Visual Simultaneous Localization and Mapping: From Geometry to Deep Learning

By

Ruihao Li

A thesis submitted for the degree of

Doctor of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

January 2018

Abstract

Visual Simultaneous Localization and Mapping (SLAM) is essential to achieve persistent autonomy for mobile robots in unknown environments, and is a key technique for enormous vision based applications, such as virtual and augmented reality. Researchers from the robotics and computer vision communities have endeavored and managed to design some efficient and versatile visual SLAM systems in the past several decades. However, how to achieve more accurate and robust localization and construct semantic map is still a challenging problem.

The work in this thesis describes several novel algorithms to solve the above problem. Several novel methods that merge data driven approaches, such as deep learning, with visual SLAM techniques are proposed in order to gain better performance. Firstly, a novel model-based SLAM method based on points and plane-patches is proposed. Evaluational experiments on multiple benchmark dataset are performed to evaluate the proposed method. Secondly, a novel indoor relocalization system based on supervised deep learning is presented. Deep neural network is successfully applied for indoor pose regression. Thirdly, in order to solve the lack of labeled datasets. A novel unsupervised deep learning based visual SLAM system (called DeepSLAM) is proposed. The novel DeepSLAM includes Mapping-Net, Tracking-Net, Loop-Net, and a graph optimization unit. Experimental evaluations prove that the proposed DeepSLAM outperforms the state-of-the-art monocular SLAMs in terms of pose estimation accuracy, and is more robust than other SLAM systems in some challenging scenes. Finally, a novel spatio-temporal deep neural network for semantic segmentation is proposed with a SLAM system to achieve semantic 3D mapping. The proposed method is verified to be effective for 3D semantic mapping.

Acknowledgment

First of all, I would like to give my honest thanks to my supervisor Prof. Dongbing Gu. He is a man full of wisdom. No matter what happens in life or work, he can always keep calm and be nice to everyone else. I like to share my daily life and study with him and ask for advice, he always offers his honest suggestions as a friend. What is more, he is a man who can always keep himself in academic frontiers and lead me into the world of robotic research. At the beginning of my Ph.D. study, I could not find the research topic, struggling every day in the office and live in darkness. I still remember the day he came to my office and gave me the paper about "Relocalization with deep learning", it was like the sunshine from dark cloud. From then on, I find my interests and dedicate myself to the research about SLAM with Deep Learning.

I would like to thank Prof. Huosheng Hu, he gives me lots of advice on my research and study. Many thanks to Prof. Zhiqiang Long, he supports me to study abroad and gives me lots of help while I am in the UK. I would like to thank Dr. Sen Wang, he spends lots of time on my final project and offers me a server to use. I really appreciate his encouragement, support and accompany when I am in trouble with the project. Many thanks to my board meeting member, Prof. Steve Sangwine and Prof. Hani Hagras. They organize board meetings for me and offer me advice on my research. Their suggestions on my Ph.D. study are always helpful for me to make progress.

I would like to gives thanks to my colleagues Mr. Robin Dowlin and Mr. Ian Dukes. They give me many supports in my experiments, especially in data collection. I appreciate the help and the accompany of my colleagues during my Ph.D. studies in the UK. They are Mr. Qiang Liu, Dr. Jianjun Gui, Dr. Chao Zhang, Mr. Yaser Alothman, Dr. Wesam Jasim, Mr. Aakash Soni, Mr. Minhuan Guo, Prof. Yan Zhuang, Prof. Yisha Liu, Prof. Yi An, Prof. Qi Li, Prof. Yuanping Xie, Prof. Guochen Wang, Prof. Jianyong Sun, Mr. Dingtian Yan, Mr. Zuyuan Zhu, etc.

I would like to thank my parents, they raise me up with their unselfish and deep love. Without their support and understanding, I can never study abroad to pursue my Ph.D. degree.

Last, but most importantly, I would like to give my thanks to my dear wife – Dr. Huichan Zhao. Many thanks for her remarkable tolerance of my absence for many years. She finished

her Ph.D. in the USA and is a post-doc now. Her research is also in robotics which means we have many common topics in our daily life. Even her research is mainly focused on soft robotics which is different from mine, she can still offer me much useful insight into my research and help me polish my academic papers which certainly occupies her lots of time to watch TV shows. She always encourages me and gives me confidence when I struggle with my research. In a word, with her accompany and understanding in my life, I can always keep passionate and energetic to the research and life.

Contents

Abstract iii

Acknowledgement v

List of Figures xiii

List of Tables xvii

Notation and Nomenclature xix

List of Acronyms xxi

1 Introduction 1

1.1 Motivation 1

1.2 Objectives and Challenges 3

1.3 Methodology and Main Contributions 5

1.4 Thesis Outline 8

2 Background Review 11

2.1 Model-based SLAM Methods 11

2.1.1 Feature-based Visual SLAM Methods 12

2.1.2 Direct Visual SLAM Methods 14

2.1.3 Summary 15

2.2 Deep Neural Networks for Visual SLAM 15

2.2.1 Convolutional Neural Network 15

2.2.2 Deep Recurrent Neural Network 17

2.2.3 Auto-Encoder 18

2.2.4 Dataset 19
2.3 Depth Estimation with Deep Learning 19
2.3.1 Supervised Methods 20
2.3.2 Unsupervised Methods 21
2.3.3 Summary 23
2.4 Pose Estimation with Deep Learning 23
2.4.1 Relocalization with Deep Learning 23
2.4.2 Ego-Motion Estimation with Deep Learning 25
2.4.3 Sensor Fusion with Deep Learning 27
2.4.4 Summary 27
2.5 Semantic Mapping with Deep Learning 27
2.5.1 Semantic Segmentation 28
2.5.2 Semantic Mapping 30
2.5.3 Summary 31

3 A Novel SLAM Algorithm based on Points and Plane-Patches 33
3.1 Introduction 33
3.2 Related Work 34
3.3 Problem Description and System Overview 36
3.4 Plane Patch Layer Construction 36
3.4.1 Plane Patches Extraction 36
3.4.2 Plane Patches Matching 38
3.5 Camera Pose Estimation with Points and Plane-patches 39
3.5.1 Feature Points Ranking 39
3.5.2 RANSAC Registration using Points and Plane Patches 41
3.6 Experimental Evaluation 42
3.6.1 Tests on Benchmark Datasets and in Our Lab 42
3.6.2 Computational Cost Analysis 46
3.6.3 Test on a Large-scale Environment 47
3.7 Conclusions 48

