REAL TIME DENSE SLAM IN UNSTRUCTURED DYNAMIC CLUTTERED 3D ENVIRONMENTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZGÜR HASTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONIC ENGINEERING

FEBRUARY 2022

Approval of the thesis:

REAL TIME DENSE SLAM IN UNSTRUCTURED DYNAMIC CLUTTERED 3D ENVIRONMENTS

submitted by **ÖZGÜR HASTÜRK** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** i**n Electrical and Electronic Engineering, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İlkay Ulusoy
Head of the Department, **Electrical and Electronics Engineering**

Prof. Dr. Aydan Müşerref Erkmen
Supervisor, **Electrical and Electronics Engineering, METU**

Examining Committee Members:

Prof. Dr. Umut Orguner
Electrical and Electronics Engineering, METU

Prof. Dr. Aydan Müşerref Erkmen
Electrical and Electronics Engineering, METU

Prof. Dr. A. Aydın Alatan
Electrical and Electronics Engineering, METU

Assist. Prof. Dr. Yakup Özkazanç
Electrical and Electronics Engineering, Hacettepe University

Assist. Prof. Dr. Oğuzhan Çifdalöz
Electrical and Electronics Engineering, Çankaya University

Date: 03.02.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Özgür, Hastürk
Signature :

ABSTRACT

REAL TIME DENSE SLAM IN UNSTRUCTURED DYNAMIC CLUTTERED 3D ENVIRONMENTS

Hastürk, Özgür
Doctor of Philosophy, Electrical and Electronic Engineering
Supervisor : Prof. Dr. Aydan M. Erkmen

February 2022, 158 pages

SLAM problem has been extensively studied by researchers in the field of robotics, however, conventional approaches in mapping assume a static environment. The static assumption is valid only in a small region; it limits the application of visual SLAM in dynamic environments. Recently proposed state of art SLAM solutions for dynamic environment use different semantic segmentation methods such as Mask R-CNN and SegNet; however, these frameworks are based on a sparse mapping framework (ORBSLAM). In addition, segmentation process increases the computational power, which makes these SLAM algorithms unsuitable for real-time mapping. Therefore, there is no effective dense RGB-D SLAM method for real-world unstructured and dynamic environments. In this study, we propose a novel real-time dense SLAM method for dynamic environments where 3D reconstruction error is manipulated for identification of static and dynamic classes having generalized Gaussian distribution. Our proposed approach requires neither explicit object tracking nor object classifier, which makes it robust to any type of moving object and suitable for real-time mapping. Our method eliminates the repeated views and uses consistent data that enhance the performance of volumetric fusion. For completeness, we compare our proposed method using different type of high dynamic dataset, which are publicly available, in order to demonstrate the versatility and robustness of our approach. Experiments shows that tracking performance of our proposed method better than other dense and dynamic SLAM approaches.

Keywords: Dynamic Mapping; Visual SLAM; Localization; 3D Reconstruction

ÖZ

YAPILANDIRILMAMIŞ DAĞINIK DİNAMİK 3 BOYUTLU ORTAMLARDA GERÇEK ZAMANLI YOĞUN ANLIK KONUMLAMA VE HARİTALAMA

Hastürk, Özgür
Doktora, Elektrik ve Elektronik Mühendisliği
Tez Yöneticisi: Prof. Dr. Aydan M. Erkmen

Şubat 2022, 158 sayfa

SLAM (Eşzamanlı Yer Belirleme ve Haritalandırma) problemi robotik alanında araştırmacılar tarafından sıklıkla çalışılan bir konu olmasına rağmen haritalama alanındaki geleneksel yaklaşımlar bulunulan ortamın durgun bir ortam olduğunu varsaymaktadır. Durgun ortam varsayımı sadece küçük alanlar için geçerli olmakta ve görsel SLAM'ın dinamik ortamdaki uygulamalarını sınırlandırmaktadır. Son zamanlarda dinamik ortamlar için önerilen son teknoloji SLAM çözümleri, Mask R-CNN ve SegNet gibi farklı anlamsal bölümleme metotları kullanmaktadır ancak bu yazılımlar seyrek haritalama yazılım çerçevelerine (ORBSLAM) dayanmaktadır. Ek olarak bölümleme süreci hesaplama gücünü arttırmakta ve bu durum SLAM algoritmalarını gerçek zamanlı haritalama için elverişsiz kılmaktadır. Bu nedenle gerçek dünyanın yapılandırılmamış ve dinamik ortamı için geçerli yoğunlukta RGB-D SLAM metodu bulunmamaktadır. Bu çalışmada; üç boyutlu (3D) yeniden yapılandırma hatalarının, genelleştirilmiş Gauss dağılımı kullanılarak statik ve dinamik sınıfların tanımlanabilmesi amacıyla manipüle edilmesi ve bu şekilde dinamik ortamlar için yeni bir gerçek zamanlı yoğun SLAM metodu oluşturulması amaçlanmıştır. Hedeflenen bu yaklaşımın belirgin obje takibi ve obje sınıflandırıcıya gerek duymaması, bu yaklaşımı herhangi bir hareketli objeye dayanıklı ve gerçek zamanlı haritalandırmaya uygun hale getirmektedir. Bu metot tekrarlanan görünümleri elemekte ve hacimsel füzyon performansını geliştiren tutarlı verileri kullanmaktadır. Tezin son kısmında; oluşturulan yaklaşımın çok yönlülüğü ve dayanıklılığının ispatlanması adına, literatürden elde edilen farklı tipte yüksek dinamik veri setleri ile hedeflenen metot karşılaştırılmıştır. Yapılan karşılaştırmalar hedeflenen metodun izleme performansının diğer yoğun ve dinamik SLAM yaklaşımlarına nazaran daha iyi olduğunu göstermiştir.

Keywords: Dinamik Haritalama; Görsel Eşzamanlı Yer Belirleme ve Haritalandırma; Konumlama; 3 Boyutlu Yeniden İnşa

To Zeynep and Tarçın,

ACKNOWLEDGMENTS

I would like to thank firstly my supervisor Professor Aydan Müşerref Erkmen for her support, guidance and friendship. It was a great honor and experience to work with her for the graduation. I wish to express my deepest gratitude her helpful criticism, never ending guidance, patience, advice and support in the progress and preparation of this thesis. I would also like to thank Professor Umut Orguner and Assistant Professor Yakup Özkazanç for their guidance in the thesis monitoring committee meetings which help me very much. In addition, I would like to thank my thesis jury members Professor Abdullah Aydın Alatan and Assistant Professor Oğuzhan Çifdalöz for their comments and suggestions, which help us very much while editing this thesis.

I would like to thank my dear family for their help and support not only during my thesis preparation but also throughout my life.

Finally, I would like to thank Zeynep Özden Yaşa Hastürk for her great support every second of my effort while writing this thesis though thanking her is not sufficient.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xv

LIST OF ABBREVIATIONS xviii

LIST OF SYMBOLS xix

CHAPTERS

1 INTRODUCTION 1

1.1 Statement of the Problem 1

1.2 Objective and Goals of the Study 3

1.3 Methodology 4

1.4 Evaluations of the Approach 5

1.5 Outline of the Thesis 6

2 LITERATURE REVIEW 9

2.1 Visual SLAM 11

3 PRELIMINARIES 17

3.1.1 Non-Linear Least-Squares (NLS) Optimization 17

3.1.2 Expectation–maximization (EM) algorithm 26

3.2 RGB-D Sensors 33

4 OUR PROPOSED METHODOLOGY 39

4.1 Justification of the Methodology and Novelties of Our Approach 39

4.2 Preliminaries and Notations 42

4.3 Depth Smoothing and Feature Matching 44

4.4 Pose Estimation 48

4.5 SDF Representation, 3D Reconstruction and Volume Visualizations 67

4.6 Dynamic Detection 74

5 EXPERIMENTAL RESULTS AND DISCUSSIONS 81

5.1 Static Environment 82

5.1.1 ICL NUIM Dataset 82

5.1.2 CoRBS 86

5.2 Dynamic Environment 89

5.2.1 TUM RGB-D Dataset 89

5.2.2 Bonn RGB-D Dynamic Dataset 104

5.2.3 VolumeDeform Dataset 107

5.2.4 CVSSP RGB-D Dataset 108

5.2.5 Outdoor mapping performance 110

6 SENSITIVITY ANALYSIS 113

7 CONCLUSION 123

7.1 Future Works 125

REFERENCES 129

APPENDICES

A. Mathematical Preliminaries 141

B. Rigid Body Motion 141

C. Transformation Matrices in Robotic Representation 148

CURRICULUM VITAE 157

LIST OF TABLES

TABLES

Table 4.1: BONN moving obstructing box dataset – change of translational ATE (RMSE cm) with respect to Regularization and Huber constant 66

Table 4.2: BONN moving obstructing box Dataset – comparison of translational ATE (RMSE cm) with changing Iteration Number and Downsample option 67

Table 5.1: Comparison of surface reconstruction accuracy (mean value) 83

Table 5.2: RMSE of the translational and rotational drift (RPE) in m/s and deg/s respectively. 87

Table 5.3: TUM Dataset – Execution Time 94

Table 5.4: TUM Dataset – Translational RPE (RMSE cm/s) 99

Table 5.5: TUM Dataset – Translational RPE (RMSE deg/s).............................. 100

Table 5.6: TUM Dataset – Translational ATE (RMSE cm).................................. 101

Table 5.7: BONN Dataset – Translational RPE - (RMSE cm/s)........................... 106

Table 5.8: VolumeDeform Dataset results 107

Table 6.1: TUM Dataset fr3/walking xyz – Translational ATE (RMS Error m) .. 117

Table 6.2: Change of Translational ATE (RMS m) with respect to weight (regularization constant, Huber constant and voxel size is constant).................... 119

Table 6.3: TUM “fr3/walking static” Sequence Translational ATE Error (RMS Error [cm]) 121

LIST OF FIGURES

FIGURES

Figure 3.1: Example of RGB-D sensors. From left to right: Microsoft Kinect, Intel RealSense D435. Courtesy of Microsoft and Intel Corporation. 34
Figure 3.2: Output images of RGB-D sensors. Left: RGB image. Right: Depth image. The brighter a pixel is, the farther away it is from the sensor. 34
Figure 4.1: Flowchart of the DUDMAP 41
Figure 4.2: DUDMAP scheme 42
Figure 4.3: Original input image , result of a depth image with adaptive bilateral filter and , invalid measurements handling 46
Figure 4.4: a) Original image b) Final image 47
Figure 4.5: VolumeDeform boxing sequence similarity ratio 47
Figure 4.6: BONN “crowd2” sequence similarity ratio with respect to threshold value. 49
Figure 4.7: Sphere tracing. The dots represent the points at which the function.... 51
Figure 4.8: Depth and color images of the same office desk 53
Figure 4.9: The coordinates of each voxel center is projected into the image plane. Note that not every projection will fall within the subset of the plane where the depth image is defined. 54
Figure 4.10: Voxels are updated with the difference between the depth image value and the distance to the respective voxel from the sensor, along the viewing direction. 55
Figure 4.11: Surface at the boundary between positive and negative values 56
Figure 4.12: Octree structure. In each successive picture the voxels are two times smaller. 68
Figure 4.13: Different levels of an octree-based model, in each successive picture the voxels are two times smaller 69
Figure 4.14: Hash table , voxel blocks 71

Figure 4.15: Gradient shaded representation of the three-dimensional in dividing cube algorithm [56] 72
Figure 4.16: Illustration of reflective, original and rotational symmetric condition 73
Figure 4.17: The 15 basic intersection topologies [55] 74
Figure 4.18: Inconsistency map of two images (EPFL RGB-D Pedestrian Dataset sequence frame 250 and 278) 75
Figure 4.19: RGB image b) Reconstruction error c) Dynamic label image 78
Figure 5.1: ICL-NUIM living room dataset containing chair, table and other type common object. 83
Figure 5.2: Dense 3D model of the scene obtained by our proposed methodology 84
Figure 5.3: Reconstructions of the objects in living room - 2 scene including error value (blue 0.025 m, green 0.045 m) 84
Figure 5.4: Frequency of residual of the reconstruction error. 85
Figure 5.5: ATE RMSE of synthetic ICL-NUIM dataset 85
Figure 5.6: CORBS Desk1 dataset containing desk, computer, book, monitor and other types common object of everyday. 86
Figure 5.7: ATE RMSE of synthetic CoRBS dataset (Desk sequence) 88
Figure 5.8: Dense 3D model of the scene obtained by our proposed methodology 88
Figure 5.9: Dense 3D model of the scene obtained by our proposed methodology (Close up view) 89
Figure 5.10: ATE / RPE of TUM fr3/walking static sequence 96
Figure 5.11: Mesh of TUM fr3/walking static sequence 96
Figure 5.12: ATE / RPE of TUM fr3/walking halfsphere sequence 97
Figure 5.13: Mesh of TUM fr3/walking halfsphere sequence 97
Figure 5.14: Scene reconstruction of fr3/walking xyz sequence 98
Figure 5.15: Comparison of estimated trajectories of of TUM fr3/walking xyz sequence 102
Figure 5.16: Comparison of estimated trajectories of of TUM fr3/walking xyz sequence (continued) 103
Figure 5.17: Relative translational error (walking-xyz) of our method 104

Figure 5.18: Bonn RGB-D Dynamic dataset containing desk, chair and moving obstructing box 105
Figure 5.19: Mesh of BONN moving obstructing box sequence 106
Figure 5.20: Estimated trajectories of of Volume Deform *boxing* and *sunflower* sequences 107
Figure 5.21: RGB-D image pair and 3D dense model of VolumeDeform *sunflower* sequence obtained by our proposed method 108
Figure 5.22:RGB-D image pair and 3D dense model of VolumeDeform *boxing* sequence obtained by our proposed metho 109
Figure 5.23: RGB-D image and final mesh of the CVSSP "dog" sequence 109
Figure 5.24: RGB-D image and final mesh of the "outdoor-1" sequence 110
Figure 5.25: outdoor-1 sequence grid fence mapping result 111
Figure 5.26: RGB-D image and final mesh of the "*outdoor*-2 " sequence a) DUDMap b) Elastic Fusion 112
Figure 6.1: Main effects plot: Variation of Absolute Trajectory Error to voxel size, regularization constant, color intensity contribution with respect to the depth, Huber constant 115
Figure 6.2: Interaction plot: Variation of Absolute Trajectory Error to voxel size, regularization constant, color intensity contribution with respect to the depth, Huber constant 118
Figure 6.3: Change of Translational ATE (RMS Error [m]) with respect to weight (regularization constant, Huber constant and voxel size is constant) 119
Figure 6.4: Quadratic model of the sensitivity analysis 121

LIST OF ABBREVIATIONS

ABBREVIATIONS

SDF	:	Signed Distance Function
ATE	:	Absolute Trajectory Error
ORB	:	Oriented FAST and Rotated BRIEF
FAST	:	Features from Accelerated Segment Test
BRIEF	:	Binary Robust Independent Elementary Features
PTAM	:	Parallel Tracking and Mapping
MCPTAM	:	Multi-Camera Parallel Tracking and Mapping
RTABMap	:	Real-Time Appearance-Based Mapping
RANSAC	:	RANdom SAmple Consensus
SIFT	:	Scale-Invariant Feature Transform
GPU	:	Graphics Processing Unit

LIST OF SYMBOLS

SYMBOLS

c_{α} : $\cos(\alpha)$
s_{α} : $\sin(\alpha)$
$[.]_{\times}$: cross product operator
$[.]^{\mathrm{Y}}$: inverse of the cross product
$R_x(\alpha)$: rotation matrix about the x-axis with angles α
R_{BA} : rotation matrix of frame A to frame B
T_A^B : homogenous transformation matrix from frame A to frame B
$\dot{q}(t)$: velocity of the point
$tr(R)$: trace of matrix R
ξ : twist coordinates
υ_n : linear velocity
ω_n : angular velocity
a_k : step size at iteration k
d_k : the direction to move along at iteration k
Q : symmetric positive definite matrix
P : real symmetric positive definite matrix
$\lambda_{min}(P)$: smallest eigenvalue of the P,
$\lambda_{max}(P)$: largest eigenvalue of the P,
i_{max} : maximum iteration number
X_{ij} : Bernoulli random variables taking values in {0, 1}
$L(\theta)$: likelihood function
η : maximum likelihood estimate
f_x, f_y : camera focal length (in pixels)
c_x, c_y : the pixel coordinate of the image center
δ_x, δ_y : projected pixel coordinate
K : Intrinsic camera calibration matrix
κ_n : radial distortion coefficient

ρ_n : tangential distortion coefficient
ψ : signed distance function.
ρ : unit-norm vector
$\eta(x)$: surface normal at x
d_i : all zeros vector except component denoted by the subscript
$\psi_t(x)$: truncated signed distance function
z_{obs} : observed depth measurement
υ_w : Color intensity contribution with respect to the depth
$\nabla_x \psi(\mathrm{x})$: gradient of signed distance function
$J_p(\xi)$: Jacobian of pose error with respect to ξ
λ : non-negative regularization factor
ρ_h : Huber constant
ρ_t : Tukey’s constant
I_s : instantaneous image of source
I_m : instantaneous image of generated model
e_c : error in color map
e_p : error in the pose registration
$h(d)$: histogram of the image
$L(D|l)$: log likelihood of the two-component mixture
$D(u)$: observed depth image
w_c : Gaussian functions for spatial weighting
w_s : Gaussian functions for range weighting
σ_s : standard deviation for spatial weighting function
σ_c : standard deviation for range weighting function
$N(u_k)$: neighborhood of pixel location u
W : normalizing factor
τ : voxel size

CHAPTER 1

INTRODUCTION

1.1 Statement of the Problem

Simultaneous Localization and Mapping (SLAM) is known as estimating the pose of a robot and the map of the scene synchronously by using noisy sensor measurements obtained from partially known environments and noisy distance sensors. In other words in SLAM, agents try to find a solution simultaneously to the questions of “What does the scene resemble?”, and “Where am I in the scene?” by using noisy range sensors.

In many SLAM approaches, it is required for an agent to run autonomously for a long time, travelling long contoured distance away from its starting position. Furthermore, some applications require 3D mapping such as underwater exploration applications, where, the agent should collect the 3D data of ocean-floors, which is an example of large 3D environment to be mapped while localizing the event. Moreover, an accurate 3D reconstruction or exploration could be highly beneficial and critical such as in urban planning of large cities with high-rises.

Underwater explorations are example of SLAM requiring a robot to run autonomously for a long time challenged by drag currents and obstacles. In addition, SLAM application in city exploration requires construction of an accurate 3D map of the entire city, crowding with close high-rises of different heights and cross sectional architects. In the literature, there are several papers for SLAM approaches to large unstructured environment; however, they are usually tested on standard datasets, which can be classified as structured environments. In addition, static environment assumption generally used in researches which become invalid with failing algorithms for unstructured dynamic environments such as undersea flooring.

For example, in [58][59], a stereo camera is used to extract a dense 3D mesh of the world and scale-invariant features are used to estimate the motion of the camera. This algorithm seems to work well in unstructured 6-DOF environment, however SIFT features should be validated in the environment in order to for achieving high efficiency of the algorithm. This technique does not utilize a dense data in the registration process therefore; error can be accumulated leading to loop closing errors while estimating the pose. Loop closure is a critical must for the localization of the agent that generally revisits previous location during exploration and mapping.

Application of SLAM algorithm in real life such as underwater explorations should address the problem of dynamic environments because the robot which operate encounter a non-static world while navigating. The environment is changing, objects are moving. Moving object causes confusion on most algorithms because of static world assumptions. Dynamic objects causes either degradation of the registration accuracy or lead to elongated object in the resulting map generating mapping ghosts. SLAM in highly dynamic environments have not been addressed fully in the literature. In short, it is not possible to find a fully encompassing framework that solves SLAM in the presence of dynamic elements in environments.

SLAM algorithms uses the sensor data in order to produce a consistent map of the environment. However, some difficulties arise in associating sensor data with map elements because of noise, sparse data and imprecision. If the sensor is not noise-free and precise, probability of generating incorrect map increases: sensors such as laser scanner produce ordered data set with minimum level clutter, while, vision sensors produces cluttered huge amount of data, which results in incorrect association. In addition, vision sensors have a limited resolution therefore; precision loss must be taken into more account, because sparse data limit point-matching process because of low level feature.

1.2 Objective and Goals of the Study

Simultaneous Localization and Mapping is to produce a consistent map of environment and to estimate the pose in the map by using noisy range sensor measurements. SLAM has been widely studied subject by researchers in the field of robotics. After the appearance of Kinect, there are many solutions, which fuse the color image and depth map. Visual SLAM produce a sparse solution by relying on points matching, whereas; direct methods can produce a dense reconstruction by minimization of the photometric error. However, none of the above methods addresses the problem of dynamic objects in the environment.

Conventional approaches in mapping assume static environment. Although static assumptions may be valid in a small region, change is inevitable when dynamic elements exist or large-scale mapping is necessary. By classifying dynamic content as outliers, a small fraction can be managed. However, SLAM problem in highly dynamic scenes is still not solved completely because there is no suggested framework found in the literature.

Another biggest difficulty in robot navigation is unstructured environments. In unstructured environments, it is not east to find discrete geometries because of noisy edge or plane. Significant research has been carried out for unstructured environments especially in the field of autonomous navigation, and a number of effective approaches have been developed using laser range finder. However, there is no effective RGB-D SLAM method for real-world unstructured and dynamic environments.

In real life, SLAM frameworks should address the problem of dynamic scene without consuming lots of computational power. Employing semantic segmentation or object classifier are time consuming, therefore, we avoid from using such techniques. Therefore, our aim is to develop a SLAM algorithm which

- can generate the dense model / mesh of environment
- can handle high dynamics in environment,
- is robust to dynamic element without requiring any type of object tracking,
- does not need any classifier or semantic information,
- is real-time operation compatible,
- does not require sophisticated sensor.

Considering these practical constraint, we propose a RGB-D based SLAM method because RGB-D cameras are specific type of sensors which can augment the image with depth information and this type of sensor has been extensively used in computer vision and computer graphics problems because of its low cost.

In this study, we propose *DUDMap*: *D*ense, *U*nstructured and *D*ynamic *Map*ping. Our approach requires neither explicit object tracking nor object classifier in contrast to recent approaches discussed in [22] and [28], which makes it robust to any type of moving object. Furthermore, we assume a dynamic environment consisting of static and dynamic classes having generalized Gaussian distribution in order to detect dynamics.

1.3 Methodology

In order to have a dense map, we reconstruct scene geometry using Signed Distance Function (SDF) instead of surfels because SDF allows us to create a dense final mesh easily. Moreover, such representation is useful in robotic applications because it defines the distance to surface.

The main contribution of this thesis is a novel and an effective SDF based SLAM algorithm that is resistant to dynamics. In addition;

- We directly use the SDF values for pose estimation where objective function is based on probabilistic camera model.
- We identify the dynamics by using image registration residual combining with Gaussian Mixture Model (GMM).
- We eliminate repeated views and use only consistent data for decreasing the required computational power.
- The number of dynamic objects does not limit our approach because we do not employ any type of moving object detection and tracking.
- Our method generates a final intense 3D mesh without using semantic information or object classifier.

1.4 Evaluations of the Approach

Our proposed method is able to operate in dynamic environments without requiring any dynamic object detection and tracking. Our experiments support our main claims, which are:

- Robustness to dynamic elements regardless of their quantity and speed of change in the environment,
- That the approach requires no explicit object tracking, and no object classifier
- That the approach generates a consistent a dense model of the environment.

We compare our method with other state of art systems using TUM dataset [1], together with other high dynamic dataset including Bonn, VolumeDeform [30], CVSSP RGB-D dataset [34] (used with permission), which are publicly available, showing the superior performance of our approach.

TUM repository has a large dataset consisting of RGB-D data and corresponding ground-truth data and it is a commonly used evaluation method for benchmarking of visual odometry and visual SLAM systems. Bonn RGB-D dataset has highly dynamic sequence with different scenario such as crowded scene, people tracking and obstructing box. Moreover, this dataset is also compatible for TUM RGB-D dataset. VolumeDeform dataset aims to reconstruct of dynamic object whose RGB-D data are obtained by using handheld consumer grade RGB-D sensor. It is generally used for evaluation of non-rigid 3D reconstruction.

In order to evaluate the outdoor performance of our method, we use commercially available ZED camera for map generation and dynamic filtering. Experiments illustrates that our method produce consistent result both in indoor and outdoor applications. These are demonstrations of real world unstructured dynamic environments of our approach.

1.5 Outline of the Thesis

The rest of this thesis is organized as follows.

Chapter 2 reviews state-of the-art visual SLAM methods that attack the problem of dynamic environments.

Chapter 3 gives the mathematical preliminaries, RGB-D sensors.

Chapter 4 is devoted to the overall methodological structure of our system, by giving details about proposed approaches for local key frame extraction, dynamic removal, 3D map representation and 3D volume visualization.

Chapter 5 focuses on experiments conducted and provides evaluation results comparing our method against other state-of-the-art methods.

Chapter 6 presents the evaluation results that analyze the sensitivity of the method to parameter changes whereas Chapter 7 provides concluding remarks and future works.

CHAPTER 2

LITERATURE REVIEW

State of art SLAM methods use probabilistic approaches by taking into account uncertainty information in sensor data. Common representation are classified as grid, raw data and feature based methods [60].

Grid based techniques represent the environments with grids representing the probability of being occupied or not occupied, during robot navigation. For example, Burgard and Thrun [61] propose a grid based mapping method with an agent with one degree of freedom sonar scanner for constructing 2D grid maps of indoor environments. In order to localize the agent itself in a large environment, it must use complex cross correlation search algorithms, which usually generate more than one location as ghost presence. In addition, using maximum likelihood search algorithm can converge to false local optima. These two problems can be solved by using a particle-based localization in the map. In addition to this approach, a multi-agent grid based SLAM implementation is proposed in [94] and mapping of an indoor environment with six mobile robots is presented in this work. The most important flaws of grid-based mapping SLAM is cycle detection when revisiting a previously present location or loop closing. For a large cycle, the minimal search-space may become huge for real-time cross-correlation, which needs high computational time on the other hand; using a tight search space may lead the agent to fail detecting the cycles or miss loop closure. Furthermore, given a large search space for scan matching, the possibility of multiple correlation modes is high [95].

Raw data or featureless techniques can be applied to outdoor environments where feature extraction is a difficult task. Iterative closest point (ICP) algorithm, commonly use featureless registration approaches, as introduced in [62]. ICP algorithm is used extensively in 3D object reconstruction. This algorithm takes two input 3D scans and tries to build a transformation matrix iteratively by minimizing the distance between two scans. Data association is the problem of associating observations from sensors to a particular area of the map being estimated within the SLAM algorithm.

A common way of matching multiple sets of 3D points is the iterative closest point algorithm. In ICP, the main assumption is overlapping point sets. The distance between the measurement and model are minimized in a least square sense by determining the best homogenous transformation. The found solution for rotation and translation are considered as transformation between two point clouds.

In ICP algorithm, it is assumed that the point sets are overlapping. The distance between closest points are minimized by determining the “best” (in least squares sense) rotation matrix and translation vector. The final transformation are considered to be the registration transformation between the two point clouds that align them properly. ICP handles the data association problem by finding the corresponding points in raw data, however, it is relatively slow because of its iterative nature [63].

Feature based methods represent the environment by abstract geometric features, therefore such methods require a data processing algorithm for extracting those features and associating features with landmarks in the map of the environment [60]. The pioneering work for feature based method, proposed by Smith and Cheeseman [64] employs an Extended Kalman Filter (EKF) based SLAM method for estimating the posterior distribution over robot position along with the positions of landmarks in a disaster environment. An initial work establishes the fact that there is a statistical relationship between each predetermined landmark location and the observation. EKF linearizes the nonlinear model and applies the conventional KF in order to generate an estimated state instead of working with the real state. EKF is used considerably for the solution of SLAM problems in the literature [65][66]. However, EKF fails to guarantee convergence because of propagating mean and covariance due to the linear approximations of the nonlinear transformation [67]. Moreover, feature extraction may not be possible for initially unknown and complex environments. Hence, in these environments and under nonlinearities, EKF based SLAM solutions may not be feasible.

Feature based approach takes advantage of reducing the search space, which removes the necessity of finding for corresponding pairs. This leads to irrelevant points not affecting the 3D registration process. In addition, feature based techniques do not need a good initial estimate or alignment of the scans. However, these methods suffer from a high computational complexity.

2.1 Visual SLAM

Parallel Tracking and Mapping (PTAM) is the first visual SLAM approach capable of handling thousands of 3D points in real-time [68]. This approach extracts features from images using a corner detection algorithm and then matching these features in different frames to obtain 3D positions. It also proposes a method to efficiently generate and store a 3D map by saving important frames into memory. However, this method focuses on accurate and fast mapping in small environments; therefore, it is not sufficient to manage large-scale maps.

Following PTAM, there are several methods similar to PTAM, which use structure from motion method. Structure from motion (SfM) is the process of estimating the 3-D structure of a scene from a set of 2-D images based on fundamental matrix, which is the relation of two images from different views. In [69], an algorithm is proposed which splits mapping and tracking in order to detect loop closures and handle large maps. In addition to SfM methods, ORB-SLAM [3] is proposed by using ORB descriptor instead of FAST feature in order to improve the accuracy of the existing PTAM scheme. Although ORB descriptor, which is utilized in ORB-SLAM2, may outperformed by other existing features it was found to be not as robust as expected. There exists always inevitable erroneous correspondences of features. Besides, the location of features are not exact due to discretization artifact. These are important reasons for ORB-SLAM's randomness and less accurate estimate. Therefore, these methods suffer from scaling drift over time.

Furthermore, other types of visual SLAM methods are based on stereo and RGB-D cameras or with visual-inertial odometry are proposed.

maplab [70] and VINS-Mono [71] have recently been introduced as inertial aided SLAM system. These are graph-based SLAM systems that use an IMU and a camera. They can generate visual maps for localization. maplab records the data during an open loop phase using only visual-inertial odometry, then loop closure detection, graph optimization and dense map reconstruction are all done offline.

All method mentioned above assume that either the camera is never obstructed or images have enough visual features to allow tracking. In real life these assumption cannot be satisfied: For example, the camera of an autonomous robot can be fully obstructed from people passing by or it can encounter a surface without visual features during motion. The following visual SLAM approaches are designed to be more robust for these cases; however, they still cannot solve the dynamic environment problem:

- MCPTAM [79] uses multiple cameras in order to increase the field of view of the system. If the visual feature can be tracked at least by one camera, then this proposed system can perform position tracking.

- RGBDSLAMv2 [80] uses an external odometry as motion estimation. In addition, in order to have a more robust odometry, built-in ROS packages like robot localization [81] can then be augmented for sensor fusion with an extended Kalman filter operating on multiple odometry sources.

RGBDSLAMv2 generates a 3D occupancy grid map (OctoMap [82]) and a dense point cloud of the environment.

- RTAB-Map, Real-Time Appearance-Based Mapping [83][84], has been introduced in order to limit the operation time during loop closure detection. In addition, it has a memory management module making it suitable for larger scale environment. Since this method can detect the loop closure within a fixed time limit, it is capable of online mapping for a longer period of runtime.

ORB-SLAM2 [3] (latest version ORB-SLAM3 [37]), S-PTAM [4] and RTAB-Map [5] are best state of art feature based visual SLAM approaches working in static environments. In order to increase the performance of such feature-based method in dynamic environment, dynamic objects are considered generally as spurious data, and are removed as outliers using RANdom SAmple Consensus (RANSAC) and robust cost function. More recently, ProSLAM [73] has been released to provide a comprehensive open source package using well know visual SLAM techniques. In generated graph, the nodes store odometry information for each map place. They also contain the RGB images, depth information and visual words. In generated graph, the edge stores the rigid 3D transformation between nodes. For detection of previously visited places, “bag of words” method is applied. The “bag-of words” method represents each image by visual words taken from a vocabulary which can be constructed either offline by using a training dataset or online and local feature descriptors are used for forming the visual words. For ORB-SLAM2 and S-PTAM, when a loop closure is detected using a special bag of words method introduced in DBoW2 [74] and generated map is optimized using bundle adjustment. Bundle adjustment describes the sum of errors between the measured pixel coordinates and the re-projected pixel coordinates. A separate thread runs for graph optimization after loop closure. In this way, it is aimed to have a higher camera tracking frame rate performance. ProSLAM detects the loop closures by direct comparison of descriptors in the map. It does not use the bag-of-words approach. For all these approaches, the time for loop closure and graph optimization increases as the map grows, which can make loop closure correction realized with a significant delay following the detection. In addition, these approaches maintain a sparse feature map because of the bundle adjustment. The sparse feature map is not visually informative and cannot directly be used for tasks such as collision-free motion planning; however, sparse SLAM is often fused with other dense methods to improve robustness, which needs more computational power.

On the other hand, targeted attempts are still being made to increase performance in dynamic scenes. For instance, DVO-SLAM [6] use photometric and depth errors instead of visual features. The joint visual odometry scene flow (VO-SF) [7] proposes an efficient solution to estimate camera motion. However, odometry-based methods either cannot recover from inaccurate image registration or lacks loop closure detection approach independent of pose estimate. In addition, RGBiD-SLAM [76], does not prefer local visual features to estimate motion, it uses photometric and depth errors over all pixels of the RGB-D images. This method performs the integration of the tracked frames into key frames, which helps to achieve a greater accuracy in the computation of relative camera transformation during loop closure. However, mean computational cost of the processes involved in the back-end pipeline (segmentation, tracking, loop detection and graph optimization) is above 600 milliseconds which is not suitable for real time operation.

SDFs have long been studied in order to represent the 3D volumes in computer graphics [24][31][32]. Newcombe et al. [24] proposed the SDF based RGB-D mapping by generating precise maps in static environments. ElasticFusion (EF) [9] is another method based on SDF, which can work in small scenarios. CoFusion [12] (CF) is a contemporary method for reconstructing several moving objects however it works with slow camera motions only and its performance deteriorates significantly with increasing camera speed. StaticFusion (SF) [13] simultaneously estimates the camera motion together with dynamic segmentation of image. However, it works only sequences without having high dynamics at the beginning. Furthermore, Kintinuous [77] and BundleFusion [78] uses truncated signed distance field (TSDF) volume for RGB-D cameras. They are capable of reconstruct the environment online with very appealing surfel-based maps, however, they require any type of recently introduced powerful Nvidia GPU and this is not a low-cost solution for dynamic SLAM. For BundleFusion, global dense optimization time on loop closure detection increases according to the size of the environment. Palazzolo [14] proposes Refusion (RF) where dynamics detection is done by using the residuals obtained from the registration on signed distance function. This approach can create a consistent mesh of the environment however highly dynamical change deteriorate mapping performance.

Some methods use consistency of motion to confirm tracked points where dynamic objects are segmented generally as spurious data since they conflict with the motion consistency of background over consecutive frames. For instance, Wang and Huang [15] segment dynamic objects using RGB optical flow. Nevertheless, the algorithm is still not robust enough for TUM high dynamic scenarios. Kim et al. [16] propose to use difference between depth images in order to eliminate the dynamics in the scene. However, this algorithm requires an optimized background estimator suitable for parallel processing. Azartash [17] uses the image segmentation for discriminating the moving region from the static background. Experimental results show that the accuracy remains almost same in low dynamic scenarios but in high dynamic accuracy deteriorates. Tan [18] uses an adaptive RANSAC for removing outliers. This method can work in dynamic situations with limited number of slowly moving object.

Other methods use classifiers to identify the dynamic objects. Kitt [19] combines the motion estimation with object detection; however, this method requires a classifier, which make this method inapplicable to online explorations. Bescos [20] proposes

DynaSLAM, which combines a prior learning by Mask R-CNN [38] and Multi-View Geometry to segment dynamic content. Multi-view geometry consists of region growth algorithm, which makes unsuitable for real-time operation even running on NVIDIA Titan GPU. Mask Fusion [36] also uses Mask R-CNN for semantic segmentation. DS SLAM [21], RDS-SLAM [39], Semantic SLAM [40] are other semantic based algorithms which use the SegNet [23]. Pose Fusion [41], implemented on Elastic Fusion, uses Open Pose CNN [42] for human pose detection, which limits this method in the non-human dynamic object scenes. Flow Fusion [43] uses optical flow residuals with PWC-Net [44] for dynamic and static human objects. However, such approaches are relying heavily on prior training methods. Therefore, if unlearned dynamic occur in camera view, estimation results are bigger. Furthermore, learning based semantic information is time- consuming with heavy computational burden.

In our work, we reconstruct our scene geometry using Signed Distance Function (SDF) instead of surfels in contrast to ElasticFusion and StaticFusion and therefore we can directly generate the mesh of the environment using such representation without using object tracking and object classifier. Moreover, number of dynamic objects or their speeds do not limit our approach.

