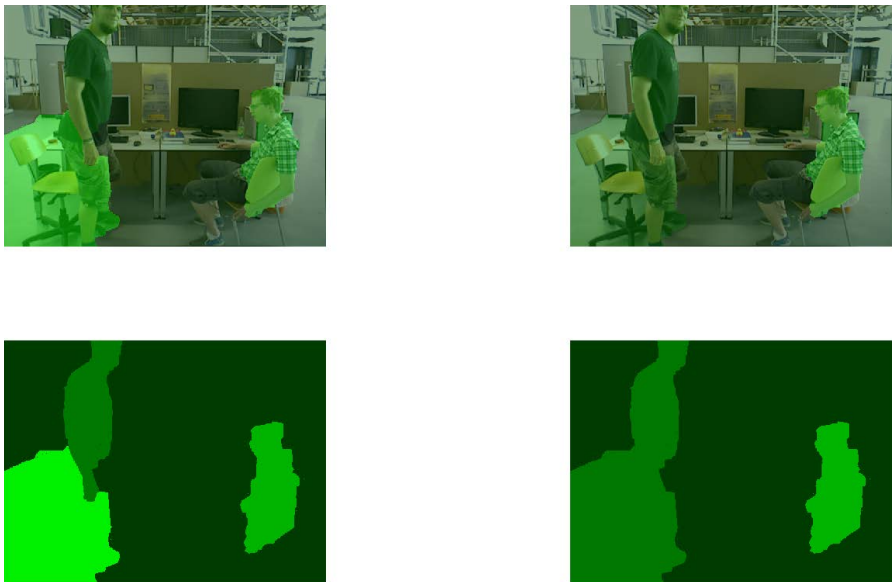


FIGURE 5.5: The left column represents the result before merging the connected neighbouring regions, while the right one represents that after merging. The lower images represent the segmented group masks in different color intensities, and the upper images are the combination of the original image and segmentation results. Best viewed in colour.

Robust Measures to Improve Results

As shown in Fig. 5.2, since under-segmentation is harder to tackle, we firstly turn it into the over-segmentation scenario by tuning the related parameters of the original method.

Secondly, we propose to look for those separated regions that always share the same motion model during the process and merge them. According to multi-view geometry [40], for those points belonging to a rigid body or the static environment, if we can obtain their positions in every two consecutive frames based on [88], we can find a fundamental matrix to describe their motion. On the other hand, for non-rigid human body, we cannot find a fundamental matrix to describe its motion as a whole; instead, each part of it may need one fundamental matrix to approximate its motion. And for all objects, their corresponding fundamental matrices are changing, either due to the motion of camera, their own movements, or both of them. For those regions whose predominant parts (70% is used in this chapter) always share the same fundamental matrices across the periods of their appearance, we will merge them into one group. A typical result is illustrated in Fig. 5.4. As we can see, after this step, over-segmented static groups can be found and combined together.

Thirdly, based on the assumption that the largest group corresponds to the static environment, we find the biggest group in each frame and remove it. And for the available benchmark dataset [61], in terms of the number of pixels, the aforementioned assumption is valid. Nonetheless, in some rare cases in practice, if the size of the second largest group is very close to the first one, it is still possible that the wrong group might be chosen.

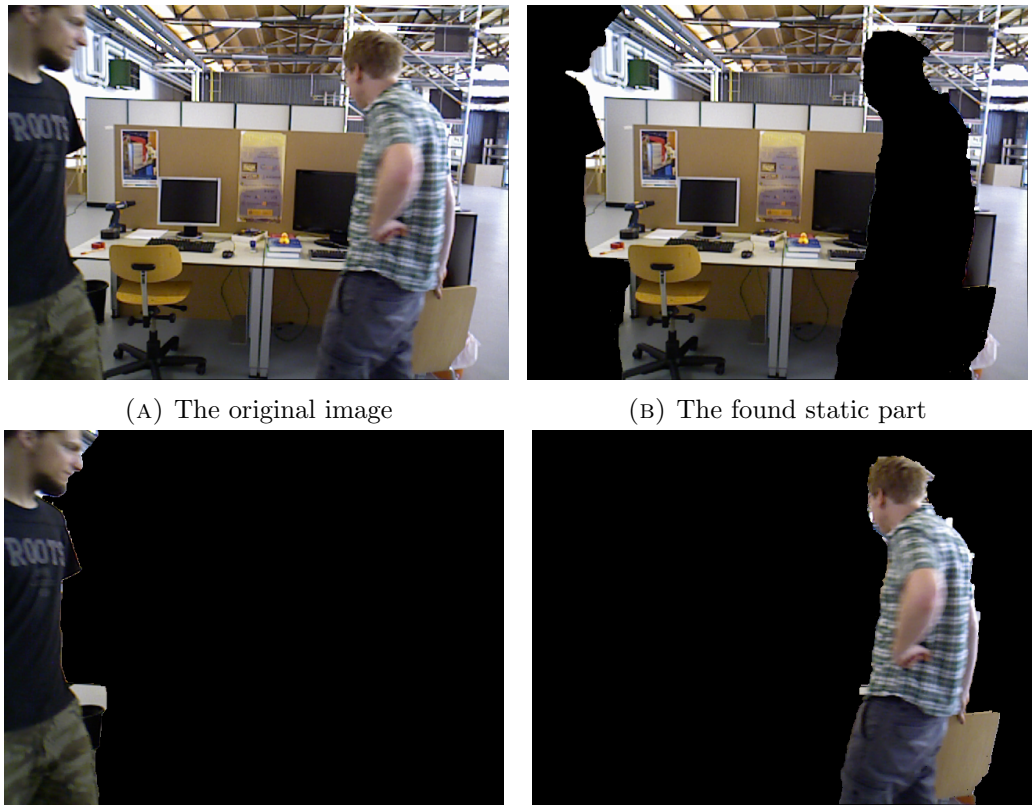
Fourthly, we check the changes of the connection between the remaining neighbouring regions during the course. For those remaining connected (the distances between the closest points from different regions are always within 1 pixel) throughout the process, it is highly possible that they belong to the same object, so we propose to agglomerate them at this step. This step can re-combine the over-segmented parts of human body together as shown in Fig. 5.5.