4 Indoor Relocalization in Challenging Environments with Dual-stream Convolutional Neural Networks 49

4.1 Introduction . . . 49

4.2 Related Work . . . 51

4.2.1 Visual Geometry Relocalization . . . 52

4.2.2 CNNs based Relocalization . . . 53

4.2.3 Encoding Method of Depth Images for CNNs . . . 54

4.3 Preliminaries . . . 55

4.3.1 Network Architecture . . . 55

4.3.2 Dual-stream CNN for Pose Regression . . . 57

4.3.3 Encoding Methods for Depth Images . . . 59

4.4 Training Mechanism for Dual-stream CNN . . . 61

4.4.1 Training Separate Streams . . . 61

4.4.2 Fine-tuning Full Connected Layer . . . 62

4.4.3 Fine tuning overall Dual-stream CNN . . . 63

4.5 Experimental Evaluation . . . 63

4.5.1 Range Images Encoding Methods . . . 64

4.5.2 Network Architecture for RGB-D Images . . . 66

4.5.3 Quantitative Analysis . . . 69

4.5.4 Qualitative Analysis based on Challenging Datasets including Fast Movement, Night-time, Dynamic Scenes . . . 71

4.6 Conclusions . . . 73

5 DeepSLAM: A Robust Monocular SLAM System with Unsupervised Deep Learning 75

5.1 Introduction . . . 75

5.2 Related Work . . . 78

5.2.1 Supervised Deep Learning for Pose Estimation . . . 78

5.2.2 Unsupervised Deep Learning for Depth and Ego-motion Estimation 79

5.3 System Overview of DeepSLAM . . . 80

5.3.1 Tracking-Net with RCNN Architecture . . . 81

5.3.2 Mapping-Net with Encoder-Decoder Architecture . . . 81

5.3.3 Training of DeepSLAM 83
5.4 Unsupervised Training based on Spatial and Temporal Geometric Consistencies 83
5.4.1 Spatial Image Loss of a Pair of Stereo Images 83
5.4.2 Temporal Image Loss of a Sequence of Monocular Imagery 85
5.4.3 Uncertainty Estimation and Outliers Rejection 87
5.5 Pose Graph Construction and Optimization 88
5.5.1 RCNN based Local Pose Graph 88
5.5.2 CNN based Global Loop Detection 89
5.6 Experimental Evaluation 90
5.6.1 Pose Accuracy Performance on KITTI 91
5.6.2 Robustness Performance on Challenging Scenes 95
5.6.3 Testing with a Low-cost Camera 97
5.6.4 Outliers Rejection 97
5.6.5 Depth Estimation and 3D Reconstruction 98
5.7 Conclusions 102

6 Semantic Scene Mapping with Spatio-Temporal Deep Neural Network for Robotic Applications 103
6.1 Introduction 103
6.2 Related Work 106
6.2.1 Spatial Image Segmentation with CNNs 107
6.2.2 Multi-stream CNNs 107
6.2.3 Semantic mapping with SLAM 108
6.3 Approach 108
6.3.1 Spatial Segmentation Network Architecture 109
6.3.2 Proposed Spatio-temporal CNN for Semantic Segmentation 111
6.3.3 Image Segmentation Prediction with Homography Matrix 112
6.3.4 3D Semantic Scene Mapping with a SLAM Algorithm 113
6.4 Experimental Evaluation 114
6.4.1 Indoor Dataset for Scene Segmentation 116
6.4.2 Training Details 116

6.4.3 Fusion Method 119
6.4.4 Quantitative Analysis 119
6.4.5 Qualitative Analysis 120
6.4.6 3D Semantic Mapping 121
6.5 Conclusions 121

7 Conclusion **123**
7.1 Research Overview 123
7.2 Contributions 125
7.3 Publication List 127
7.4 Future Work 128

Bibliography **131**

List of Figures

1.1 The structure of this thesis. 9

2.1 Some model-based SLAM methods. 13
2.2 Three popular deep learning architectures. 16
2.3 Depth estimation with unsupervised deep learning. 20
2.4 Pose estimation with deep Learning. 24
2.5 Semantic mapping with deep learning. 29

3.1 System overview of the proposed SLAM algorithm. 35
3.2 Plane patches matching approach. 38
3.3 Different types of feature points. 40
3.4 Comparison between estimated trajectory using our method and ground truth trajectory. 44
3.5 3D map reconstructed from benchmark datasets using our method. 45
3.6 Trajectory and 3D map with the proposed SLAM algorithm in our building. 47

4.1 Architecture overview of the proposed dual-stream CNN for indoor relocalization. 54
4.2 Selected network layers for different network inputs. 56
4.3 Different encoding methods of depth images. 59
4.4 Predicted trajectory using our method in our network building. 66
4.5 Cumulative histograms of relocalization error with different network architectures for RGB-D images as input. 68
4.6 Challenging scenes for indoor relocalization with the dual-stream CNN. . . 69

4.7 Cumulative histogram of relocalization error for challenging indoor test datasets with the dual-stream CNN. 72

5.1 Testing framework of our proposed DeepSLAM. 77
5.2 Training scheme of the proposed DeepSLAM system. 79
5.3 Spatial and temporal constraints used to formulate the spatial and temporal losses. 82
5.4 Pose Graph with local and global connections. 89
5.5 Trajectories on KITTI sequences 03-07, 10 using our DeepSLAM system (best viewed in color). KITTI sequences 00-02, 08, 09, 11-21 are used for our network training. Trajectories with SfMLearner [1], VISO2-M [2] and VISO2-S [2] are also plotted for comparison. 92
5.6 Trajectories on KITTI dataset using our DeepSLAM system. There is no ground-truth for these trajectories. We plotted trajectories with stereo ORB-SLAM (ORB-SLAM-S) for reference. ORB-SLAM-S uses images with 1242×376. KITTI sequences 00-08 are used for network training. 94
5.7 Robustness performance of our DeepSLAM on some challenging environments in RobotCar dataset. 96
5.8 Testing on our self-collected dataset using a low-cost ZED camera. 97
5.9 Geometric mask and photometric mask for outliers rejection. 98
5.10 The estimated depth maps with our Mapping-Net. 99
5.11 Reconstructed 3D map with our DeepSLAM system. 101

6.1 3D semantic maps. 104
6.2 System overview of the proposed semantic mapping system. 106
6.3 Spatio-temporal neural network architecture. 108
6.4 Visual comparison on our manually labelled dataset (Essex Indoor). 110
6.5 Image segmentation prediction with homography matrix. 112
6.6 Train the network with the "problem" dataset. 118
6.7 Segmentation results of the networks using "problem" dataset for training. . 118
6.8 3D semantic map of the second floor in the network building of our university. 119