CHAPTER 3

PRELIMINARIES

In order to localize the sensor and to generate consistent mesh of the environment, we propose a Simultaneous Localization and Mapping method. In SLAM system, pose estimation problem is usually formulated as nonlinear optimization or nonlinear least square problem where sum of squared errors are minimized. Therefore, in this section, we try to provide an overview of nonlinear least square optimization with commonly used methods.

3.1 Non-Linear Least-Squares (NLS) Optimization

For a given vector function $f: R^m \rightarrow R^n$ with $\geq m$, the objective is to minimize $\|f(x)\|$. In particular, our aim is to find

$$x^* = argmin_{\boldsymbol{x}} F(x) \tag{3.1}$$

where

$$F(x) = \sum_{i=1}^{m} \left(f_i(x)\right)^2 = \frac{1}{2}\|f(x)\|^2 = \frac{1}{2} f(x)^T f(x) \tag{3.2}$$

n residuals are defined using the vector function as $f_i(x) : R^m \rightarrow R$ and the residuals are usually non-linear and non-convex functions.

The general form of iterations in order to solve this optimization problem is in the form of

$$x_{k+1} = x_k + a_k d_k \tag{3.3}$$

x_k and x_{k+1} are current iteration and next iteration. a_k is the step size at iteration k and d_k is the direction to move along at iteration, which is a descent approach.

3.1.1 Descent Methods

Definition: **(Descent direction)** For a given $x \in R^n$, a direction $d \in R^n$ is called a descent direction if $\exists\ a > 0$ such that

$$f(x + ad) < f(x). \tag{3.4}$$

Let $f(x + ad) = g(a)$, then using Taylor's expansion

$$g(a) = g(0) + g'(0)a + O(a) \tag{3.5}$$

$$f(x + ad) = f(x) + a\nabla f(x)^T d + O(a) \tag{3.6}$$

Since $\lim_{a\to 0} \frac{|O(a)|}{a} = 0$, $\exists\ a > 0$, $\frac{|O(a)|}{a} < |\nabla f(x)^T d|$, then $f(x + ad) - f(x) < 0$.

Hence, d is a descent direction.

For a given $x \in R^n$, let $g(a) = f(x + a\frac{d}{\|d\|})$, the rate of change of f is

$$g' = \frac{\prec \nabla f(x), d \succ}{\|d\|} \tag{3.7}$$

By the Cauchy-Schwarz inequality

$$-\frac{1}{\|d\|} . \|\nabla f(x)\| . \|d\| \leq \frac{\prec \nabla f(x), d \succ}{\|d\|} \leq \frac{1}{\|d\|} . \|\nabla f(x)\| . \|d\| \tag{3.8}$$

Therefore, the direction with the maximum rate of decrease is along -$\nabla f(x)$.

Let us go back to general form of our iterative algorithm with $d = -\nabla f(x)$, then we get

$$x_{k+1} = x_k - a_k \nabla f(x_k) \tag{3.9}$$

If $\{x_k\}$ is the sequence generated by the descent algorithm, then,

$$x_{k+1} = x_k - a_k \nabla f(x_k) \tag{3.10}$$

$$x_{k+2} = x_{k+1} - a_{k+1} \nabla f(x_{k+1}) \tag{3.11}$$

$$\prec x_{k+1} - x_k, x_{k+2} - x_{k+1} \succ = a_k a_{k+1} \prec \nabla f(x_k), \nabla f(x_{k+1}) \succ \tag{3.12}$$

Let $g_k(a) = f(x_k - a\nabla f(x_k))$, a_k minimizes this function if and only if

$$\frac{dg_k}{da} a_k = 0 \tag{3.13}$$

However, using chain rule,

$$\frac{dg_k}{da} a_k = \prec -\nabla f(x_k), \nabla f(x_k - a_k \nabla f(x_k) \ \succ = \prec -\nabla f(x_k), \nabla f(x_{k+1}) \ \succ \tag{3.14}$$

Therefore,

$$\prec x_{k+1} - x_k, x_{k+2} - x_{k+1} \succ = 0 \tag{3.15}$$

This equation implies that $x_{k+2} - x_{k+1}$ is orthogonal to $x_{k+1} - x_k$. Observe that the method of steepest descent moves in orthogonal steps.

Let $A \in R^{nxn}$ be symmetric matrix with real entries and let $x \in R^{nx1}$ (column vector),

$$Q = x^T A x = [x_1 \quad \cdots \quad x_n] \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}. \tag{3.16}$$

Then, $Q = x^T A x$ is said to be a quadratic form and the quadratic form is positive definite $Q > 0$ when $x \neq 0$. Using symmetric positive definite matrix Q, we can form a quadratic function in the form of

$$f(x) = \frac{1}{2} x^T Q x - b^T x \tag{3.17}$$

Then, the derivative of $x^T A x$

$$D(x^T Q x) = x^T (Q + Q^T) = 2x^T Q \tag{3.18}$$

$$D(b^T x) = x^T (Q + Q^T) = b^T \tag{3.19}$$

If a quadratic form $x^T A x$ is given with $A^T \neq A$, then because the transposition of a scalar equals itself,

$$(x^T A x)^T = x^T A^T x = x^T A x \tag{3.20}$$

we obtain

$$x^T A x = \frac{1}{2} x^T A x + x^T A^T x \tag{3.21}$$

$$x^T A x = \frac{1}{2} x^T (A+A^T) x \tag{3.22}$$

Since $(A+A^T)^T = Q^T = A+A^T = Q$

$$x^T A x = \frac{1}{2} x^T (A+A^T) x \triangleq \frac{1}{2} x^T (Q) x \tag{3.23}$$

Let us go back to general form of our iterative algorithm with $d = -\nabla f(x)$, then we get

$$x_{k+1} = x_k - a_k \nabla f(x_k) \tag{3.24}$$

Our aim to find a a_k to minimize $f(x_{k+1})$ where $f(x) = \frac{1}{2} x^T Q x - b^T x$.

Let

$$g(a) = f\big(x_k - a\nabla f(x_k)\big) \tag{3.25}$$

$$g(a) = \frac{1}{2} f\big(x_k - a\nabla f(x_k)\big)^T Q f\big(x_k - a\nabla f(x_k)\big) - b^T\big(x_k - a\nabla f(x_k)\big) \tag{3.26}$$

$g(a)$ is quadratic and $g(a) = \alpha a^2 + \beta a + \gamma$ and $argmin\ g(a) = -\frac{\gamma}{2\alpha}$

where

$$\alpha = \frac{1}{2} \nabla f^T\big(x_k - Q\nabla f(x_k)\big) \tag{3.27}$$

$$\gamma = (b^T - x_k{}^T Q)\nabla f(x_k) = -\nabla f^T(x_k)\nabla f(x_k) \tag{3.28}$$

Finally,

$$a_k = \frac{\nabla f^T(x_k)\nabla f(x_k)}{\nabla f^T(x_k) Q \nabla f(x_k)} \tag{3.29}$$

The descent algorithm (known as steepest descent) with exact line search for quadratic functions becomes

$$x_{k+1} = x_k - \frac{\nabla f^T(x_k)\nabla f(x_k)}{\nabla f^T(x_k)Q\nabla f(x_k)}\nabla f(x_k) \tag{3.30}$$

$$\nabla f(x_k) = Qx_k - b \tag{3.31}$$

Rayleigh's Inequalities. $P \in R^{nxn}$ with real symmetric positive definite matrix, Then

$$\lambda_{min}(P)\|x\|^2 \leq x^T P x \leq \lambda_{max}(P)\|x\|^2 \tag{3.32}$$

Where $\lambda_{min}(P)$ and $\lambda_{max}(P)$ are respectively the smallest and largest eigenvalue of the P.

Using Rayleigh's inequality

$$\lambda_{min}(Q)\nabla f^T(x_k)\nabla f(x_k) \leq \nabla f^T(x_k)Q\nabla f(x_k) \leq \lambda_{max}(Q)\nabla f^T(x_k)\nabla f(x_k) \tag{3.33}$$

$$\nabla f^T(x_k)Q^{-1}\nabla f(x_k) \leq \frac{1}{\lambda_{min}}\nabla f^T(x_k)\nabla f(x_k) \tag{3.34}$$

Finally , we get,

$$0 < a \leq \frac{2}{\lambda_{max}(Q)} \tag{3.35}$$

First order necessary condition. If x^* is an unconstrained local minimizer of a differentiable function $R^n \to R$, then we must have $\nabla f(x^*) = 0$.

Proof: Consider $y \in R^n$. If we define $g(a) = f(x^* + ay)$ and $g: R^n \to R$. Using chain rule,

$$\frac{dg}{da}(a) = y^T \nabla f(x^* + ay) \tag{3.36}$$

$$\frac{dg}{da}(0) = y^T \nabla f(x^*) \tag{3.37}$$

Using the definition of limit

$$\frac{dg}{da}(0) = \lim_{a \to 0} \frac{f(x^* + ay) - f(x^*)}{a} \geq 0 \tag{3.38}$$

Because the local optimality of x^*, $f(x^* + ay) \geq f(x^*)$. Finally we get,

$$y^T \nabla f(x^*) \geq 0. \tag{3.39}$$

Since y is arbitrary, if $-y$ is used, then

$$-y^T \nabla f(x^*) \geq 0. \tag{3.40}$$

$$y^T \nabla f(x^*) - y^T \nabla f(x^*) \geq 0 \tag{3.41}$$

This inequality implies that $\nabla f(x^*) = 0$

3.1.1.1 Newton Methods

If Taylor's expansion is used around the current estimate x_k, we get,

$$f(x) \approx f(x_k) + \nabla f^T(x_k)(x - x_k) + \frac{1}{2}(x - x_k)\nabla^2 f^T(x - x_k) \triangleq q(x) \quad (3.42)$$

First order necessary condition implies that $q(x^*) = 0$. Hence,

$$\nabla f(x_k) + \nabla^2 f(x_k)x + \nabla^2 f(x_k)x_k = 0 \quad (3.43)$$

$$x = x_k - [\nabla^2 f(x_k)]^{-1}\nabla f(x_k) \quad (3.44)$$

If $\nabla^2 f(x_k) \geq 0$, then $f(x_k)$ is convex and hence our stationary point is a global optimum. Newton's method picks this point as the next iterate

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1}\nabla f(x_k) \quad (3.45)$$

Let us show that $-[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)$ is a descent direction. Define a function $h(a)$

$$h(a) = f(x - a[\nabla^2 f(x)]^{-1}\nabla f(x)) \quad (3.46)$$

$$h'(a) = [-([\nabla^2 f(x)]^{-1}\nabla f(x))]^T \nabla f(x - a[\nabla^2 f(x)]^{-1}\nabla f(x)) \quad (3.47)$$

$$h'(0) = -\nabla f^T(x)[\nabla^2 f(x)]^{-1}\nabla f(x) \quad (3.48)$$

If $\nabla f(x) \neq 0$ and $[\nabla^2 f(x)]^{-1} > 0$, then $h'(0) < 0$. This result shows that $-[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)$ is in the descent direction.

3.1.2 Levenberg-Marquardt Modification

Gauss-Newton method solves the optimization problem quite fast because it has quadratic convergence to local minimum. However, in some circumstances it can show a poor performance especially under poor initial parameter selection. Gradient Descent (GD), on the other hand, shows slow linear convergence, but always decreases the function for a sufficiently small step size.

If $[\nabla^2 f(x)]^{-1} \geq 0$ or $\nabla^2 f(x)$ is singular, the Newton's direction may not be a descent direction or may not be well defined. The basic idea is to make $\nabla^2 f(x)$ positive defi

In order to overcome the problem of both Gauss-Newton and Gradient Descent, the Levenberg-Marquardt (LM) algorithm can be used. This method smoothly switches between both methods through an adaptive parameter.

Lemma 2. Let A and nxn matrix with eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ and let $\mu \in R$. Then eigenvalues of $A + \mu I$ are $\lambda_1 + \mu, \lambda_2 + \mu, \dots, \lambda_n + \mu$

Proof. Let λ_i be an eigenvalue of A with eigenvector ϑ_i

$$A\vartheta_i = \lambda_i \vartheta_i \tag{3.49}$$

$$(A + \mu I)\vartheta_i = A\vartheta_i + \mu\vartheta_i = \lambda_i\vartheta_i + \mu\vartheta_i = (\lambda_i + \mu)\vartheta_i \tag{3.50}$$

Therefore $\lambda_i + \mu$ is an eigenvalue of $A + \mu I$.

Levenberg-Marquardt modification is finally in the form of

$$x_{k+1} = x_k - [\nabla^2 f(x_k) + \mu_k I]^{-1} \nabla f(x_k). \tag{3.51}$$

For large values of μ we get short step in the steepest descent direction and for small values of μ, we can get (almost) quadratic final convergence (regular Newton method). Therefore, the damping parameter influences both the direction and the size of the step, and choice of initial μ value should be related to $A_0 = \nabla^2 f(x_k)$ as

$$\mu = max_i(A_{0_{ii}}) \tag{3.52}$$

After having completed the rule and search direction, next step is to choose the initial parameters. The initial point is usually selected randomly. The final step is to choose the stopping criteria if we have found a local minima is found.

The stopping criteria can be selected as

$\lvert x_{k+1} - x_k \rvert < \epsilon)$	Difference between iteration is small.
$\lVert \nabla f(x_k) \rVert \leq \epsilon$	$\epsilon > 0$ is a small prescribed threshold
$\lvert x_{k+1} - x_k \rvert \leq \epsilon(\lVert x \rVert + \epsilon)$	Gradual change from relative step size ϵ
$\lvert f(x_{k+1}) - f(x_k) \rvert < \epsilon)$	Improvements in function value are saturating
$i < i_{max}$	Safeguard against an infinite loop

3.2 Expectation–maximization (EM) algorithm

If we have two coins: Coin 1 and Coin 2, and each has its own probability of seeing "Head" on any one flip. Now, select a coin at random and flip that one coin m times and repeat this process n times.

$$\begin{array}{cccc} X_{11} & \cdots & X_{1m} & Y_1 \\ \vdots & \ddots & \vdots & \vdots \\ X_{n1} & \cdots & X_{nm} & Y_n \end{array}$$

The X_{ij} are Bernoulli random variables taking values in {0, 1} where

$$X_{ij} = \begin{cases} 1 & if\ jth\ flip\ for\ ith\ coin\ is\ Head \\ 0 & if\ jth\ flip\ for\ ith\ coin\ is\ Tail \end{cases} \tag{3.53}$$

Y_i takes values in {1, 2} and indicate which coin was used on the nth trial. Note that all the X's are independent and, in particular

$$X_{i1}, X_{i2}, \dots, X_{im} | Y_i = j = Bernouili(p_j) \tag{3.54}$$

We can write out the joint probability density function of all nm + n random variables and formally come up with maximum likelihood estimation (MLE) for p_1 and p_2. If we call these MLEs $\widehat{p_1}$ and $\widehat{p_2}$.

$$\widehat{p_1} = \frac{total\ number\ of\ H\ of\ Coin1}{total\ number\ of\ Coin\ 1\ flipped} \tag{3.55}$$

$$\widehat{p_2} = \frac{total\ number\ of\ H\ of\ Coin2}{total\ number\ of\ Coin\ 2\ flipped} \tag{3.56}$$

Now suppose that the Y_i are not observed but we still want MLEs for p_1 and p_2. The data set now consists of only the X's and is "incomplete". The goal of the EM Algorithm is to find MLEs for p_1 and p_2 in this case. Basic notation for the EM Algorithm for this coin example is summarized below.

- X be observed data, generated by some distribution depending on some parameters. Here, X represents something high-dimensional. (In the coin example it is an n × m matrix.) These data may or may not be iid. (In the coin example it is a matrix with iid observations in each row.) X will be called an "incomplete data set".

- Y be some “hidden” or “unobserved data” depending on some parameters. Here, Y can have some general dimension. (In the coin example, Y is a vector.)

- Z = (X, Y) represent the “complete” data set. We say that it is a “completion” of the data given by X.

The distribution of Z depends on some parameter θ and that we can write the probability density function for Z as

$$f(z|\theta) = f(x, y|\theta) = f(y|x, \theta)f(x|\theta) \tag{3.57}$$

$L(\theta)$ is usually used to denote a likelihood function and it always depends on some random variables which are not shown by this notation. Because there are many groups of random variables, $L(\theta|Z)$ or $L(\theta|X)$ is used to denote the complete likelihood and incomplete likelihood functions, respectively.

$$L(\theta|Z) = L(\theta|X, Y) = f(X, Y|\theta) \tag{3.58}$$

$$L(\theta|X) = L(\theta|X) \tag{3.59}$$

Jensen’s Inequality: Let X be a random variable with mean $\mu = E[X]$ and let g be a convex function. Then

$$g(E[X]) \leq E[g(X)] \tag{3.60}$$

To prove Jensen’s inequality, visualize the convex function g and a tangent line at the point $(\mu, g(\mu))$. Let the line $l(x)$ be

$$l(x) = m(x - \mu) + g(\mu) \tag{3.61}$$

By convexity of g, the line $l(x)$ is always below g.

$$m(x - \mu) + g(\mu) \leq g(x) \tag{3.62}$$

We can rewrite this inequality using the random variable X to get

$$m(X - \mu) + g(\mu) \leq g(X) \tag{3.63}$$

By taking the expected value of both side

$$g(\mu) \leq E[g(X)] \tag{3.64}$$

which is the desired result

$$g(E[X]) \leq E[g(X)] \tag{3.65}$$

If g is concave, then the negative of g is convex.

Let we have some data X with joint probability density function $f(X|\theta)$ and let $l(\theta)$ denote the log-likelihood.

$$l(\theta) = ln\, f(X|\theta) \tag{3.66}$$

We would like to find a new θ that satisfies

$$l(\theta) \geq l\left(\hat{\theta}^{n}\right) \tag{3.67}$$

where $\hat{\theta}^{n}$ is the n^{th} iteration guess.

$$l(\theta) - l(\hat{\theta}^n) = \ ln\, f(X|\theta) - ln\, f(X|\hat{\theta}^n) \tag{3.68}$$

$$= ln \left\{ \int f(X|y,\theta)\, f(y|\theta) dy \right\} - ln\, f(X|\hat{\theta}^n) \tag{3.69}$$

Introduce some hidden variables Y,

$$l(\theta) - l(\hat{\theta}^n) = ln \left\{ \int \frac{f(X|y,\theta)\, f(y|\theta)}{f(y|X,\hat{\theta}^n)} f(y|X,\hat{\theta}^n) dy \right\} - ln\, f(X|\hat{\theta}^n) \tag{3.70}$$

$\left\{ \int \frac{f(X|y,\theta)\, f(y|\theta)}{f(y|X,\hat{\theta}^n)} f(y|X,\hat{\theta}^n) dy \right\}$ is the expectation with respect to the distribution $Y|X,\hat{\theta}^n$. So, applying Jensen's inequality, we have

$$l(\theta) - l(\hat{\theta}^n) \geq \ \int ln \left(\frac{f(X|y,\theta)\, f(y|\theta)}{f(y|X,\hat{\theta}^n)} \right) f(y|X,\hat{\theta}^n) dy - ln\, f(X|\hat{\theta}^n) \tag{3.71}$$

$$= \int ln(1)\, f(y|X,\hat{\theta}^n) dy = 0 \tag{3.72}$$

$$l(\theta) \geq l(\hat{\theta}^n) + \int ln \left(\frac{f(X|y,\theta)\, f(y|\theta)}{f(y|X,\hat{\theta}^n)\, f(X|\hat{\theta}^n)} \right) f(y|X,\hat{\theta}^n) dy \tag{3.73}$$

$$\int ln \left(\frac{f(X|y,\hat{\theta}^n)\, f(y|\hat{\theta}^n)}{f(y|X,\hat{\theta}^n)\, f(X|\hat{\theta}^n)} \right) f(y|X,\hat{\theta}^n) dy \tag{3.74}$$

$$= \int ln \left(\frac{f(X,y|\hat{\theta}^n)}{f(X,y|\hat{\theta}^n)} \right) f(y|X,\hat{\theta}^n) dy \tag{3.75}$$

$$= \int ln(1)\, f\big(y|X,\hat{\theta}^n\big)dy = 0 \tag{3.76}$$

We have that $l\big(\hat{\theta}^n\big) + \int ln\left(\frac{f(X|y,\theta)\, f(y|\theta)}{f\big(y|X,\hat{\theta}^n\big)\, f\big(X|\hat{\theta}^n\big)}\right) f\big(y|X,\hat{\theta}^n\big)dy$ is bounded $l(\theta)$ and that it is equal to this upper bound when $\theta = \hat{\theta}^n$.

If we maximize $l\big(\hat{\theta}^n\big) + \int ln\left(\frac{f(X|y,\theta)\, f(y|\theta)}{f\big(y|X,\hat{\theta}^n\big)\, f\big(X|\hat{\theta}^n\big)}\right) f\big(y|X,\hat{\theta}^n\big)dy$, we may improve towards maximizing $l(\theta)$.

Maximizing

$$\int ln\left(\frac{f(X|y,\theta)\, f(y|\theta)}{f\big(y|X,\hat{\theta}^n\big)\, f\big(X|\hat{\theta}^n\big)}\right) f\big(y|X,\hat{\theta}^n\big)dy \tag{3.77}$$

with respect to θ is equivalent to maximizing

$$\int lnf(X|y,\theta)\, f(y|\theta)\, f\big(y|X,\hat{\theta}^n\big)dy \tag{3.78}$$

$$= E_Y[lnf(X,Y|\theta)|X\hat{\theta}^n] \tag{3.79}$$

So, if we could compute this expectation, maximize it with respect to θ, call the result $\hat{\theta}^{n+1}$ and iterate, we can improve towards finding the θ that maximizes the likelihood. These expectation and maximization steps are precisely the Expectation–maximization (EM) algorithm.

The EM Algorithm is a numerical iterative for finding an MLE of θ. The rough idea is to start with an initial guess for θ and to use this and the observed data X to "complete" the data set by using X and the guessed θ to postulate a value for Y , at which point we can then find an MLE for θ in the usual way. The actual idea though is slightly more sophisticated. We will use an initial guess for θ and postulate an entire distribution for Y , ultimately averaging out the unknown Y .

Let η be maximum likelihood estimate and X be complete data , EM iteratively alternates between making guesses about x , and finding the η that maximizes probability $p(x|\ \eta\ )$ over θ. In order to use EM, there should be given observed data y , density probability $p(y|\eta)$, complete data x and parametric density $p(x|\ \eta\ )$.

It is assumed that complete data x can be modeled by random variable X having density $p(x|\ \eta\ )$ over the data set Ω ($\eta \in \Omega$). In this case, X is not directly observed, y which is realization of random vector Y is observed. Y depends on X.

Given the observed data y , our aim is to find maximum likelihood estimate (MLE) of η

$$\hat{\eta}_{MLE} = argmax_{\eta\in\Omega}\, p(y|\eta) \tag{3.80}$$

However, it is easier to calculate η that maximizes the log-likelihood of y, because log() function is monotonically increasing:

$$\hat{\eta}_{MLE} = argmax_{\eta\in\Omega} \log p(y|\eta) \tag{3.81}$$

In some circumstances where, it is difficult to assess both likelihood, the EM algorithm makes a guess about the complete data and solve for η that maximized log-likelihood. Then, it is easy to make better guess about complete data. EM algorithm consists of two steps namely E-step and M-step however let us break down into five steps:

Step 1: Make an initial estimate $\eta^{(m)}$for η

Step 2: Calculate the conditional probability distribution $p(x|y,\eta^{(m)})$ for completed data x and observed data y

Step 3: Calculate conditional expected log-likelihood

$$\int logp(x|\eta)\, p\big(x\big|y,\eta^{(m)}\big)dx = E_{\left(X\middle|y,\eta^{(m)}\right)}\log p(X|\eta)$$

where integral is over the set $X(y)$ and $X(y)$ is assumed to be independent from η.

Conditional expected log-likelihood depends on η as a free parameter and current guess $\eta^{(m)}$ calculated in Step 2.

Step 4: Find the η that maximizes conditional expected log-likelihood.

Obtained result is $\eta^{(m+1)}$

Step 5: Assign $m = m + 1$ and g oto Step 2. Iterate until stopping criterion is satisfied.

Steps 2 and 3 are called the E-step for expectation, and Step 4 is called the M-step for maximization.

Expectation Maximization algorithm is used to estimate dynamic label image with initial guess obtained by pose estimation solved by Levenberg Marquart method.

3.3 RGB-D Sensors

In our approach, we sequentially fuse the RGB-D info into a signed distance field because RGB-D cameras are specific type of sensors, which can augment the image with depth information. Therefore, this type of sensor has been extensively used in computer vision and computer graphics problems in order to find novel solutions. The depth information provides a extensive information to find a solution of object detection, semantic segmentation, shape analysis, pose estimation and 3D reconstruction.

With the appearance of Kinect in 2010 as an user input device for the gaming, many cheap RGB-D sensors became available on the market such as Kinect, Asus Xtion Pro and Intel RealSense. Most recent developed stereo RGB-D sensor is ZED camera and it is compatible to recently launched Jetson board of the Nvidia which makes it a powerful tool for applications in the field of robot vision. Figure 3.1 shows some examples of RGB-D sensors.

The output of these sensors is depicted in Figure 3.2. The left image in the figure is an RGB image, i.e., each pixel stores the color of a point in the real world. The right image is a depth image, i.e., each pixel stores the coordinate of that point with respect to the reference frame of the sensor.

Figure 3.1: Example of RGB-D sensors. From left to right: Microsoft Kinect, Intel RealSense D435. Courtesy of Microsoft and Intel Corporation.

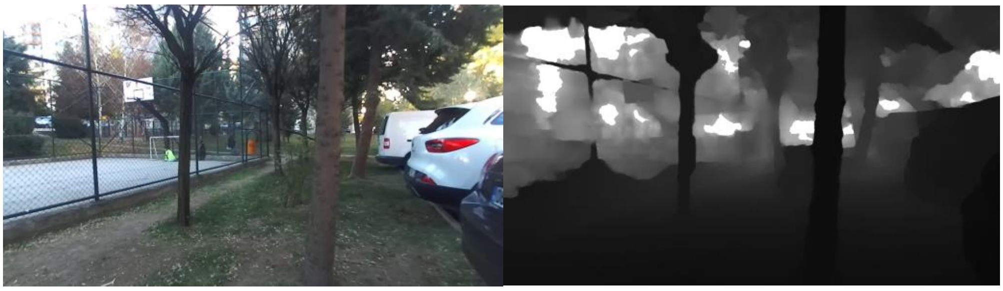


Figure 3.2: Output images of RGB-D sensors. Left: RGB image. Right: Depth image. The brighter a pixel is, the farther away it is from the sensor.

Using RGB-D sensor, measured 3D (X, Y, Z) positions of sensed surfaces can be directly computed from the intrinsic RGBD camera parameters and the measured depth image values. The Z coordinate is directly taken as the depth value and the (X, Y) coordinates are computed using the pinhole camera model. In a typical pinhole camera model, 3D (X, Y, Z) points are projected to (x, y) image locations, e.g., for the image columns the x image coordinate is

$$x = f_x \frac{X}{Z} + c_x - \delta_x. \tag{3.82}$$

However, for a depth image, this equation is re-organized to "back-project" the depth into the 3D scene and recover the (X, Y) coordinates as shown by equation

$$X = \frac{x + \delta_x - c_x}{f_x} Z \tag{3.83}$$

$$Y = \frac{y + \delta_y - c_y}{f_y} Z \tag{3.84}$$

$$Z = Z \tag{3.85}$$

where Z denotes the sensed depth at image position (x, y), (f_x, f_y) denotes the camera focal length (in pixels), (c_x, c_y) denotes the pixel coordinate of the image center, i.e., the principal point, and (δ_x, δ_y) denote adjustments of the projected pixel coordinate to correct for camera lens distortion. Intrinsic camera calibration matrix $K \in R^3$ is written as

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.86}$$

In practice, the pinhole camera model is not sufficient to fully model the behavior of imperfect real-world lenses. In order to remove the radial and tangential lens distortion, there are various closed-form approximations, While higher-degree approximations are more expressive, they become more difficult to calibrate and are often numerically unstable. The following distortion function is very common and effective for consumer-grade cameras.

$$\varrho_o(X) = \begin{bmatrix} x\left(\dfrac{1+\kappa_1 r^2+\kappa_2 r^4+\kappa_3 r^6}{1+\kappa_4 r^2+\kappa_5 r^4+\kappa_6 r^6}\right) + 2\rho_1 xy + \rho_2(r^2+2x^2) \\ y\left(\dfrac{1+\kappa_1 r^2+\kappa_2 r^4+\kappa_3 r^6}{1+\kappa_4 r^2+\kappa_5 r^4+\kappa_6 r^6}\right) + 2\rho_2 xy + \rho_1(r^2+2y^2) \\ 1 \end{bmatrix} \tag{3.87}$$

Here, $\kappa_1, \kappa_2, \ldots, \kappa_6$ and ρ_1, ρ_2 are for radial and tangential distortion coefficients, and $r^2 = x^2 + y^2$ respectively. In this study, we use the following distortion model if the coefficients are available. This coefficient are invariant to scaling of an image.

$$\varrho_1(X) = \begin{bmatrix} x(1+\kappa_1 r^2+\kappa_2 r^4+\kappa_3 r^6) + 2\rho_1 xy + \rho_2(r^2+2x^2) \\ y(x(1+\kappa_1 r^2+\kappa_2 r^4+\kappa_3 r^6)) + 2\rho_2 xy + \rho_1(r^2+2y^2) \\ 1 \end{bmatrix} \tag{3.88}$$

Studies of accuracy for the Microsoft Kinect sensor show that a Gaussian noise model provides a good fit to observed measurement errors on planar targets where the distribution parameters are mean 0 and standard deviation $\sigma_Z = \frac{m}{2f_x b} Z$ for depth measurements where $\frac{m}{f_x b} = -2.85x10^{-3}$ is the linearized slope for the normalized disparity empirically found in [3]. Since 3D the coordinates are a function of both the pixel location and the depth, their distributions are also known as shown below:

$$\sigma_X = \frac{x+\delta_x-c_x}{f_x}\sigma_Z = \frac{x+\delta_x-c_x}{2f_x}(2.85x10^{-3})Z^2 \qquad (3.89)$$

$$\sigma_Y = \frac{y+\delta_y-c_y}{f_y}\sigma_Z = \frac{y+\delta_y-c_y}{f_y}(2.85x10^{-3})Z^2 \qquad (3.90)$$

$$\sigma_Z = \frac{m}{2f_x b}Z = \frac{(2.85x10^{-3})Z^2}{2} \qquad (3.91)$$

These equations indicate that 3D coordinate measurement uncertainty increases as a quadratic function of the depth for all three coordinate values. However, the quadratic coefficient for the (X, Y) coordinate standard deviation is at most half that in the depth direction, i.e., $(\sigma_X, \sigma_Y) \approx 0.5\sigma_Z$ at the image periphery where $\frac{x-c_x}{f_x} \approx 0.5$, and this value is significantly smaller for pixels close to the optical axis.

For example, consider a “standard” Primesense sensor having no lens distortion and typical factory-set sensor values: $f_x, f_y = f$ = 586±30 for focal length, (640; 480) for image (x, y) dimension, and (c_x, c_y) = (320; 240) for the image center. In this case the ratios

$$(\frac{x+\delta_x-c_x}{f_x}, \frac{y+\delta_y-c_y}{f_y}) = (0,0) \qquad (3.92)$$

at the image center and

$$(\frac{x+\delta_x-c_x}{f_x}, \frac{y+\delta_y-c_y}{f_y}) = (0.548 \pm 0.028, 0.411 \pm 0.221) \qquad (3.93)$$

at the (x, y) positions on the image boundary.

With this in mind, the $(X,\ Y,\ Z)$ coordinates of a depth image are modeled as measurements from a non-stationary Gaussian process whose mean is 0 at all points but whose variance changes based on the value of the triplet (x, y, Z).

CHAPTER 4

OUR PROPOSED METHODOLOGY

4.1 Justification of the Methodology and Novelties of Our Approach

SLAM algorithms uses the sensor data in order to produce a consistent map of environment. In real life, SLAM frameworks should address the problem of dynamic scene without consuming much computational power. Employing semantic segmentation or object classifier are time consuming, therefore, we refrain using such techniques in our proposed SLAM algorithm. Our aim is to develop a SLAM algorithm which

- can generate the dense model/mesh of environment
- can handle high dynamics in environment,
- is robust to dynamic element without requiring any type of object tracking,
- does not need any classifier or semantic information,
- is real-time operation compatible,
- does not require sophisticated sensor.

Considering these practical constraints, we propose a RGB-D based SLAM method because RGB-D cameras are specific type of sensors, which can augment the image with depth information. Moreover, this type of sensor has been extensively used in computer vision and computer graphics problems because of its low cost.

In order to generate a dense model of environment, we use signed distance field (SDF) representation because such representation is useful in robotic applications because it defines the distance to surface.

Our proposed approach requires neither explicit object tracking nor object classifier therefore it is robust to any type of moving object. We identify the dynamic part using the image registration residuals, which are obtained by the pose estimation on the SDF using Levenberg-Marquart method.

Figure 4.1 depicts the flowchart of our SDF-based RGB-D dense SLAM method. We start the process by obtaining the RGB-D image. Initially, we assign labels as static and pose as identity. We continue the process by measuring the similarity ratio of the consecutive images. If the image passes the similarity test, next process is the pose estimation. We try to solve this optimization problem by Levenberg-Marquart method until the pose difference is less than 0.001 or iteration number is predefined level. In order to measure the pose difference, we use the norm of pose vector. Using the calculated norm, we can calculate the image registration residuals and this is an initial guess for dynamic label. Then we continue the process by calculation of the final dynamic label or image using Expectation Maximization algorithm. We re-perform pose estimation using this final image and this process continues until the last frame is processed.

The Signed Distance Function (SDF), also referred to as the Signed Distance Transform, or simply Distance Transform has been widely applied to the processing or visualization of volumetric 3D data. The SDF is usually implemented as a voxel-based (or pixel-based in the two-dimensional case) representation, in which each cell contains the distance to the nearest surface in the scene. The signed part indicates whether the voxel (or pixel) is on the outside (positive) or inside (negative) an object.

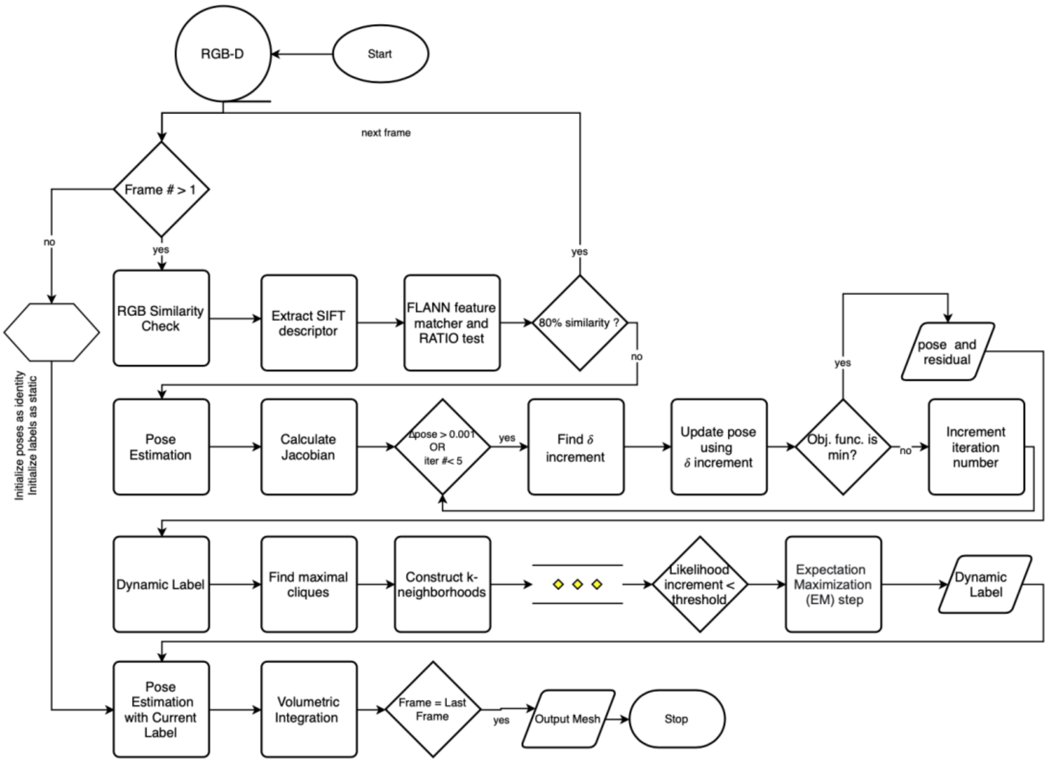

Start
RGB-D
next frame
Frame # > 1
no
yes
RGB Similarity Check
Extract SIFT descriptor
FLANN feature matcher and RATIO test
80% similarity ?
yes
no
Initialize poses as identity
Initialize labels as static
Pose Estimation
Calculate Jacobian
Δpose > 0.001 OR iter #< 5
yes
Find δ increment
Update pose using δ increment
Obj. func. is min?
yes
no
pose and residual
Increment iteration number
Dynamic Label
Find maximal cliques
Construct k-neighborhoods
Likelihood increment < threshold
Expectation Maximization (EM) step
Dynamic Label
Pose Estimation with Current Label
Volumetric Integration
Frame = Last Frame
yes
Output Mesh
Stop

Figure 4.1: Flowchart of the DUDMAP

In this study, we represent the geometry using SDF. In order to reconstruct the scene, we fuse incrementally RGB-D data into SDF and geometry is stored in voxel grid. First, camera pose is estimated using SDF and SDF is updated based on newly computed camera pose. In the literature, most of the volumetric fusion techniques, for example, KinectFusion [24] uses synthetic depth images and aligns them using Iterative Closest Point. However, we use the camera pose directly on the SDF because SDF encrypts the 3D geometry of the environment.