Through these steps, we can reasonably combine some segregated regions to produce more elegant segmentation results of the static environment and moving objects. Fig. 5.6 has shown one of the final segmentation results after applying our merging procedures.

5.3.2 Dense Visual SLAM

After densely separating the moving objects from the images, we can employ dense SLAM to estimate camera/robot poses only using the remaining static parts.

The overall process of dense SLAM is as follows: after obtaining visual odometry by using both dense colour and depth information, selecting keyframes and detecting loop-closures, dense SLAM makes use of G2O [33], which presents a general theoretical framework along with various practical implementations for graph-based SLAM, for pose optimisation, and outputting pose trajectories. Interested reader can refer to [59] for more details.



(A) The original image (B) The found static part

(C) Found people1 (D) Found people2

FIGURE 5.6: The original image and segmentation results (from the *walking_static* sequence).

There are several practical reasons why we need to densely remove the moving objects before using dense SLAM. Firstly, to improve robustness, dense SLAM proposes a fast dense image registration method based on joint optimisation of the colour and depth errors of all the available pixels. Although t-distribution has also been employed to deal with large errors, as we will see, those pixels corresponding to the moving objects in the scenarios can impose inevitable negative effects on the optimisation results. Secondly, dense SLAM proposes an entropy-based method for keyframe selection and loop-closure validation to reduce drift. However, without firstly densely removing the moving objects, the entropy value will be spoiled. As the result, unexpected keyframes may be selected and false loop closures may be found. Therefore, to some degree, dense SLAM is specially designed for static scenarios, and very susceptible to moving objects.

And the overall process of our proposed method is summarised in Algorithm 6.

Algorithm 6: Dense moving object segmentation based robust dense SLAM algorithm

Inputs: a sequence of RGB-D data;

Output: camera/robot poses;

Assumption: the largest group corresponds to the static group;

1. ***Dense moving object detection and segmentation***: perform [50], from which we can obtain the preliminary segmentation results. If they are under-segmented, we can adjust the related parameters to change it into over-segmentation;
2. ***Merge separated regions based on fundamental matrix***;
3. ***Remove the static region***: find the largest region and remove it;
4. ***Agglomerate remaining connected regions***;
5. ***Dense SLAM***: after separating the moving objects out, we put the obtained static regions in every image as the input to dense visual SLAM [59] to estimate the camera/robot poses.

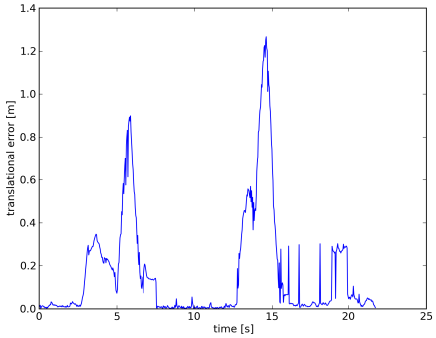
5.4 SLAM Results for The Benchmark Dataset

To show the effectiveness of Algorithm 6, we have chosen to compare the SLAM results on some challenging benchmark sequences (the *walking* series) provided by the TUM [61] benchmark dataset we have employed before. Moreover, to further support our analysis, we have shown the SLAM results both before and after moving object segmentation in this section as a comparison.

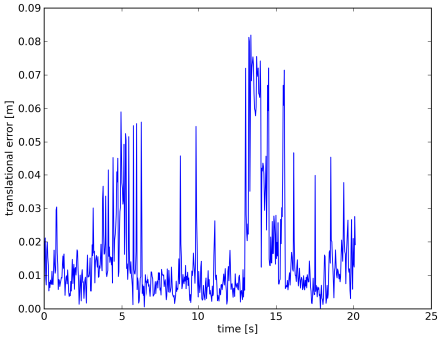
5.4.1 RGB-D SLAM Benchmark and Methods Involved

As we know, although many good results have been reported for most of the TUM dataset [59], the *walking* sequences are still among the most challenging ones that have not been fully solved yet. According to [61], they are specially designed for evaluating the robustness of visual SLAM and odometry algorithms when there are non-rigid moving objects dynamically occupying large parts of the visible scene. The major difficulties of these sequences lie in that motion blur, large featureless zones and large human movements are pervasive in the images. As we will see, the traditional sparse feature-based SLAM could not fully address them [87], nor could the state-of-the-art method in RGB-D SLAM known as dense SLAM [59] produce meaningful results.