6.9 Visual comparison on Cityscapes dataset. 122

List of Tables

2.1 Some important model-based SLAM methods. 12
2.2 Some important depth estimation methods with deep learning. 20
2.3 Some novel pose estimation methods with deep learning. 23
2.4 The milestone semantic perception methods with deep learning. 28

3.1 Comparison of ATE RMSE between different SLAM methods. 43
3.2 Average processing time for different procedures of our method. 46
3.3 Feature and RANSAC feature inlier composition from 100 pairs of frames using ICL-NUIM office dataset. 46

4.1 Relocalization performance comparison with different depth image encoding methods using PoseNet. 63
4.2 Results of different network architectures with RGB-D images as inputs. . . 65
4.3 Quantitative analysis. 67
4.4 Indoor relocalization results in challenging scenes with different methods. . 70

5.1 Tracking results on KITTI dataset with our proposed DeepSLAM system. Except for our DeepSLAM, two data-driven learning methods (ESP-VO [3] and SfMlearner [1]) and three model-based methods (ORB-SLAM, VISO2-M and VISO2-S) are added into the table for comparison. Note that monocular VISO2-M and ORB-SLAM did not work with image size 416×128, so their results with image size 1242×376 are shown here. Our DeepSLAM and SfMLearner use images with 416×128. The best results among data-driven learning methods are made in bold. 93
5.2 Robustness performance on challenging scenes of RobotCar dataset. 95

5.3 Depth estimation results on KITTI. 100

6.1 Segmentation performance comparison with different fusion methods. . . . 114
6.2 Per-class segmentation IoU (%) on Cityscapes dataset. 115
6.3 Per-class segmentation IoU (%) on the Essex Indoor test dataset when trained with the "problem" dataset. 117

Notation and Nomenclature

$\mathbf{p}_i$	point in 3D space
$\overline{\mathbf{p}}$	mean value of points in 3D space
Σ	covariance matrix
λ_i	eigenvalue
th_p	curvature threshold to judge plane
π	vector to represent a plane
$\mathbf{n}$	plane normal
$\mathbf{T}$	transformation matrix
$\rho(\pi)$	minimum representation of a plane
$\mathbf{Q}_c$	uncertainty of measured plane
th_{co}	coplanarity threshold for coplanarity criterion
th_{ov}	overlap threshold for overlap criterion
$\mathbf{c}_l$	center of the plane patch re-projected into the 2D image
a_l	size of the plane patch re-projected into the 2D image from local frame
$\mathbf{t}$	translation
R	rotation
$\mathbf{X}_i$	homogeneous position representation of feature point
W_i	weight of point
$E_{\mathbf{p}}$	the Euclidean loss
x_n	labelled ground-truth vector
$E_{\mathbf{p}}$	positional Euclidean loss
$E_{\mathbf{q}}$	rotational Euclidean loss
H_i	horizontal distance of corresponding pixels between stereo images
B	baseline of stereo camera

f	camera focal length
$\widehat{D}_i$	predicted depth map
I'_l	synthesized left image
I'_r	synthesized right image
$SSIM(\cdot)$	Structural SIMilarity (SSIM) metric
$f_s(\cdot)$	SSIM metric to evaluate the quality of a synthesized image
λ_s	weight for SSIM part
$\|\cdot\|_1$	L1 norm
w	image width
Q	disparity map
$L^p_{l,r}$	left-right photometric consistency loss
$L^d_{l,r}$	disparity consistency loss
L^o	left-right pose consistency loss
λ_p	positional weight
λ_r	rotational weight
$\widehat{\varphi}$	estimated rotation represented by Euclidean angle
K	camera intrinsics matrix
E^k_p	photometric error between temporal images
$L^p_{k,k+1}$	photometric losses of image pair I_k, I_{k+1} from image sequences
$L^g_{k,k+1}$	geometric losses in the monocular image sequence
M_g	mask of the corresponding geometric error map
$\sigma_{k,k+1}$	uncertainty of pose estimation and depth estimation
$S(\cdot)$	Sigmoid function
λ_e	normalizing factor between the geometric and photometric errors
q_{th}	percentile to construct masks
d_{cos}	cosine distance of two feature vectors from an image pair
r	stride used by the dilated convolution
$x[i]$	1-D input signal with length K
$y[i]$	output of the dilated convolution
$[u,v]$	2D coordinate of a pixel in the keyframe

List of Acronyms

SLAM	Simultaneous Localisation and Mapping
VO	Visual Odometry
PTAM	Parallel Tracking and Mapping
DTAM	Dense Tracking and Mapping
DoF	Degree-of-Freedom
GPS	Global Positioning System
GT	Ground Truth
SfM	Structure-from-Motion
IMU	Inertial Measurement Unit
VINS	Vision-aided Inertial Navigation System
INS	Inertial Navigation System
DL	Deep Learning
CNN	Convolutional Neural Network
CNNs	Convolutional Neural Networks
DNN	Deep Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
RCNN	Recurrent Convolutional Neural Network
FCN	Fully Convolutional Network
AR	Augment Reality
VR	Virtual Reality
IoU	Intersection over Union
PCA	Principal Component Analysis
RANSAC	Random Sample Consensus

CPU	Central Processing Unit
GPU	Graphic Processing Unit
RMSE	Root-Mean-Square Error
ATE	Absolute Trajectory Error
SURF	Speed Up Robust Features
SIFT	Scale-Invariant Feature Transform
FAST	Features from Accelerated Segment Test
BRIEF	Binary Robust Independent Elementary Features
ORB	Oriented Fast and Rotated BRIEF
BoWs	Bag of Words
MND	Minimized Normal + Depth
ICP	Iterative Closest Point
HHA	Horizontal disparity, Height above ground and Angle with gravity
SGD	Stochastic Gradient Descent
ILSVRC14	Large-Scale Visual Recognition Challenge 2014
ReLU	Rectified Linear Unit
LRN	Local Response Normalization
DSO	Direct Sparse Odometry
CRFs	Conditional Random Fields
ASPP	Atrous Spatial Pyramid Pooling
PSPNet	Pyramid Scene Parsing Network
CMoDE	Convoluted Mixture of Deep Experts
API	Application Program Interface
UAV	Unmanned Autonomous Vehicle
ROS	Robot Operating System

Chapter 1

Introduction

This research is focused on the robust localization and semantic mapping problems by relying only on low-cost cameras. Both model-based SLAM methods and data-driven based SLAM methods are studied, and some novel approaches are proposed. This chapter presents the research motivation, objectives, challenges, research methodologies, thesis contributions and thesis outline.