Figure 4.2 depicts the important steps of our proposed method. We first apply a depth filter in order to eliminate significant amounts of noise in raw depth images. In order to eliminate redundant data in fusion process, we trim repeated camera views by measuring the similarity ratio of RGB images. We then perform pose estimation and continue the process by detecting the dynamic elements in the scene.

The subsequent subsections provide the details of each block in our proposed system. Section 4.2 reviews the preliminaries for the proposed system and Section 4.3 explains the depth smoothing process and similarity test. Section 4.4 is devoted to the pose estimation algorithms. Section 4.5 and Section 4.6 respectively focuses on 3D volume visualization and dynamic detection from image registration or pose estimation residuals.

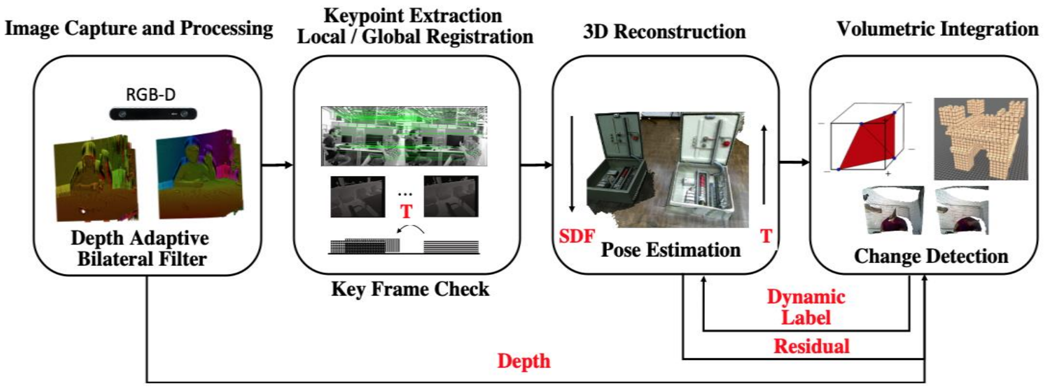


Figure 4.2: DUDMAP scheme

4.2 Preliminaries and Notations

In our approach, we denote a 3D point as $[X,Y,Z]\in R^3$, rotation of the camera as $R\in SO(3)$, and translation as $T\in R^3$, respectively. At time t, RGB-D frame contains an RGB image and a depth map. The homogenous point $\mathbf{X}=(x, y, z,1)^T$ can be computed by assuming a pinhole camera model with intrinsic parameters f_x, f_y, c_x and c_y (focal length and optical center) such as

$$\boldsymbol{X} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \left[\frac{x - c_x}{f_x} z, \frac{y - c_y}{f_y} z, \mathrm{z}, 1\right]^T \tag{4.1}$$

The 3D point corresponding to a pixel is reconstructed using following equations.

$$\frac{x - c_x}{f_x} z = u \tag{4.2}$$

$$\frac{y - c_y}{f_y} z = v \tag{4.3}$$

The pixel coordinates becomes

$$\left[\frac{u f_x}{z} + c_x, \frac{v f_y}{z} + c_y\right]^T. \tag{4.4}$$

In a rigid body motion, the common representation matrix is the homogenous transformation matrix H consisting of a 3x3 rotation matrix and 3x1 translation vector T is widely known to the robotics community as

$$H_{4x4} = \begin{bmatrix} R_{3x3} & T_{3x1} \\ 0_{1x3} & 1_{1x1} \end{bmatrix} \tag{4.5}$$

which is used in the transformation of a point $\vec{X}$ under motion as

$$\mathrm{X}' = H_{4x4}\mathrm{X} \tag{4.6}$$

The rotation matrix R has nine parameters and if we were to estimate the camera motion, we would have to solve these nine parameters by forming a constrained optimization problem, which can be very slow to implement. The Lie algebra allows us lower dimensional linear space for rigid body motion representation making it popular in computer vision problems. Actually, R is an orthonormal matrix which has in fact a 3 parameter representation and these 3 parameters are in general found using Euler angles.

We use a Lie algebra SE(3) representation as twist coordinates ξ as in [23] because the rigid motion has 6 degrees of freedom while transformation matrix T has 12 parameters. Using the Lie algebra representation rigid body motion can be written as

$$\xi = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & \upsilon_1 \\ \omega_3 & 0 & -\omega_1 & \upsilon_2 \\ -\omega_2 & \omega_1 & 0 & \upsilon_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.7}$$

4.3 Depth Smoothing and Feature Matching

Commercially available RGB-D cameras usually produce invalid depth measurements. In addition, there exist significant amounts of noise in raw depth images. Bilateral filter as indicated in [92] modifies the weighting to account for variation of intensity thereby effectively carrying out a robust smoothing operation. Depth smoothing helps to reduce the disoccluded areas, which should be filled before the operation. Moreover, smoothing of depth image is beneficial for avoiding holes in the raw depth image and for removing the sharp discontinuities from depth image. The most significant source of such deviation is quantization noise, which arises when the disparity is estimated within a finite precision. The standard deviation of noise in depth measurement is given as proportional to the square of the depth. In this study, we use a depth adaptive bilateral filtering method, which is more effective to smooth depth images than the bilateral filtering.

Let $D(u)$ be an observed depth image where u denotes the location of a pixel. The depth estimation smoothed by the depth adaptive bilateral filtering presented in [93] is

$$\widehat{D}(u) = \frac{D(u)}{W} \sum_{N(u_k)} \left[w_s(u - u_k) w_c\big(D(u) - D(u_k)\big) \right] \tag{4.8}$$

where w_c and w_s are Gaussian functions for spatial and range weighting with standard deviations σ_s and σ_c. $N(u_k)$ is the the neighborhood of u. W is used for normalizing factor to have a total sum of 1 over $N(u_k)$. w_c and w_s are given as

$$w_s = e^{-\frac{(u-u_k)^2}{2\sigma_s{}^2}} \tag{4.9}$$

$$w_c = e^{-\frac{(D(u)-D(u_k))^2}{2\sigma_c{}^2}} \tag{4.10}$$

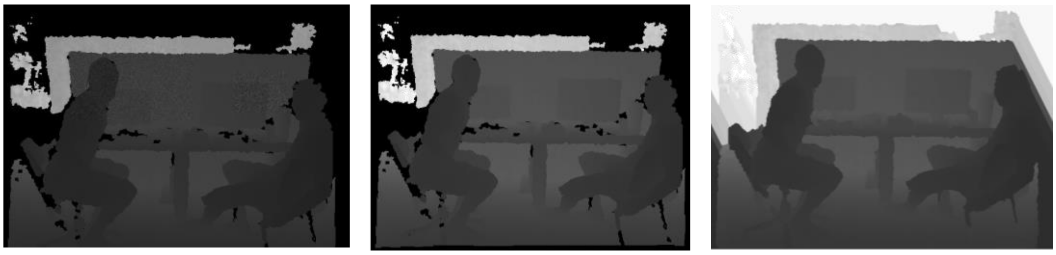
Unlike the bilateral filter, here the values of σ_c for the depth image are not fixed but vary with the depths. It can be approximated as

$$\sigma_c = \alpha D(u)^2 \tag{4.11}$$

where α is constant and its value depends on the camera parameters. In our experiments, α is set to be 12 and σ_s is 4 (in pixels). Figure 4.3 show the original input image and result of a depth image with adaptive bilateral filter, respectively.

We can see that depth adaptive bilateral filter for the depth image is more effective to remove the noise. Foreground are appropriately smoothed while preserving depth discontinuity features since the proposed filter is adaptive to the variation of depth. After filtering, invalid measurements due to out-of-range measurement or reflecting surfaces are tried to be eliminated using depth smoothing filter and the resulting image is depicted in also Figure 4.3.

In this study, we use a depth adaptive bilateral filtering method because it modifies the weighting to account for variation of intensity. Figure 4.4 depicts the original depth image, smoothed and filtered image respectively. In addition, we change the zero values in the original depth images by neighboring 5x5 pixel mean value in the smoothing process.


Input depth image

Adaptive bilateral filter result

Handling of invalid measurement

Figure 4.3: Original input image , result of a depth image with adaptive bilateral filter and , invalid measurements handling

SDF fusion is an averaging process therefore, it is important not to use redundant data in the fusion process because small error renders the SDF model unclear. In order to eliminate redundant camera views, we perform similarity ratio test based on feature matching. A typical feature matcher consists of the following steps: extracting local feature, matching features by using nearest-neighbor approach and selecting good correspondences.

In the literature, Scale Invariant Feature Transform (SIFT) is being proposed for extracting keypoints and is widely used in different applications. SIFT feature-matching works well for scaled images but fails for some cases such as faces with pose changes [8]. Application of feature matching method FLANN with SIFT descriptor overcomes such disadvantages of SIFT. In similarity analysis, we use FLANN based feature matching with SIFT descriptor and we use RATIO [2] to select good correspondences that compares the lowest feature distance and the second lowest feature distance for recognizing good ones. Similarity ratio of the VolumeDeform “boxing”sequence is depicted in Figure 4.5. Since the ratio is not high, which indicates low degree of similarity, all the frames are included in the mapping process.

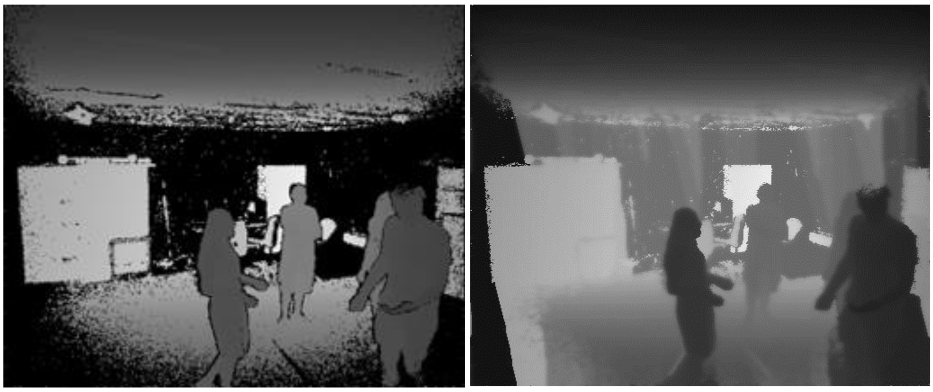


Figure 4.4: a) Original image b) Final image

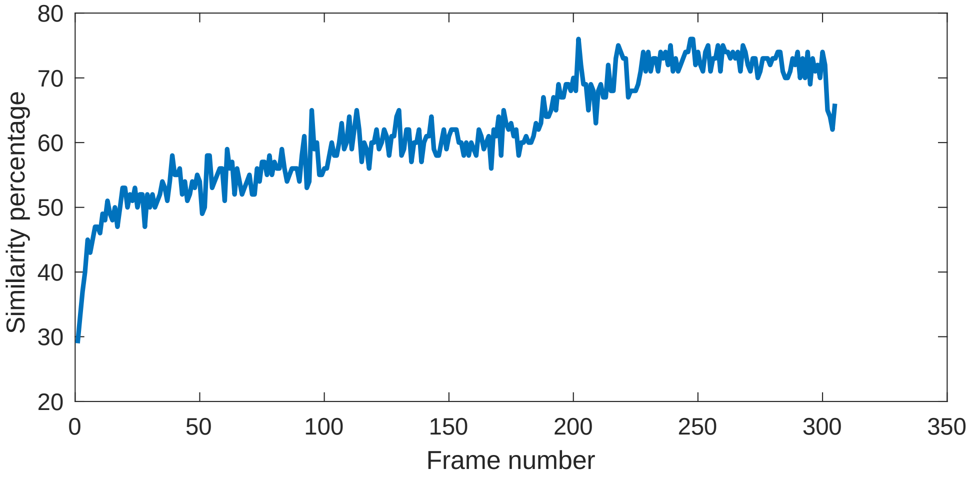


Figure 4.5: VolumeDeform boxing sequence similarity ratio

On the other hand, BONN dataset “crowd2” sequence is a high dynamic sequence having 895 frames. Figure 4.6 illustrates the BONN “crowd2” sequence similarity ratio with respect to threshold value. If 80% similarity threshold is utilized, 106 frames are skipped, which results in 11.8% decrease in computational time. Absolute translational error increases only 2.2%, while rotational relative pose error increases by 0.3%. In low dynamic sequences, the number of similar frames is higher, which decreases the unnecessary computational power. This is the novel enhancement in this thesis we provide to existing methods in the literature for the betterment of the performance. We use the 84% similarity threshold because our aim is to have 5% decrease in computational cost.

On the other hand, BONN dataset “crowd2” sequence is a high dynamic sequence having 895 frames. Figure 4.6 illustrates the BONN “crowd2” sequence similarity ratio with respect to threshold value. If 80% similarity threshold is utilized, 106 frames are skipped, which results in 11.8% decrease in computational time. Absolute translational error increases only 2.2%, while rotational relative pose error increases by 0.3%. In low dynamic sequences, the number of similar frames is higher, which decreases the unnecessary computational power. This is the novel enhancement in this thesis we provide to existing methods in the literature for the betterment of the performance. We use the 84% similarity threshold because our aim is to have 5% decrease in computational cost.

4.4 Pose Estimation

The Signed Distance Function (SDF), also referred as simply Distance Transform has been widely applied to the processing or visualization of volumetric 3D data. Commonly used in the field of computer graphics as an acceleration structure for speeding up ray-casting operations [16] it can also be used as a 3D model representation. Other application area where SDF is widely used include collision detection and haptic feedback.

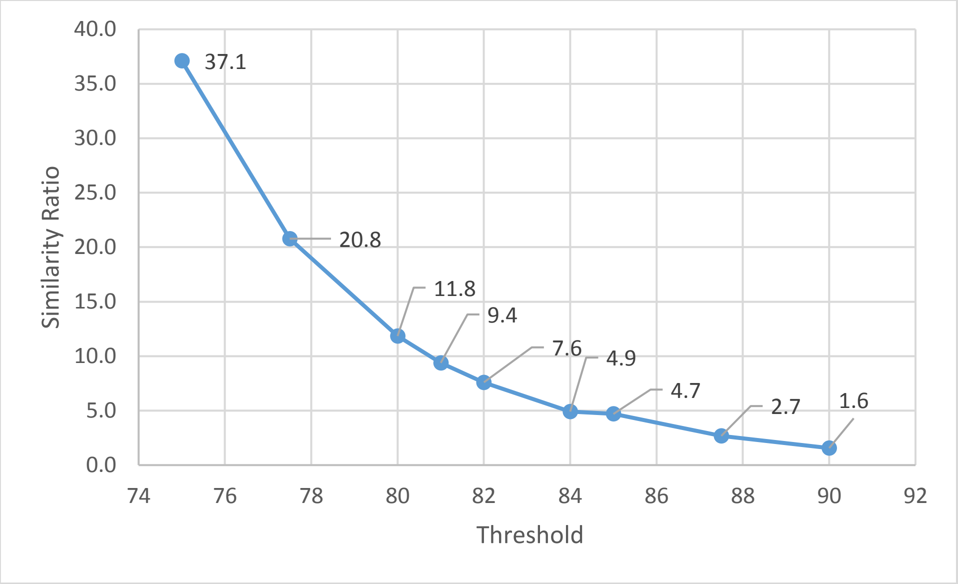


Figure 4.6: BONN "crowd2" sequence similarity ratio with respect to threshold value.

The SDF is usually implemented as a voxel-based representation, in which each cell contains the distance to the nearest surface in the scene. The signed part indicates whether the voxel is on the outside (positive) or inside (negative) an object.

Let ψ be a function $\psi(x): R^N \to R$ which maps the N-dimensional space to a scalar value. For example, let N=2 and consider the circle equation $x^2 + y^2 = r^2$ or more consistent form $\|x\|_2^2 = r^2$.

The circle equation is equivalent to $\|x\|_2 - r = 0$. $\|.\|_2$ is the L2-norm (Euclidean distance). If we focus on the first term of the equation and plot , the obtained plot is a smoothly varying gradient that becomes lighter (higher-valued) further away from the origin, in every direction. If we now subtract r from $\|x\|_2$, we get the the distance is relative to the edge of the circumference. It is a positive value whenever outside, negative whenever inside and zero exactly on the edge of the circle. This means that the common definition for a circle conforms precisely to the definition of a signed distance function.

SDF represent a surface, where we have to test in order to determine where a given ray intersects this surface . Given a ray,

$$\alpha\rho = \alpha \begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \end{bmatrix} \tag{4.12}$$

where ρ is a unit-norm vector and α a scalar. We want to find a scalar $\psi(\alpha^{*}\rho) = 0$. This process is known as sphere tracing. In this technique, a ray is iteratively rescaled by adding the current value of the SDF to α and it is similar to successive approximation of roots by Secant Method.

SDF is defined as the distance to the nearest surface, each step along the ray can be thought of as, moving to the edge of the largest sphere that fits the current point in space. An illustration of the algorithm is given in Figure 4.7. When searching in this way, for a surface, it is practical to have an early stopping condition at some smallest allowed step-size is set. This early stopping (at a positive value) can speed up rendering, but can also be used to dilate objects, making them appear arbitrarily thicker. Conversely, late stopping can be used to make objects thinner.

SDF represents the distance to the nearest surface, however, it has some drawbacks with its original form. It could not encode the surfaces with high geometric details. SDF is dependents on heavily on normal vector. For an SDF with a analytical expression, the components of the normal vector are simply obtained by partial derivatives of the SDF with respect to each spatial dimension.

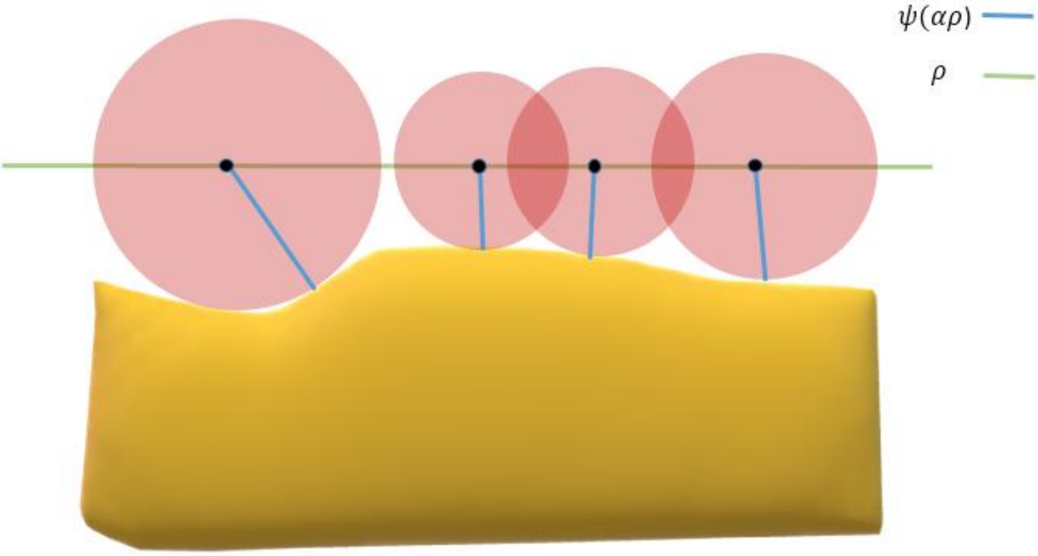


Figure 4.7: Sphere tracing. The dots represent the points at which the function $\psi(\alpha^*\rho)$ is evaluated and the blue lines represent the scalar value returned by the function $\psi(\alpha^*\rho)$

$$\eta(x) = \nabla_x \psi(x)^T = \begin{bmatrix} \frac{\partial\psi(x)}{\partial x_1} \\ \frac{\partial\psi(x)}{\partial x_2} \\ \frac{\partial\psi(x)}{\partial x_3} \end{bmatrix} \tag{4.13}$$

An explicit expression will often not be available and the gradient vector can be found by finite differences in such cases.

The first derivative of the SDF with respect to position produces the gradient toward the surface, and second derivative produces the curvature which is a measure of how this gradient changes with position:

$$Curv(x) = \nabla^2{}_{xx}\psi(x) \tag{4.14}$$

$$\nabla^2{}_{xx}\psi(x) = \frac{\partial^2\psi(x)}{\partial {x_1}^2} + \frac{\partial^2\psi(x)}{\partial {x_2}^2} + \frac{\partial^2\psi(x)}{\partial {x_3}^2} \tag{4.15}$$

Finite-difference approach to evaluating gradients requires several memory look-ups. An approximation that gives an indication of curvature, and also has the benefit of being normalized in the range [0; 1] is the projection of adjacent gradient vectors onto each-other:

$$\nabla^2{}_{x,i}\psi(x) \approx 1 - \frac{\eta(x+d_i)^T\eta(x)}{\|\eta(x+d_i)\|_2 \cdot \|\eta(x)\|_2} \tag{4.16}$$

Where $\eta(x)$ is the surface normal at x, d_i is the displacement vector having all zeros except component denoted by the subscript. This equation implies that if two nearby points in space have gradients oriented in different directions, the measure of curvature will be high.

We have defined SDF with its properties, now we can continue on method that can be used for calculation of SDF from raw depth images. The input is the depth data information obtained by RGB-D sensor and is usually an ordinary gray-scale image. Each pixel in a depth image stores a numeric value that corresponds to the distance at which a surface was measured along. Figure 4.8 shows a depth and color image of an office. The images are part of a publicly available dataset [96].

In order to create a SDF from a depth image, we first initialize a discrete voxel grid (x,y,z) in 3D with resolution of τ where each voxel represents a cube in space, each cube measuring τ meters in length, width and height.

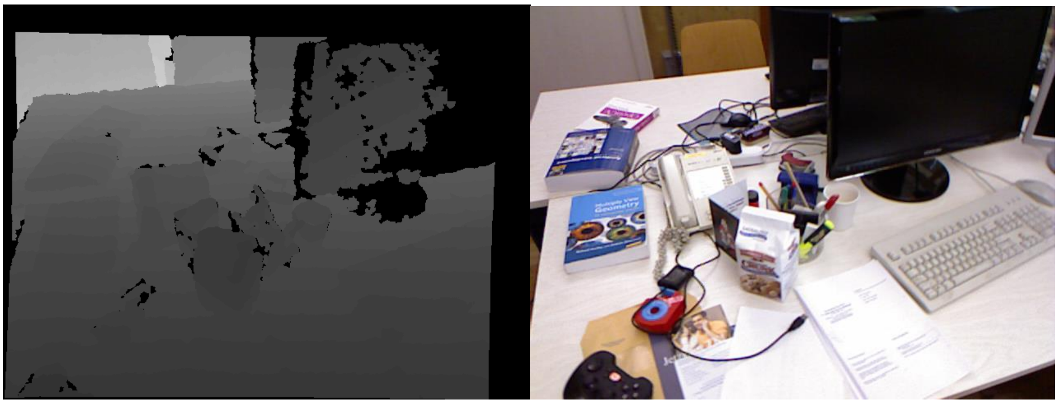


Figure 4.8: Depth and color images of the same office desk

Let $\pi: R^3 \rightarrow R$ be a (vector-valued) function that perspective-projects 3D points to the image plane, or formally

$$\pi(u,v,z)\left[\frac{uf_x}{z}+c_x,\frac{vf_y}{z}+c_y\right]^T \tag{4.17}$$

The spatial coordinates to the center of each voxel are then perspective projected using π into the image plane, which is illustrated in Figure 4.9.

The difference between this value and the distance to the voxel (measured along the view axis) is then computed. The resulting difference is stored in the voxel itself. Since SDF represents a surface, this produces values that are positive, zero or negative depending on whether the center of the voxel is outside, on or behind surfaces, respectively.

By interpolating between voxels as shown Figure 4.10 and we can get a surface at the boundary between positive and negative values (Figure 4.11). In order to avoid false surface, we truncate the values that can be written into the voxel grid at a small positive value D_{max} and a small negative value D_{min}. This approach helps us to allow local changes without the need for updating distance values in remote voxels. Note that the negative and positive limits can be unsymmetric around zero. Choosing large positive limit improves collision avoidance; however, minimum value determines the minimum thickness of the reconstructed object.

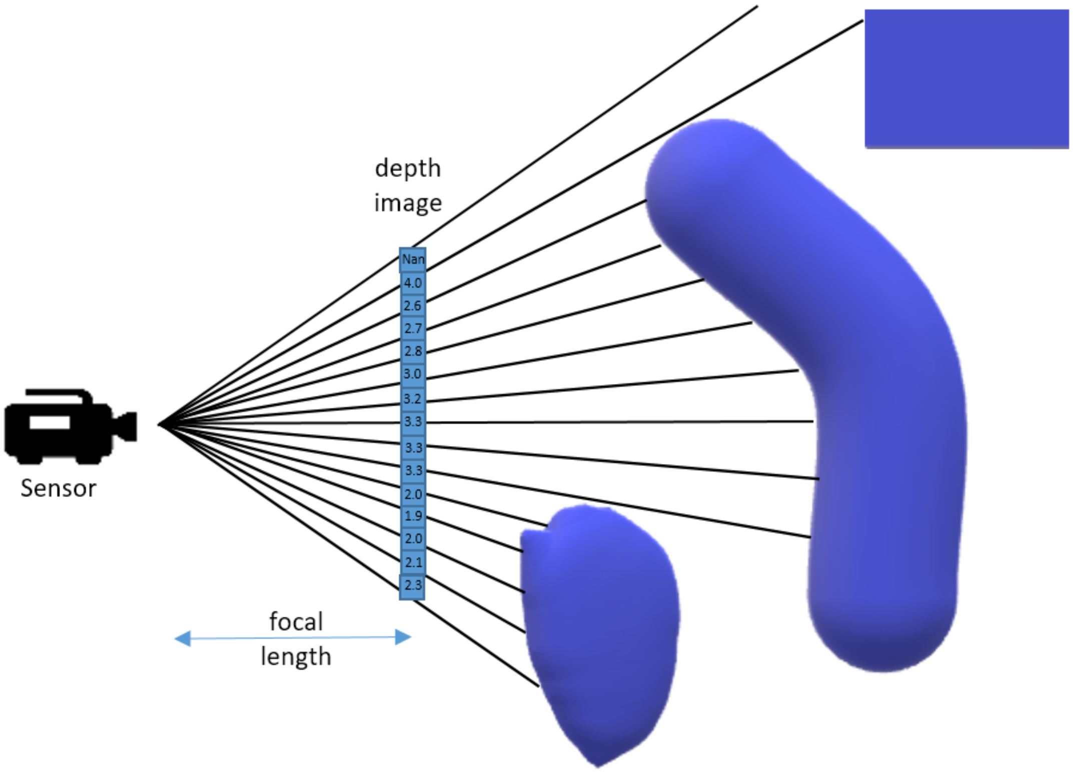


Figure 4.9: The coordinates of each voxel center is projected into the image plane. Note that not every projection will fall within the subset of the plane where the depth image is defined.

In addition, new depth value obtained by distance sensor in the form of video stream affect the data used to represent signed distance function $\psi(x)$. Instead of putting the new values, it is common to use a weighted average of those depth values.

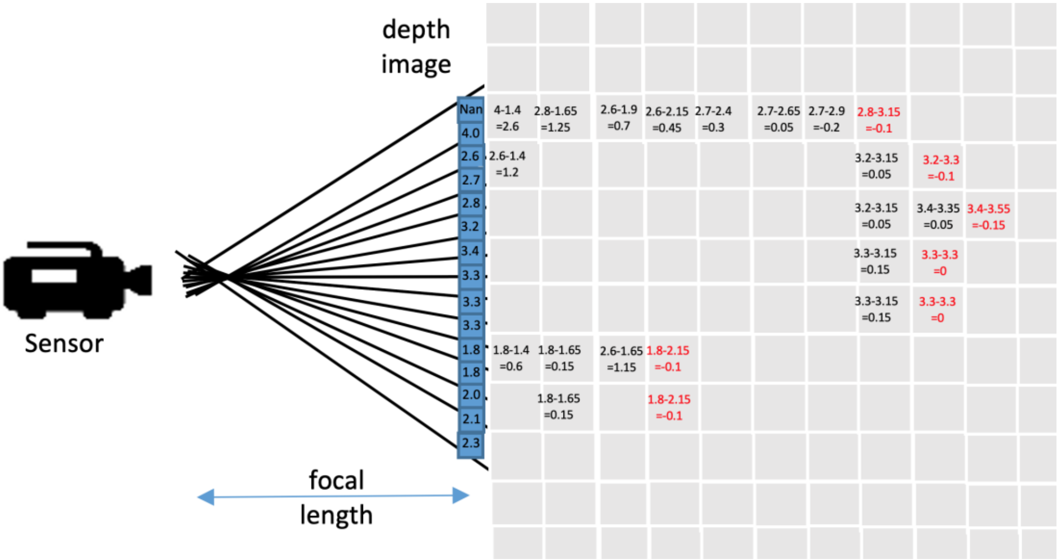


Figure 4.10: Voxels are updated with the difference between the depth image value and the distance to the respective voxel from the sensor, along the viewing direction.

Let $W(w)$ be a function with $w = (w_1, w_2, w_3)^T$ representing the weight of the data stored in truncated signed distance function and let $\psi_t(x)$ be the tri-linear interpolation between the nearest 8 values if x is within the boundary. The update of a single element using $\psi_t(x)$ and $W(w)$ is done by

$$\psi_t(x)_{n+1} = \frac{\psi_t(x)_n W(w)_n + \psi_t(x)_{n+1} W(w)_{n+1}}{W(w)_n + W(w)_{n+1}} \tag{4.18}$$

$$W(w)_{n+1} = \min(W(w)_n + W(w)_{n+1}, W_{max}) \tag{4.19}$$

$W(w)$ can be calculated using the error model $err(Z)$ of distance sensor.

$$W(w) = \frac{cos\theta}{err(Z)} \tag{4.20}$$

For structured light sensors, we are certain about measurements perpendicular to surfaces that are close to the sensor and that error increases based on an error model for the sensor that varies with distance.

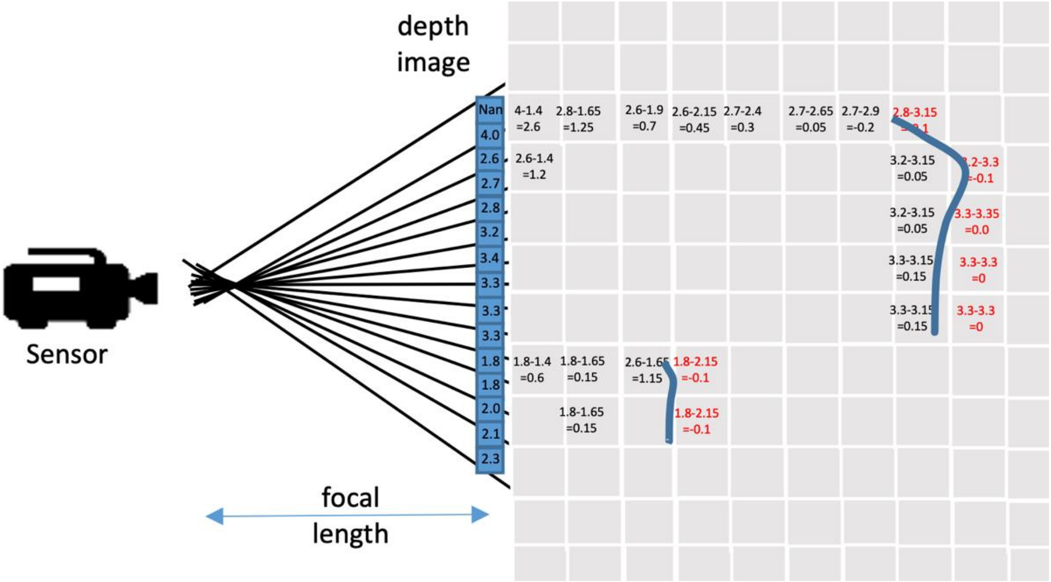


Figure 4.11: Surface at the boundary between positive and negative values

However , calculation of $W(w)$ is computational extensive because of calculation of $cos\theta$ and estimation of surface normals. We set to $W = 1$ and this causes weigthed update to a rolling average due to saturation of W_{max}.

In order to use color information directly for pose estimation, we store color values for each voxel. The color values are updated for each RGB channel by using same rolling average process as

$$R(x)_{n+1} = \frac{R(x)_n W(w)_n + R(x)_{n+1} W(w)_{n+1}}{W(w)_n + W(w)_{n+1}} \tag{4.21}$$

$$G(x)_{n+1} = \frac{G(x)_n W(w)_n + G(x)_{n+1} W(w)_{n+1}}{W(w)_n + W(w)_{n+1}} \tag{4.22}$$

$$B(x)_{n+1} = \frac{B(x)_n W(w)_n + B(x)_{n+1} W(w)_{n+1}}{W(w)_n + W(w)_{n+1}} \tag{4.23}$$

where $R(x), G(x), B(x)$ are red, green and blue values of color, respectively,

Algorithm for SDF initialization and SDF update with rolling average are given below.

Algorithm 1. SDF initialization and SDF update

```
Input : Integer coordinates of each voxel (x), maximum truncation distance, w
for all integer coordinates x
    Assign maximum truncation distance
        $\psi_t(x) \leftarrow d_{max}$
end for
for all w
    Assign w as zero
        $W(w) \leftarrow 0$
end for
for all integer coordinates x
    Assign w as integer coordinates and initial weigth as 1
        $w \leftarrow x \quad weigth \leftarrow 1$
    Update of a single element of $\psi_t(x)$
        $\psi_t(x)_{n+1} \leftarrow \frac{\psi_t(x)_n W(w)_n + \psi_t(x)_{n+1} W(w)_{n+1}}{W(w)_n + W(w)_{n+1}}$
        Update of a single element of $W(w)$
            $W(w)_{n+1} \leftarrow \min(W(w)_n + W(w)_{n+1}, W_{max})$
        Update red, green and blue channel
            $R(x)_{n+1} = \frac{R(x)_n W(w)_n + R(x)_{n+1} W(w)_{n+1}}{W(w)_n + W(w)_{n+1}}$
            $G(x)_{n+1} = \frac{G(x)_n W(w)_n + G(x)_{n+1} W(w)_{n+1}}{W(w)_n + W(w)_{n+1}}$
            $B(x)_{n+1} = \frac{B(x)_n W(w)_n + B(x)_{n+1} W(w)_{n+1}}{W(w)_n + W(w)_{n+1}}$
        end for
```

Representing surfaces using signed distance function is extremely easy because it provides an efficient mechanism in consistent surface estimation. However, there exists some special case where using TSDF for surface representation is limited or special requirements exist. For example, surface representation using TSDF is memory intensive because required memory for TSDF volume scales cubically and it depends on the grid resolution. For example, a voxel grid of resolution 512^3 covering a bounded volume of about 4 m^3. In order to solve this memory requirement, we use Voxel Hashing method proposed by Nießner [10] which is memory and computational power efficient. In this technique, surface data is stored only densely where the measurements are observed and data can be exchanged easily efficiently through hash table in both ways. We do not need memory constrained voxel grid using this voxel hashing method and this supports real-time performance without giving up finer quality reconstruction.

TSDF has intensive calculations. Fusing new data, all pixels are projected to 3D coordinates which requires 640 x 480 = 307200 operations for a VGA resolution.