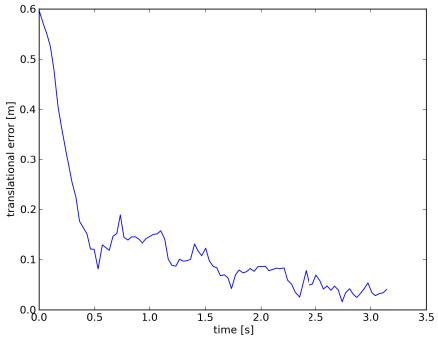
In Chapter 3, we proposes a sparse feature-based two-view motion segmentation method [87]. By combining with pose-graph SLAM, it has obtained a meaningful result just for the first sequence. In addition, we have also chosen dense SLAM as another reference.



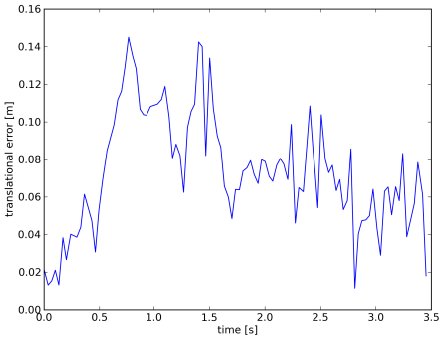
(A) RPE when directly applying dense SLAM to the *walking_static* sequence



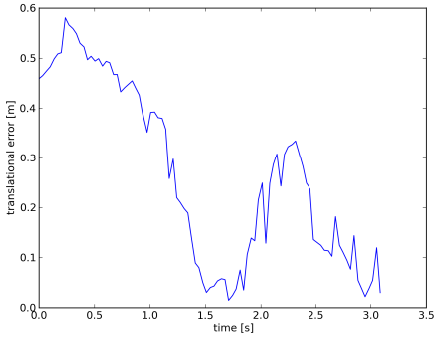
(B) RPE when applying dense SLAM to the segmented *walking_static* sequence



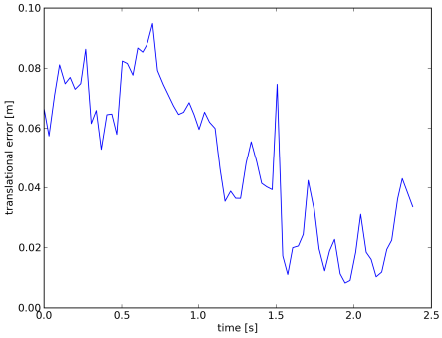
(C) RPE when directly applying dense SLAM to the *walking_halfsphere* sequence section



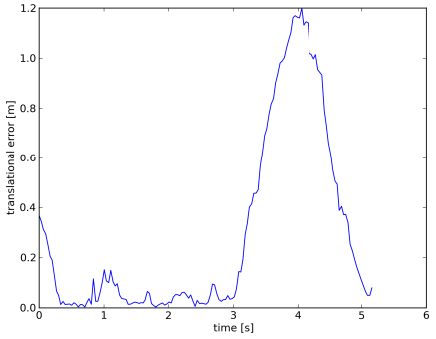
(D) RPE when applying dense SLAM to the segmented *walking_halfsphere* sequence section



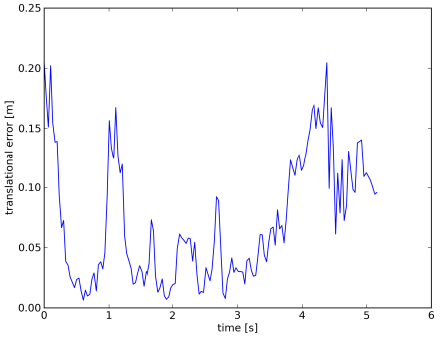
(E) RPE when directly applying dense SLAM to the *walking_xyz* sequence section



(F) RPE when applying dense SLAM to the segmented *walking_xyz* sequence section



(G) RPE when directly applying dense SLAM to the *walking_rpy* sequence section



(H) RPE when applying dense SLAM to the segmented *walking_rpy* sequence section

FIGURE 5.7: Comparison of dense SLAM results before and after moving object segmentation using the four *walking* sequences.

Therefore, including our own method, there are three methods altogether involved in this chapter for comparison. Nonetheless, only for the first sequence the results of the three methods are presented; while for the rest sequences we mainly compare the results obtained by Algorithm 6 with those by directly applying dense SLAM to the original videos.