1.1 Motivation

Visual Simultaneous Localization and Mapping (SLAM) is essential to achieve persistent autonomy for vision-based mobile robots, especially in unknown and GPS-denied environments. It is also a key enabler for enormous vision based applications, such as Virtual Reality (VR) and Augmented Reality (AR). In the past decades, significant progress on visual SLAM techniques has been made and some efficient and versatile visual SLAM systems have been developed.

There are two key objectives for a visual SLAM system to achieve. One is localization, and the other one is mapping. For a camera in an unknown environment, localization is to locate the camera itself and obtain the 6-DoF pose information in the world coordinate. Mapping is to construct the map of the environment with the 3D feature points. The pose change between two image frames is called ego-motion, and with accumulated ego-motions, camera localization can be calculated. Localization can help maintain a consistent and accurate map, and mapping can help to achieve more accurate localization. For a typical visual SLAM system, it is mainly consisted of two parts: front-end and back-end. Front-end mainly

includes feature extraction, feature matching, and ego-motion estimation. Back-end mainly includes loop closure detection, graph construction and optimization. Loop closure detection is also known as place recognition, and it is to detect whether the camera visits a previous visited scene. Relocalization is to locate the camera itself in a pre-visited environment. The back-end of SLAM system can help the reduction of accumulated localization errors and the maintenance of a consistent 3D map. What is more, if the system can perceive semantic information of the environment, such as people, desk, road, sky etc., mapping can be made into semantic mapping.

Most of existing visual SLAM methods explicitly model camera projections, motions and environments based on visual geometry. Therefore, they are referred to as model-based SLAM. They can be divided into feature-based methods [4–6] and direct methods [7–9] according to the means image information is used. Specifically, feature-based visual SLAM methods extract sparse features, such as points and lines, from the images for feature matching and ego-motion estimation, while direct methods directly use dense (or semi-dense) image pixels for motion estimation under the assumption of photometric consistency. Loop closure detection and back-end optimization can be incorporated with both methods to form a full visual SLAM system.

The state-of-the-art model-based visual SLAM algorithms have made a great success in the past decade. Superior performance on localization and mapping, for example, has been demonstrated by both feature-based [6] and direct [9] methods. However, they still face many challenging issues, in particular when being deployed in large-scale environments, or under extreme lighting conditions. Nowadays system robustness [10] and high level (semantic) perception [10, 11] are the demanding tasks for visual SLAM systems. It becomes increasingly challenging to solve these problems by solely relying on model-based methods. One of the reasons is that the high-dimensional images carry significant “redundant” information and the real world has the complex appearance, which is difficult to be manually modeled in a precise manner.

Deep learning can automatically learn effective feature representations from data and has successfully demonstrated its capability for some challenging visual perception tasks. Unavoidably the evolution of visual SLAM from model-based methods to deep learning methods occurs. Recently the attempts to merge visual SLAM with deep learning include the

depth map of the environment from a monocular image by using deep learning [12], the visual odometry by using deep learning for a comparable performance [13]; the semantic map estimation by using deep learning [11]. These recent advances promise a huge potential for visual SLAM systems to bring in adaptive and learning capability to address the challenging issues.

In conclusion, the motivation for this research is to develop novel visual SLAM algorithms with both model-based methods and deep learning methods, that can achieve more accurate and robust performance, and perceive semantic information at the same time.

1.2 Objectives and Challenges

Based on the motivation stated in the previous section, the objectives of this thesis are to leverage the advantages of both model-based SLAM methods and deep learning methods to develop novel visual localization and semantic mapping algorithms, and provide comprehensive insights on the performance by the means of theoretical and experimental analyses.

The work will focus on the following specific objectives:

- **High accuracy**: For SLAM systems, the accuracy performance of localization is one of the most important criteria. This is also the basic requirement for most robotic operations. Therefore, one objective of this thesis is to achieve high localization accuracy for different robotic applications.

- **High robustness**: For autonomous robots, when faced with featureless or challenging scenes, most visual SLAM algorithms will lose tracking and mapping. It is intolerable for real-time robotic applications. So how to enhance the robustness of visual SLAM systems is also one of our objectives.

- **Semantic perception**: For the robots-environment interface, perceiving semantic information of the environment is necessary, and it can help robots to carry out some high level tasks. What is more, object-level SLAM or semantic SLAM can also enhance the robustness and accuracy of localization. Therefore, one of our objectives is to empower robots the ability of semantic perception with commercial cameras.

- **Low cost and small size camera**: For many robotic applications in our daily life, it is always preferred to achieve localization and mapping with a low cost and small

size sensor. The cost here includes price and power. Commercial cameras are ideal in this aspect. So one of the objectives of this thesis is to keep or even improve the performance of localization and mapping, while relying only on low-cost and small-size visual sensors.

In order to attain the aforementioned objectives, several specific challenges have been identified:

1. **How to estimate the 6-DoF pose and construct the 3D map robustly.**

   With visual SLAM technology for robotic applications, we not only need robot location and scene mapping, but also need robust camera poses with absolute scale and 3D map with dense information. What is more, the application of visual SLAM usually happens in an unknown and featureless environment. Therefore, it is a big challenge on how to use the advanced technology to estimate 6-DoF pose and construct the 3D map robustly.

   Manually designing more robust hand-crafted features is one solution. The other solution is to make use of the power of deep learning to learn features itself. Deep learning methods can learn very efficient image features in an end-to-end way. A neural network accepts input from one end, and produces output at the other end. The learning that optimizes the network weights by considering the inputs and outputs directly is called end-to-end learning., and demonstrate very robust and astonishing performance in object recognition. However, how to transfer the usage of deep learning methods from object recognition to pose regression is still an undefined problem, and pose regression is that a neural network can output 6-DoF poses directly.

2. **How to tackle the problem of labelled dataset shortage with unsupervised deep learning.**

   Supervised deep learning methods need a large amount of labeled data for network training. For object recognition, the community has ImageNet dataset as the benchmark for training and testing. For pose estimation, there is still no ImageNet-scale labeled dataset for the community at the moment. Therefore, how to develop unsupervised deep learning approaches and provide solutions for the lack of labeled dataset is a big challenge.

3. **How to construct semantic map in SLAM with deep learning.**

   Perceiving semantic information of the environment is necessary for interacting robots

in the environment. Deep learning methods have also achieved great success in semantic image segmentation. How to introduce deep learning methods into visual SLAM to achieve semantic SLAM is a very challenge and meaningful topic.

In order to develop a successful visual SLAM system, the challenges identified have to be addressed. This thesis aims to accomplish these.