Furthermore, during TSDF update and rendering, each pixel requires $\frac{|d_{max}-d_{min}|}{\tau}$ operations. $|d_{max}-d_{min}|$ is the maximum ray length and τ is the voxel size. Therefore, ray casting is known as the most computationally intensive operation in any type of dense RGB-D SLAM system. Since each pixel is independent from each other, GPU can be utilized in parallel to have a real time performance. However, TSDF operations can be split into independent task for parallel processing.

TSDF can encode surface at sub-voxel accuracy by interpolation, however, it can fail at sharp corners and edges. Therefore, such type of structures requires special effort or selecting a suitable voxel element size and truncation distance. Truncation distance expresses a prior information about the average thickness of object in the environment. If our interest are tables, chairs or objects like that, it is easy to decide the truncation distance, however, it is not easy to know which type of object encountered for. In our experiments choosing larger value results less accurate reconstruction, while, smaller distance value leads to more accurate construction with more detail.

Studies of accuracy for the Kinect sensor show that assuming a Gaussian noise model for the normalized disparity provides good fits to observed measurement errors on planar targets.

$$p(z_{obs}|z_{true}) \propto exp^{\left[-\frac{(z_{true}-z_{obs})^2}{\sigma^2}\right]} \tag{4.24}$$

In principle, the noise of the any disparity based distance sensors is quadratically proportional to the distance, $\sigma \propto {depth_{true}}^2$(Equation 3.164). However, in our current implementation, we assume a fixed σ over all pixels. Assuming independent and identical distributed pixels with Gaussian noise in depth values , the likelihood of observing a depth image D from the pose of the camera $(Rx_{i,j}+T)$ becomes

$$p(D|(R,T)) = \prod_{i,j} p\left(D_{i,j}\middle|(R,T)\right) = \prod_{i,j} e^{\left(\frac{-\psi\left(Rx_{i,j}+T\right)^2}{\sigma^2}\right)} \tag{4.25}$$

Our aim is to find a camera rotation R^* and translation T^*, which maximize the likelihood of observing a depth image.

$$(R^*, T^*) = \underset{R,T}{\text{argmax}} \prod_{i,j} e^{\left(\frac{-\psi\left(Rx_{i,j}+T\right)^2}{\sigma^2}\right)} \tag{4.26}$$

In order to find the camera poses that maximizes this likelihood, we define a pose error function $E(R,T)$ by taking the negative logarithm of (4.26) in order to simplify the calculation we define the error function $E(R,T)$ as

$$E(R,T) = \sum_{i,j} \psi\left(Rx_{i,j} + T\right)^2 \tag{4.27}$$

A rigid-body motion can be described in Lie algebra with the 6-dimensional twist coordinates $\xi = (\omega_1,\omega_2,\omega_3,\upsilon_1,\upsilon_2,\upsilon_3)$. If we rewrite the error function (7) then it becomes

$$E_p(\xi) = \sum_{i,j} \psi_{i,j}\left(\hat{\xi}\right)^2 \tag{4.28}$$

$$\psi_{i,j}(\xi) = \psi\left(Rx_{i,j} + T\right). \tag{4.29}$$

The vector $\xi = (\omega_1,\omega_2,\omega_3,\upsilon_1,\upsilon_2,\upsilon_3)$ can be converted to into the corresponding Lie group SE(3) by computing $T = exp^{\xi}$, where:

$$\xi = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & \upsilon_1 \\ \omega_3 & 0 & -\omega_1 & \upsilon_2 \\ -\omega_2 & \omega_1 & 0 & \upsilon_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.30}$$

If image registration is correct with the 3D model, the projected colors should be consistent as well. We incorporate this color consistency condition by adding color error function $E_c(\xi)$, which we define in (4.31). Since there is no absolute reference of the image for comparison, color values stored in the voxels are used. Using color intensities of the pixels $I(p_i)$ and corresponding voxels $V_I(\hat{\xi})$, then the error function becomes

$$E_c(\xi) = \sum_{i,j} \left(V_I(\hat{\xi}) - I(p_i) \right)^2 . \tag{4.31}$$

Then , we define the joint error function is given in (4.32) with υ_w being the intensity contribution with respect to the depth

$$E(\xi) = E_p(\hat{\xi}) + \upsilon_w E_c(\hat{\xi}) \tag{4.32}$$

and we start by linearizing ψ around the initial pose estimate $\hat{\xi}$ using the Jacobian matrix. The values of the Jacobian indicate the change of the error when translating along or rotating around the respective component of ξ. In addition, the Jacobian matrix is the derivative of our signed distance function with respect to rigid body transformation parameters $\xi = (\omega_1, \omega_2, \omega_3, \upsilon_1, \upsilon_2, \upsilon_3)$. We can compute the gradient of SDF with respect to parameters as

$$\nabla_x \psi(\mathrm{x}) = \frac{\partial \psi(\mathrm{x})}{\partial x} = \left[\frac{\partial}{\partial x_1} \psi(\mathrm{x}) \quad \frac{\partial}{\partial x_2} \psi(\mathrm{x}) \quad \frac{\partial}{\partial x_3} \psi(\mathrm{x}) \right] \tag{4.33}$$

This gradient can be calculated by numerically differentiating ψ(x) using central difference over x_1, x_2 and x_3. However, this term does not include any derivation with respect to ξ. In order to have a complete expression for Jacobian of pose error $J_p(\xi)$, we use the chain rule:

$$J_p(\xi) = \frac{\partial \psi_{i,j}(\hat{\xi})}{\partial \hat{\xi}} = \frac{\partial \psi_{i,j}(\hat{\xi})}{\partial \hat{\xi}} \frac{\partial (\hat{\xi})}{\partial \xi} \tag{4.34}$$

Equation (4.34) needs for an expression how the position of given point change with our transformation parameters $\xi = (\omega_1, \omega_2, \omega_3, \upsilon_1, \upsilon_2, \upsilon_3)$ and this can be obtained by analyzing $\hat{\xi} = e^{\xi \Delta t} x$ with respect to ξ. For a given point $\hat{\xi}$, then we have;

$$\frac{\partial (\hat{\xi})}{\partial \xi} = \begin{bmatrix} 0 & \upsilon_3 & -\upsilon_2 & 1 & 0 & 0 \\ -\upsilon_3 & 0 & \upsilon_1 & 0 & 1 & 0 \\ \upsilon_2 & -\upsilon_1 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.35}$$

In (4.34), $\frac{\partial \psi_{i,j}(\hat{\xi})}{\partial \hat{\xi}}$ is computed numerically by evaluating the gradient. We compute the derivatives using a 3x3 Sobel-Feldman operator, which is a discrete differentiation operator . Equation 3.33 is the example of the derivative filter.

$$SoFe \in R^3 = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{4.36}$$

$$SoFe \in R^3 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{4.37}$$

We compute the Jacobian for color error term $J_c(\xi)$ as

$$J_c(\xi) = \frac{\partial\psi_{i,j}\left(\sum_{i,j}\left(V_I(\hat{\xi}) - I(p_i)\right)^2\right)}{\partial\hat{\xi}}\frac{\partial(\hat{\xi})}{\partial\xi} \tag{4.38}$$

$$\frac{\partial(\hat{\xi})}{\partial\xi} = \begin{bmatrix} 0 & \upsilon_3 & -\upsilon_2 & 1 & 0 & 0 \\ -\upsilon_3 & 0 & \upsilon_1 & 0 & 1 & 0 \\ \upsilon_2 & -\upsilon_1 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.39}$$

We have already derive the joint error function as

$$E(\xi) = \underbrace{\sum_{i,j}\psi_{i,j}(\hat{\xi})^2}_{E_p(\xi)} + \upsilon_w \underbrace{\sum_{i,j}\left(V_I(\hat{\xi}) - I(p_i)\right)^2}_{E_c(\xi)} \tag{4.40}$$

We adopt to use the Levenberg Marquardt algorithm because Gauss-Newton cannot calculate the best optimal estimate, resulting in non-minimum function value. Levenberg-Marquardt algorithm can handle this problem in standard form of

$$[A + \lambda I]\Delta\xi = -\mathrm{b} \tag{4.41}$$

where λ is the non-negative correction factor updated at each iteration. Levenberg-Marquart method solves Equation (4.40) using Jacobian as

$$\left(J^T J + \lambda(J^T J)\right)\delta = J^T f \tag{4.42}$$

In our case, we want to try to find the pose that minimizes the combination of pose and color error in Equation (4.32). After we compute the Jacobians, we write the matrix $J^T J$ and the vector $J^T f$

$$
\begin{aligned}
J^T J &= \sum_i J_{p,i}{}^T J_{p,i} + \upsilon_w \sum_i J_{c,i}{}^T J_{c,i} \\
J^T f &= \sum_i J_{p,i}\, \psi_{i,j}(\hat{\xi}) + \upsilon_w \sum_i J_{c,i} \left(V_I(\hat{\xi}) - I(p_i) \right)
\end{aligned}
\tag{4.43}
$$

Algorithm 2 summarizes the pose estimation process we just elaborated.

Algorithm 2. Pose estimation algorithm

```
Input : Joint error function
Output : Pose
begin
  Initialize parameters $c_x, c_y, f_x, f_y$
  Calculate Jacobian
      $J_p(\xi) = \frac{\partial \psi_{i,j}(\hat{\xi})}{\partial \hat{\xi}} \frac{\partial(\hat{\xi})}{\partial \xi}$
      $J_c(\xi) = \frac{\partial \psi_{i,j}\left(\sum_{i,j}\left(V_I(\hat{\xi}) - I(p_i)\right)^2\right)}{\partial \hat{\xi}} \frac{\partial(\hat{\xi})}{\partial \xi}$
  Initialize non-negative correction factor as Gramian of Jacobian
  while (pose difference) > 0.001 or iteration # <5  do
      Find $\delta$ increment  for  $\left(J^T J + \lambda(J^T J)\right)\delta = J^T f$
      Update pose with increment
      if  objective function is minimum
          return pose
      else
          Update correction factor if needed
          Increment iteration number
end
```

We solve (4.42) iteratively until difference $(\xi(k+1)-\xi(k))$ is small enough or the maximum iteration number is reached. In order to increase real time performance, we conduct all calculations on the GPU in parallel since the matrix $J^T J$ and the vector $J^T f$ are independent of each other. Even though, a maximum iteration number are performed or optimization parameters yields values below the pre-defined threshold, in order to improve the convergence probability, we select to scale the contribution of new data based on a weighting function. To improve the basin of convergence for the solution, we scale the contribution of each measurement, based on a weighing function. This produces the standard iteratively reweighed least-squares algorithm. As a weighting function a possible choice are either Huber estimator or the Tukey's estimator. Huber and Tukey's estimator are given respectively as

$$\rho_h(\mathrm{x}) = \begin{cases} 1.0 & if\ |x| \leq \rho_h \\ \dfrac{\rho_h}{|x|} & otherwise \end{cases} \tag{4.44}$$

$$\rho_t(\mathrm{x}) = \begin{cases} \left[1-\left(\dfrac{x}{\rho_t}\right)^2\right]^2 & if\ |x| \leq \rho_t. \\ 0 \quad otherwise \end{cases} \tag{4.45}$$

Here, ρ_h and ρ_t are small constants typically smaller than a tenth of the size of voxel. We use Huber estimator because of computational advantage. Furthermore, Huber estimator produces small residuals and is convex in contrast to the Tukey function, which Huber estimator does not cause new local minima [90][91]. Finally, the matrix $J^T J$ become

$$J^T J = \rho\left(\psi(\hat{\xi})\right)\sum_i {J_{p,i}}^T J_{p,i} + \upsilon_w \rho\left(\psi(\hat{\xi})\right)\sum_i {J_{c,i}}^T J_{c,i} \tag{4.46}$$

$$J^T f = \rho\left(\psi(\hat{\xi})\right)\sum_i J_{p,i}\,\psi_{i,j}(\hat{\xi}) + \upsilon_w\,\rho\left(\psi(\hat{\xi})\right)\sum_i J_{c,i}\left(V_I(\hat{\xi}) - I(p_i)\right) \tag{4.47}$$

In Equation (4.41), λ is a non-negative regularization term and I is the identity matrix. For λ, initial value is usually selected as 1/10 of the voxel size, however, in order to select the Huber and regularization term, sensitivity analysis is performed. Table 4.1 shows the obtained absolute trajectory error by using the BONN moving obstructing box dataset. In this analysis, a voxel size of 0.01 m is used. The best value of trajectory error is 0.298 m which is obtained when Huber constant and regularization term is 0.02 and 0.1 respectively.

Table 4.1: BONN moving obstructing box dataset – change of translational ATE (RMSE cm) with respect to Regularization and Huber constant

Absolute Translation Error (RMSE meter)		Regularization			
		0.002	0.005	0.02	0.1
Huber	0.005	0.504	0.503	0.493	0.476
	0.002	0.737	0.320	0.600	0.554
	0.02	0.323	0.320	0.322	0.298
	0.01	0.335	0.400	0.410	0.362

SDF calculation and update process are independent for each voxel and most of the calculation in registration is done for each pixel in the depth image. The matrix $J^T J$ and the vector $J^T f$ are the results of sums over all the pixels. In order to increase the speed of solution and avoid from local minima, a coarse to fine iteration level of detail is used. We compute the derivatives of SDF while constructing the Jacobians. In order to increase the iteration number capability, we use a sub-sampling ¼ of the pixels. If we use ¼ downsampled image and original image, absolute trajectory error almost the same level. Table 4.2 shows the absolute trajectory error obtained by BONN moving obstructing box dataset original and downsampled image. Using this technique, we can process larger number of inexpensive calculation without changing the convergence. However, increasing the iteration number from 2 to 3, which leads to increase in computational cost, also increases absolute trajectory error.

Table 4.2: BONN moving obstructing box Dataset – comparison of translational ATE (RMSE cm) with changing Iteration Number and Downsample option

ATE (rmse) m	Iteration Number			
	1	2	3	6
Original	0.312	0.265	0.308	0.293
¼ Downsample	0.313	0.263	0.306	0.293

4.5 SDF Representation, 3D Reconstruction and Volume Visualizations

With the appearance of 3D sensor such as laser scanners, 3D reconstruction has received high degree of attention. This enables today indoor building model as well as outdoor navigation and geometry aware inspection. However, generally sensors produce large data of 3D point and, therefore, processing and visualization of 3D data become a challenging task. Many 3D reconstruction algorithm in robotics process data on an iterative manner by accessing the sample in order to generate the observed surface. Accessing a given point defined by 3D coordinate system becomes a challenging job because amount of sample in the scene or database which affects the complexity of the overall system. Most common approach to handle data is to group them on a grid and store in a array in order to access them quickly. However, memory requirements depend on the cube of size, which hinder large scale reconstruction. For example, representing a scene in the form cube 10x10x10 m^3 at

1 cm resolution, would require 3.7 GB memory, assuming 4 byte for each grid. Common remedy for this problem is to use octree for representing the occupied cells [45][46]. An octree is a tree data structure where each internal node can have at most 8 children, as shown in Figure 4.12.

The main drawback of the octree representation of 3D object in visualization is that there exists only a single primitive element, cube, therefore the precision heavily depends on the size of this cube. Figure 4.13 illustrates the different levels of an octree-based model. Each picture shows a deeper level of the octree compared to the previous one, i.e., in each successive picture the voxels are two times smaller.

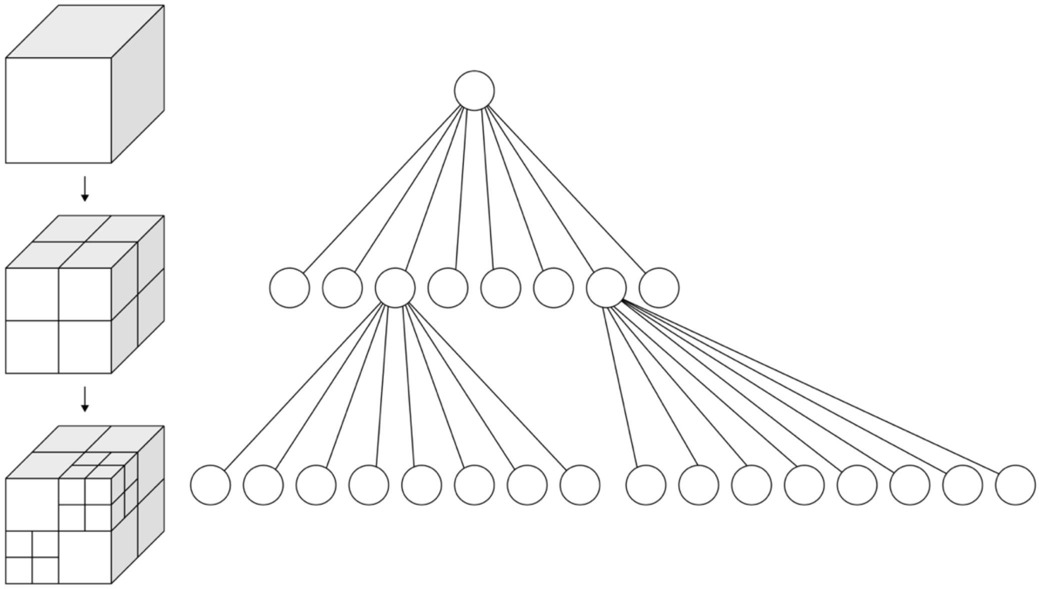


Figure 4.12: Octree structure. In each successive picture the voxels are two times smaller.

In addition to the primitive element size, it is necessary to navigate the full octree when an arbitrary voxel information is requested. Virtual memory or paging is introduced by Bridson [47], which enables significant reduction in memory requirements as well as data to be stored on external storage. However, memory overhead is still a significant problem. In [48], it is proposed to encode the location of voxel with a hash function in order to enable large scene to be managed with limited memory. Nießner [10] proposes a new voxel hashing scheme which is memory and computational power efficient. In this technique, surface data are stored only where the measurements are observed and data can be exchanged easily efficiently through hash table in both ways.

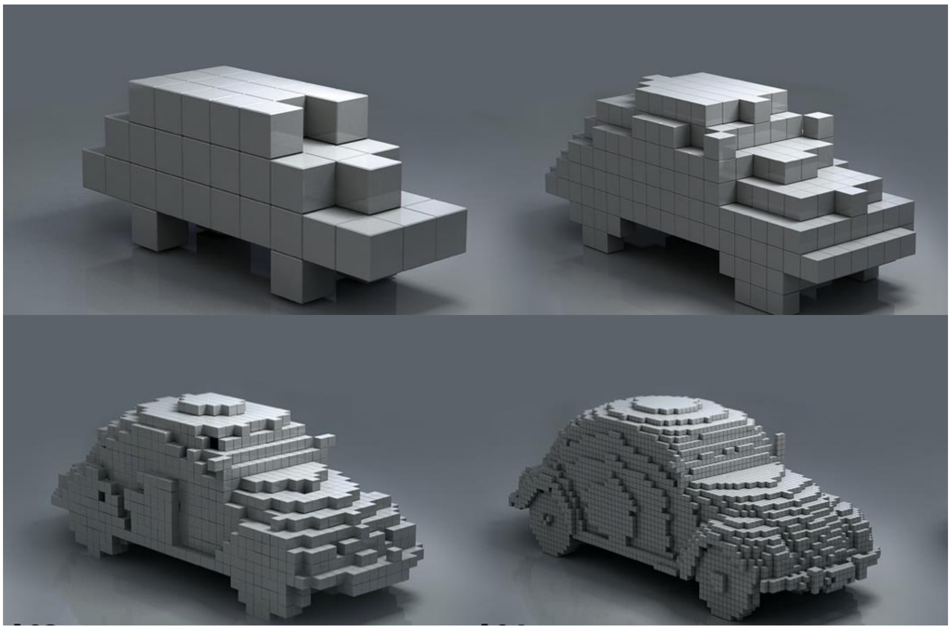


Figure 4.13: Different levels of an octree-based model, in each successive picture the voxels are two times smaller

In order to decrease the memory requirement and large scale mapping, we use a voxel hashing scheme similar to [10]. In this way, only voxels corresponding to surfaces or obstacles are indexed.

Figure 4.14 illustrates our voxel hashing data structure. The idea is to grid the environment with small voxel blocks composed of 8x8x8 voxels. Each voxels stores SDF, color and weight as

```
struct Voxel          {

                          float sdf; /** Signed distance function */
                          uchar3 color;  /** Color */
                          unsigned char weight; /**SDF weight*/
                       }
```

Using GPU accelerated hash table, voxels blocks can be retrieved quickly. Coordinates of a 3D point is calculated by multiplication and rounding operation. Mapping from world coordinate to hash value is done using the following hashing function.

$$H(x,y,z)=(x\cdot p1 \oplus y\cdot p2 \oplus z\cdot p3) \bmod M \tag{4.48}$$

where p1 , p2 , and p3 are large prime numbers (in our case 73856093, 19349669, 83492791 respectively, based on [48], M is the number of buckets, $\oplus$ is the logical XOR operator, and mod is the modulo operator. In small hash table, there exists collisions, however, even with large hash tables and excellent hash functions, collisions are inevitable. Using large prime numbers reduces the risk of collision.

Volume visualization is an important field of computer science, which provides key elements in order to discover and explore unseen structures of a volumetric data and allow users to get visual insight of complex datasets. History of volume visualization begins in 1970’s , when primary use of volumetric data in 3D medical imaging.

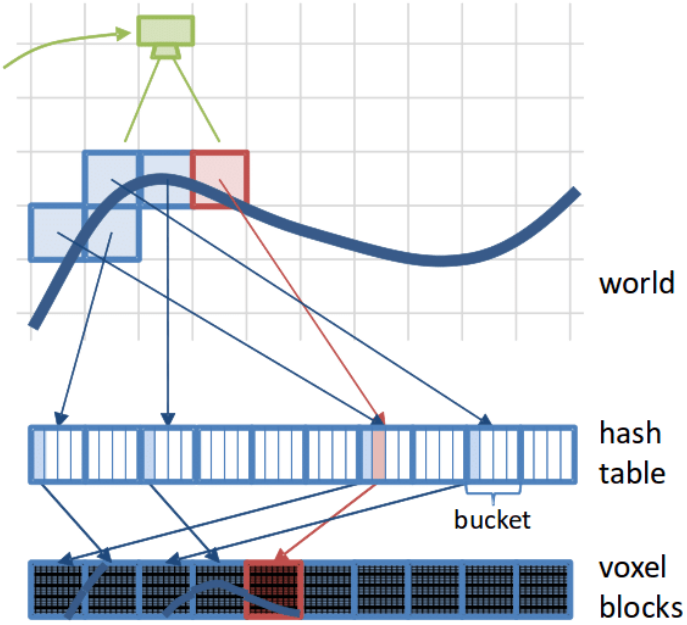


Figure 4.14: Hash table , voxel blocks

Volume visualization pipeline consist of data acquisition, enhancement of the data, reconstruction of the 3D voxel model and shading of the 2D projection. An object with volumetric data is represented by a voxels which are counterpart of pixel in 3D. It can store numeric value of associated independent variables such as density, color, material, deformation of real objects. The data reference for a voxel can be a discrete sample of medical imaging or synthetic computational model such as fluid dynamic analysis. Independent from the source of data, volume can be stored by using primitive models ranging from 1-D element such as point or particles to 2-D polygon meshes or curved surface as well as 3-D volume elements voxels [51].

In order to extract the primitives from the volume several methods are available. For example, cuberille method where space is meshed into equal cubes by three orthogonal set of equally spaced parallel planes [52]. In addition, cloud of points can be formed by the dividing cubes algorithm [53] which subdivides the voxels into smaller cubes that lie on the surface of the object and projects the intensity calculated for each cube onto the viewing plane, forming a gradient shaded representation of the three-dimensional object Figure 4.15.

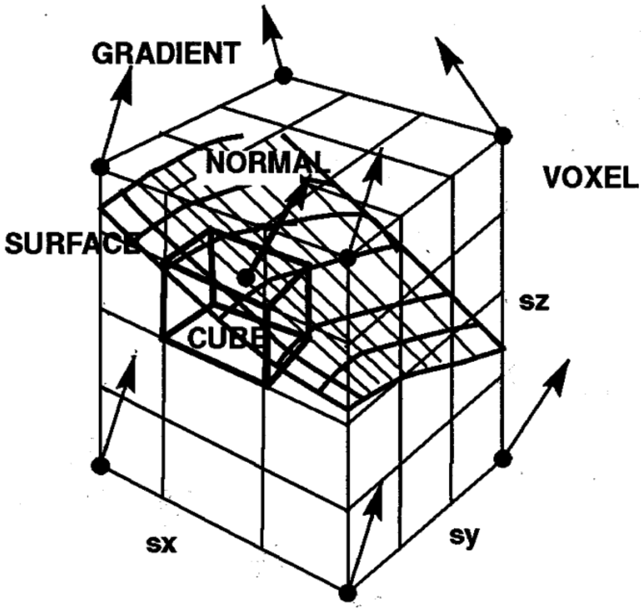


Figure 4.15: Gradient shaded representation of the three-dimensional in dividing cube algorithm [56]

In addition, fine polygon mesh can be generated by marching cubes algorithm which is originally described by Lorensen and Cline [54] which, takes as its input a regular scalar volumetric data set that has a scalar value residing at each lattice point of a rectilinear lattice in 3D space. This algorithm uses a volume unit called a cell and a look up table of possible polygon shape. Each of the vertices of a cell can be marked as free or occupied, therefore there exists (256) 28 possible configurations. However, possible configuration decreases to 15 if reflection and symmetric constraints are considered. Illustration of reflective and symmetric conditions are depicted in Figure 4.16. The 15 unique cube–isosurface intersection scenarios result when considering both of these symmetries are shown in Figure 4.17. Furthermore, Cases 11 and 14 are considered as mirror symmetric, use of all three conditions (reflective, symmetric and mirror symmetric) in union results in 14 basic topologies.

In order to have more accurate surfaces, the final value is obtained by interpolating between the voxels connected to the edge where the surface might be. For instance, consider the Case 1 in Figure 4.17, if the voxel values are stored in a range [0,255] and surface threshold is 128. Also, suppose that value stored in the bottom left voxel is 150 and the bottom right voxel has a value of 100, then linear interpolation yields

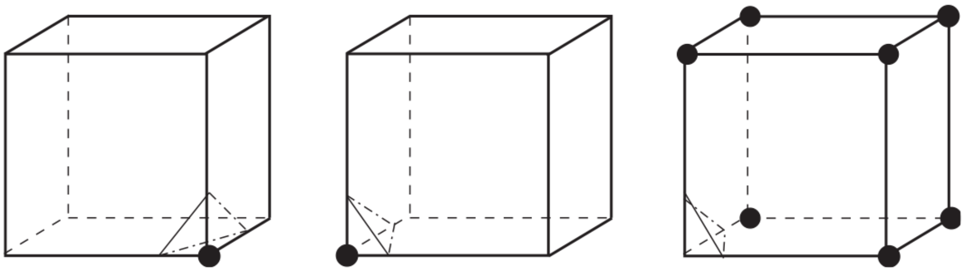


Figure 4.16: Illustration of reflective, original and rotational symmetric condition

$$\frac{150-128}{200-100}=0.32$$

Hence, the vertex is placed at a fraction of 0.32 between voxels. This step results in a better fit of the surface.

In short, we use the discrete voxel grid in order to represent the SDF. Signed distance value is calculated by trilinear interpolation of eight neighboring pixels. We project each voxel onto the image plane instead of ray casting because this process is suitable for parallel processing since each voxel is independent of its neighbors. Since the operation has to be carried out for each voxel, GPU is used for this operation. Finally, we implement marching cubes algorithm [25] in order to extract the triangle mesh. In RGB-D mapping approaches, storing the SDF in a 3D grid requires large amount memory. Therefore, we use a special memory allocation technique proposed by Nießner in [10]. In this technique, we only allocate the voxels in required areas, which, enables scanning the large areas with limited memory.

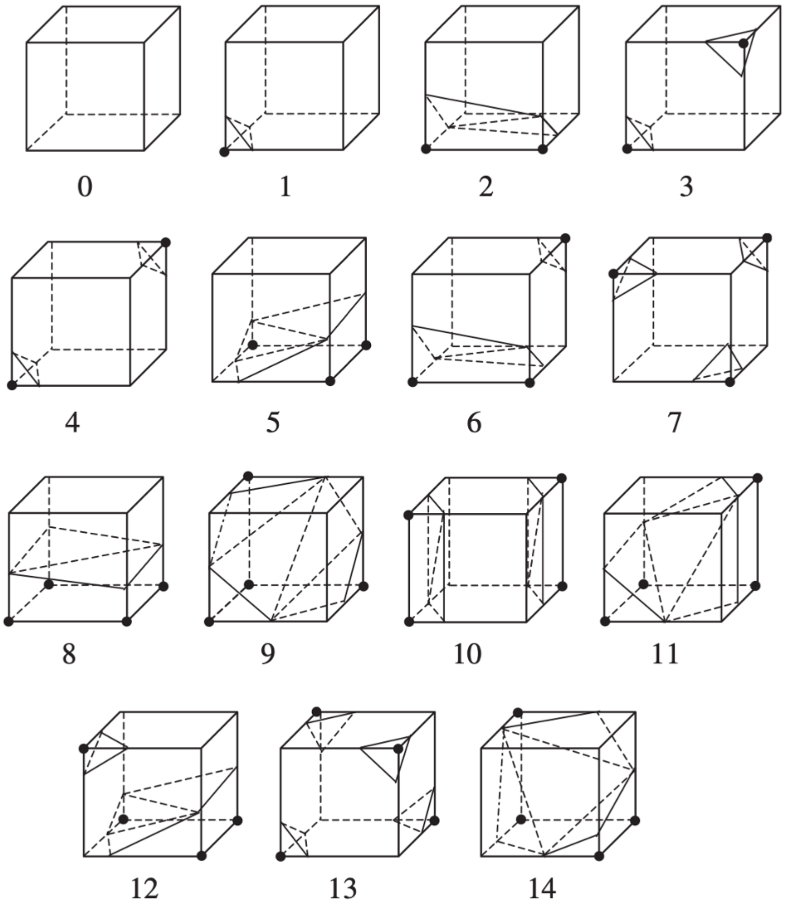


Figure 4.17: The 15 basic intersection topologies [55]

4.6 Dynamic Detection

Let I_m and I_s be the instantaneous image of generated model and source respectively, The error in color map denoted by e_c as

$$e_c = |I_{s\leftarrow m} - I_s| \tag{4.49}$$

If the images I_m and I_s are not accurately registered and if there is change in the geometry, the resulting error would not be zero (Figure 4.18). In general, minimizing (4.47) results in a sufficient image registration. SDF represents the distance to nearest surface and therefore we select to use SDF as an error function. The error in the depth can be written as

$$e_p(\hat{\xi}) = \sum_{i=1}^{N} \left\| \psi_i(\hat{\xi}) \right) \|^2. \tag{4.50}$$

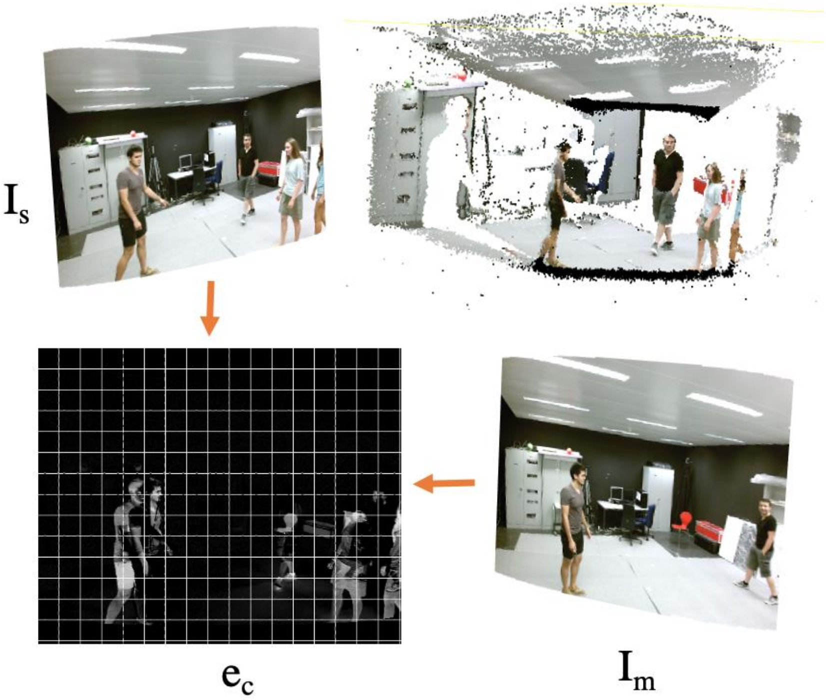


Figure 4.18: Inconsistency map of two images (EPFL RGB-D Pedestrian Dataset sequence frame 250 and 278)

In Equation (4.48), N is pixel number, ψ is the signed distance function and $\hat{\xi}$ is the matrix exponential multiplied by the 3D point corresponding to the i^{th} pixel p_i computed using homogenous point equation in (4.1).

After performing an initial registration using Equation (4.48), we compute for each pixel, its residual as defined in (4.49).

$$r_i = \left\| \psi_i(\hat{\xi}) \right\|^2 \tag{4.51}$$

The residual obtained after image registration is used as for dynamic detection. Our aim is to compute the binary labeling for each element according to occurred changes. For example, $l_i = 0$ indicates consistency and $l_i = 1$ shows the presence of change in corresponding voxel i.

If $h(d)$ is the histogram of the image, our problem is in the form of binary classification problem using a dynamic label threshold. Then, the probability density function can be defined as the combination of two density functions related class label as

$$p(D) = \sum_{i=1}^{2} P_i \, p_i(D|l_i) \tag{4.52}$$

using class conditional densities and prior probabilities. In order to calculate an estimate of dynamic change, we maximize $p(l|D)$

$$l^* = {}^{argmax}_{\;\;\;\; l} \{L(D|l)\} \tag{4.53}$$

where $L(D|l)$ is the log likelihood of the two-component mixture and it can be written as

$$L(D,l) = \sum_{x=0}^{L-1} h(d) \ln p(D|l)\,. \tag{4.54}$$

The final log-likelihood function is in the form of

$$L(D,U,l) = \sum_{i=1}^{2} \sum_{d=0}^{L-1} h(d) u_i(d) \mathrm{l\,n}\, \{P_i p_i(d|l_i)\} \tag{4.55}$$

$$p(d|l) = H(y) \frac{2}{\rho_c \sqrt{2\pi}} e^{-\frac{y^2}{2{\rho_c}^2}} \tag{4.56}$$

In (4.53) $u_i(d)$ is the indication of static or dynamic component.