5.4.2 Challenging Sequences

We firstly apply Algorithm 6 and dense SLAM to four challenging *walking* sequences in the TUM dataset, whose details are illustrated as follows:

walking_static: in this sequence, the camera is being kept in place manually, therefore its movement is small, while two people are moving around the table with large motions.

walking_halfsphere: in this sequence, the camera is moving on a small half sphere whose diameter is about one meter, while two persons are walking around in the office scene.

walking_xyz: in this sequence, the camera's movement is small, and two people are moving around the table.

walking_rpy: in this sequence, the camera mainly rotates, and two people are walking around the table. Since large parts of the visible scenes are dynamic, it constitutes a very difficult task.

5.4.3 Comparison of the Experimental Results

As we have done in previous chapters, following the rules proposed by [14, 59], we have quoted both the root mean square error (RMSE) of relative pose error (RPE) in meters per second and the RMSE of absolute trajectory error (ATE) for comparison.

Among the four sequences, The sparse method in Chapter 3 could only obtain a meaningful result for the first one: the RMSE of RPE is $0.084m/s$, and the RMSE of ATE is $0.161m$ [87]. And for parts of the other three sequences, the overall comparison of the results obtained by Algorithm 6 in this chapter (represented as MS_DSLAM) versus those of dense SLAM is summarised in Table 5.1. In addition, two representative detailed comparison of RPE of the four sequences is shown in Fig. 5.7, and similar results can be observed in the ATE case.

As we can see, the applicability of the traditional sparse feature-based SLAM to challenging dynamic scenarios is limited [87], nor could dense SLAM [59], which is known as

the state-of-the-art method in RGB-D SLAM, produce satisfactory results as it does in static scenarios, while Algorithm 6 has demonstrated its effectiveness and robustness.

5.4.4 Discussion

As we know, sparse feature-based methods as a classical choice for SLAM in static environments can usually produce satisfactory results. However, as noted in Chapter 3, when it comes to blurred images and featureless regions in dynamic scenarios, both the quantity and quality of detectable features decreases dramatically, thus jeopardising the applicability of this kind of methods in this case. Similarly, the results also show that dense SLAM is very sensitive to moving objects, as our previous analysis has indicated.

TABLE 5.1: Comparison of the RPE (m/s) & ATE (m) Results of Dense SLAM Versus Those of Algorithm 6 (MS_DSLAM) Using the Four Sequences

Seq.	**RPE**		**ATE**	
	DenseSLAM	***MS_DSLAM***	***DenseSLAM***	***MS_DSLAM***
1	0.309	0.022	0.470	0.024
2	0.175	0.080	0.116	0.055
3	0.321	0.055	0.202	0.040
4	0.477	0.088	0.515	0.076

5.5 Summary

Motion segmentation is of great importance for conducting SLAM in dynamic scenarios. In this chapter, we propose a practical moving object segmentation method that can densely segment rigid and non-rigid objects in a unified manner by building upon the latest achievements in computer vision. Combined with dense SLAM, it constitutes a new way for robust dense SLAM in dynamic scenarios as well as multibody SLAM. Results from some challenging real data have shown that this method is quite promising.

Nevertheless, Algorithm 6 is still not robust enough for some challenging dynamic scenarios. Therefore, in the near future, we aim to find more robust and efficient ways to perform moving object segmentation, and by integrating Algorithm 6 with object tracking to construct a complete solution for robust SLAM in dynamic scenarios.

Chapter 6

Conclusions and Future Work

In the previous chapters, from spare to dense, from motion segmentation to moving object segmentation, we have detailed our proposed 4 kinds of motion segmentation methods and their corresponding solution frameworks, and a series of tests through simulation and experiments using real data and benchmark dataset have been performed to demonstrate their advantages over state-of-the-art methods in handling challenging dynamic scenarios for RGB-D SLAM. As the results have shown, appropriate motion segmentation methods can outperform traditional robust measures including RANSAC and robust kernels in terms of efficiency and robustness and therefore more suitable to be integrated with those available SLAM backe-end solution frameworks.

6.1 What We Have Done: Motion Segmentation based Robust RGB-D SLAM

In Chapter 3, a distance-based motion segmentation algorithm using only two frames of RGB-D data is proposed, and results from both simulated and real data show its efficiency and reliability. To further verify its usability in multibody SLAM scenarios, we firstly utilize it to solve a simulated multibody RGB-D SLAM problem, and then apply it to segment a real RGB-D dataset collected by ourselves. Based on the good results of our motion segmentation algorithm, we can obtain satisfactory SLAM results for the simulated problem, and the segmentation results using real data also enable us to attain visual odometry for each motion group thus facilitating the following steps to solve the practical multibody RGB-D SLAM problems.

In the first part of Chapter 4, another 2-view sparse feature-based motion segmentation algorithm for RGB-D data is proposed which offers us a unified way to handle outliers

and dynamic scenarios. Together with the pose-graph SLAM framework, they constitute an effective and robust solution that enable us to conduct RGB-D SLAM in static environments as well dynamic ones, although traditionally they have been divided into different categories and treated separately using different kinds of methods. Through comparisons with RANSAC using simulated data and testing with different benchmark RGB-D sequences against the state-of-the-art method in RGB-D SLAM, we show that our solution is efficient and effective in handling general static and dynamic scenarios, some of which have not be achieved before.