1.3 Methodology and Main Contributions

Addressing the challenges summarised above, this thesis proposes several novel model-based SLAM techniques and deep learning methods, and evaluates the proposed methods by performing large amounts of evaluational experiments. At the beginning, we try to design more robust features manually to improve the robust of visual SLAM system (challenge 1). Therefore, a novel model-based SLAM method based on points and plane-patches is proposed. However, when facing with some challenging scenes, such as featureless, dynamic, nighttime environments, the proposed SLAM method can hardly work. Deep learning can learn the efficient features automatically and we decide to introduce deep learning to solve pose regression problems. A novel indoor relocalization system with supervised deep learning is presented. However, the method requires the ground truth of camera poses for conducting the supervised training, and currently obtaining ground truth datasets in practice is typically difficult and expensive, and the amount of existing labeled datasets for supervised training is still limited. Developing unsupervised methods maybe a solution to the limitation of labelled datasets, therefore, a novel SLAM system with unsupervised deep learning is proposed to tackle challenge 2. Consequently we can train them with easily collected unlabelled datasets and apply them to localization and mapping scenarios. Finally, in order to address challenge 3, a novel spatio-temporal CNN is combined with SLAM to construct semantic 3D map.

The key contributions of this thesis are the novel approaches that combine deep learning methods and model-based SLAM. The main contribution of this thesis are summarised as below:

(1) SLAM based on points and plane-patches (Chapter 3)

We proposed a novel RGB-D SLAM algorithm. The novelty of the proposed algorithm lies in the use of both feature points and plane patches for pose estimation. A plane patch

is defined as a small-sized patch constructed by using a feature point with small curvature. The feature points with small curvature are called plane points. The remaining feature points are classified as either smooth points which represent smooth surfaces or structural points which represent edges or corners of some objects. Two criteria (coplanarity and overlap) are used to match the plane patches from two frames. The proposed algorithm is able to weight various types of feature points in order to improve the robustness of SLAM algorithms in favouring plane patches but not the isolated feature points, and maintain the real-time performance of SLAM algorithms in favouring small plane patches, but not the large size of planes. We evaluate the proposed algorithm on multiple benchmark datasets. A large scale experiment is also presented to show the robustness of our algorithm. This part is based on the following publication:

- Ruihao Li, Qiang Liu, Jianjun Gui, Huosheng Hu, Dongbing Gu. A Novel RGB-D SLAM Algorithm Based on Points and Plane-Patches. In Automation Science and Engineering (CASE), 2016 IEEE International Conference on, pp. 1348-1353. IEEE, 2016.

(2) Relocalization with supervised deep learning (Chapter 4)

We proposed an indoor relocalization system using a dual-stream Convolutional Neural Network (CNN) with both color images and depth images as the network inputs. Aiming at the pose regression problem, a deep neural network architecture for RGB-D images is introduced, a training method by stages for the dual-stream CNN is presented, different depth image encoding methods are discussed and a novel encoding method is proposed. By introducing the range information into the network through a dual-stream architecture, we not only improved the relocalization accuracy by about 20% compared with the state-of-the-art deep learning method for pose regression, but also greatly enhance the system robustness in challenging scenes such as large scale, dynamic, fast movement and night-time environments. To the best of our knowledge, this is the first work to solve the indoor relocalization problems based on deep CNNs with RGB-D camera. The method is first evaluated on the Microsoft 7-Scenes dataset to show its advantage in accuracy compared with other CNNs. Large scale indoor relocalization is further presented using our method. The experimental results show that 0.3m in position and 4° in

orientation accuracy could be obtained. Finally, this method is evaluated on challenging indoor datasets collected from motion capture system. The results show that the relocalization performance is hardly affected by dynamic objects, motion blur or night-time environments. This part is based on the following publications:

- Ruihao Li, Qiang Liu, Jianjun Gui, Huosheng Hu, Dongbing Gu. Indoor Relocalization in Challenging Environments with Dual-stream Convolutional Neural Networks. IEEE Transactions on Automation Science and Engineering (T-ASE), 2017.
- Ruihao Li, Qiang Liu, Jianjun Gui, Huosheng Hu, Dongbing Gu. Night-time Indoor Relocalization Using Depth Image with Convolutional Neural Network. In Automation and Computing (ICAC), 2016 22nd International Conference on, pp. 261-266. IEEE, 2016.

(3) A novel SLAM system with unsupervised deep learning (Chapter 5)

We proposed DeepSLAM, a novel unsupervised deep learning based visual SLAM system. The DeepSLAM training is fully unsupervised, but requires stereo datasets for recovering the depth scale. The DeepSLAM testing takes a monocular image sequence as the input. Therefore, it is a monocular SLAM paradigm. DeepSLAM consists of several essential components, including Mapping-Net, Tracking-Net, Loop-Net, and a graph optimization unit. Specifically, the Mapping-Net is an encoder and decoder architecture for describing the structure of environment while the Tracking-Net is a Recurrent Convolutional Neural Network architecture for capturing the camera motion. The Loop-Net is a pre-trained binary classifier for detecting loop closures. DeepSLAM can simultaneously generate pose estimate, depth map and outliers-rejection mask. We evaluate its performance on various datasets, and find that DeepSLAM outperforms the state-of-the-art monocular SLAMs in terms of pose estimation accuracy, and is more robust than other SLAM systems in some challenging scenes. This part is based on the following publications:

- Ruihao Li, Sen Wang, Dongbing Gu. DeepSLAM: A Robust Monocular SLAM System with Unsupervised Deep Learning. IEEE Transactions on Robotics (TRO), 2018. (Under Review)

- Ruihao Li, Sen Wang, Zhiqiang Long, Dongbing Gu. UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning. 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018. (Accepted)

(4) Semantic scene mapping with deep learning (Chapter 6)

We proposed a semantic pixel-wise mapping system for potential robotic applications. The system includes a novel spatio-temporal deep neural network for semantic segmentation and a Simultaneous Localisation and Mapping (SLAM) algorithm for 3D point cloud map. Their combination yields a 3D semantic pixel-wise map. The proposed network consists of Convolutional Neural Networks (CNNs) with two streams: spatial stream with images as the input and temporal stream with image differences as the input. Due to the use of both spatial and temporal information, it is called spatio-temporal deep neural network, which shows a better performance in both accuracy and robustness in semantic segmentation. Further only keyframes are selected for semantic segmentation in order to reduce the computational burden for video streams and improve the real-time performance. Based on the result of semantic segmentation, a 3D semantic map is built up by using the 3D point cloud map from a SLAM algorithm. The proposed spatio-temporal neural network is evaluated on both Cityscapes benchmark (a public dataset) and Essex indoor benchmark (a dataset we labelled ourselves manually). Compared with the state-of-the-art spatial only neural networks, the proposed network achieves better performances in both pixel-wise accuracy and Intersection over Union (IoU) for scene segmentation. This part is based on the following publication:

- Ruihao Li, Dongbing Gu, Qiang Liu, Zhiqiang Long, Huosheng Hu. Semantic Scene Mapping with Spatial-temporal Deep Neural Network for Robotic Applications. Cognitive Computation, 2017.