After dynamic label identification and updating the label grid (Algorithm 2), a second pose estimation and registration are performed using the newly obtained label set (Algorithm 3). However, we must filter out dynamic labels that originated from noise (Figure 4.19). In order to that, we compare the SDF value of new observation with the previous static reconstruction and compute the differenceδ_L. Applying a threshold θ, we obtain the label grid such that

$$l_i = dynamic \text{ if } \delta_{\mathrm{L}} > \theta. \tag{4.57}$$

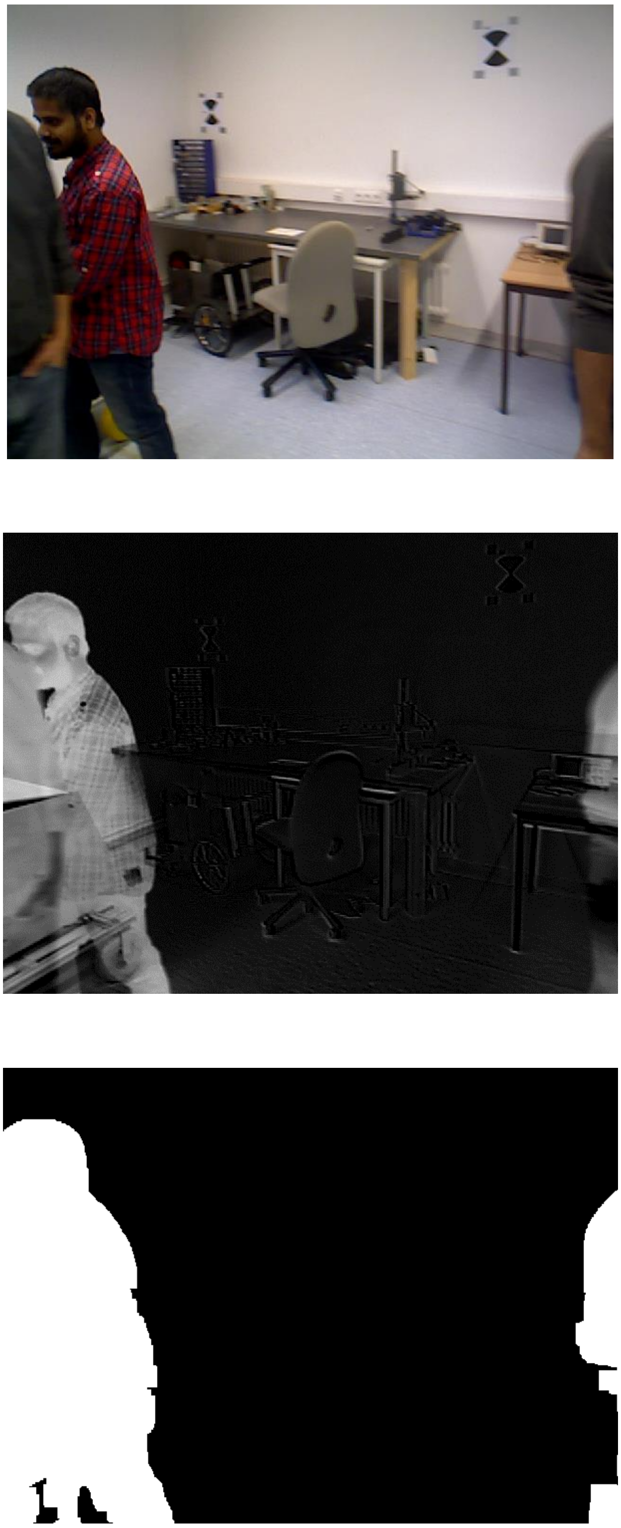


Figure 4.19: RGB image b) Reconstruction error c) Dynamic label image

Algorithm 3. Dynamic labelling algorithm

```
Input : Depth image, prior segmentation from residual error, initial label class

Output : Segmented depth image with label
Initialize parameters
Find maximal cliques
Construct k-neighborhoods
Partition into parallel threads
do each EM iteration
    for each neighborhood of the subgraph do in parallel
        E-step
        M-step
    end for
    Update parameters
while Likelihood increment < threshold
return Label set
```

Algorithm 4. DUDMAP

```
Input : Depth image, RGB image
Output : Artificial camera view, mesh (optional)
Initialize parameters for sensor, camera tracking, SDF
if frame number = initial frame
    Initialize poses as identity
    for $i \in N$ (number of pixel)
        Initialize labels as static
    end for
    Pose estimation using matrix exponential
else
    RGB similarity check
    Pose estimation using matrix exponential
    Generate label set
    Re-pose estimation using matrix exponential with label set
Volume integration
    Update parameters
while   Frame number < total number of frame
Extract mesh
return  Final mesh
```

CHAPTER 5

EXPERIMENTAL RESULTS AND DISCUSSIONS

Our proposed method is able to operate in dynamic environments without requiring any dynamic object detection and tracking. Our experiments support our main claims, which are:

- Robustness to dynamic elements regardless of their quantity and speed of change in the environment,
- That the approach requires no explicit object tracking, and no object classifier
- That the approach generates a consistent a dense model of the environment.

The experiments were conducted on a workstation computer Intel i7 running at 3.20 GHz and a GeForce 1070 GPU using Ubuntu 16.04. Our default parameters have been determined empirically so that a sensitivity analysis is performed on changes of parameters the system is sensitive to. Our experiments demonstrate many evaluation scenarios.

1. In order to evaluate the consistent final mesh generating performance, a static environment is used with ground truth selected as repository datasets namely ICL NUIM and CoRBS (Section 5.1).

2. In real life, there are a lot of dynamic movements (e.g., of cars, people, animals) that affect the visual odometry calculations. In order to evaluate the performance of our proposed algorithm in dynamic environment, we compare our method with other state of art repositories using TUM dataset [1], together with other high dynamic dataset including Bonn [14],

VolumeDeform [30], CVSSP RGB-D dataset [34] (used with permission), which are publicly available in order to show the superior performance of our approach (Section 5.2).

3. In addition, outdoor performances of our method are also evaluated using commercially available ZED camera for map generation and dynamic filtering (Section 5.3).

5.1 Static Environment

The focus of this part is to show that our proposed approach can generate 3D mesh of scene in static environment because this is the key part of mapping algorithms. Two datasets which are ICL NUIM and CoRBS are used in this part.

5.1.1 ICL NUIM Dataset

The ICL-NUIM dataset provides a benchmarking environment for RGB-D, Visual Odometry and SLAM algorithms. In addition, all data contained by ICL NUIM dataset are compatible with the evaluation tool provided by the TUM RGB-D dataset. There are two different sequences, living room and office room scene with a corresponding ground truth. Since the living room has 3D surface ground truth together, it is a perfect tool for evaluation of camera trajectory and reconstruction of SLAM performance. Figure 5.1 illustrates the sample RGB image of ICL-NUIM living room dataset containing a chair, a table and another type common object used in daily life. We use “living_room_traj2_frei” sequence, which has 882 frames for 30 sec for 3D mesh generation. Table 5.1 shows the comparison of surface reconstruction accuracy of the SLAM algorithms and Figure 5.2 illustrates the resulting dense 3D model. Our proposed method (DUDMAP) achieves 0.004 m mean value which is half of the Elastic Fusion (Figure 5.4). In addition, Figure 5.4 also shows that among 4782720 element, the maximum absolute distance error is

0.045 m and average error is 0.004 m. Surface reconstruction quality is measured using CloudCompare program and frequency of residual of the reconstruction error is given in Figure 5.3. The other successful methods including Kintinuous and RGB-D SLAM have slightly higher accuracy value 0.009 m and 0.031 m respectively. Therefore, our proposed methodology have a superior performance in 3D reconstruction of standardized static environment.

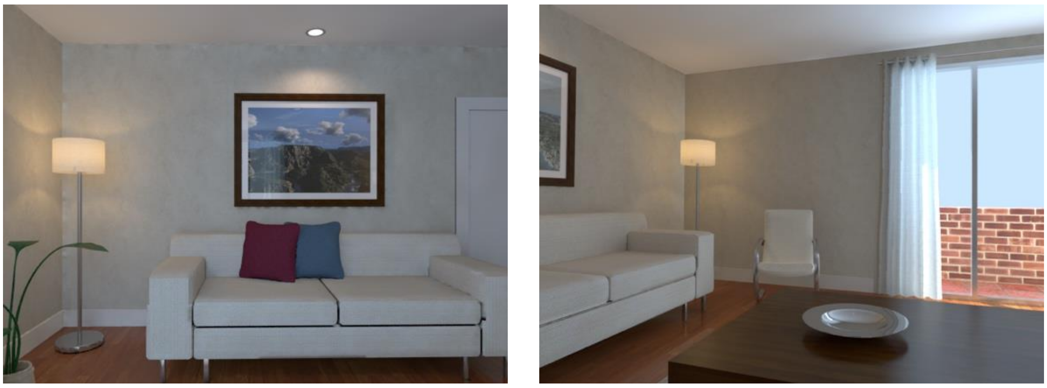


Figure 5.1: ICL-NUIM living room dataset containing chair, table and other type common object.

Table 5.1: Comparison of surface reconstruction accuracy (mean value)

DVO SLAM	0.119 m
RGB-D SLAM	0.031 m
MRSMap	0.098 m
Kintinuous	0.009 m
ElasticFusion	0.008 m
DUDMAP	0.004 m

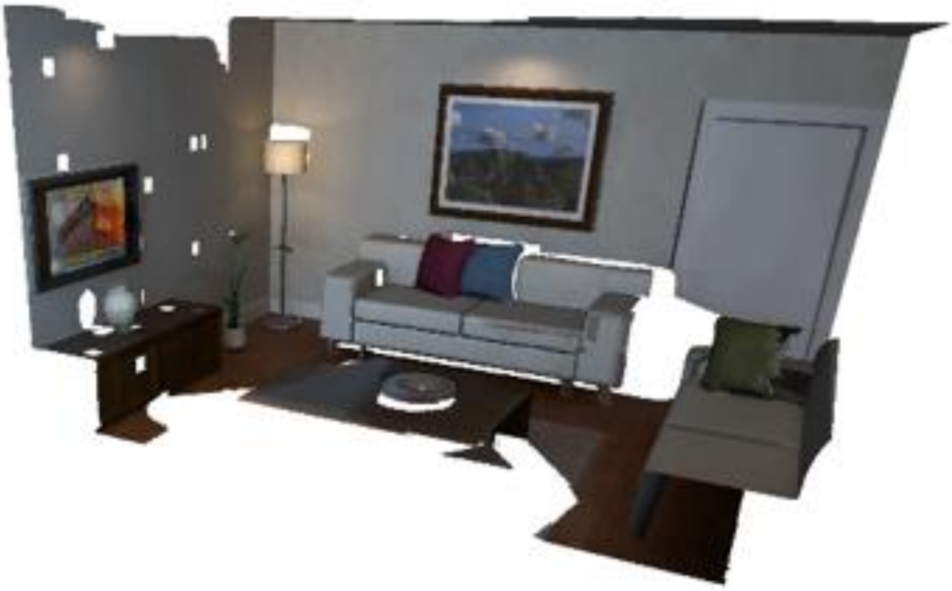


Figure 5.2: Dense 3D model of the scene obtained by our proposed methodology

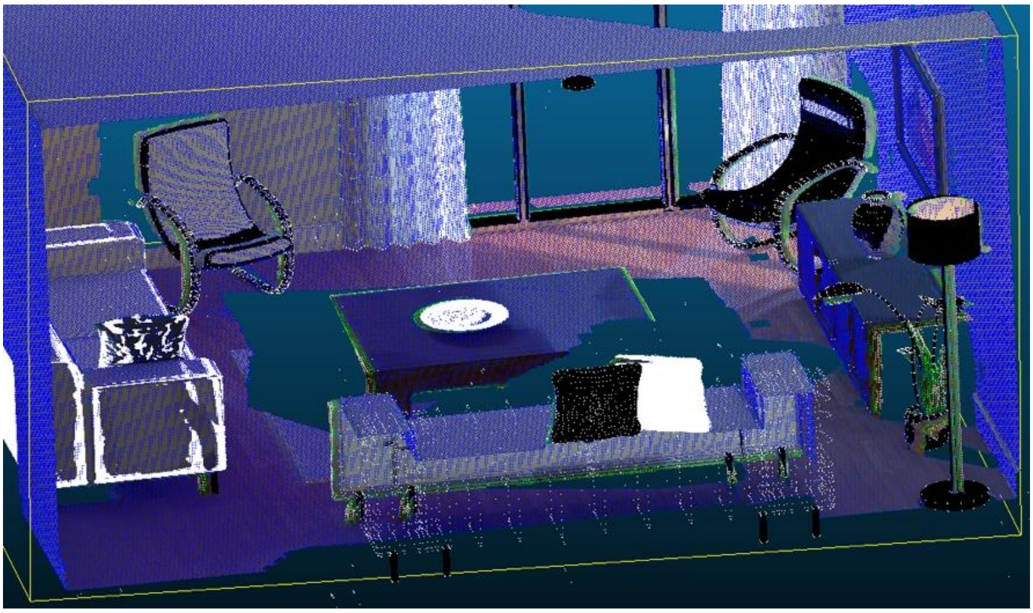


Figure 5.3: Reconstructions of the objects in living room - 2 scene including error value (blue 0.025 m, green 0.045 m)

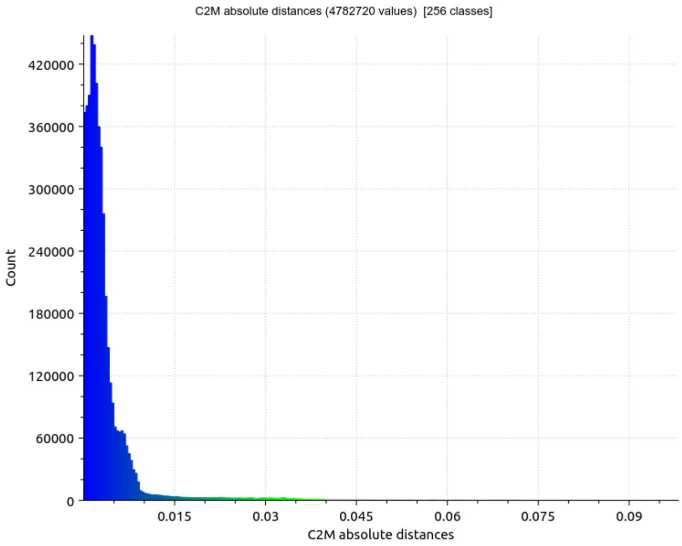


Figure 5.4: Frequency of residual of the reconstruction error.

Figure 5.5 shows the obtained trajectory of the ICL NUIM dataset-living room 2 sequence. The resulting absolute translational error is 0.004 m in RMS. The translational and rotational error are 0.007 m and 0.149 deg both in RMS.

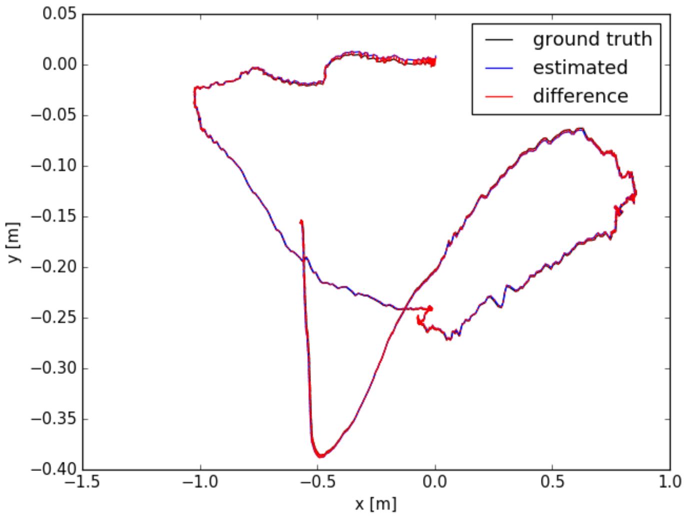


Figure 5.5: ATE RMSE of synthetic ICL-NUIM dataset

5.1.2 CoRBS

Comprehensive RGB-D Benchmark for SLAM (CoRBS) dataset is the combination of real depth and color data together with a ground truth trajectory. This dataset allows the user to independently evaluate the localization as well as the mapping portion of RGB-D SLAM systems using real data. In addition, it provides ground truth for the trajectory obtained by using an external motion capture system for the scene geometry via an external 3D scanner, yielding a sub-millimeter precision. In order to evaluate the 3D volume reconstruction and trajectory estimation performance of our proposed methodology in static environment, we use the D1 (Desk) sequence (Figure 5.6). It has 611 frames obtained by Kinect V2 camera with a resolution of 640x480 pixels. The duration of sequence is 23.4 sec. The average translation/rotational velocity in the sequence is 0.24 m/s and 35 deg/s with maximum of 0.6 m/s and 80 deg/s. The total bounding box of the 3D scene is 1.18 × 2.32 × 0.76 [m x m x m].

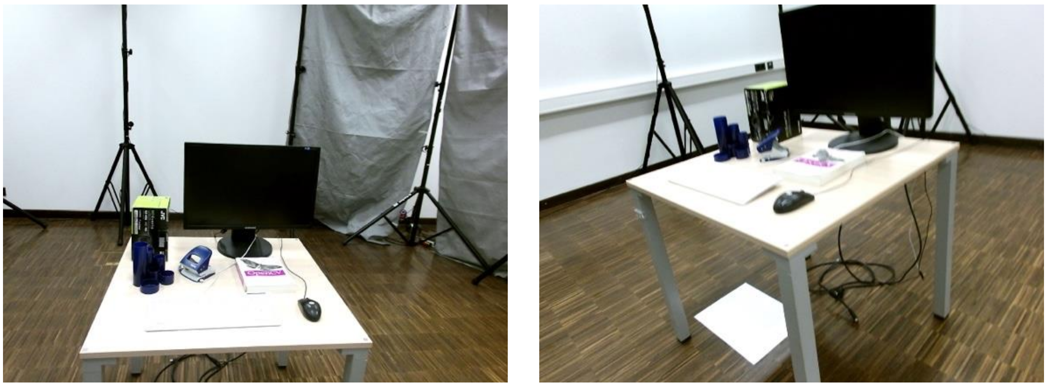


Figure 5.6: CORBS Desk1 dataset containing desk, computer, book, monitor and other types common object of everyday.

Figure 5.7 shows the obtained ATE / RPE plot of the D1 sequence. The resulting absolute translational error is 0.007061 m in RMS. The translational and rotational error are 0.015041 m and 0.914027 deg both in RMS. Figure 5.8 and Figure 5.9

illustrates the surface reconstruction capability of the our method. In Figure 5.9 both the ground truth model and resulting mesh is shown. We generate successfully the mesh of the environment with the consistent color of the objects. However, there exists some distortion because of the selected voxel size.

We compare against the state-of-the- art algorithm DVO as well as ICP which is used e.g. in KinectFusion, Elastic Fusion and other modern SLAM systems. We include DNA-SLAM, which is Dense Noise Aware Simultaneous Localization because this method is referred to its high accurately estimation. It uses a sophisticated weighting scheme for reducing noise characteristics in dense motion estimation and this weighting approach decreases the drift compared to DVO. Table 5.2 depicts the RMSE of the translational and rotational drift (RPE) in m/s and deg/s respectively for different sequences of the CoRBS dataset. The best results are depicted in bold. The trajectories estimated with ICP, DVO and DNA-SLAM exhibit a lower accuracy than our method (DUDMAP). Compared to other methods, the relative translational as well as the rotational error are substantially reduced in most sequences. For the sequences D1 and E1, we achieve the best accuracy.In all sequences, out method is closer to the ground truth than other methods. In summary, our method is seen to outperform in most of the sequences especially in rotational aspect because of the direct pose estimation and joint error weighting scheme for pose and intensity.

Table 5.2: RMSE of the translational and rotational drift (RPE) in m/s and deg/s respectively.

	D1 sequence		E1 sequence	
Algorithm	E_{trans}	E_{rot}	E_{trans}	E_{rot}
ICP [88]	0,04	2,08	0,08	4,42
DVO [87]	0,06	2,54	0,03	1,59
DNA-SLAM [89]	0,03	0,97	0,03	1,42
DUDMAP	**0,02**	**0,91**	**0,01**	**0,68**

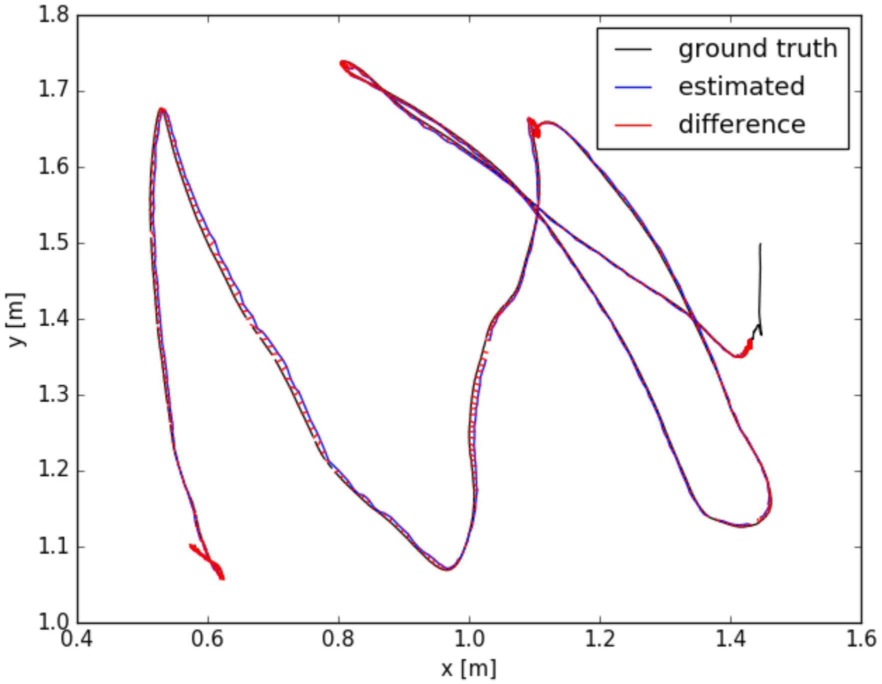


Figure 5.7: ATE RMSE of synthetic CoRBS dataset (Desk sequence)

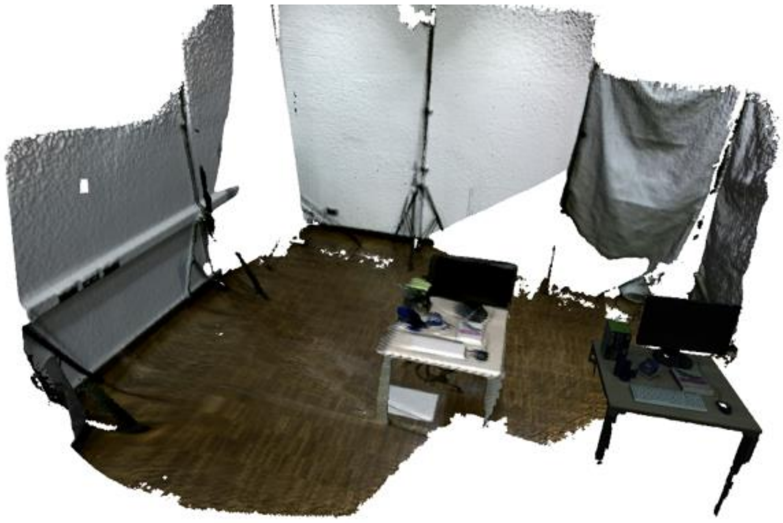


Figure 5.8: Dense 3D model of the scene obtained by our proposed methodology

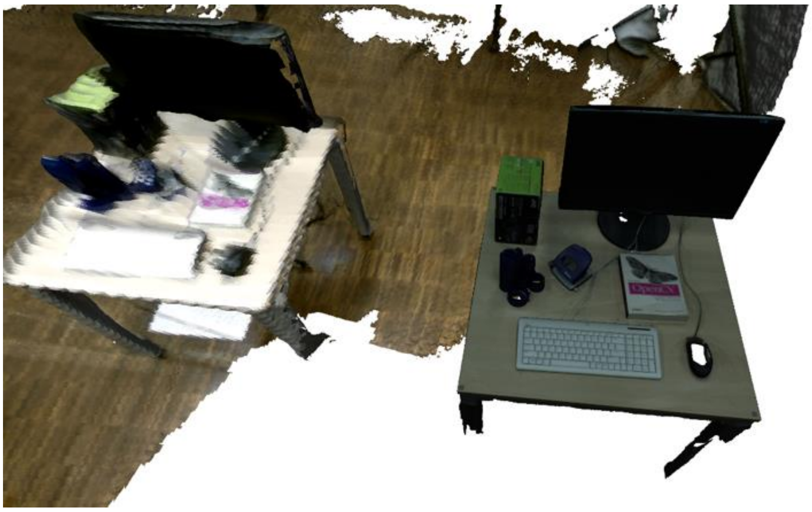


Figure 5.9: Dense 3D model of the scene obtained by our proposed methodology (Close up view)

5.2 Dynamic Environment

5.2.1 TUM RGB-D Dataset

In this dataset, there is a person sitting moving his/her arms: thus the sitting sequence has a relatively low dynamics. Whereas walking sequences are highly dynamic and complex because moving objects cover almost all camera views. In this dataset, the evaluation is performed through the metrics proposed in [1] as translational, rotational relative pose error (RPE) and translational absolute trajectory error (ATE). The obtained results of dense visual SLAM methods are listed in Table 5.4 – Table 5.6. In the TUM dataset, the ground-truth trajectory is obtained from a high-accuracy motion-capture system with eight high-speed tracking cameras (100 Hz). Therefore, a quantitative evaluation is possible regarding the accuracy of pose estimation. However, the TUM dataset has no exact 3D model of the environment, therefore we can evaluated the 3D reconstruction performance results of our method qualitatively. Qualitative results are shown in Figure 5.11, Figure 5.13 and Figure

5.14. Figure 5.12 also shows the scene reconstruction result of fr3/walking xyz sequence obtained by using Elastic Fusion, DynaSLAM and DS-SLAM.

In TUM dataset, fr3 denotes that the dataset sequence it belongs to, is freiburg3; sitting and walking represent two different character states, sitting is low dynamics example while walking is a high dynamics example; xyz, rpy, static, and half halfsphere stand for four types of camera ego-motions. For example, sit means that the person is sitting, and xyz means the camera moves along the x-y-z-axis. Basic properties of the sequences which used in evaluation is listed below.

- ‘freiburg3_sitting static’ sequence has a duration of 23.64 sec and average translational velocity is 0.011 m/s while and average angular velocity is 1.699 deg/s.

- ‘freiburg3_sitting xyz’ sequence has a duration of 42.51 sec and average translational velocity is 0.132 m/s while and average angular velocity is 3.562deg/s.

- ‘freiburg3_walking_static’ sequence has a duration of 24.83s sec and average translational velocity is 0.282 m/s while and average angular velocity is 1.388 deg/s.

- ‘freiburg3_walking_xyz’ sequence has a duration of 28.84 sec and average translational velocity is 0.208 m/s while and average angular velocity is 5.490 deg/s.

- ‘freiburg3 walking halfsphere’ sequence has a duration of 35.82 sec and average translational velocity is 0.221 m/s while and average angular velocity is 18.267 deg/s.

It is noted that, since only part of the human body is moving in the low dynamic dataset, when the visual odometry is tracking, the static part of the human body still provides pose estimation information. While the ultimate goal of this section is to further construct a static map by eliminating the influence of dynamic objects, our focus is on the end effect of static mapping.

As shown in Table 5.4, our proposed scheme achieves an average translation RPE error of 0.045 m/s, that is considerably lower than other dense methods such as VOSF, Elastic Fusion, Static Fusion and Mask Fusion. Our aim is to develop a dense RGB-D SLAM algorithm without using high computational power in dynamic environments. According to Table 5.4 – Table 5.6, our method achieves smaller relative and translational error than other dense method. For all high dynamic sequences, our method reaches the lowest RPE errors except for the “fr3/walk stat” sequence. In highly dynamic scene, our proposed method produces better results for the following reasons:

ElasticFusion is not capable of dynamics in the sequences. Hence, dynamic object deteriorates the 3D mesh and pose estimation. CoFusion works well for slow camera motions but its performance deteriorates noticeably when the speed of the camera increases. StaticFusion works sequences with limited dynamics at the beginning and therefore, it produces large errors on highly dynamic environment. In general, existing high dynamic in the scene leads to blurry motion in the image, resulting inconsistent mesh.

In addition, according to Table 5.4 till Table 5.6, we conclude that undoubtedly the semantic based Visual SLAM methods have better results based on ATE and RPE criteria. However, such methods do not provide a dense model and are relying heavily on the semantic segmentation prior result from the learning techniques. If an unlearned condition or object exists in the camera view, the estimation result is highly influenced.

Table 5.3 compares the execution time of our proposed method with semantic based SLAM algorithms. Semantic SLAM methods in Table 5.3 use either SegNet or Mask R-CNN. Mask R-CNN is a Convolutional Neural Network (CNN) and state-of-the-art in terms of image segmentation. SegNet, is designed to be an efficient architecture for pixel-wise semantic segmentation. It is primarily motivated by road scene understanding applications which require the ability to model appearance (road, building), shape (cars, pedestrians) and understand the spatial-relationship (context) be- tween different classes such as road and side-walk.

Segmentation stage takes 34 ms on average, however, Mask R-CNN require longer time about 200 ms. All semantic method except Mask Fusion are based on ORBSLAM, therefore it is included in timing analysis. Figure 5.14, Figure 5.15 and Figure 5.16 show that DynaSLAM has a satisfactory tracking performance. The addition of the multi-view geometry stage based on region growth algorithm and background inpainting process in DynaSLAM introduce a delay, makes this method unsuitable for real time operation. Table 5.3 also compares the CPU and GPU performance of the SegNet. If SegNet is run on CPU instead of GPU, segmentation process takes 2582 ms which is almost 68 times longer than GPU. If a lightweight semantic segmentation such as Seg.Net is used, as in DS-SLAM and RDS-SLAM, the required time for per frame for segmentation decreases from 200 ms to 30 ms. However, an unlearned dynamics in the camera field of view results in pose error, leading to moving object to be mapped as static object. Our method without using any semantic label criteria runs almost at constant rate regardless of the moving object type and speed without requiring high end graphics unit.

Figure 5.14 shows that a person remains in the model, because, the model built has an artifact in the “walking xyz” sequences. This situation also occurs in “walking halfsphere” (Figure 5.13) and “walking static”(Figure 5.10) sequences because the camera is tracking a person initially and finally the camera never look again, hence, it is not possible to identify voxels free.

Figure 5.17 also confirms such a case. It is clear that the translational error is higher at the beginning when the camera tracks the person. Because of the high dynamic initially, the translational error is above 0.3 m. One second later the translational error is as low as 0.03 m, however, a new high dynamical motion results in an increase in the translational error. Thus, the high translational error results in a artifact in the resulting mesh of the environment.

Figure 5.15 and Figure 5.16 shows- the estimated trajectories of of TUM fr3/walking xyz sequence of SLAM systems. In Figure 5.15, it is clear that ORBSLAM has the worst trajectory estimation in sparse SLAM systems. The performance of our proposed method differs from RDS-SLAM, Semantic SLAM, RDS-SLAM, DynaSLAM and DS-SLAM especially at coordinate point (-0.8 m, -3.3 m) and at the triangle area with corner points (-0.8 m, -3.0 m) , (-0.75m, -2.75 m) and (-0.7 m, -2.85 m). In Figure 5.16, it is clear that VO-SF and Elastic Fusion cannot estimate the trajectory successfully. Although, Flow Fusion and Pose Fusion use the semantic segmentation method, their performances are not as good as those shown in Figure 5.15. Since Flow Fusion and Pose Fusion are built on Elastic Fusion, performance of combination of Elastic Fusion with semantic methods is not better that ORBSLAM based semantic systems.

According to Figure 5.15 and Figure 5.16, among all dense, non-semantic SLAM system, DUDMap provides the most consistent trajectory estimation, which is consistent with result of Table 5.6. If scene generation capability given in Figure 5.14 is considered, sparse and semantic SLAM systems such as DynaSLAM and DS-SLAM generate a blurry model of the environment, however our proposed method provides a clear model. In short, semantic SLAM systems have the best trajectory estimate with blurry 3D model, however, dense method have more clear 3D model with less accurate trajectory.

Table 5.3: TUM Dataset – Execution Time

Method	Semantic	GPU	Dynamic Label	Time for per frame (ms)
ORBSLAM3	-	-	-	22 – 30
DS-SLAM	SegNet	P4000	38 ms	Feature extraction > 9.3 Consistency check > 29 Segmentation> 38 Total > 75 ms
		Intel i7-8750 CPU only	2582 ms	Total > 2600 ms [21]
DynaSLAM	Mask R-CNN	Tesla M40	200 ms	Multi-view geometry > 200 Background inpaint> 120
RDS-SLAM	SegNet	RTX 2080Ti	30 ms	Total>300 ms
DUDMap	-	GTX 1070	8.4 ms	Similarity Check:< 7.1 Pose Estimation< 10.3 Dynamic Label< 8.4 Total < 50 ms
Elastic Fusion	-	GTX 780Ti	-	<66 ms
Mask Fusion	Mask R-CNN	GTX TitanX	200 ms	<60 ms

Figure 5.15 and Figure 5.16 illustrate the estimated trajectory result of fr3/walking xyz sequence obtained by state of art visual SLAM system. Trajectory results is consistent with the Table 5.4 – Table 5.6. Semantic based Visual SLAM methods except Pose Fusion and Flow Fusion have better results in ATE and RPE criteria.

Our proposed method can compete with Semantic SLAM and RDS-SLAM, however, DynaSLAM and DS-SLAM have the best estimate. However, our method have the best result among the dense and CNN-free method.

Table 5.4, Table 5.5 and Table 5.6 illustrate the translational RPE (RMSE cm/s), translational RPE (RMSE deg/s) and Translational ATE (RMSE cm) of the TUM fr3 dataset. Among the dense methods, all of them show similar performance in low dynamic dataset except Elastic Fusion. Elastic Fusion provides an average 1.25 cm/s translation RPE, 0.45 deg/s translational RPE and 1.5 cm translational ATE in low dynamic sequences. The error values for second best method , Co Fusion , are 1.35 cm/s translation RPE, 0.7 deg/s translational RPE and 1.9 cm translational ATE.Our method provides an average 2.6 cm/s translation RPE, 0.9 deg/s translational RPE and 1.8 cm translational ATE in low dynamic sequences. However, high dynamic deteriorates the performance of Elastic Fusion and CoFusion. In high dynamics, our proposed method is the best by providing 4.6 cm/s translation RPE, 1.4 deg/s translational RPE and 4 cm translational ATE. Refusion takes the second place with 5.2 cm/s translation RPE, 2.2 deg/s translational RPE and 5.4 cm translational ATE. Mask Fusion the third method attained with 5.8 cm/s translation RPE, 1.6 deg/s translational RPE and 5.9 cm translational ATE. For all dense method, walk/xyz and walk/half are the most challenging sequences.

All sparse SLAM method illustrated in Table 5.4, Table 5.5 and Table 5.6 give the superior results than our proposed method. In low dynamic, sparse SLAM methods show similar performance as ORBSLAM because these methods are based on ORBSLAM. Addition of segmentation step increases the dynamic mapping performance. A similar case can be seen in Mask Fusion which is based on Elastic Fusion and addition of segmentation process Mask R-CNN increases the performance of Mask Fusion.