Significant progress has been made to improve the robustness upon the traditional RGB-D SLAM algorithms in the last few years. However, when faced with challenging scenarios composed of large moving objects and image blur caused by quick camera motion, almost all of the available sparse and dense methods for SLAM will fail. In the second part of Chapter 4, we analyze the major difficulties involved in some of the challenging RGB-D benchmark datasets and propose a more robust and adaptive motion segmentation algorithm that can handle semi-dense features. By combining this algorithm with the available pose-graph SLAM, we can obtain a robust SLAM solution framework that is capable of dealing with some of the extremely difficult scenarios. The results from the most challenging sequences of the available benchmark show that our method has better performance than state-of-the-art.

In Chapter 5, based on the latest achievements in computer vision and RGB-D SLAM, a practical way for dense moving object segmentation and thus a new framework for robust dense RGB-D SLAM in challenging dynamic scenarios is put forward. As the state-of-the-art method in RGB-D SLAM, dense SLAM is very robust when there are motion blur or featureless regions, while most of those sparse feature-based methods could not handle them. However, it is very susceptible to dynamic elements in the scenarios. To enhance its robustness in dynamic scenarios, we propose to combine dense moving object segmentation with dense SLAM. Since the object segmentation results from the latest available algorithm in computer vision are not satisfactory, we propose some effective measures to improve upon them so that better results can be achieved. After dense segmentation of dynamic objects, dense SLAM then can be employed to estimate the camera/robot poses. Quantitative results from available challenging benchmark sequences have proved the effectiveness of our method.

6.2 Future Work: Beyond Motion Segmentation

As we know, motion segmentation constitutes a crucial step for conducting SLAM in dynamic scenarios. However, motion segmentation itself cannot solve all the problems

on our way.

For motion segmentation itself, we can only expect that it successfully separate the inputs into different groups according to their motions. However, how to ensure the resulting quality of automatic data association, how to form a bigger picture of the related scenarios in a spacial-temporal sense based on the outputs, are largely unaddressed. As a result, much more work is still needed, although some related practical measures and solution frameworks have been proposed in this direciton.

Firstly, we have tried to make use of optical flow results to deal with motion blur, but is it the best way to handle motion blur? Is it possible for us to fully address the challenges of motion blur and textureless regions only using the available dense methods?

Secondly, the mapping of non-rigid moving objects is still an open problem, although some researchers from computer vision are making promising progress.

Thirdly, it seems that, to fully address those problems encountered while performing SLAM in dynamic scenarios, we need to combine motion segmentation with moving object tracking and learning. However, what is the best way to combine motion detection, segmentation, tracking and learning together to form an efficient pipeline to handle general scenarios robustly?

Lastly, but not leastly, SLAM has adopted a bottom-up way, i.e., from small pieces of noisy senser data to form an overall filtered picture of the environment and the camera/robot poses. However, how far is us to semantic SLAM by following this bottom-up way? Is it possible to conduct human-robot-interaction related research based on moving object segmentation and long term accumulative learning, instead of solely relying on some pre-formulated knowledge as most available semantic SLAM related work has done?

We look forward to investigating these regards in detail in the near future.

Appendix A

Analysis of The Computational Cost of Algorithm 3

Suppose we have N_p pairs of shadow points belonging to N_g groups. Although the number of motion groups is unknown before segmentation, and may change a little bit during the process, in practice, it is very small when compared with the number of points. Therefore, here we just regard it as an unknown small constant. In addition, since in practice N_p is almost always larger than 7, we only consider situations under such conditions. And the unit of the cost is comparing the relative distance once or comparing the side once.

If $N_g = 1$, there is only one motion group. For the basic components, we just need to perform 3 times of comparisons to obtain them; for the rest points, each of them need to be compared with the basic components $(3+1)$ times. As a result, the corresponding total cost will be $3 + 4(N_p - 3) = 4N_p - 9$.

If $N_g \geq 2$, there are many possibilities for the sequence of the formed groups, so are the computational costs. Here we just consider two extremes. In the simplest case, most of the points belong to the first group and the rest groups contain just their respective basic components, i.e., 3 pairs of points. For the basic components, we just need to perform 3 times of comparisons to obtain them; for the rest points, each of them need to be compared with the basic components $(3 + 1)$ times. So, in total the cost for the first groups is $3 + 4(N_p - 3N_g) = 4N_p - 12N_g + 3$. For the other groups, each of them need to compare with basic components in its former groups before setting up a new group. In the least situation, we will refer to the next group just after one comparison and discovering that the result does not satisfy the grouping conditions. In this case, the cost for the i-th group will be $3 + 3(i - 1) = 3i$, and for $i = 2, ..., N_g$, in total it will

be $3(N_g+2)(N_g-1)/2$. Adding the cost associated with the scenario when $i=1$ into it, we will get $4N_p+1.5N_g^2-10.5N_g$.