1.4 Thesis Outline

The overview of the structure of this thesis is shown in Figure 1.1. This chapter and chapter 2 form the base of this work. The main work is presented in Chapters 3, 4, 5 and 6. Chapter 7 concludes the thesis.

The details of the chapters are outlined as below:

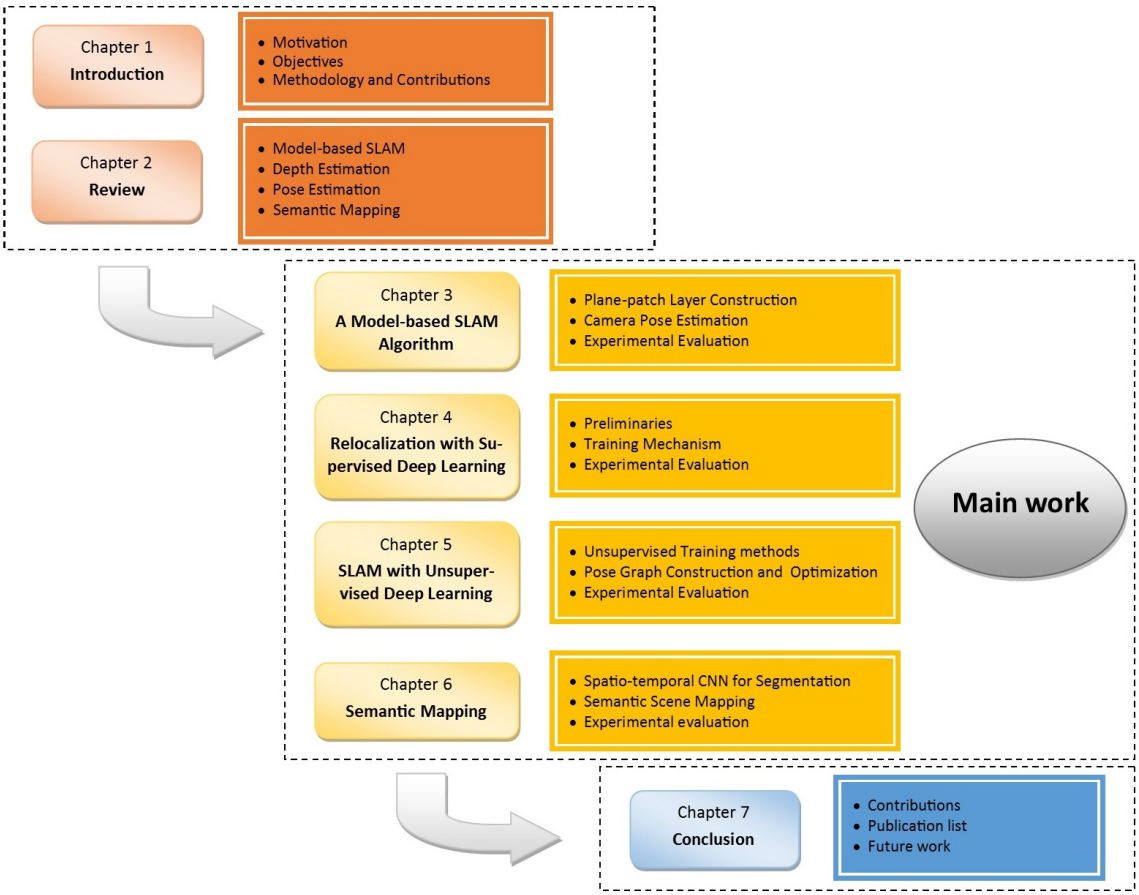


Figure 1.1: The structure of this thesis.

- Chapter 2 reviews previous work related to this research. Model-based SLAM methods are introduced firstly. The two main methods—-feature-based methods and direct based methods are discussed separately. Secondly, some different kinds of Deep Neural Network architectures for visual SLAM are introduced. Thirdly, literatures about depth estimation with deep learning are reviewed. Supervised learning methods and unsupervised learning methods are introduced. Fourthly, pose estimation with deep learning methods are reviewed in perspective of relocalization, ego-motion estimation and sensor fusion. Finally, semantic mapping methods which combine deep learning and SLAM are discussed to provide a broad overview.

- Chapter 3 presents a novel RGB-D SLAM algorithm based on points and plane patches. Related works are introduced first. Then plane patch layer construction including plane patches extraction and plane patches matching is presented. Two criteria (coplanarity and overlap) used to match the plane patches are proposed. Moreover, camera pose estimation

methods with points and plane-patches are given, the pose estimation methods use both feature points ranking and RANSAC registration. The evaluation experiments are then given based on multiple public benchmark datasets and self-collected datasets. Finally, the conclusion and future works are presented.

- Chapter 4 introduces an indoor relocalization system using supervised deep learning techniques. Related works are given first. Then the designed dual-stream CNN architecture for indoor pose regression problem is introduced. Following that, different depth image encoding methods are discussed and a novel encoding method is proposed. Then the training method by stages for the proposed network is given. In the end, the evaluation experiments based on the public dataset and our own dataset are presented to prove the accurate and robust performance of our proposed system.

- Chapter 5 proposes a novel unsupervised deep learning based visual SLAM system called DeepSLAM. Related works are given first. Then the system overview is presented, and the system consists of Mapping-Net, Tracking-Net, Loop-Net, and a graph optimization unit. Following that, the loss function which drives the networks training is introduced. Temporal losses, spatial losses and outlier rejection mechanism are carefully designed for the system. Then the loop detection with Loop-Net and the pose graph construction are presented. Finally, in order to prove the pose estimation and depth estimation performance of DeepSLAM, the corresponding evaluation experiments are performed based on some public benchmark datasets.

- Chapter 6 describes a semantic pixel-wise mapping system for potential robotic applications. A related literature review is given first. The system that includes a novel spatio-temporal deep neural network and a SLAM algorithm is introduced then. A 3D semantic pixel-wise map is generated. To evaluate system performance in both pixel-wise accuracy and Intersection over Union (IoU), the proposed spatio-temporal neural network is performed on both Cityscapes benchmark and Essex indoor benchmark.

- Chapter 7 concludes the work of this thesis and major contributions. After listing the academic papers published or submitted during this research, ideas for future work are summarized.