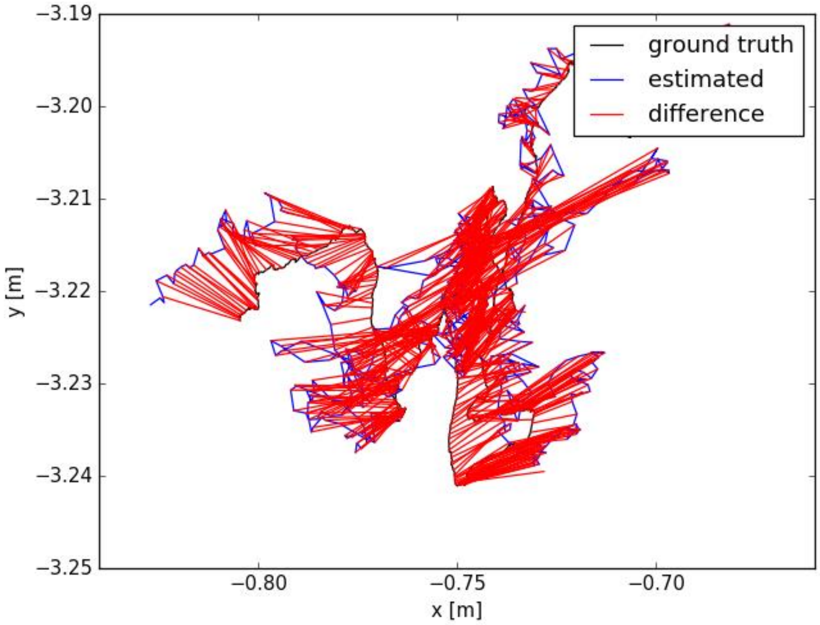


Figure 5.10: ATE / RPE of TUM fr3/walking static sequence

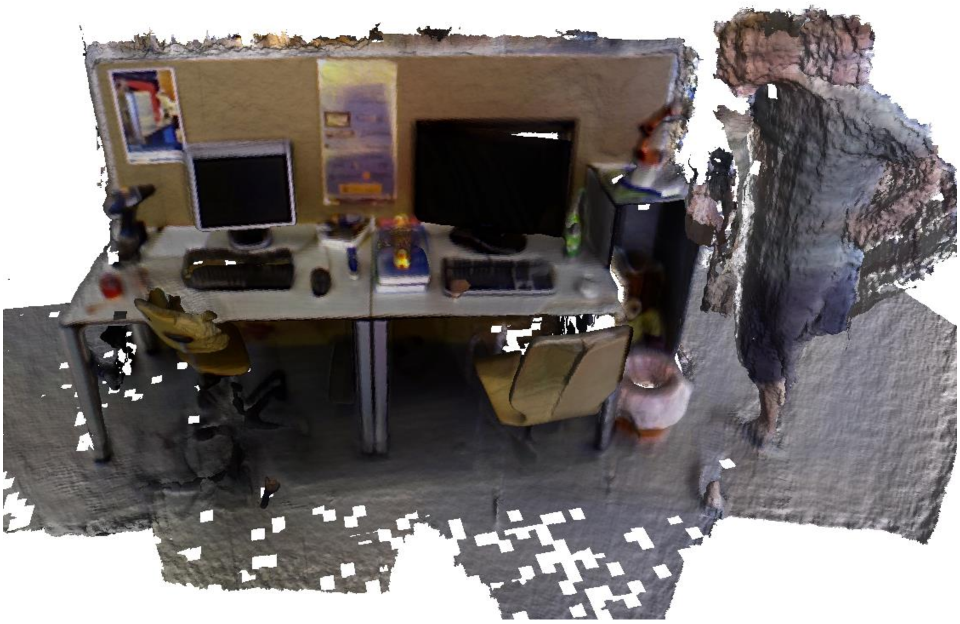


Figure 5.11: Mesh of TUM fr3/walking static sequence

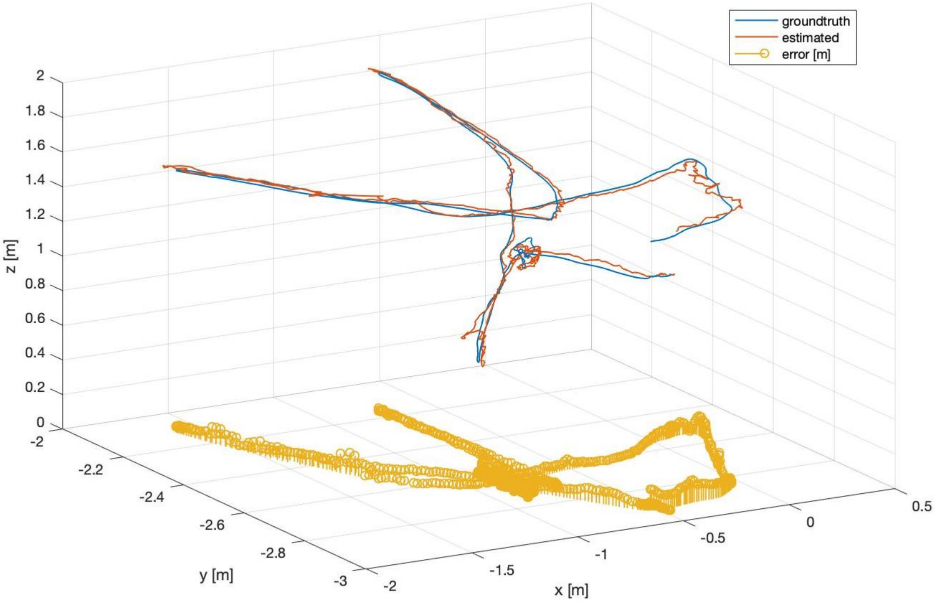


Figure 5.12: ATE / RPE of TUM fr3/walking halfsphere sequence

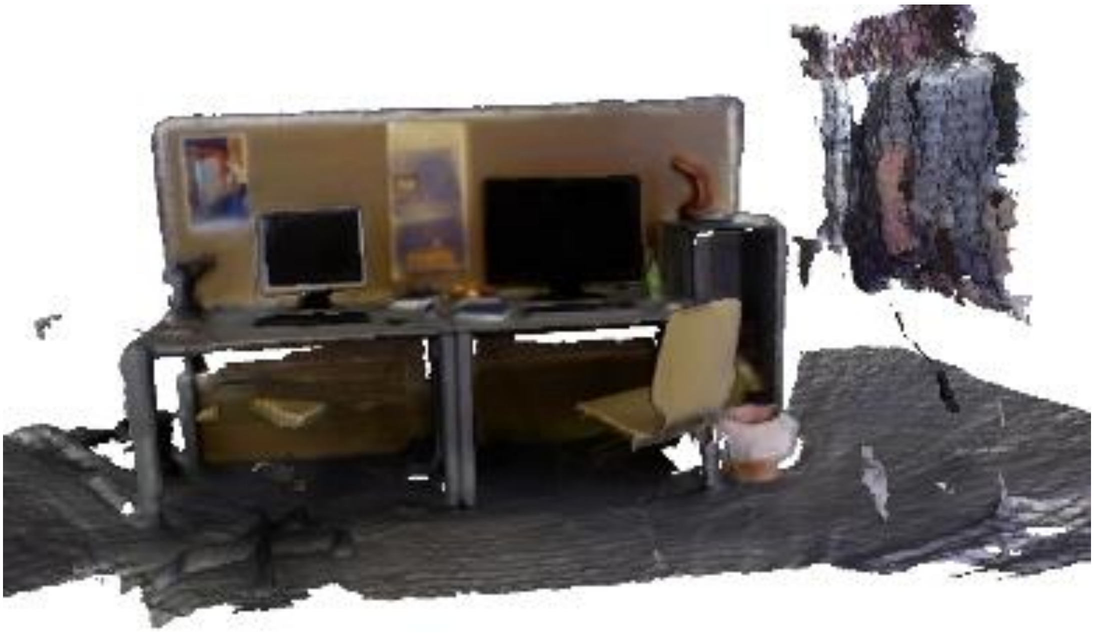


Figure 5.13: Mesh of TUM fr3/walking halfsphere sequence

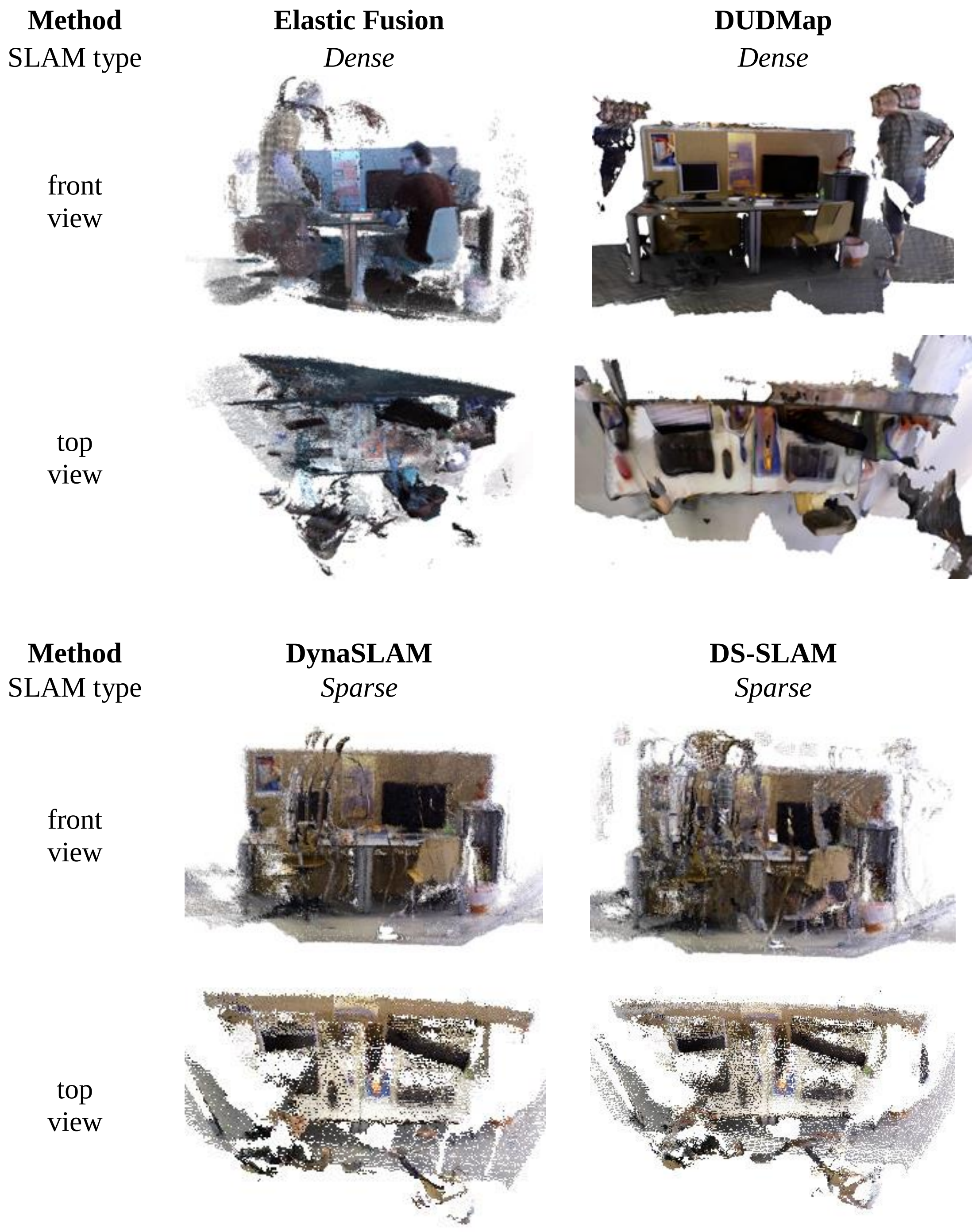


Figure 5.14: Scene reconstruction of fr3/walking xyz sequence

Table 5.4: TUM Dataset – Translational RPE (RMSE cm/s)

Mapping Type		Dense Map								Sparse Map			
CNN Utilization		No CNN							Segmentation CNN				
Dynamic	**Sequence**	VOSF	EF	CF	RF	SF	ORBSLAM	DUDMAP	MF	DS-SLAM	Dyna-SLAM	Semantic SLAM	RDS-SLAM
Low	sit static	2,4	0,9	1,1	2,1	1,1	1	1,5	1,7	0,8	1,3	0,9	1,2
Low	sit xyz	5,7	1,6	2,7	3,8	2,8	-	3,7	4,6	-	-	-	-
High	walk stat	10,1	26	22,4	1,7	1,3	78	2,6	3,9	1	0,9	1	2,2
High	walk xyz	27,7	24	32,9	11,8	12,1	42,6	10,3	9,7	3,3	2,1	2,2	4,3
High	walk half	33,5	20,5	40	6,4	20,7	32,7	4,7	9,3	3	2,9	2,8	4,8
Mean Error	All sequences	15,9 18,4*	14,6 17,9*	19,82 24,1*	5,2 5,5*	7,6 8,8*	30,9*	4,6	5,8	2,0*	2,4*	2,3*	3,2*

* Mean value excluding sit/xyz, All methods are given with corresponding paper in the reference set.

Table 5.5: TUM Dataset – Translational RPE (RMSE deg/s)

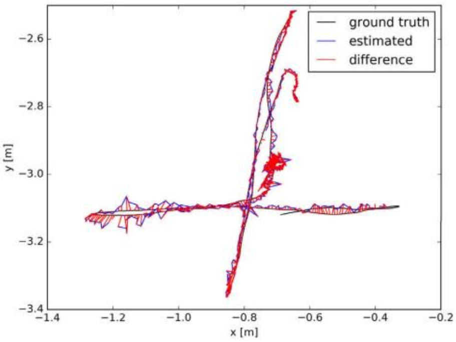
Mapping Type		Dense Map								Sparse Map			
CNN Utilization		No CNN							Segmentation CNN				
Dynamic	**Sequence**	VOSF	EF	CF	RF	SF	ORBSLAM	DUDMAP	MF	DS-SLAM	Dyna-SLAM	Semantic SLAM	RDS-SLAM
Low	sit static	0,7	0,3	0,4	0,6	0,4	0,3	0,4	0,5	0,3	0,3	0,3	0,3
Low	sit xyz	1,4	0,6	1	1,3	0,9	-	1,2	1,3	-	-	-	-
High	walk stat	1,7	4,8	4	1,1	0,4	6	0,6	0,8	0,3	0,3	0,3	0,5
High	walk xyz	5,1	4,8	5,6	2,7	2,7	7,9	2,3	2	0,8	0,7	0,6	0,9
High	walk half	6,7	6,4	13	3	5	7,2	2,3	3,4	0,8	0,8	0,7	1,9
Mean Error	All sequences	3,1 3,6*	3,4 4,1*	4,8 5,8*	2,2 1,9*	1,9 2,1*	5,4*	1,4 1,4*	1,6 1,7*	0,6*	0,5*	0,48*	1,2*

* Mean value excluding sit/xyz, All methods are given with corresponding paper in the reference set.

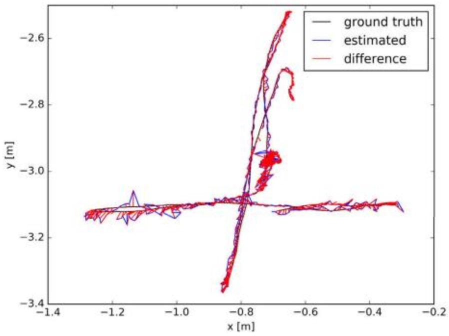
Table 5.6: TUM Dataset – Translational ATE (RMSE cm)

Mapping Type		Dense Map								Sparse Map			
CNN Utilization		No CNN							Segmentation CNN				
Dynamic	**Sequence**	VOSF	EF	CF	RF	SF	ORBSLAM	DUDMAP	MF	DS-SLAM	Dyna-SLAM	Semantic SLAM	RDS-SLAM
Low	sit static	33	0,8	1,1	0,9	1,3	0,9	1	2,1	0,7	1,1	0,8	0,8
Low	sit xyz	11	2,2	2,7	4	4	-	2,6	3,1	-	-	-	-
High	walk stat	33	29	55	1,7	1,4	36	1,8	3,5	0,8	0,7	0,8	2
High	walk xyz	87	91	69	9,9	12	92	7,4	10	2,5	1,6	1,6	5,7
High	walk half	74	64	80	11	39	65	7,1	11	3	3	2,5	8
Mean Error	All sequences	48 56,8*	37 46*	41 51*	5,4 5,9*	12 13,4*	48,4*	4 4,3*	5,9 6,7*	1,8*	1,6*	1,4*	4,1*

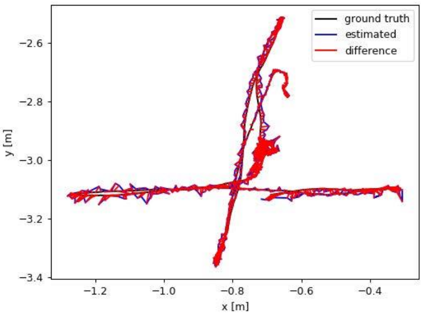
* Mean value excluding sit/xyz, All methods are given with corresponding paper in the reference set.



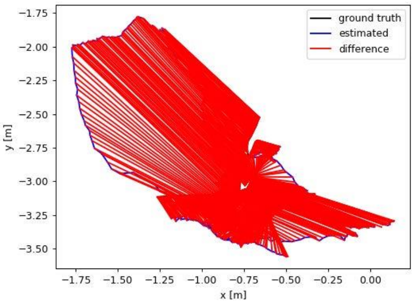
DynaSLAM
Sparse, Mask R-CNN



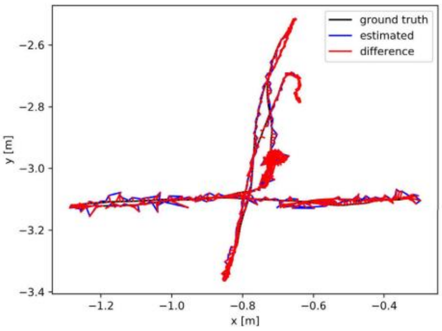
DS-SLAM
Sparse, SegNet CNN



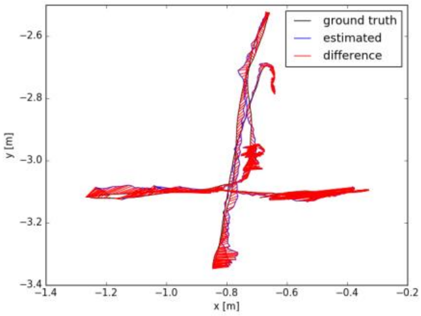
RDS-SLAM
Sparse, Mask R-CNN / SegNet



ORBSLAM3
Sparse, No CNN

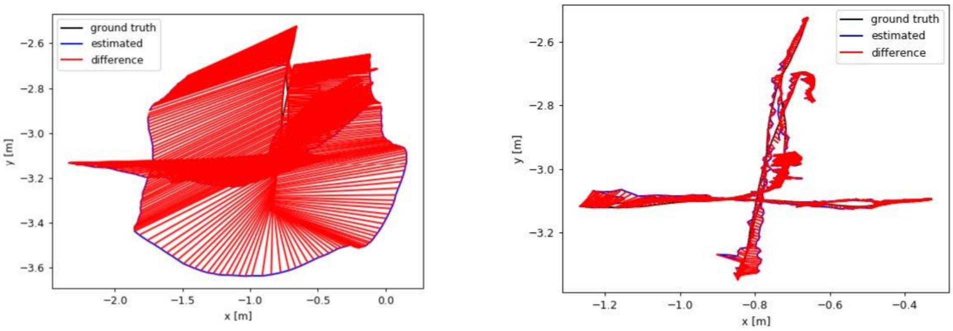


Semantic SLAM
Sparse, BlitzNet CNN



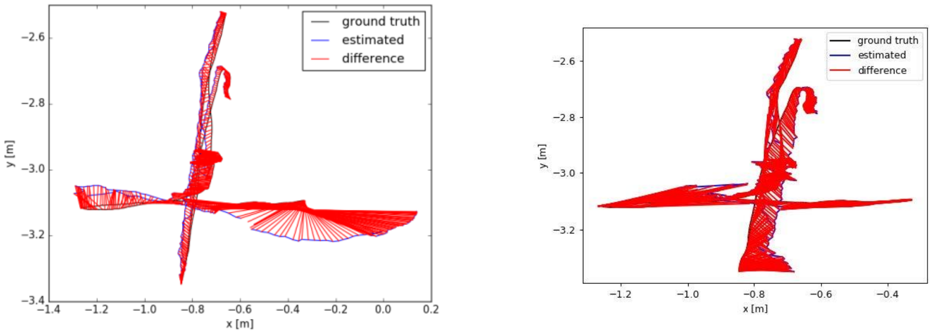
DUDMAP
Dense, No CNN

Figure 5.15: Comparison of estimated trajectories of of TUM fr3/walking xyz sequence



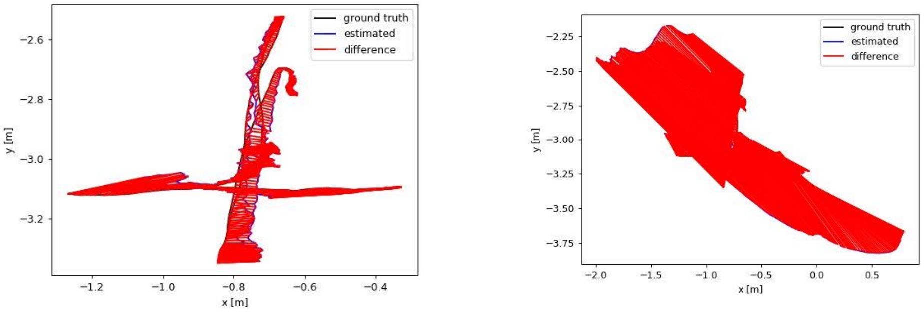
Elastic Fusion
Dense, No CNN

Pose Fusion
Dense, Open Pose CNN



Refusion
Dense , No CNN

Static Fusion
Dense, No CNN



Flow Fusion
Dense, Pwc.Net CNN

VO-SF
Dense, No CNN

Figure 5.16: Comparison of estimated trajectories of of TUM fr3/walking xyz sequence (continued)

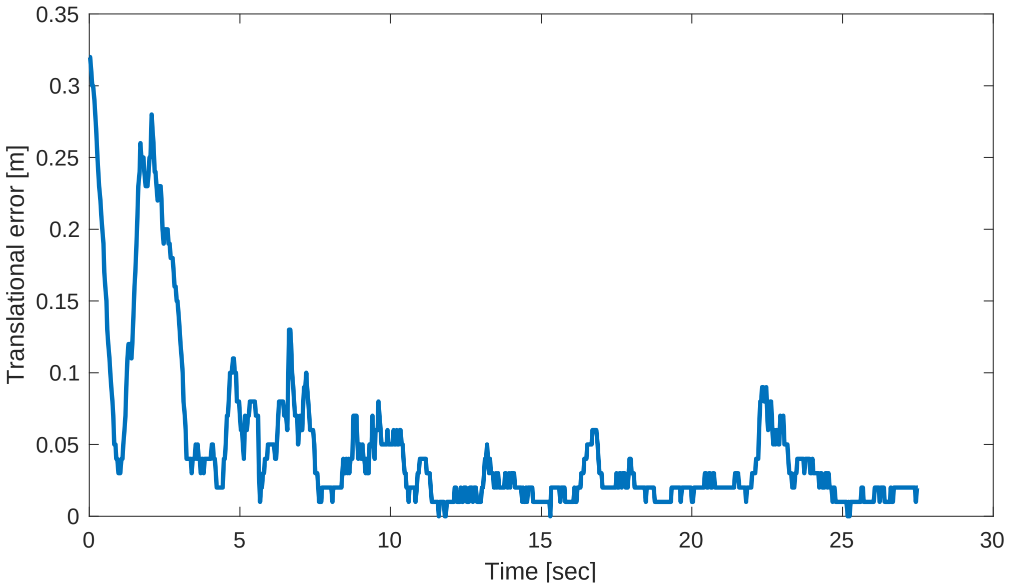


Figure 5.17: Relative translational error (walking-xyz) of our method

5.2.2 Bonn RGB-D Dynamic Dataset

In order to further evaluate the accuracy of camera pose tracking, we compare our approach with the three state of the art SLAM systems namely, DynaSLAM, Refusion and Static Fusion on the dynamic scenes of Bonn dataset published in [14]. Results are obtained by running available open source implementations for each method This dataset has 24 dynamic sequences, where people perform different tasks, such as manipulating boxes or playing with balloons. When an robot is moved to an arbitrary location and if is can't locate itself against the map, then this situation known as "kidnapped robot problem". For example, "moving_obstructing_box (see Figure 5.18) " scene assesses the kidnapped camera problem, where the camera is moved to a different location whereas "balloon_tracking" has uniformly colored balloon having no features on it. These tasks often obstruct the camera, creating particularly challenging situation when mapping. For each sequence, there exists the ground truth pose of the sensor, recorded with an Optitrack Prime 13 motion capture system. In addition, the sequences are in the same format as the TUM RGB-D Dataset.

Table 5.7 shows that in balloon tracking DynaSLAM outperforms the other methods. However, it has poor performance on the obstructing box scene. Since DynaSLAM is the combination of neural network and geometric approach, the available semantic information on the scene helps to increase the performance. DynaSLAM with the combined neural network and geometric approach performs best in the sequence with people and known object. This is due to the heavy bias of having people in sequence, therefore the segmentation of people always helps the DynaSLAM achieving better results. In obstruction box sequence, on the other hand, our proposed method performs best because our method does not employ any type of object classifier or tracking. If a voxel is empty in enough time, then we mark it as free. The speed of dynamic elements becomes important, because, it allows us to capture the static part of environment in the first frames.

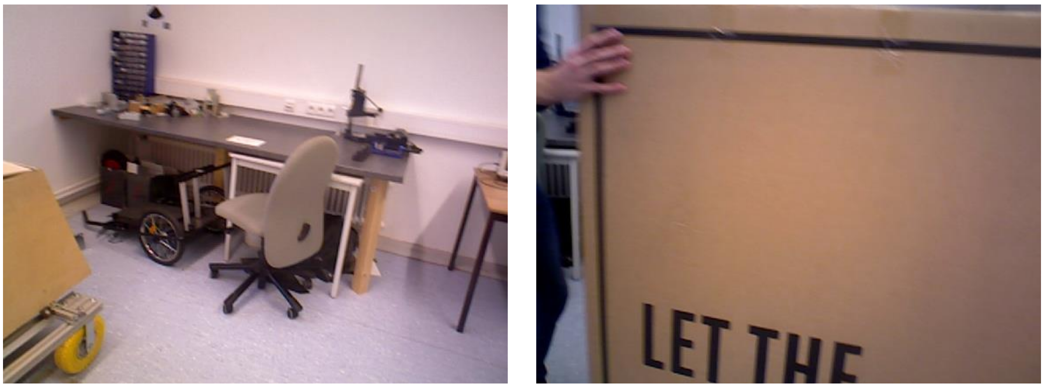


Figure 5.18: Bonn RGB-D Dynamic dataset containing desk, chair and moving obstructing box

Figure 5.19 shows the resulting mesh of BONN moving obstructing box sequence and the camera view is often obstructed by the box which creates particularly challenging situations for mapping approaches. However, it is clear that our method can generate the 3D model of the environment without having to track of the box or walking people.

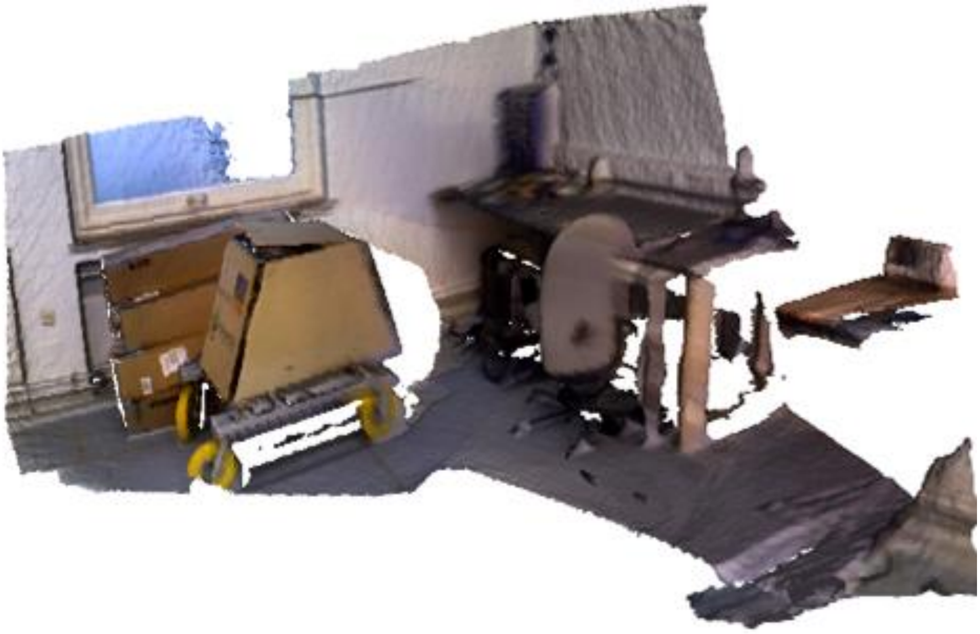


Figure 5.19: Mesh of BONN moving obstructing box sequence

Table 5.7: BONN Dataset – Translational RPE - (RMSE cm/s)

Dynamic	Sequence	RF [14]	SF [13]	DynaSLAM [20]	DUDMAP
High	balloon tracking2	0,32	0,37	0,19	0,27
High	obstruction box	0,34	0,33	0,54	0,17

5.2.3 VolumeDeform Dataset

VolumeDeform is a RGB-D dataset for the purpose of real-time non-rigid reconstruction and is used for evaluation of the non-rigid object reconstruction algorithms at real-time rates [30]. Since dynamic data sets for evaluating RGB-D SLAM method with exact trajectory are limited, this dataset is used to measure the elimination capability of our method to handle dynamic parts in the scene. Results of pose error and trajectory error are listed in Table 5.8 and Figure 5.20. Obtained results using boxing, sunflower and frame sequences are illustrated Figure 5.21 and Figure 5.22.

Table 5.8: VolumeDeform Dataset results

Dynamic	Sequence	Trans, RPE RMSE (cm/s)	Trans, RPE RMSE (deg/s)	Trans, ATE RMSE (cm)
High	boxing	0,32	0,37	0,19
High	sunflower	0,34	0,33	0,54

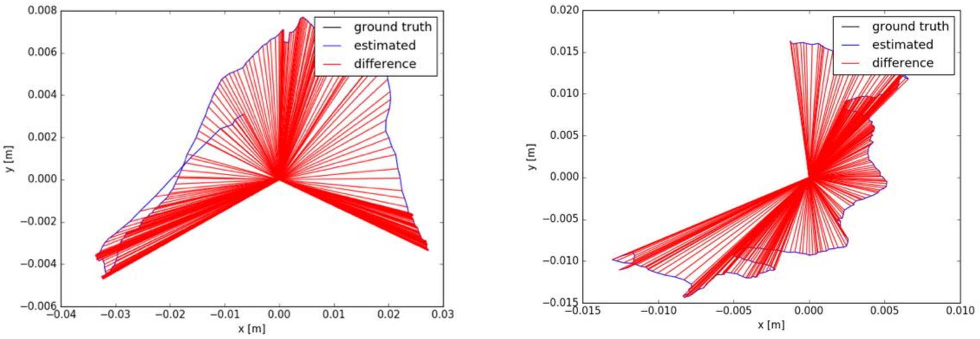


Figure 5.20: Estimated trajectories of of Volume Deform *boxing* and *sunflower* sequence

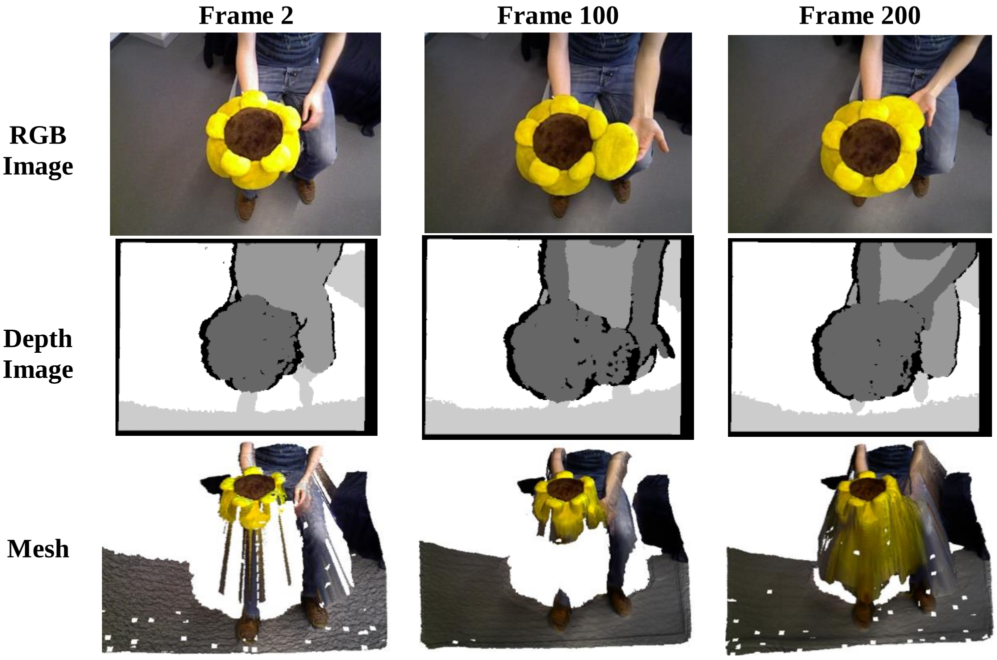


Figure 5.21: RGB-D image pair and 3D dense model of VolumeDeform *sunflower* sequence obtained by our proposed method

5.2.4 CVSSP RGB-D Dataset

“CVSSP Dynamic RGBD dataset has RGBD sequences of general dynamic scenes captured using the Kinect V1/V2 as well as two synthetic sequences”[33]. This dataset is designed for non-rigid reconstruction. “dog” sequence is selected because there exists little clearly distinct geometry in the environment with non-rigid dynamic object. In this sequence , the dynamic part is the movement of the arm of the person and the head of the dog. The exact value of the trajectory and reference 3D model of the environment are not provided, therefore we evaluated the 3D mesh result, qualitatively. As the frame number increases, our proposed method successfully eliminates dynamic in the frame (Figure 5.23).

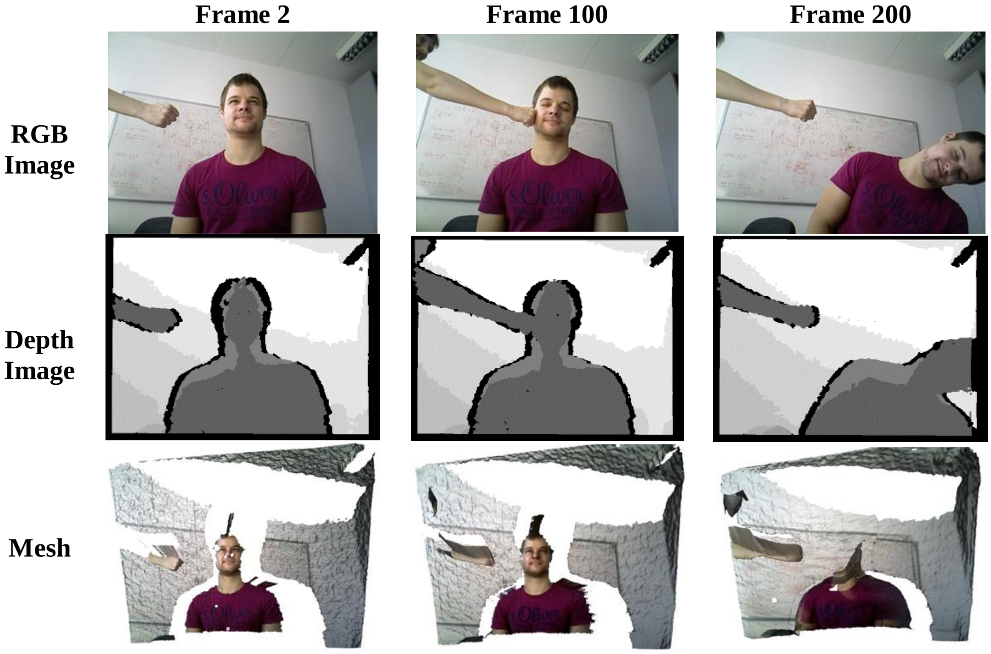
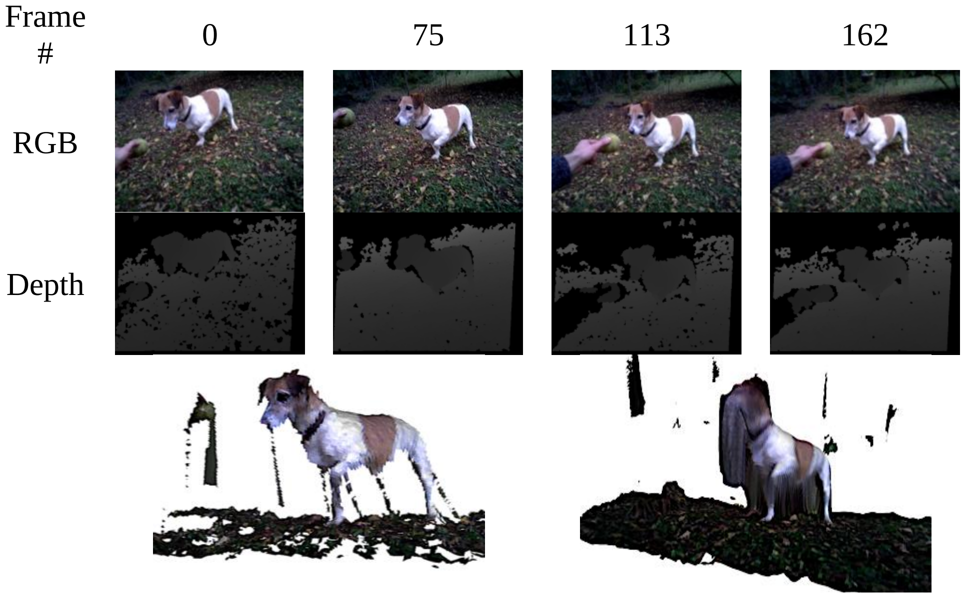


Figure 5.22:RGB-D image pair and 3D dense model of VolumeDeform *boxing* sequence obtained by our proposed method



Mesh @ 2. frame

Mesh @ 180. frame

Figure 5.23: RGB-D image and final mesh of the CVSSP “dog” sequence

5.2.5 Outdoor Mapping Performance

We used the ZED camera [35] in hand-held setup for acquiring RGB-D images. We captured frame in resolution of 1280x720 with rate of 30 fps. In order to measure the 3D mapping performance of our proposed approach, default camera properties and standard settings are used without calibration or lens distortion correction. 0.01 m voxel size with minimum 0.3 m depth sensor setting is used. Our method successfully created the mesh of the environment with some distortions. For instance, 0.01 mm voxel size results in coarse map especially in missing wire grid fence and part of the fence door (Figure 5.24)

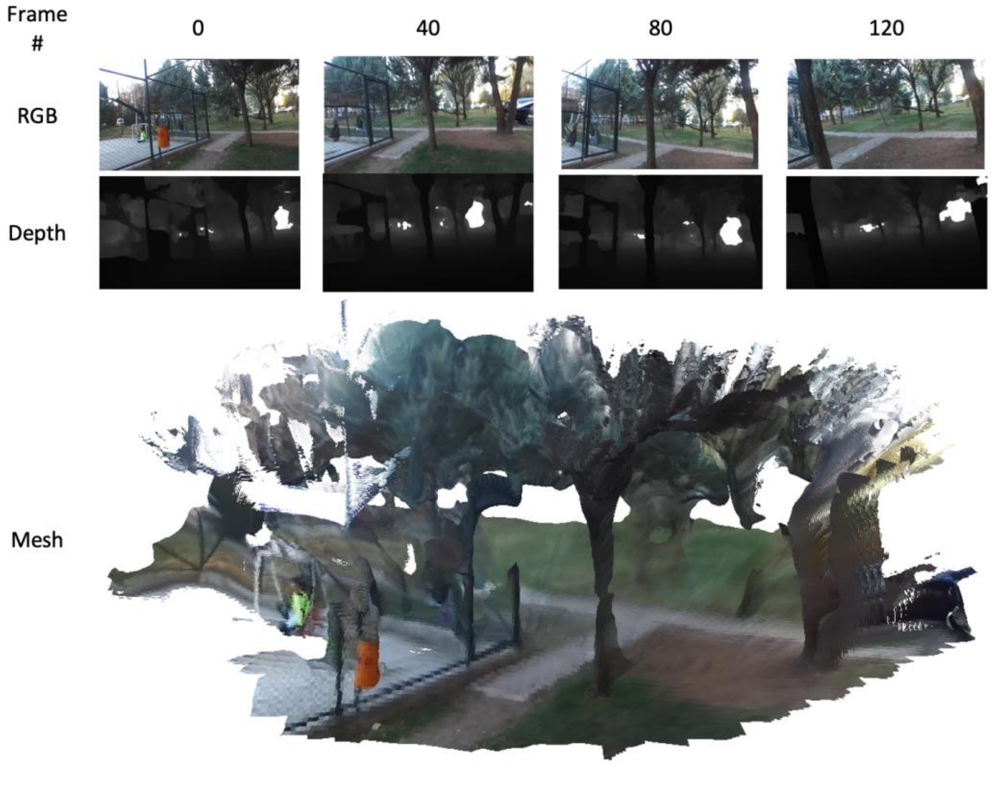


Figure 5.24: RGB-D image and final mesh of the “outdoor-1” sequence

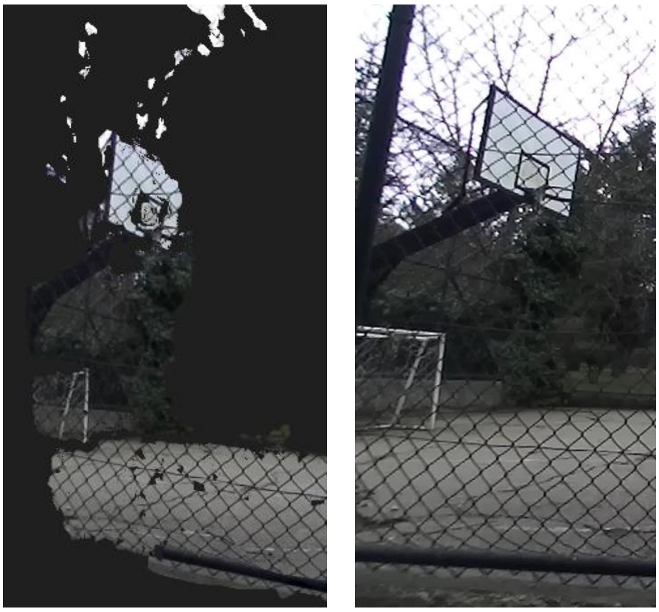


Figure 5.25: outdoor-1 sequence grid fence mapping result

Using smaller voxel size increases the mapping performance helps to maintain grid fence as in Figure 5.25. If an autonomous robot is flying around thin branches, telephone lines or chain link fencing, an detailed map is required in order to avoid from the collision because those are the main collision areas for outdoor autonomous drones.