When we come to the other extreme where most of the points belong to the last group, similarly we can see that for the basic components from group $i=1,...,N_g-1$, their cost in total is $\sum_{i=1}^{N_g-1} 3+3\times4\times(i-1)=6N_g^2-15N_g+9$. While for the points belonging to the last group, each of them need to be compared with the basic components in the prior groups before reaching the last one, therefore the maximum corresponding cost is $4N_g$. For the basic components in the last group, their cost is $3\times4\times(N_g-1)+3$. Consequently, in total, the cost will be $6N_g^2-15N_g+9+4N_g(N_p-3N_g)+3\times4\times(N_g-1)+3=4N_gN_p-6N_g^2-3N_g$.

Putting two extremes together, we can see that the computation cost c is bounded: $4N_p+1.5N_g^2-10.5N_g\leq c\leq 4N_gN_p-6N_g^2-3N_g$. Since the number of groups can be regarded as a constant, the computation cost is linear to the number of pairs of points, as stated in Section 3.2.3 of this paper.

Bibliography

[1] Sebastian Thrun. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.

[2] John J Leonard and Hugh F Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, 7(3):376–382, 1991. ISSN 1042-296X.

[3] Pieter Abbeel Van Den Berg, Jur and Ken Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information, 2011. ISSN 0278-3649.

[4] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006. ISSN 1070-9932.

[5] Udo Frese. Interview: Is slam solved? *Künstl Intell*, 24:255–257, 2010 2010. doi: 10.1007/s13218-010-0047-x.

[6] Ian D. Reid Nicholas D. Molton Davison, Andrew J. and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007. ISSN 0162-8828.

[7] Shoudong Huang and Gamini Dissanayake. Convergence and consistency analysis for extended Kalman filter based SLAM. *IEEE Transactions on Robotics*, 23:1036–1049, 2007. ISSN 15523098. doi: 10.1109/TRO.2007.903811.

[8] Luis Goncalves, Enrico Di Bernardo, Dave Benson, Marcus Svedman, Jim Ostrowski, Niklas Karlsson, and Paolo Pirjanian. A visual front-end for simultaneous localization and mapping. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 44–49. ISBN 078038914X. doi: 10.1109/ROBOT.2005.1570094.

[9] Hauke Strasdat, J M M Montiel, and Andrew J Davison. Scale Drift-Aware Large Scale Monocular SLAM. *Robotics: Science and Systems*, 2:5, 2010. doi: 10.1.1.165.7975. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.7975&rep=rep1&type=pdf.

[10] Niko Sünderhauf and Peter Protzel. Towards a robust back-end for pose graph SLAM. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1254–1261, 2012. ISBN 9781467314039. doi: 10.1109/ICRA.2012.6224709.

[11] Winston Churchill and Paul Newman. Experience-based navigation for long-term localisation. *The International Journal of Robotics Research*, 32(14):1645–1661, 2013. ISSN 0278-3649.

[12] Aisha Walcott-Bryant, Michael Kaess, Hordur Johannsson, and John J. Leonard. Dynamic pose graph SLAM: Long-term mapping in low dynamic environments. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1871–1878, 2012. ISBN 9781467317375. doi: 10.1109/IROS.2012.6385561.

[13] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. ISSN 0001-0782.

[14] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Robust odometry estimation for rgb-d cameras. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[15] Kuen-Han Lin and Chieh-Chih Wang. Stereo-based simultaneous localization, mapping and moving object tracking. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3975–3980. IEEE, 2010. ISBN 1424466741.

[16] Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889–916, 2007. ISSN 0278-3649.

[17] Motilal Agrawal, Kurt Konolige, and Luca Iocchi. Real-time detection of independent motion using stereo. In *Application of Computer Vision, 2005. WACV/-MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, volume 2, pages 207–214. IEEE, 2005. ISBN 0769522718.

[18] Kemal E Ozden, Konrad Schindler, and Luc Van Gool. Multibody structure-from-motion in practice. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(6):1134–1141, 2010. ISSN 0162-8828.

[19] Abhijit Kundu, K Madhava Krishna, and CV Jawahar. Realtime multibody visual slam with a smoothly moving monocular camera. In *Computer Vision (ICCV),*

2011 IEEE International Conference on, pages 2080–2087. IEEE, 2011. ISBN 145771101X.

[20] Anastasios Roussos, Chris Russell, Ravi Garg, and Lourdes Agapito. Dense multi-body motion estimation and reconstruction from a handheld camera. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 31–40. IEEE, 2012. ISBN 1467346608.

[21] Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 127–136. IEEE, 2011. ISBN 145772183X.

[22] Shoudong Huang, Heng Wang, Udo Frese, and Gamini Dissanayake. On the number of local minima to the point feature based SLAM problem. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2074–2079, 2012. ISBN 9781467314039. doi: 10.1109/ICRA.2012.6224876.

[23] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME-Journal of Basic Engineering*, 82:35–45, 1960. ISSN 00219223. doi: 10.1115/1.3662552. URL http://fluidsengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1430402.