Chapter 2

Background Review

In this chapter, some related work of this research is introduced to provide a comprehensive and consistent overview. Section 2.1 reviews related works of model-based visual SLAM methods which include both feature-based and direct methods. Three types of deep learning networks and available training data sets are given in Section 2.2. Depth estimation methods with deep learning are surveyed in Section 2.3, followed by the review of pose estimation methods with deep learning in section 2.4. Section 2.5 introduces the state-of-the-art semantic mapping methods with deep learning.

2.1 Model-based SLAM Methods

Model-based SLAM methods explicitly model camera projections, motions and environments based on multiple view geometry and photometric consistency. They can be divided into feature-based methods and direct methods. Feature-based methods extract and match feature points from 2D images, and then compute and optimize camera pose along with the position of these feature points in 3D. In contrast, direct methods do not need to extract feature points but instead directly use pixels in the image to compute 6-DoF camera poses by minimizing photometric errors. There exist plenty of model-based methods, but we just focus on a few below which perform extremely well in terms of localization and mapping accuracy.

Table 2.1: Some important model-based SLAM methods.

Methods	Year	Reference
Feature-based SLAM methods	2003	MonoSLAM [14]
	2007	PTAM [4]
	2014	RGB-D SLAM [5]
	2015	ORB-SLAM [6]
Direct SLAM methods	2011	DTAM [7]
	2014	SVO [8]
	2014	LSD-SLAM [15]
	2017	DSO [9]

2.1.1 Feature-based Visual SLAM Methods

MonoSLAM [14] proposed by Davison et al. is one of the earliest real-time visual SLAM systems with a monocular camera. Different from Structure from Motion (SfM) approaches that lack real-time performance, MonoSLAM adopts a probability framework and creates sparse but consistent 3D feature points for the map. By combining a general camera motion model and feature initialization, MonoSLAM achieved 3D localization and mapping with 30Hz real-time performance on a standard PC. MonoSLAM bridged pure vision and autonomous robotics closer and provided some new potential applications for Augmented Reality (AR).

However, tracking and mapping in MonoSLAM system are intimately linked and operated in one single thread. In other words, the 6-DoF camera pose and the 3D map points are updated together at every frame. It has the limitation on the number of sparse features the algorithm could handle due to the use of the large volume of images. In order to tackle this problem, Klein et al. proposed a parallel tracking and mapping (PTAM) system [4] which separates tracking and mapping into two parallel threads. The mapping thread is updated according to keyframes and is performed using computationally expensive bundle adjustment technology. The tracking thread is updated at framerate to produce 6-DoF camera pose estimation based on the built 3D map. PTAM was successfully performed with a hand-held camera but could only work in a small environment.

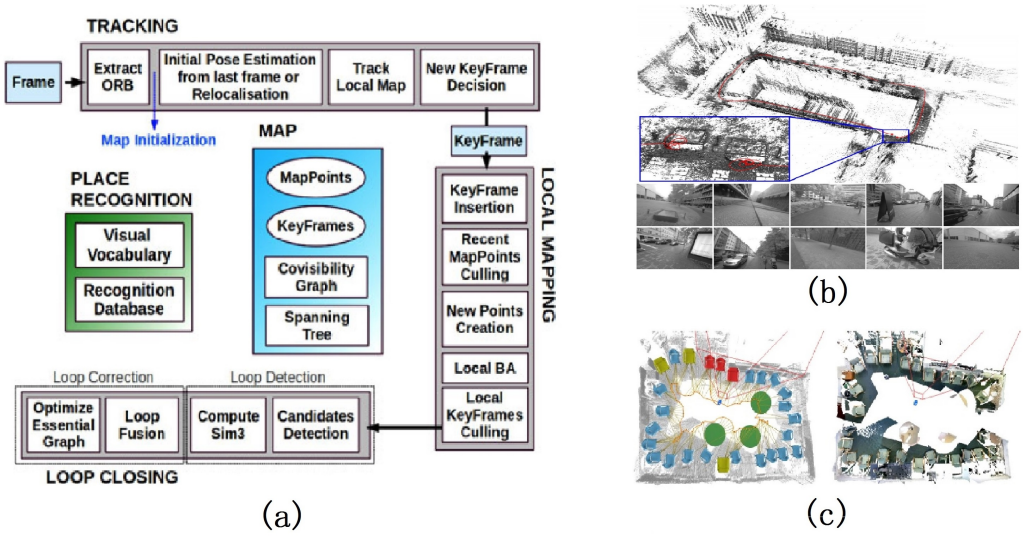


Figure 2.1: Some model-based SLAM methods. (a) Feature-based SLAM method–ORB-SLAM (image from [6]) (b) Direct SLAM methods–Direct Sparse Odometry (DSO) (image from [9]) (c) Semantic SLAM method–SLAM++ (image from [16])

ORB-SLAM proposed by Mur-Artal et al. [6] is one of the most successful feature-based SLAM systems so far. They first proposed a place recognition system [17] with ORB features based on Bag-of-Words (BoW) technology. ORB [18] is a rotational invariant and scale aware feature, and could be extracted in high frequency. The proposed place recognition algorithm can run efficiently, resulting in the real-time implementation of both relocalization and loop closing in visual SLAM systems. Then building upon the main ideas of PTAM [4] and the ORB place recognizer [17], they proposed ORB-SLAM [6] with monocular cameras, which could be performed in large-scale environments and has demonstrated a superior performance. Afterwards, they extended ORB-SLAM from monocular cameras to stereo and RGB-D cameras [19].

Endres et al. proposed RGB-D SLAM [5], which is based on feature points. The proposed RGB-D SLAM can generate dense and accurate 3D maps. In recent years, there appears a new kind of sensors called event camera or Dynamic and Active-pixel Vision (DAVIS) sensor. The corresponding SLAM algorithms [20] [21] were proposed for 6-DoF motion tracking and 3D reconstruction and impressive performance was demonstrated, especially in some challenging scenarios.

The shift from low-level point features to high-level objects is also observed in emerging semantic SLAM. Salas-Moreno et al. [22] presented a planar SLAM system which could

detect planes in environments and yield a planar map. They also proposed a SLAM system called SLAM++ [16], which can detect objects, such as chairs and desks, and then utilize these objects for localization. However, only limited number of objects, such as planes, desks and chairs are extracted, and specific supervised off-line learning is required.

2.1.2 Direct Visual SLAM Methods

Different from above feature-based methods, direct methods do not use sparse features, instead using all pixels in an image to estimate 6-DoF camera poses by penalizing some photometric errors for each overlapping image pair.