In the second sequence, we captured frame in resolution of 1280x720 with rate of 10 fps using default camera properties. 0.02 m voxel size and 16 m maximum depth settings are used in this sequence. As can be seen from the Figure 5.26, the final mesh has no artifact of the walking person in the scene. However, result of Elastic Fusion has traces of the walking person.

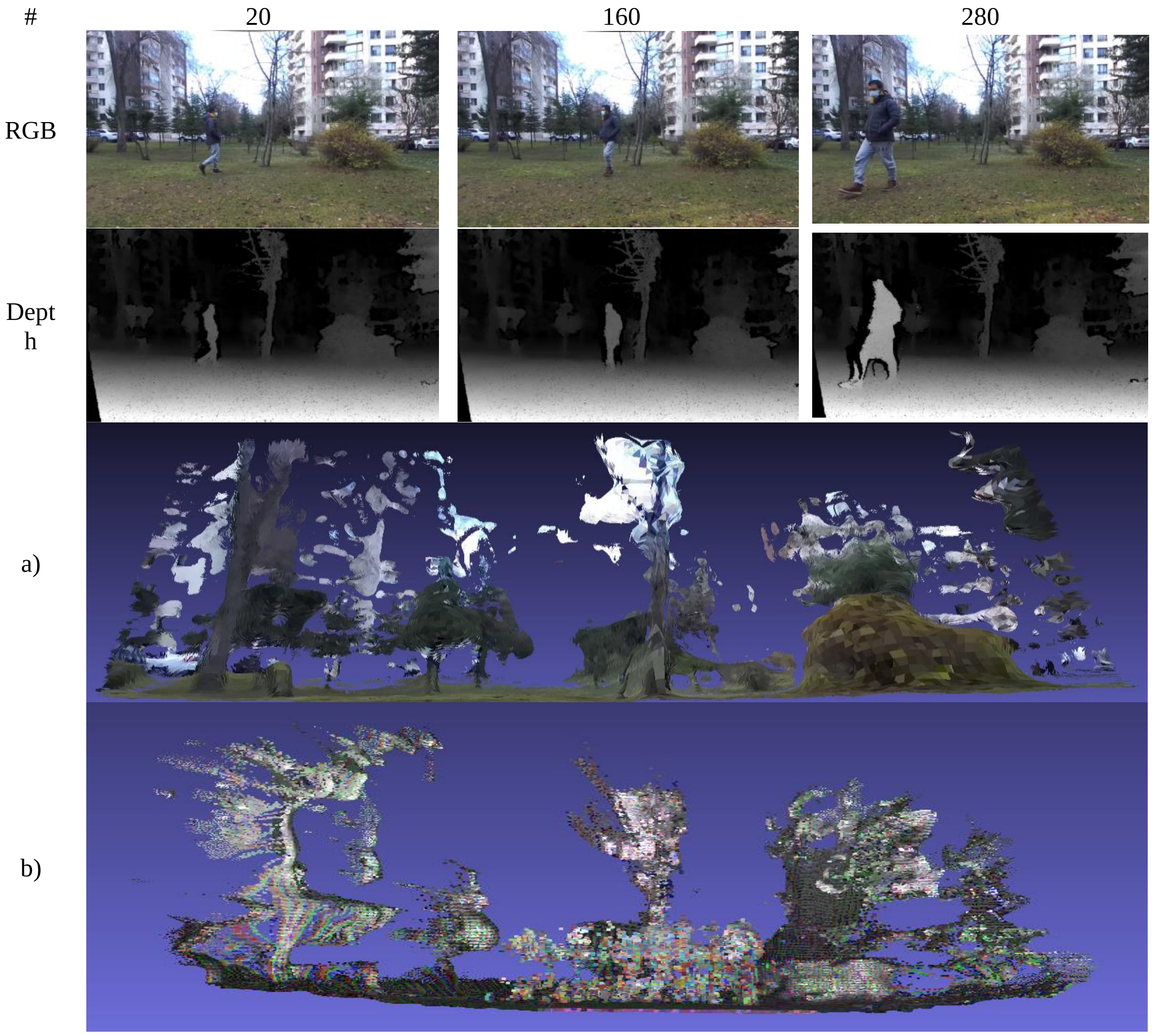


Figure 5.26: RGB-D image and final mesh of the “*outdoor*-2 ” sequence a) DUDMap b) Elastic Fusion

CHAPTER 6

SENSITIVITY ANALYSIS

In order to systematically measure the effect the input parameters to SLAM tracking performance, a design of experiments study is conducted because it is an efficient method for studying the relationship between multiple input variables and key output variables. In addition, it is a structured approach for collecting data and making discoveries.

In order to limit the memory requirement, the voxel size is the most important parameter because representing a scene in the form of cube 10 x10 x 10 m^3 in size at 1 cm resolution, would require 7.4 GB memory, storing 8 byte for each grid. Our aim is to limit the memory size to 4 GB maximum, we represent the environment in the form of a rectangular prism of size 10 x10 x 5 m^3 at 1 cm resolution, which is the most detailed version. For voxel size, we use 0.01 m, 0.02 m and 0.05 m, which uses 3.7 GB, 0.5 GB, and 0.03 GB memory, respectively.

Levenberg-Marquart method is the combination of Gradient Descent and Gauss Newton. For example, if we select regularization constant or damping parameter as zero, then we get regular Newton method. Using large values of regularization constant tends to dampen the solution and it is important once the solution is close to optimal.

Huber function is a penalty method, which is dependent on the residual. Using Huber function, errors close to zero (small errors) are scaled quadratically, while the large values will be scaled linearly. This allows us better adjustment when image is close to its correct alignment.

If image registration is correct with the 3D model, the projected colors should be consistent as well. We incorporate this color consistency condition by adding color error function. Therefore, weight ratio, which is the contribution of the intensity with respect to the depth, is also important.

In this section, the sensitivity of the proposed methodology to voxel size, regularization constant, color intensity contribution with respect to the depth, Huber constant will be examined. Moreover, the required computational time is measured while changing the voxel size. In order to assess the performance of the our method fr3/walking xyz dataset is selected for the error and timing analysis because, in this sequence, camera is tracking a person at the beginning and finally camera never revisits again, which results in artifact in resulting mesh. In addition, most of the state art system use this sequence for performance analysis.

6.1 Design of Experiments

In order to systematically measure the effect the input parameters (voxel size, regularization constant, color intensity contribution with respect to the depth, Huber constant) to SLAM tracking performance, a design of experiments study is conducted. In this study, 4 factors and 3 levels design of experiment which is 3×4 factorial design with 81 treatment combinations are considered because this factorial design allows us to discover the main effects and interactions.

For Huber constant, the general use is the 1/5 of the voxel grid, therefore we use 0.005, 0.025 and 0.01 Huber constant levels. Regularization level is selected using Gramian of Jacobian as 0.002, 0.02 and 0.01, respectively. The resulting performance index which is Absolute Tracking Error (ATE [m] in rms) are tabulated in Table 6.1 and main effects plot is illustrated in Figure 6.1 shows. Comparing all changes in Table 6.1 and Figure 6.1, it is clear that using smaller voxel helps to improve the tracking, however, we know that using smaller voxels increases the memory requirements and computational complexity. In addition, the 0.1 weight ratio seems to be the best.

Main effects plots shows the effects of one independent variable on the dependent variable and it is useful when we have several categorical variables. Figure 6.1 shows the main effect, which displays the means for each group within a categorical variable. According to Figure 6.1, we should select Huber constant as 0.01, Regularizarion constant is 0.02, voxel size as 0.01 m and weight as 0.1. These values are consistent with Table 6.1 because the minimum tracking error 0.08 m which is obtained when Huber constant, regularization constant, voxel size and weight are 0.01,0.02, 0.01 and 0.1, respectively.

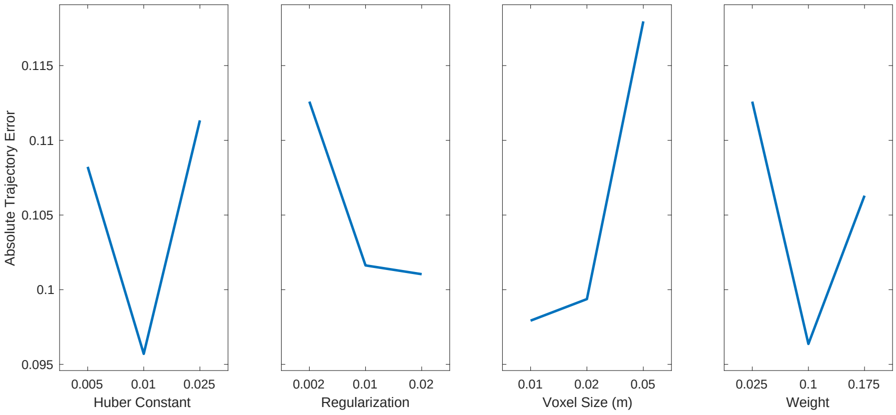


Figure 6.1: Main effects plot: Variation of Absolute Trajectory Error to voxel size, regularization constant, color intensity contribution with respect to the depth, Huber constant

Figure 6.2 illustrates the interaction plot, which shows how the relationship between one categorical factor and a continuous response depends on the value of the second categorical factor. This is opposed to the “main effect” which is the action of a single independent variable on the dependent variable. In Figure 6.2, the levels of variable is displayed on horizontal axis and line represent the means of each level of other variable. As an example, let us analyze the greyscale subplot, which shows the voxel size – Huber constant interaction in the tracking error. We have 3 levels of input for voxel size and Huber constant which are (0.01,0.02,0.05) and (0.025, 0.01, 0.005), respectively. According to this subplot, it is clear that tracking error always decreases with decreasing the voxel size except Huber constant is 0.025. The same scenario exists in voxel size-weight interaction. If weight is not selected as 0.025, using smaller voxels improves the tracking performance.

In Huber constant-regularization interaction, using 0.01 or 0.02 produces almost same result. Huber constant- voxel size have the similar tracking performance if voxel size is not equal to 0.05. In addition, Figure 6.2 shows that interaction of Huber constant for other variables is minimum for 0.01 Huber constant.

The interaction of weight - regularization is almost same if 0.01 or 0.02 regularization constant is used. Weight-Huber constant interaction has minimum if 0.1 weight and 0.01 Huber constant are selected, however, it has a strange properties because response of tracking error are completely complement when 0.025 and 0.05 Huber constant is selected. In weight-voxel size interaction, using large value of voxel produce minimum error when the weight is 0.01 however, this minimum error value is still larger than error in other voxel sizes.

The regularization-voxel size interaction shows that using smaller voxel size leads to minimum tracking error and 0.01 m and 0.02 m voxel size produces the similar result. In regularization-Huber constant plot, 0.01 Huber constant is the best and it has similar trend with 0.005 Huber value however, 0.005 Huber has a poor performance when regularization is 0.01 or 0.02. In general, using interaction plots, we can conclude 0.01 Huber constant, 0.02 regularization are the best choice without doubt.

Table 6.1: TUM Dataset fr3/walking xyz – Translational ATE (RMS Error m)

			weight		
			0.025	0.1	0.175
Huber Constant 0.005	Regularization 0.002	Voxel 0.01 m	0.098	0.096	0.102
		Voxel 0.02 m	0.101	0.096	0.113
		Voxel 0.05 m	0.102	0.116	0.171
	Regularization 0.02	Voxel 0.01 m	0.095	0.091	0.097
		Voxel 0.02 m	0.099	0.093	0.108
		Voxel 0.05 m	0.102	0.109	0.165
	Regularization 0.01	Voxel 0.01 m	0.097	0.093	0.100
		Voxel 0.02 m	0.100	0.093	0.109
		Voxel 0.05 m	0.102	0.104	0.170
Huber Constant 0.01	Regularization 0.002	Voxel 0.01 m	0.110	0.083	0.092
		Voxel 0.02 m	0.105	0.086	0.094
		Voxel 0.05 m	0.092	0.108	0.097
	Regularization 0.02	Voxel 0.01 m	0.108	**0.080**	0.088
		Voxel 0.02 m	0.105	0.085	0.092
		Voxel 0.05 m	0.092	0.094	0.114
	Regularization 0.01	Voxel 0.01 m	0.108	**0.085**	0.089
		Voxel 0.02 m	0.105	0.094	0.093
		Voxel 0.05 m	0.092	0.081	0.112
Huber Constant 0.025	Regularization 0.002	Voxel 0.01 m	0.118	0.104	0.090
		Voxel 0.02 m	0.101	0.107	0.097
		Voxel 0.05 m	0.365	0.095	0.101
	Regularization 0.02	Voxel 0.01 m	0.117	0.103	0.090
		Voxel 0.02 m	0.101	0.106	0.096
		Voxel 0.05 m	0.100	0.096	0.102
	Regularization 0.01	Voxel 0.01 m	0.117	0.104	0.089
		Voxel 0.02 m	0.101	0.106	0.097
		Voxel 0.05 m	0.107	0.094	0.102

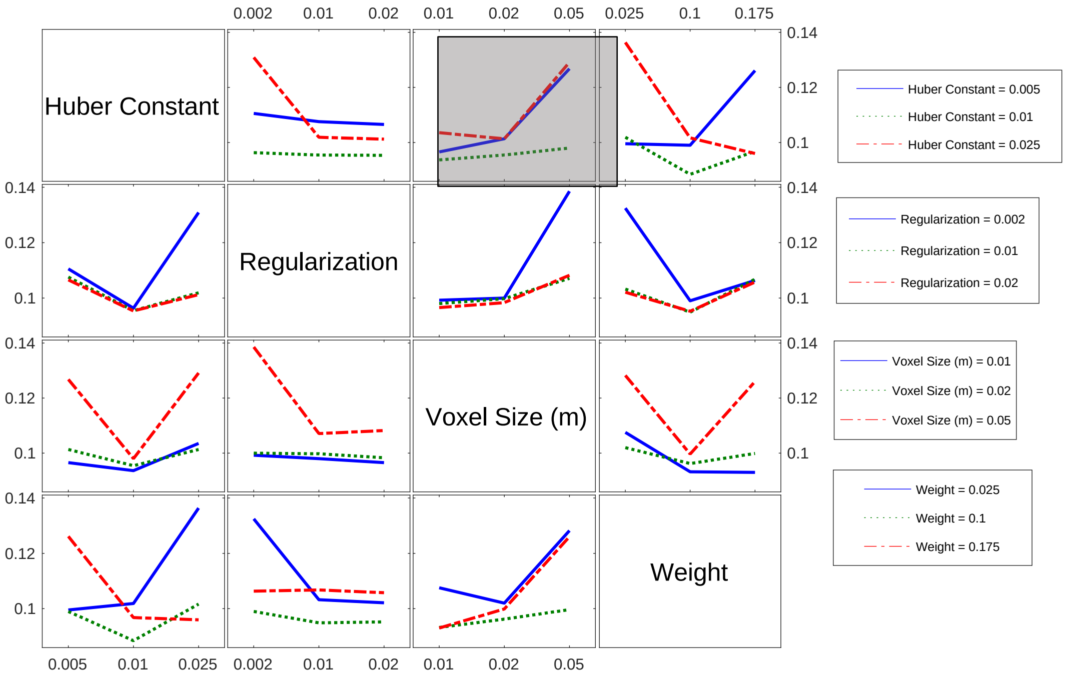


Figure 6.2: Interaction plot: Variation of Absolute Trajectory Error to voxel size, regularization constant, color intensity contribution with respect to the depth, Huber constant

Using the Table 6.1, we can find the minimum tracking error as 0.08 m and it is obtained when Huber constant, regularization constant, voxel size and weight are 0.01,0.02, 0.01 and 0.1, respectively. In main effect plot, we can easily see that the minimum tracking error is obtained when the weight is 0.1. However, in order to find the best value for weight, a detailed investigation is performed for weight value between 0.1 and 0.175. Table 6.2 shows the absolute tracking error for weight value between 0.1 and 0.175. In this analysis, Huber constant, regularization constant, voxel size are 0.01,0.02, 0.01, respectively. The minimum error is obtained when weigth is 0.1325. Therefore , we select to use 0.01,0.02, 0.01 and 0.1325 values for parameters.

Table 6.2: Change of Translational ATE (RMS m) with respect to weight (regularization constant, Huber constant and voxel size is constant)

weight	absolute translational error (RMS Error [m])
0.1	0.0800
0.11	0.0764
0.12	0.0739
0.125	0.0731
0.1275	0.0734
0.13	0.0729
0.1325	0.0727
0.14	0.0731
0.15	0.0797
0.175	0.0880

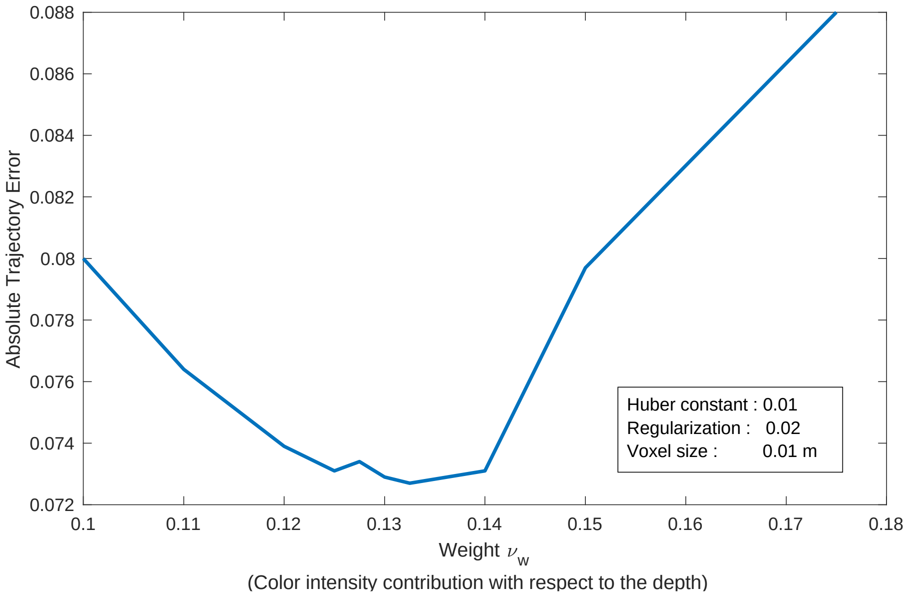
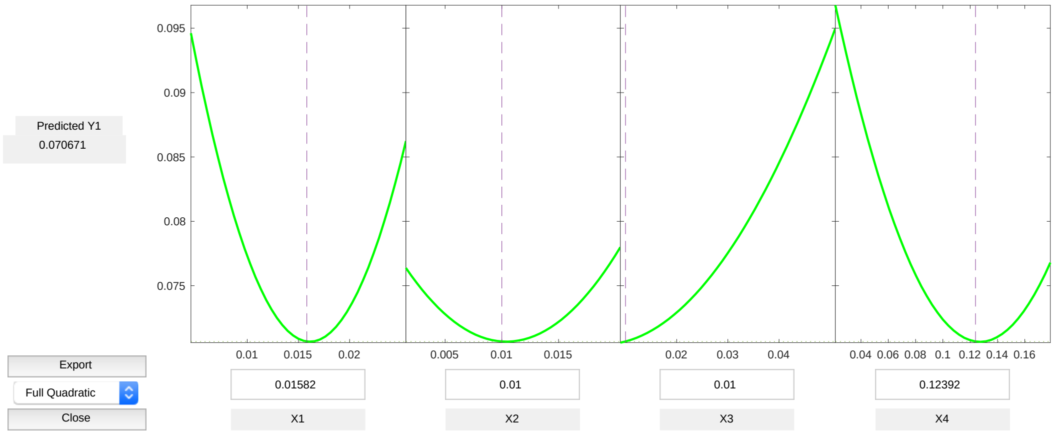


Figure 6.3: Change of Translational ATE (RMS Error [m]) with respect to weight (regularization constant, Huber constant and voxel size is constant)

In order to consider the constant, linear, interaction, and squared terms, a quadratic model is formed. We used MATLAB rstool to examine relationship in full quadratic mode. Figure 6.4 shows the response of the given input (X1, X2, X3, X4) and output (Y1). (X1, X2, X3, X4) denote Huber constant, regularization constant, voxel size and weight and (Y1) represents the tracking error. Using this quadratic model, we can choose (0.015,0.01,0.01,0.124) for minimum value 0.071 m. However, if we use the these values, the resulting error is 0.089 m, which is not consistent therefore, we use the vales (0.01,0.02, 0.01 and 0.1325) for Huber constant, regularization constant, voxel size and weight.

In this section, in addition to the sensitivity of the proposed methodology to the voxel size , the contribution weight of the intensity with respect to the depth, Huber constant and regularization constant, the required time for per image is also analyzed (see Table 6.3).

According to Table 6.3, in all case, using larger voxel, dramatically decreases the required calculation time which makes that the proposed scheme more suitable for real-time applications. However , using larger voxel, increases the absolute translational error. Using larger ratio of the intensity information with respect to the depth information decreases the RMSE error , however, such situation is not valid for all cases. Therefore, utilization of application specific constant increases the performance of the proposed scheme.



Predicted Y1
0.070671

Full Quadratic

X1 0.01582

X2 0.01

X3 0.01

X4 0.12392

Figure 6.4: Quadratic model of the sensitivity analysis

Table 6.3: TUM “fr3/walking static” Sequence Translational ATE Error (RMS Error [cm])

Voxel Size (m)	**weight (υ)**	**Translational ATE Error (cm)**	**Mean Time for per frame (ms)**
0.005	0.01	1.05	230
0.01	0.01	1.77	116
0.02	0.01	1.67	76
0.05	0.01	2.16	62
0.005	0.025	0.74	231
0.01	0.025	0.79	116
0.02	0.025	1.06	77
0.05	0.025	0.90	62
0.005	0.04	0.69	230
0.01	0.04	0.81	115
0.02	0.04	0.77	76
0.05	0.04	0.95	62

CHAPTER 7

CONCLUSION

SLAM techniques estimate jointly a map of an unknown environment and the robot pose within such map, only from the data streams of its on-board sensors. Visual SLAM, where the main sensor is a camera, have received high-level attention in recent years. The basic configuration is monocular camera which has practical advantages however it has also several drawbacks such as the depth estimation and scale uncertainty. With the appearance of Kinect in 2010, there are many advanced RGB-D SLAM systems which use the depth images or fuse the color and depth information. Using complicated setups like RGB-D cameras, such issues are solved and the robustness of visual SLAM systems can be greatly improved. However, most visual methods perform poorly in dynamic environments. In these techniques, dynamic objects are considered typically as spurious data and removed as outliers using RANSAC and robust cost function. Dynamic environments, on the other hand, are widespread characteristics of many robotic applications.

Conventional approaches in mapping assume that the environment is static. Although, the static assumption holds true in a single mapping run in small scale scenarios, change is inevitable when dynamic elements exist or large-scale mapping necessary. This approach generally succeeds in ignoring moving objects by setting their corresponding key points and the use of distant key frames. However, when dealing with dynamic environments, the system becomes less accurate as the objects, that have remained static in several key frames, are mapped in the reconstruction.

Another biggest issues in robot navigation is unstructured environments. In unstructured environments, it is not easy to find discrete geometries because of noisy edge or plane. Significant research has been carried out for unstructured environments especially in the field of autonomous navigation, and a number of effective approaches have been developed. However, there is no effective RGB-D SLAM method for real-world unstructured and dynamic environments.

In summary, the research community has addressed SLAM from many different angles. However, the vast majority of the approaches and datasets assume a static environment. Consequently, they can only manage small fractions of dynamic content by classifying them as outliers. Although the static assumption holds for some robotic applications, it limits the applicability of visual SLAM in many relevant cases, such as intelligent autonomous systems operating in populated real-world environments over long periods. By classifying dynamic content as outliers, a small fraction can be managed. However, SLAM problem in highly dynamic scenes is still not solved completely because there is no suggested framework found in the literature.

In our work, we reconstruct our scene geometry using Signed Distance Function (SDF) and therefore we can directly generate the mesh of the environment using such representation without using object tracking and object classifier. Moreover, number of dynamic objects or their speeds do not limit our approach.

Our proposed method, dense SDF-based dynamic mapping approach, can operate in environments where high dynamics exist without depending on moving objects. In addition, if a static object moves, the corresponding voxels are removed successfully from the mesh. After performing a complete evaluation of our proposed method for several sequences of the TUM, Bonn and VolumeDeform dataset, our method has demonstrated to have an improved pose estimation capability even though there exists dynamic elements in the scene. In addition, in order to evaluate the outdoor performance of our method, we use commercially available ZED camera for map generation and dynamic filtering. Experiments illustrate that our method produce consistent result without generating an artifact of dynamic object both in indoor and outdoor applications. These are demonstrations of real world dynamic environments of our approach.

User-friendliness and the unconstrained aspect of our method is seen in representing surfaces using truncated signed distance function which is extremely easy and it provides an efficient mechanism in consistent surface estimation. However, surface representation using TSDF is memory intensive because required memory for TSDF volume scales cubically and it depends on the grid resolution. Hence, special care has to be taken for efficient memory usage considering the performance. In addition, TSDF has intensive calculations. For example, for fusing a data in VGA format, it requires basically 0.3 million operations. However, since each pixel is independent from each other, GPU can be utilized in parallel to have a real time performance.

7.1 Future Works

If a voxel is empty in enough time, then we mark it as free voxel. The speed of dynamic elements becomes important because, the environment should allow us to capture the static part of scene in the first frames. Therefore, it is so challenging to generate the static part of environments having very high dynamics at the beginning. We compare our method with other state of art systems using repositories which are available in the literature. These RGB-D datasets have no build-in tool to evaluate the dynamic handling capability of state of art SLAM systems. However, there should be limit for dynamic, which can be handled by our proposed method. This limit can be instantaneous rotation or translation of moving object or any type of metric, which should be identified.

Our proposed method can operate in environments where high dynamics exist without depending on moving objects. We find a dynamic label image from the image registration residuals. Actually, we have a silhouette of the moving object and using this information, it is possible to extract the dynamic object and to find the some properties of dynamic elements such as speed. These open questions can be taken as a guideline for further work on the approach for visual SLAM methods.

In addition, the SDF can encode surface interfaces at sub-voxel accuracy through interpolation, however, it can fail at sharp corners and edges, therefore they are not straightforward to extract from a SDF representation and such type of structures requires special effort such as selecting a suitable voxel element size and truncation distance. Truncation distance expresses a prior information about the average thickness of object in the environment. Using a feature-preserving algorithm with adaptive variable voxel size can improve the surface extraction on sharp corners; however, it is left as a future work.

Visual SLAM allows robots to find the location of itself with reference to its surrounding environment. With the help of this technology, a device can capable of geographical understanding, for example, it can determine the shape of an area. Such property is an important feature for Augmented Reality applications because it enables pairing of RGB image and allows autonomous objects such as automobiles and quadrotors to track the environment. Moreover, Visual SLAM overcomes the problem of GPS limitation and finally RGB-D sensor becomes no longer as camera, it turns into an artificial eye that can measure the distance to surface and pair the images with Visual SLAM technology.

Visual SLAM is a foundation to get truly AR because localizing the object in the map or capturing 3D scene is not sufficient, however, using deep learning and cloud computing, we have next level of adoption. In short, Visual SLAM can localize the object, however we need the semantic information about the object derived from the artificial intelligence.

If we load a device with Visual SLAM technology, then we can get the user experience by the real time application of augmented reality. Integration of Visual

SLAM with Augmented Reality helps us to combine global location with digital elements producing in real time. For example, imagine if you are walking and passing a restaurant. We could use our smartphone or smart device such as glasses to view menus about the restaurant by combining the street where the restaurant is located with AR. Moreover, we could get the 3D video content of any store. This is the new opportunity for marketing and it can be easily adopted to shopping and entertainment.

Metaverse is used to represent the hypothetical synthetic environment which is the combination of "transcending (*meta*)" and the "universe (*verse*)". It is a link between the digital world and physical system. This phenomena "metaverse" is evolving and better results are obtained by introduction of newly portable devices. This is the truest form of the technology: It is intuitive, offering a plethora of possibilities for developers and users of tomorrow. Even though, state of art visual SLAM system including our proposed method, DUDMAP, already provide a foundation for three-dimensional recognition, the metaverse requires more understandings about more complex environments for integration of digital object and real life. HoloLens and Oculus has already special product in order to have basics of virtual reality technology to explore the potential of mixed reality interaction. Apple announced ARKit for 3D keypoints tracking. In metaverse, the virtual universe in constructed by acquiring the 3D structure of a scene and this information helps us to build digital twin construction, which is an important role for connecting artificial intelligence to get conversion with the physical world. Therefore, it is very vital to ensure the accuracy of object registration, and the interaction with the physical world in the metaverse. Therefore, it is expected to have more precise and computationally effective SLAM algorithms in the metaverse.

In the metaverse, human and their digital twins or representatives called as avatars will connect and co-exist. In order to form a proper connection between physical and digital environments, a deep understanding of both world is necessary. In real world, our eyes provides us spatial information and we can built a 3D reconstruction with knowing the exact location of each object. Likewise, the metaverse needs to correct 3D reconstruction of the scene. In order to achieve this goal, Visual SLAM allows us to create 3D structure of an unknown environment and it should solve simultaneously the challenging problems of unknown space, uncontrollable camera motion and camera drift in real-time. Therefore, it is clear that a new RGB-D dataset containing fast and slow camera motions and varying degrees of dynamic elements would be greatly appreciated by researchers if made available.

REFERENCES

[1] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), 2012, pp. 573– 580.

[2] D.G. Lowe, ”Distinctive image features from scale-invariant keypoints,” Gonput.Vis. 60(2), pp. 91-110, 2004

[3] Mur-Artal, R.; Tardós, J.D. ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Trans. Robot. 2017, 33, 1255–1262

[4] Pire, T., Fischer, T., Castro, G., De Cristoforis, P., Civera, J., and Jacobo Berlles, J. (2017). S-PTAM: Stereo Parallel Tracking and Mapping. Robotics and Autonomous Systems, 27-42.

[5] Labbe, M. and Michaud, F. (2013). Appearance-based loop closure detection for online large-scale and long-term operation. IEEE Transactions on Robotics, 29,734-745.

[6] Kerl, C., Sturm, J., and Cremers, D. (2013). Dense visual SLAM for RGB-D cameras. In Proceedings, IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2100- 2106.

[7] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, “Fast odometry and scene flow from RGB-D cameras based on geometric clustering,” in IEEE Intl. Conf. on Robotics and Automation (ICRA), 2017, pp. 3992–3999.

[8] Diaz-Chito, Katerine, Aura Hernez-Sabatnd Antonio M. L. ”A reduced feature set for driver head pose estimation.” Applied Soft Computing 45, pp. 98-107, 2016.

[9] Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., and Leutenegger, S. (2016). ElasticFusion: Real-time dense SLAM and light source estimation. The International Journal of Robotics Research, 35, 1697-1716.

[10] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D Reconstruction at Scale using Voxel Hashing. Proc. of the SIGGRAPH Asia, 32(6), 2013.

[11] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, “ElasticFusion: Dense SLAM without a pose graph,” Robotics: Science and Systems (RSS), 2015.

[12] M. Runz and L. Agapito, “Co-fusion: Real-time segmentation, tracking and fusion of multiple objects,” in IEEE Intl. Conf. on Robotics and Automation (ICRA), 2017, pp. 4471–4478.

[13] R. Scona, M. Jaimez, Y.R. Petillot, M. Fallon, and D. Cremers., Staticfusion: Background reconstruction for dense RGB-D SLAM in dynamic environments. In Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA), 2018.

[14] E. Palazzolo, J. Behley, P. Lottes, P. Gigue`re, and C. Stachniss. “Refusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals”. CoRR, abs/1905.02082, 2019

[15] Y. Wang and S. Huang, “Motion segmentation based robust RGB-D SLAM,” in WCICA, pp. 3122–3127, IEEE, 2014.

[16] H. Kim and J. H. Kim, “Effective background model-based RGB-D dense visual odometry in a dynamic environment,” IEEE Transaction on Robotics, vol. 32, no. 6, pp. 1565–1573, 2016.

[17] H. Azartash, K.-R. Lee, and T. Q. Nguyen, “Visual odometry for RGB-D cameras for dynamic scenes,” in Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on. IEEE, 2014, pp. 1280–1284.

[18] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, “Robust monocular SLAM in dynamic environments,” in Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on. IEEE, 2013, pp. 209–218.

[19] Kitt, F. Moosmann, and C. Stiller, “Moving on to dynamic environments: Visual odometry using feature classification,” in Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. IEEE, 2010, pp. 5551–5556.

[20] B. Bescos, J. M. Facil, J. Civera, and J. Neira, “Dynslam: Tracking, mapping and inpainting in dynamic scenes,” arXiv preprint arXiv:1806.05620, 2018.

[21] Wu Y, Luo L, Yin S, Yu M, Qiao F, Huang H, Shi X, Wei Q, Liu X. An FPGA Based Energy Efficient DS-SLAM Accelerator for Mobile Robots in Dynamic Environment. Applied Sciences. 2021; 11(4):1828. https://doi.org/10.3390/app11041828

[22] C.; Liu, Z.; Liu, X.-J.; Xie, F.; Yang, Y.; Wei, Q.; Fei, Q. DS-SLAM: A semantic visual slam towards dynamic environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1168–1174.