[24] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proc. of 8th National Conference on Artificial Intelligence/14th Conference on Innovative Applications of Artificial Intelligence*, volume 68, pages 593–598, 2002. ISBN 0262511290. doi: 10.1.1.16.2153. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:FastSLAM+:+A+Factored+Solution+to+the+Simultaneous+Localization+and+Mapping+Problem#0.

[25] Michael Montemerlo, Sebastian Thrun, Daphne Roller, and Ben Wegbreit. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 1151–1156. LAWRENCE ERLBAUM ASSOCIATES LTD, 2003. ISBN 1045-0823.

[26] Bill Triggs, Philip F Mclauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. *Vision algorithms: theory and practice*, pages 153–177, 2000.

[27] Feng Lu and Evangelos Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4:333–349, 1997. ISSN 0929-5593, 1573-7527. doi: 10.1023/A:1008854305733. URL http://link.springer.com/article/10.1023/A:1008854305733$\delimiter"026E30F$nhttp://link.springer.com/article/10.1023/A:1008854305733?LI=true$\delimiter"026E30F$nhttp://link.springer.com/content/pdf/10.1023/A:1008854305733.

[28] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2:31–43, 2010. ISSN 1939-1390. doi: 10.1109/MITS.2010.939925.

[29] Gibson Hu. *Towards Reliability and Scalability in Feature Based Simultaneous Localization and Mapping*. PhD thesis, Faculty of Engineering and Information Technology Centre for Autonomous Systems, University of Technology Sydney, 2014.

[30] Ayoung Kim and Ryan Eustice. Pose-graph visual SLAM with geometric model selection for autonomous underwater ship hull inspection. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pages 1559–1565, 2009. ISBN 9781424438044. doi: 10.1109/IROS.2009.5354132.

[31] Kaustubh Pathak, Andreas Birk, Narunas Vaskevicius, Max Pfingsthorn, Sören Schwertfeger, and Jann Poppinga. Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation. *Journal of Field Robotics*, 27:52–84, 2010. ISSN 15564959. doi: 10.1002/rob.20322.

[32] Luca Carlone, Rosario Aragues, José A Castellanos, and Basilio Bona. A fast and accurate approximation for planar pose graph optimization. *The International Journal of Robotics Research*, pages 0278364914523689–, 2014. ISSN 0278-3649. doi: 10.1177/0278364914523689. URL http://ijr.sagepub.com/content/early/2014/05/01/0278364914523689?papetoc.

[33] Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3607–3613, 2011. ISBN 1612843867.

[34] Niko Sunderhauf and Peter Protzel. Switchable constraints for robust pose graph slam. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1879–1884. IEEE, 2012. ISBN 1467317373.

[35] Yasir Latif, César Cadena, and José Neira. Robust graph slam back-ends: A comparative analysis. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2683–2690. IEEE, 2014.

[36] Davide Scaramuzza and Friedrich Fraundorfer. Visual Odometry [Tutorial], 2011. ISSN 1070-9932.

[37] Friedrich Fraundorfer and Davide Scaramuzza. Visual odometry: Part II: Matching, robustness, optimization, and applications. *IEEE Robotics and Automation Magazine*, 19:78–90, 2012. ISSN 10709932. doi: 10.1109/MRA.2012.2182810.

[38] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. ISSN 0920-5691.

[39] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110:346–359, 2008. ISSN 10773142. doi: 10.1016/j.cviu.2007.09.014.

[40] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. ISBN 0521540518.

[41] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):698–700, 1987. ISSN 0162-8828.

[42] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 1, 2004. ISSN 1063-6919. doi: 10.1109/CVPR.2004.1315094.

[43] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012. ISSN 0278-3649.

[44] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow, 1981. ISSN 00043702.

[45] Ameesh Makadia, Christopher Geyer, and Kostas Daniilidis. Correspondence-free structure from motion. *International Journal of Computer Vision*, 75:311–327, 2007. ISSN 09205691. doi: 10.1007/s11263-007-0035-2.

[46] Charles Bibby and Ian Reid. Simultaneous localisation and mapping in dynamic environments (slamide) with reversible data association. In *Proceedings of Robotics: Science and Systems*.

[47] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the ekf-slam algorithm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3562–3568. IEEE. ISBN 1424402581.

[48] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011. ISBN 145771101X.

[49] Konrad Schindler, David Suter, and Hanzi Wang. A model-selection framework for multibody structure-and-motion of image sequences. *International Journal of Computer Vision*, 79:159–177, 2008. ISSN 09205691. doi: 10.1007/s11263-007-0111-7.

[50] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1, 2013. ISSN 0162-8828.

[51] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstrained video. In *2013 IEEE International Conference on Computer Vision*, 2013.

[52] Ravi Garg, Anastasios Roussos, and Lourdes Agapito. Dense variational reconstruction of non-rigid surfaces from monocular video. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1272–1279. IEEE, 2013. ISBN 1063-6919.

[53] Roberto Tron and René Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 1–8, 2007. ISBN 1424411807.