[23] Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 2017, 39, 2481–2495.

[24] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. An Invitation to 3D Vision: From Images to Geometric Models. Springer Verlag, 2003.

[25] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohli, J. Shotton, S. Hodges, and A.W. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In ISMAR, pages 127–136, 2011.

[26] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. Computer Graphics, 21(4):163–169, 1987.

[27] J. W. Li, W. Gao, and Y. H. Wu, “Elaborate scene reconstruction with a consumer depth camera,” International Journal of Automation and Computing, pp. 1–11, 2018.

[28] A. Taneja, L. Ballan and M. Pollefeys, "Geometric change detection in urban environments using images", IEEE transactions on pattern analysis and machine intelligence, vol. 37, no. 11, pp. 2193-2206, 2015.

[29] Martin Rünz, Maud Buffier, and Lourdes Agapito. “MaskFusion: Real-time Recognition, Tracking and Reconstruction of Multiple Moving Objects”. In Proceedings of the Inter- national Symposium on Mixed and Augmented Reality

[30] T. Bagautdinov, F. Fleuret, and P. Fua. Probability occupancy maps for occluded depth images. In Computer Vision and Pattern Recognition (CVPR), 2015. https://www.epfl.ch/labs/cvlab/data/data-rgbd-pedestrian/

[31] Innmann M., Zollhöfer M., Nießner M., Theobald C., Stamminger M.: VolumeDeform: "Real-Time Volumetric Non-rigid Reconstruction” 14th European Conference on Computer Vision (ECCV) doi: 10.1007/978-3-319-46484-8_22 https://cloud9.cs.fau.de/index.php/s/46qcNZSNePHx08A

[32] F. Steinbrücker, J. Sturm and D. Cremers, "Volumetric 3D mapping in real-time on a CPU," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 2021-2028, doi: 10.1109/ICRA.2014.6907127.

[33] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. 2017. BundleFusion: Real-Time Globally Consistent 3D Reconstruction Using On-the-Fly Surface Reintegration. ACM Trans. Graph. 36, 3, Article 24 (June 2017), 18 pages. DOI:https://doi.org/10.1145/3054739

[34] C. Malleson, J. Guillemaut and A. Hilton, "Hybrid Modeling of Non-Rigid Scenes From RGBD Cameras," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 8, pp. 2391-2404, Aug. 2019, doi: 10.1109/TCSVT.2018.2863027.

[35] https://www.stereolabs.com

[36] M. Runz, M. Buffier and L. Agapito, "MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects," 2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Munich, Germany, 2018, pp. 10-20, doi: 10.1109/ISMAR.2018.00024.

[37] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM," jul 2020 [Online]. Available: http://arxiv.org/abs/2007.11898

[38] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.

[39] Y. Liu and J. Miura, "RDS-SLAM: Real-Time Dynamic SLAM Using Semantic Segmentation Methods," in IEEE Access, vol. 9, pp. 23772-23785, 2021, doi: 10.1109/ACCESS.2021.3050617.

[40] Y. Fan et al., "Semantic SLAM With More Accurate Point Cloud Map in Dynamic Environments," in IEEE Access, vol. 8, pp. 112237-112252, 2020, doi: 10.1109/ACCESS.2020.3003160.

[41] T. Zhang, Y. Nakamura, "PoseFusion: Dense RGB-D SLAM in Dynamic Human Environments." Xiao J.; Kröger T.; Khatib O. Proceedings of the 2018 International Symposium on Experimental Robotics, pp.772-780, 2018. hal-01893144

[42] Z. Cao, G. Hidalgo, T. Simon, S. -E. Wei and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 1, pp. 172-186, 1 Jan. 2021, doi: 10.1109/TPAMI.2019.2929257.

[43] T. Zhang, H. Zhang, Y. Li, Y. Nakamura and L. Zhang, "FlowFusion: Dynamic Dense RGB-D SLAM Based on Optical Flow," 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 2020, pp. 7322-7328, doi: 10.1109/ICRA40945.2020.9197349.

[44] D. Sun, X. Yang, M. Liu and J. Kautz, "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 8934-8943, doi: 10.1109/CVPR.2018.00931.

[45] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, “Adaptively sampled distance fields: A general representation of shape for computer graphics,” 2000.

[46] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachiss, and W. Burgard, “OctoMap: An efficient probabilistic 3D mapping framework based on octrees,” Autonomous Robots, 2013.

[47] R. E. Bridson, Compuational Aspects of Dynamic Surfaces. PhD thesis, Standford, CA, USA, 2003.

[48] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross, “Optimized spatial hashing for collision detection of deformable objects,” Proceedings of Vision, Modeling, Visualization (VMV 2003), pp. 47–54, 2003.

[49] Blanco, J. (2012). A tutorial on SE(3) transformation parameterizations and on-manifold optimization.

[50] Ivancevic, Vladimir & Ivancevic, Tijana. (2011). Lecture Notes in Lie Groups.

[51] Shirley, P. and Neeman, H., "Volume Visualization at the Center for Supercomputing Research and Development", Proceedings of the Chapel Hill Workshop on Volume Visualization, Chapel Hill, NC, May 1989, 17-20.

[52] X Choi, Y. K., & Hahn, J. K. (2007). Shrink-wrapped isosurface from cross sectional images. IEICE transactions on information and systems, E90-D(12), 2070–2076. https://doi.org/10.1093/ietisy/e90-d.12.2070

[53] Cline, H. E., Lorensen, W. E., Ludke, S., Crawford, C. R. and Teeter, B. C., "Two Algorithms for the Three-Dimensional Reconstruction of Tomograms", Medical Physics, 15, 3 (May/June 1988), 320-327.

[54] Lorensen W, Cline H. Marching cubes: a high resolution 3D surface construction algorithm. Computer Graphics 1987;21(4):163–9.

[55] Timothy S. Newman, Hong Yi, A survey of the marching cubes algorithm, Computers & Graphics, Volume 30, Issue 5, 2006, Pages 854-879, ISSN 0097-8493, https://doi.org/10.1016/j.cag.2006.07.021.

[56] Lorensen, W. E., "Extracting Surfaces from Medical Volumes," SIGGRAPH 94

[57] David B. Kirk and Wen-mei W. Hwu. 2010. Programming Massively Parallel Processors: A Hands-on Approach (1st. ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[58] S. Se, D. Lowe, and J. Little. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. International Journal of Robotics Research, 21(8), August 2002.

[59] S. Se, H. Ng, P. Jasiobedzki, and T. Moyung. Vision based Modeling and Localization for Planetary Exploration Rovers. In International Astronautical Congress. Proceedings IAC 2004., October 2004.

[60] Weingarten, J. Feature-based 3D SLAM. PhD Thesis, Swiss Federal Institute of Technology Lausanne, EPFL, no 3601, Dir.: Roland Siegwart,(2006).

[61] W. Burgard, A.B. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. Artificial Intelligence, 114(1-2):3–55, 1999.

[62] Besl, P. and McKay, N. D. (1992). A method for registration of 3-d shapes. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 14(2):239256.

[63] T. Bailey. Mobile Robot Localisation and Mapping in Extensive Outdoor Environments. PhD thesis, Univ. of Sydney, 2002.

[64] R. Smith, M. Self, and P. Cheeseman, “Estimating uncertain spatial relationships in robotics”, Autonomous Robot Vehicles, Springer, Germany, pp. 167–193, 1990.

[65] A. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, “A Solution to the Simultaneous Localization and Map Building Problem”. IEEE Transaction on Robotic and Automation, vol. 17, pp. 229–241, 2001.

[66] H. F. Durrant-Whyle, and T. Bailey, ”Simultaneous Localization and Mapping: Part I“. IEEE Robotic Automation Magazine, vol. 13, pp. 99– 108, 2006.

[67] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: A factoredsolution to the simultaneous localization and mapping problem”, in Proceedings of the National Conference on Artificial Intelligence, Edmonton, Canada, pp. 593–598, August 2002.

[68] W. Burgard, A.B. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. Artificial Intelligence, 114(1-2):3–55, 1999.

[69] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige. Double window optimisation for constant time visual SLAM. In IEEE International Conference on Computer Vision (ICCV), 2011.

[70] Schneider, T., Dymczyk, M., Fehr, M., Egger, K., Lynen, S., Gilitschenski, I., and Siegwart, R. (maplab: An open framework for research in visual-inertial mapping and localization. IEEE Robotics and Automation Letters, 3:1418- 1425.

[71] Yi, L., Fei, G., Tong, Q., Wenliang, G., Tianbo, L., William, W., Zhenfei, Y., and Shaojie, S. (2017). Autonomous aerial navigation using monocular visual-inertial fusion. Journal of Field Robotics, 35 :23-51.

[72] Pire, T., Fischer, T., Castro, G., De Cristoforis, P., Civera, J., and Jacobo Berlles, J. (2017). S-PTAM: Stereo Parallel Tracking and Mapping. Robotics and Autonomous Systems, 27-42.

[73] Schlegel, D., Colosi, M., and Grisetti, G. (2017). ProSLAM: Graph SLAM from a programmer's perspective.

[74] Galvez-Lopez, D. and Tardos, J. D. (2012). Bags of binary words for fast place recognition in image sequences.IEEE Transactions on Robotics, 28,1188-1197

[75] Kerl, C., Sturm, J., and Cremers, D. (2013). Dense visual SLAM for RGB-D cameras. In Proceedings, IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2100- 2106.

[76] Gutierrez-Gomez, D., Mayol-Cuevas, W., and Guerrero, J. J. (2016). Dense RGB-D visual odometry using inverse depth. Robotics and Autonomous Systems, 75:571-583.

[77] Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J. J., and McDonald, J. (2015). Real-time large-scale dense RGB-D SLAM with volumetric fusion. The International Journal of Robotics Research, 34, 598-626.

[78] Dai, A., Niener, M., Zollofer, M., Izadi, S., and Theobalt, C. (2017). Bundlefusion: Real-time globally consistent 3D reconstruction using on the fly surface reintegration. ACM Transactions on Graphics.

[79] Harmat, A., Trentini, M., and Sharf, I. (2015). Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments. Journal of Intelligent & Robotic Systems, 78, 291-317.

[80] Endres, F., Hess, J., Sturm, J., Cremers, D., and Burgard, W. (2014). 3-D mapping with an RGB-D camera. IEEE Transactions on Robotics, 30,177-187.

[81] Moore, T. and Stouch, D. (2014). A generalized extended Kalman filter implementation for the Robot Operating System. In Proceedings International Conference on Intelligent Autonomous Systems. Springer.

[82] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). Octomap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots, 34,189- 206.

[83] Labbe, M. and Michaud, F. (2013). Appearance-based loop closure detection for online large-scale and long-term operation. IEEE Transactions on Robotics, 29,734-745.

[84] Labbe, M. and Michaud, F. (2014). Online global loop closure detection for large-scale multi-session graph based SLAM. In Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2661-2666.

[85] O. Wasenmüller, M. Meyer and D. Stricker, "CoRBS: Comprehensive RGB-D benchmark for SLAM using Kinect v2," 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), 2016, pp. 1-7, doi: 10.1109/WACV.2016.7477636.

[86] A. Handa, T. Whelan, J. McDonald and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 1524-1531, doi: 10.1109/ICRA.2014.6907054.

[87] Kerl, C., Sturm, J., Cremers, D.: Dense visual slam for RGB-D cameras. In: International Conference on Intelligent Robot Systems (IROS). (2013)

[88] Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Robotics-DL tentative, International Society for Optics and Photonics (1992) 586–606

[89] Wasenmüller O., Ansari M.D., Stricker D. (2017) DNA-SLAM: Dense Noise Aware SLAM for ToF RGB-D Cameras. In: Chen CS., Lu J., Ma KK. (eds) Computer Vision – ACCV 2016 Workshops. ACCV 2016. Lecture Notes in Computer Science, vol 10116. Springer, Cham. https://doi.org/10.1007/978-3-319-54407-6_42

[90] R.A. Newcombe, S. Lovegrove, and A.J. Davison. DTAM: Dense tracking and mapping in real-time. In Proc. of the Intl. Conference on Computer Vision (ICCV), pages 2320– 2327, 2011.

[91] M.Werlberger,W. Trobin, T. Pock, A.Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 Optical Flow. In Proc. of the British Machine Vision Conference (BMVC), September 2009.

[92] J. W. Li, W. Gao, and Y. H. Wu, "Elaborate scene reconstruction with a consumer depth camera," International Journal of Automation and Computing, pp. 1–11, 2018.

[93] C. Tomasi, R. Manduchi. Bilateral filtering for gray and color images. In Proceedings of the 6th International Conference on Computer Vision, IEEE, Bombay, India, pp. 839– 846, 1998.

[94] Birk, A., & Carpin, S. (2006). Merging occupancy grids from multiple robots. Proceedings of the IEEE, 94(7), 1384–1397.

[95] T. Bailey. Mobile Robot Localisation and Mapping in Extensive Outdoor Environments. PhD thesis, Univ. of Sydney, 2002.

[96] J. Sturm, S. Magnenat, N. Engelhard, F. Pomerleau, F. Colas, W. Bur- gard, D. Cremers, and R. Siegwart. Towards a benchmark for rgb-d slam evaluation. In Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf. (RSS), June 2011

APPENDICES

A. Mathematical Preliminaries

The vec operator that stacks all the columns of an M ×N matrix to form a MN ×1 vector

$$vec\left(\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}\right) = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} \tag{A.1}$$

The cross product operator $[.]_\times$ maps a 3 × 1 vector to skew-symmetric matrix :

$$\omega = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad [\omega]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \tag{A.2}$$

The inverse cross product operator $[.]\vee$ is the inverse of the cross product as:

$$\begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}^{\vee} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{A.3}$$

B. Rigid Body Motion

In 3D reconstruction, the main aim is to capture the geometry and appearance of the objects. In rigid bodies, poses of moving rigid bodies are related point by point with changing location and orientation between poses. In rigid body motion, the distance between any two points remains the same and orientation of the body is conserved. Two particular groups $SO(3)$ and $SE(3)$ are particularly interested in robotics society. $SO(3)$ is the special orthogonal group that represents rotations and SE(3) is the special Euclidean group that represents rigid body motions.

In rigid body motion, two reference frames which are body frame and inertial frame are used. Body frame remains fixed to the body whereas the inertial frame is fixed. For example, let us consider the a point p on the rigid body after the body undergoes a rotation of angle α about the inertial z-axis. The relation between the initial coordinates and final coordinates is the rotation matrix. Similarly rotate the frame about the new y and z axes with angles β and θ, we have a net orientation R (α, β, θ) and the angles (α, β, θ) is used to represent the rotation sequence. The angles (α,β,θ) are known as the ZYZ Euler angles. Since all rotations performed about the principal axes of the moving frame, elementary rotations are defines as

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \tag{A.4}$$

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{A.5}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.6}$$

If we first rotate the B frame about the z-axis of frame B by an angle α, then rotating about the (new) y-axis of frame B by an angle β, and then rotating about the (once again, new) z-axis of frame B by an angle γ. Then the final form becomes

$$R_{BA} = R_z(\alpha)R_y(\beta)R_z(\gamma) \tag{A.7}$$

$$R_{BA} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.8}$$

$$R_{BA} = \begin{bmatrix} c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma & -c_\alpha c_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta \\ s_\alpha c_\beta c_\gamma + c_\alpha s_\gamma & -s_\alpha c_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta \\ -s_\beta c_\gamma & s_\beta s_\gamma & c_\beta \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \text{(A.9)}$$

where c_α and s_α denotes $\cos(\alpha)$ and $\sin(\alpha)$, respectively and similarly for the other terms.

$SO(3)$ is Special Orthogonal Group which is defined as

$$SO(3) \ = \{R \in R^{3x3} | R^T R = RR^T = I, \det(R) = 1\} \quad \text{(A.10)}$$

where R is rotational matrix, I is identity matrix and $\det(.)$ is the determinant.

If we have a rotation R ∈ $SO(3)$, the Euler angles can be calculated by solving equation (3.10) for α, β, and γ. If $\sin(\beta) \neq 0$, the solutions become

$$\beta = atan2(\sqrt{{r_{31}}^2 + {r_{32}}^2}, r_{33}) \quad \text{(A.11)}$$

$$\alpha = atan2(\frac{r_{23}}{s_\beta}, \frac{r_{13}}{s_\beta}) \quad \text{(A.12)}$$

$$\gamma = atan2(\frac{r_{32}}{s_\beta}, \frac{-r_{31}}{s_\beta}) \quad \text{(A.13)}$$

where $atan2(y, x)$ uses the sign of both x and y to determine the quadrant. ZYZ Euler angles are an example of a local parameterization of $SO(3)$.

The equivalent axis representation, the singularities in the parameterization occur at R = I, the identity rotation. In particular, we note that (α, β, γ) of the form (α, 0,−α) yields (α, 0,−α) = I. The singularities are referred as the absence of solution to the inverse problem of finding the Euler angles. Therefore, there are infinitely many representations of the identity rotation in the ZYZ. In order to overcome this problem, another Euler angle sequence such as ZYX and YZX Euler angle parameterizations can be utilized because they have the advantage of not having a singularity at the identity, R = I. However, they have singularities at other, different, orientations. As an example , in ZYX Euler angles, the singularity occurs when θ = −π/2. This situation is a fundamental topological fact that singularities can never be eliminated in any 3-dimensional representation of $SO(3)$.

The Lie-algebra of $SO(3)$ is termed $so(3)$ and is defined by

$$so(3) = \left\{ A = [\omega_x] = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \middle| A^T = -A \right\} \tag{A.14}$$

In robotics, any rotation of a body about a given axis is a common motion. If $\omega \in R^3$ be a unit vector which specifies the direction of rotation and $\theta \in R$ be the angle of rotation in radians. It is clear that every rotation corresponds to $R \in SO(3)$. Now, consider the point on the rotating body which has constant unit velocity about axit ω. Then the velocity of the point, $\dot{q}$, can be defined as

$$\dot{q}(t) = \omega \times q(t) = [\omega]_\times q(t) \tag{A.15}$$

Since this equation is a time-invariant linear differential equation, then

$$q(t) = e^{[\omega]_\times t}\, q(0) \tag{A.16}$$

where $q(0)$ is the initial position and $e^{[\omega]_\times t}$ is the matrix exponential and it can be written as

$$e^{[\omega]_\times t} = I + [\omega]_\times t + \frac{([\omega]_\times t)^2}{2!} + \frac{([\omega]_\times t)^3}{3!} + \cdots \tag{A.17}$$

Then, if the point is rotated about the axis ω with an unit velocity for θ units of time, then final rotation becomes

$$R(\omega, \theta) = e^{[\omega]_\times \theta} \tag{A.18}$$

Equation (3.19) is not useful from because it is an infinite series and, hence, it requires high computational power.

Lemma 1 : Given $[\omega]_\times \epsilon \, so(3)$, following equations are valid and higher powers of $[\omega]_\times$ can be calculated.

$$([\omega]_\times)^2 = \omega\omega^T - \|\omega\|^2 I \tag{A.19}$$

$$([\omega]_\times)^3 = -\|\omega\|^2 [\omega]_\times \tag{A.20}$$

Using this lemma with $\|\omega\| = 1$, then

$$e^{[\omega]_\times \theta} = I + \left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \cdots\right)[\omega]_\times + \left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} \cdots\right){[\omega]_\times}^2 \tag{A.21}$$

Therefore

$$e^{[\omega]_\times \theta} = I + [\omega]_\times \sin(\theta) + {[\omega]_\times}^2 (1 - \cos(\theta)) \tag{A.22}$$

This formula (3.23) is known as Rodrigues' formula and it is an efficient method for computing matrix exponential. Indeed, this equation verify that $e^{[\omega]_\times \theta}$ is indeed a rotation matrix.

Let $R = e^{[\omega]_\times \theta}$ and $[\omega]_\times \epsilon\, so(3)$, using Rodrigues' formula we get

$$[e^{[\omega]_\times \theta}]^{-1} = e^{-[\omega]_\times \theta} = e^{{[\omega]_\times}^T \theta} = [e^{[\omega]_\times \theta}]^T \tag{A.23}$$

This equation implies that $R^T = R^{-1}$, it is then easy to show that $R^T R = I$ and $\det(R) = 1$. Then we can conclude that given a skew-symmetric matrix, exponentials of skew symmetric matrix are orthogonal.

$$e^{[\omega]_\times \theta}\ \epsilon\ SO(3) \tag{A.24}$$

Equation (3.25) indicate that the exponential map converts skew symmetric matrices into orthogonal matrices. The skew symmetric matrix is a representation of an axis of rotation and this map generates a rotation about a given axis by a given or specified amount θ. This is the relationship between skew-symmetric matrices and orthogonal matrices.

Each rotation matrix can be represented by a matrix exponential of some skew-symmetric matrix. In other words, the mapping exp : $so(3) \rightarrow SO(3)$ is surjective

Let $R\ \in\ SO(3)$, then there exist ω $\in\ R^3$, $\|\omega\| = 1$ and θ ∈ R such that $R = e^{[\omega]_\times \theta}$

Proposition

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = e^{[\omega]_\times \theta} \tag{A.25}$$

This statement declares that the exponential map is surjective onto $SO(3)$.

$$e^{[\omega]_\times \theta} = I + [\omega]_\times \sin(\theta) + {[\omega]_\times}^2 (1 - \cos(\theta)) \tag{A.26}$$

$$e^{[\omega]_\times \theta}$$

$$= \begin{bmatrix} 1-\left(1-\cos(\theta)\right)({\omega_2}^2+{\omega_3}^2) & \omega_1\omega_2(1-\cos(\theta))-\omega_3\sin(\theta) & \omega_1\omega_3(1-\cos(\theta))+\omega_2\sin(\theta) \\ \omega_1\omega_2(1-\cos(\theta))+\omega_3\sin(\theta) & 1-\left(1-\cos(\theta)\right)({\omega_1}^2+{\omega_3}^2) & \omega_2\omega_3(1-\cos(\theta))-\omega_1\sin(\theta) \\ \omega_1\omega_3(1-\cos(\theta))-\omega_2\sin(\theta) & \omega_2\omega_3(1-\cos(\theta))+\omega_1\sin(\theta) & 1-(1-\cos(\theta))({\omega_1}^2+{\omega_2}^2) \end{bmatrix}$$

$$= \begin{bmatrix} {\omega_1}^2(1-\cos(\theta))+\cos(\theta) & \omega_1\omega_2(1-\cos(\theta))-\omega_3\sin(\theta) & \omega_1\omega_3(1-\cos(\theta))+\omega_2\sin(\theta) \\ \omega_1\omega_2(1-\cos(\theta))+\omega_3\sin(\theta) & {\omega_2}^2(1-\cos(\theta))+\cos(\theta) & \omega_2\omega_3(1-\cos(\theta))-\omega_1\sin(\theta) \\ \omega_1\omega_3(1-\cos(\theta))-\omega_2\sin(\theta) & \omega_2\omega_3(1-\cos(\theta))+\omega_1\sin(\theta) & {\omega_3}^2(1-\cos(\theta))+\cos(\theta) \end{bmatrix}$$

$$trace\left(e^{[\omega]_\times \theta}\right) = trace(R) \tag{A.27}$$

$$r_{11} + r_{22} + r_{33} = 2\cos(\theta) + 1 \tag{A.28}$$

In order to verify that this equation has a solution, we use the property that trace of R is equal to the sum of its eigenvalues. Since R preserves lengths and det R = +1, its eigenvalues have magnitude 1 and occur in complex conjugate pairs. Then,

$$-1 \le trace(R) \le 3 \tag{A.29}$$

$$\theta = \cos^{-1}\frac{trace(R) - 1}{2} \tag{A.30}$$

$\theta \mp 2\pi n$ or $-\theta \mp 2\pi n$ could be chosen in order to avoid ambiguity. Equating the off-diagonal terms, then we get

$$r_{32} - r_{23} = 2\omega_1\sin(\theta) \tag{A.31}$$

(A.32)

$$r_{21} - r_{12} = 2\omega_3 \sin(\theta)$$

$$r_{13} - r_{31} = 2\omega_2 \sin(\theta) \tag{A.33}$$

Choosing $\theta \neq 0$,

$$\omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \frac{1}{2\sin(\theta)} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{A.34}$$

If $\theta + 2\pi$ is selected, the rotation axis would have been ω and the exponential map is a many to one map from R^3 onto $SO(3)$. For example, if $R = I$, then $trace(R)$=3 , $\theta = 0$ and ω is arbitrary. If $R \neq I$, the above construction shows that there are two distinct ω. This proves that the exponential map is surjective onto $SO(3)$.

Rigid body motion representation forms a Special Euclidean group $\mathbb{SE}(\mathbf{3})$, which is also known as Lie group [49]. A Euclidean transformation in 3D consists of a translation and rotation. There exist two main advantage of using Lie algebra for describing the rigid body motion. They allow us to have global description without suffering from singularities. Such type of singularities are unavoidable if Euler angle is chosen to represent the rotation. The second advantage is that Lie algebra provides a geometric description of rigid body motion which simplifies the analysis of mechanism of rigid body motion.

C. Transformation Matrices in Robotic Representation

Special Euclidean $\mathbb{SE}(3)$ group is defined as a noncommutative product of 3D rotations and 3D translations. The commonly used approach is to use 4x4 transformation matrix T_A^B in order to transform a 3D point from one coordinate frame

A to into another coordinate frame B. Special Euclidean group $\mathbb{SE}(3)$ is formally defined as follows:

$$\mathbb{SE}(3) = \left\{ \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} | R \in \mathbb{SO}(3), t \in R^3 \right\} \tag{A.35}$$

where t is a 3D translation vector and R is rotation matrix that belongs to the Special Orthogonal group $\mathbb{SO}(3)$. We can write the full homogenous transformation matrix T_A^B from frame A to frame B and the respective inverse transformation T_B^A which maps back from frame B to frame A, as

$$T_B^A = [T_A^B]^{-1} = \begin{bmatrix} {R_A^B}^T & -{R_A^B}^T t_A^B \\ 0 & 1 \end{bmatrix} \tag{A.36}$$

This matrix representation allows to easily transform a 3D point p_A in frame A to the respective 3D point $p_B$in coordinate frame B:

$$p_B = R_A^B p_A + t_A^B \tag{A.37}$$

Consider the motion of a rigid body rotated about a line in the z direction, through the point (0, p, 0). If we let θ denote the amount of rotation, then the new orientation and the new coordinates for the origin become

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.38}$$

$$P_z(\theta) = \begin{bmatrix} 0 \\ p \\ 0 \end{bmatrix} \tag{A.39}$$

The homogeneous representation of the configuration of the rigid body is given by

$$T_B^A = \begin{bmatrix} \cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta & \cos(\theta) & 0 & p \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.40}$$

For this case, when the angle $\theta = 0$, T_B^A gives that the relative displacement between the two frames is a pure translation along the y-axis. In addition, successive transformation can be written as,

$$T_A^C = T_B^C T_A^B \tag{A.41}$$

The notion of the exponential mapping introduced for SO(3) can be generalized to the Euclidean group, SE(3). If $\omega \in R^3$ be a unit vector which specifies the direction of rotation and let $\theta \in R$ be the angle of rotation in radians. If the link rotates with unit velocity, then the velocity of the tip point p(t) can be written as

$$\dot{p}(t) = \omega \times (p(t) - q) \tag{A.42}$$

Where q is a point on the axis of rotation. If this equation is converted to homogeneous coordinates using ζ

$$\zeta = \begin{bmatrix} [\omega]_\times & v \\ 0 & 0 \end{bmatrix} \tag{A.43}$$

$$v = -\,\omega \times q \tag{A.44}$$

Equation $\dot{p}(t)$ can then be rewritten as

$$\begin{bmatrix} \dot{p} \\ 0 \end{bmatrix} = \begin{bmatrix} [\omega]_\times & -\,\omega \times q \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} = \zeta \begin{bmatrix} p \\ 1 \end{bmatrix} \xrightarrow{yields} \dot{p} = \zeta p \tag{A.45}$$

The solution of the differential equation is

$$p(t) = e^{\zeta t}\, p(0) \tag{A.46}$$

where $p(0)$ is the initial position of the point and $e^{\zeta t}$ is the matrix exponential and it can be written as

$$e^{\zeta t} = I + \zeta t + \frac{(\zeta t)^2}{2!} + \frac{(\zeta t)^3}{3!} + \cdots \tag{A.47}$$

If prismatic joint with unit velocity v is modeled, then the velocity of a point is

$$\dot{p}(t) = v \tag{A.48}$$

Then the solution of this equation is

$$p(t) = e^{\zeta t}\, p(0) \tag{A.49}$$

where

$$\zeta = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix} \tag{A.50}$$

The 4×4 matrix ζ given in equations () and () represents the generalization of the skew-symmetric matrix $\zeta \in so(3)$.

If $\omega = 0$, then the homogeneous coordinates ζ becomes

$$\zeta = \begin{bmatrix} [\omega]_\times & v \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix} \tag{A.51}$$

Then, it is easy to show that

$$\zeta^2 = \zeta^3 = \cdots = 0 \tag{A.52}$$

So that,

$$e^{\zeta\theta} = I + \zeta\theta \tag{A.53}$$

Therefore,

$$e^{\zeta\theta} = \begin{bmatrix} I & v\theta \\ 0 & I \end{bmatrix} \; if \; \omega = 0 \tag{A.54}$$

This equation show that $e^{\zeta\theta} \in SE(3)$

If $\omega \neq 0$ and $\|\omega\| = 1$, let us use a rigid transformation in the form of

$$h = \begin{bmatrix} I & \omega \times v \\ 0 & I \end{bmatrix} \tag{A.55}$$

Using the calculation of Lemma 1, we get

$$h^{-1}\zeta h = \begin{bmatrix} I & -\omega \times v \\ 0 & I \end{bmatrix} \begin{bmatrix} [\omega]_\times & v \\ 0 & 0 \end{bmatrix} \begin{bmatrix} I & \omega \times v \\ 0 & I \end{bmatrix} \tag{A.56}$$

$$h^{-1}\zeta h = \begin{bmatrix} [\omega]_\times & \omega\omega^T v \\ 0 & 0 \end{bmatrix} \tag{A.57}$$

Now, we can use the identity and cross product operator

$$e^{\hat{\zeta}\theta} = e^{h(\hat{\zeta}\theta)h^{-1}} = he^{(\hat{\zeta}\theta)}h^{-1} \text{ and } [\omega]_{\times}\,\omega = \omega \times \omega = 0 \tag{A.58}$$

$$e^{\zeta\theta} = \begin{bmatrix} e^{\omega\theta} & \left(I - e^{[\omega]_{\times}\theta}\right)(\omega \times v) + \omega\omega^{T} v\,\theta \\ 0 & 1 \end{bmatrix} \tag{A.59}$$

which is an element of SE(3).

Finally, The Lie algebra of SE(3) is given by

$$se(3) = \left\{ \begin{bmatrix} \omega & v \\ 0 & 0 \end{bmatrix} \middle| \omega \in so(3), v \in R^{3} \right\} \text{and } \omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{A.60}$$

The exponential map of $\mathbb{SE}(3)$, $\mathbb{se}(3) \rightarrow \mathbb{SE}(3)$), is given by

$$e^{\left(\begin{bmatrix} \omega & v \\ 0 & 0 \end{bmatrix}\right)} = \left(\begin{bmatrix} e^{\omega} & Av \\ 0 & 1 \end{bmatrix}\right) \tag{A.61}$$

where

$$A = I + \frac{1 - cos\|\omega\|}{\|\omega\|^{2}}\omega + \frac{\omega - sin\|\omega\|}{\|\omega\|^{3}}\omega^{2} \tag{A.62}$$

and $e^{\omega}$is given by the Rodriguez' formula,

$$\exp(\omega) = I + \frac{sin\|\omega\|}{\|\omega\|}w + \frac{1 - cos\|\omega\|}{\|\omega\|^{2}}\omega^{2} \tag{A.63}$$

Lie Group algebra is an minimal representation for 3D transformations and is highly suitable for numerical optimization because it uses 6 parameters so-called twist coordinates $\xi \in R^6$ for 6 DOF (3 rotation and 3 translation)

$$\xi = [\upsilon_1 \quad \upsilon_2 \quad \upsilon_3 \quad \omega_1 \quad \omega_2 \quad \omega_3]^T \in R^6 \tag{A.64}$$

where $v^T = [\upsilon_1 \quad \upsilon_2 \quad \upsilon_3]^T$ denotes the linear velocity and $\omega^T = [\omega_1 \quad \omega_2 \quad \omega_3]^T$ encodes the angular velocity. Using the cross product operator $[.]_X$; the twist can be represented as

$$\hat{\xi} = \begin{bmatrix} [\omega]_\times & \upsilon \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & \upsilon_1 \\ \omega_3 & 0 & -\omega_1 & \upsilon_2 \\ -\omega_2 & \omega_1 & 0 & \upsilon_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.65}$$

The logarithm and the exponential map transform elements from a Lie group to its associated Lie algebra using a transformation matrix $T \in \mathrm{SE}(3)$, and its twist $\xi \in \mathrm{se}(3)$ as

$$T = \exp(\xi) : \ se(3) \rightarrow SE(3) \tag{A.66}$$

$$\xi = \log(\mathrm{T}) : \ SE(3) \rightarrow se(3) \tag{A.67}$$

Small motions with $\xi \approx 0$, exponential map can be approximated as $\mathrm{T} \approx \mathrm{I} + \hat{\xi}$. Closed form solution can be calculated using Rodriguez' formula. The logarithm map for obtaining the twist $\xi = [v^T, \omega^T]^T$ has the following closed form representation:

$$\theta = \|\omega\| = \mathrm{acos}\left(\frac{tr(R) - 1}{2}\right) \tag{A.68}$$

$$\log(R) = \frac{\theta}{2sin\theta}(R - R^T) \tag{A.69}$$

$$\omega = \log(R)^{\vee} \tag{A.70}$$

$$v = \left(I - \frac{1}{2}[\omega]_{\times} + \frac{1 - \frac{\theta\cos(\frac{\theta}{2})}{2\sin(\frac{\theta}{2})}}{\theta^2}[\omega]_{\times}{}^{2} \right) t \tag{A.71}$$

where the vee operator $(.)^{\vee}$, is the inverse of the cross product, which extracts the 3D vector from a skew symmetric matrix.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Hastürk, Özgür
Nationality: Turkish (TC)
Date and Place of Birth: 18 March 1986, Ankara
Marital Status: Married
Phone: +90 530 067 77 47
email: ozgurhasturk@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
MS	METU Electrical and Electronics Enginering	2011
BS	METU Mechanical Engineering	2008

WORK EXPERIENCE

Year	Place	Enrollment
2008-Present	ROKETSAN	System Algorithms Design Engineer
2007 July	ROKETSAN	Intern Eng. Student
2007 August	ASELSAN	Intern Eng. Student

FOREIGN LANGUAGES

Advanced English, Intermediate German

PUBLICATIONS

1. Hastürk Ö., "Applications of slider chain inversion in control actuation systems," 2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), 2016, pp. 1579-1584, doi: 10.1109/AIM.2016.7576995.

2. A. Aydogan and O. Hasturk, "Adaptive LQR stabilization control of reaction wheel for satellite systems," 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), 2016, pp. 1-6, doi: 10.1109/ICARCV.2016.7838849.

3. Aydogan A., Hasturk O. and Rogers E. (2018) Dynamic modeling and computed torque control ol flexure jointed TVC systems. Proceedings of the ASME Dynamic Systems and Control Conference, vol. 3, Atlanta, USA.

4. A. Aydogan, O. Hasturk, E. Rogers, Robust H∞ Computed Torque Control of Flexible Joint TVC Systems, IFAC-PapersOnLine, Volume 52, Issue 12 2019, Pages 454-459, https://doi.org/10.1016/j.ifacol.2019.11.285.

5. Hastürk Ö, Erkmen, A.M., DUDMap: 3D RGB-D mapping for dense, unstructured, and dynamic environment. International Journal of Advanced Robotic Systems. May 2021. doi:10.1177/17298814211016178

6. Hastürk Ö., "A novel electromechanical actuator for missile jet vane thrust control," 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), 2015, pp. 1298-1302, doi: 10.1109/AIM.2015.7222718.

7. Hastürk, Ö.; Erkmen, A.M.; Erkmen, I. ; Proxy-Based Sliding Mode Stabilization of a Two-Axis Gimbaled Platform. In Proceedings of the World Congress on Engineering and Computer Science, San Francisco, CA, USA, 19–21 October 2011.