[54] Samunda Perera and Nick Barnes. A simple and practical solution to the rigid body motion segmentation problem using a rgb-d camera. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 494–500. IEEE, 2011. ISBN 145772006X.

[55] Tat-Jun Chin, David Suter, and Hanzi Wang. Multi-structure model selection via kernel optimisation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3586–3593. IEEE, 2010. ISBN 1424469848.

[56] Uwe Franke, Clemens Rabe, Hernán Badino, and Stefan Gehrig. *6d-vision: Fusion of stereo and motion for robust environment perception*, pages 216–223. Springer, 2005. ISBN 3540287035.

[57] Konrad Schindler, U James, and Hanzi Wang. *Perspective n-view multibody structure-and-motion through model selection*, pages 606–619. Springer, 2006. ISBN 3540338322.

[58] Hauke Strasdat, José MM Montiel, and Andrew J Davison. Visual SLAM: Why filter? *Image and Vision Computing*, 30:65–77, 2012. ISSN 02628856. doi: 10.1016/j.imavis.2012.02.009.

[59] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 2100–2106, 2013. ISBN 2153-0858.

[60] Maurice F Fallon, Hordur Johannsson, Michael Kaess, David M Rosen, Elias Muggler, and John J Leonard. Mapping the mit stata center: Large-scale integrated visual and rgb-d slam, 2012.

[61] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012.

[62] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002. ISSN 0920-5691.

[63] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 519–528. Ieee, 2006. ISBN 0769525970.

[64] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. ISSN 0920-5691.

[65] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *International Symposium on Robotics Research (ISRR)*, 2011.

[66] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[67] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the rgb-d slam system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1691–1696. IEEE, 2012. ISBN 146731403X.

[68] Sunglok Choi, Taemin Kim, and Wonpil Yu. Performance evaluation of ransac family. In *British Machine Vision Conference (BMVC)*, 2009.

[69] Etienne Vincent and Robert Laganiére. Detecting planar homographies in an image pair. In *Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on*, pages 182–187. IEEE, 2001. ISBN 9539676940.

[70] Marco Zuliani, Charles S Kenney, and BS Manjunath. The multiransac algorithm and its application to detect planar homographies. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–153–6. IEEE, 2005. ISBN 0780391349.

[71] Youbing Wang and Shoudong Huang. An efficient motion segmentation algorithm for multibody rgb-d slam. In *Australasian Conference on Robotics and Automation (ACRA)*, 2013.

[72] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007. ISBN 142441749X.

[73] Nikolas Engelhard, Felix Endres, Jürgen Hess, Jürgen Sturm, and Wolfram Burgard. Real-time 3d visual slam with a hand-held rgb-d camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, volume 2011, 2011.

[74] Peter J Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. ISSN 0003-4851.

[75] Yasir Latif, Cesar Dario Cadena Lerma, and José Neira. Robust loop closing over time. In *Robotics: Science and Systems*, 2012.

[76] Edwin Olson and Pratik Agarwal. Inference on networks of mixtures for robust robot mapping. In *Robotics: Science and Systems*, 2012.

[77] Stephen M Smith. Asset-2: Real-time motion segmentation and object tracking. *Real-Time Imaging*, 4(1):21–40, 1998. ISSN 1077-2014.

[78] Ondřej Chum, Jiří Matas, and Josef Kittler. *Locally optimized RANSAC*, pages 236–243. Springer, 2003. ISBN 3540408614.

[79] Hanzi Wang, Tat-Jun Chin, and David Suter. Simultaneously fitting and segmenting multiple-structure data with outliers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(6):1177–1192, 2012. ISSN 0162-8828.

[80] Jurgen Sturm, Wolfram Burgard, and Daniel Cremers. Evaluating egomotion and structure-from-motion approaches using the tum rgb-d benchmark. In *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*.

[81] Jorg Stuckler and Sven Behnke. Integrating depth and color cues for dense multi-resolution scene mapping using rgb-d cameras. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, pages 162–167. IEEE, 2012. ISBN 1467325104.

[82] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):978–994, 2011. ISSN 0162-8828.

[83] Hanzi Wang, Jinlong Cai, and Jianyu Tang. Amsac: An adaptive robust estimator for model fitting. In *IEEE International Conference on Image Processing (ICIP)*.

[84] Niko Sunderhauf. *Robust Optimization for Simultaneous Localization and Mapping*. Phd thesis, 2012.

[85] René Vidal, Yi Ma, and Shankar Sastry. Generalized principal component analysis (gpca). In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 1, pages I–621–I–628 vol. 1, 2003. ISBN 0769519008.

[86] Jingyu Yan and Marc Pollefeys. *A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate*, pages 94–106. Springer, 2006. ISBN 3540338381.

[87] Youbing Wang and Shoudong Huang. Motion segmentation based robust rgb-d slam. In *World Congress on Intelligent Control and Automation (WCICA)*, 2014.

[88] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. Motion detail preserving optical flow estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34 (9):1744–1757, 2012. ISSN 0162-8828.