Newcombe et al. proposed a dense tracking and mapping (DTAM) system [7]. As the depth of each pixel in an image is estimated, DTAM generates a dense 3D map for each frame. Afterwards, Newcombe et al. [23] proposed KinectFusion using RGB-D cameras which was successfully demonstrated in dense registration and mapping. KinectFusion relies on a truncated signed distance function (TSDF) for pixel grid representation and utilizes Iterative Closest Point (ICP) for aligning depth images. Both DTAM and KinectFusion run in room scale environments with a commercial GPU for real-time performance.

Whelan et al. presented Elasticfusion [24] based on surfel representation with a RGB-D camera. By using frame-to-model tracking and non-rigid deformation, Elasticfusion performs the time windowed surfel-based dense data fusion. The dense global consistent map is obtained without the need for pose graph optimization or post-processing steps. A GPU is also required for camera tracking and dense mapping in order to achieve real-time performance.

In order to increase the efficiency of dense based methods, Engel et al. proposed semi-dense visual odometry (SVO) [25] which runs real-time on CPU. SVO does not use all pixels in the image but uses the pixels with a non-negligible image gradient. The semi-dense inverse depth map is estimated and 6-DoF camera motion is tracked with the alignment of estimated depth maps. Forster et al. also presented a similar approach called SVO [8]. Engel et al. then improved SVO [25] by introducing Large-Scale Direct Monocular SLAM (LSD-SLAM) [15] which could run in large-scale environments with CPU. LSD-SLAM employs $\mathfrak{sim}(3)$ to detect scale drifts and provides a probabilistic solution to handling the noisy depth prediction during tracking. Recently, Engel et al. further improved the direct method and proposed

Direct Sparse Odometry (DSO) [9]. DSO combines photometric errors with geometric errors and optimizes all the model parameters jointly. The demonstrated performance includes high accuracy in tracking and mapping, and robustness in some featureless environments.

Pascoe et al. proposed NID-SLAM [26] which is also a direct method for monocular cameras. Instead of penalizing photometric errors like most direct methods, NID-SLAM chooses to use Normalized Information Distance (NID) metric to estimate the camera motion. NID-SLAM demonstrated its robustness to appearance changes.

2.1.3 Summary

Model-based visual SLAM methods have successfully demonstrated their powerful capabilities in pose estimation and 3D map construction. Particularly the feature-based representative ORB-SLAM [6] and the direct representative DSO [9] both achieve high accuracy in the large-scale environment, and real-time performance with commercial CPUs. However, their robustness still tends to be struggling when they face some featureless environments or other challenging scenes, e.g, serious image blur. Further, they do not have a learning capability to be adaptive to specific circumstances. The success of deep learning in image processing and recognition sheds some light on the improvement of robustness performance over time through the continuous acquisition of data.

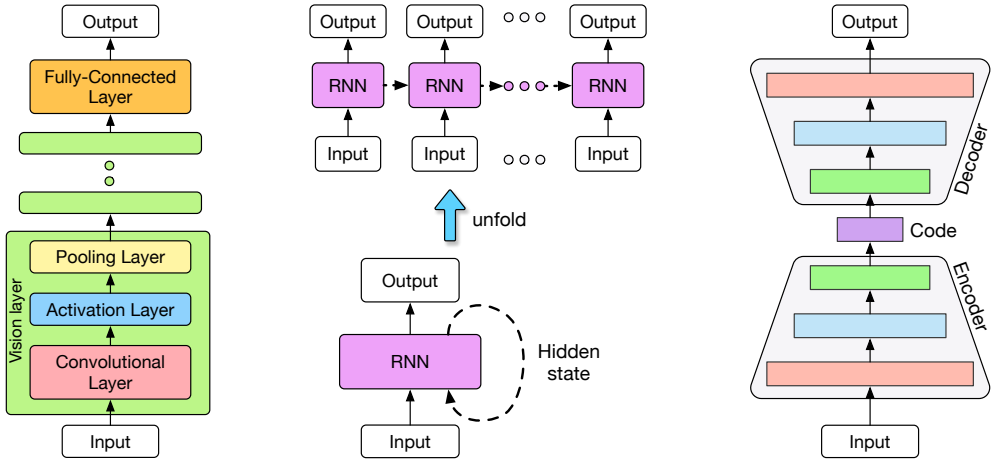
2.2 Deep Neural Networks for Visual SLAM

Model-based methods represent the input images with manually designed features and search for the best pose which matches the features between image frames. Deep learning directly learns good representations of the input images at multiple levels. The representation could be unknown features, depth, or even ego-motion between two frames for SLAM problems.

In this section, three types of deep neural networks (DNNs) are briefly illustrated, which have been already found in deep learning SLAM methods. For more information on deep learning, the reader is referred to [27].

2.2.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is one of the most popular deep neural network architectures to date [28]. A CNN mainly consists of vision layers (e.g., convolutional layer, activation layer, pooling layer) and common layers (e.g., fully-connected layer), as shown



(a) CNN. (b) RNN. (c) Auto-Encoder.

Figure 2.2: Three popular deep learning architectures: Convolutional Neural Network, Recurrent Neural Network and Auto-Encoder.

in Fig. 2.2(a). Dropout layers and normalization layers (e.g., batch normalization layer) are also frequently incorporated. A loss function, such as Softmax and Euclidean loss, drives the training by minimizing the differences between the predictions and the labels.

Convolutional layers are a core component of CNNs although the specific structures of CNNs vary from case to case. Given an input tensor $\mathbf{x}$, they produce an output feature map $\mathbf{h}^k$ by the convolution of an input image with a linear filter $\mathbf{W}^k$:

$$\mathbf{h}^k_{ij} = (\mathbf{W}^k * \mathbf{x})_{ij} + \mathbf{b}_k \tag{2.1}$$

where $\mathbf{b}_k$ is a bias term, $*$ is the matrix multiplication operation. Thanks to the local connectivity and parameter sharing of the convolutional layers, CNNs vastly reduce the number of parameters and are more efficient than multi-layer perception.

Another important layer of CNN is pooling, which is a form of non-linear down-sampling and always occurs after convolutional layer. Pooling partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum, average or stochastic value according to the type of pooling operation. It provides a form of translation invariance and reduces computation for upper layer and dimensionality of intermediate representations. More generally, assuming the input volume size is $\mathbf{W}_1 \times \mathbf{H}_1 \times \mathbf{D}_1$, given two hyperparameters: the spatial extent $\mathbf{F}$ and the stride $\mathbf{S}$, then the output volume size is $\mathbf{W}_2 \times \mathbf{H}_2 \times \mathbf{D}_2$, where:

$$\mathbf{W}_2 = (\mathbf{W}_1 - \mathbf{F})/\mathbf{S} + 1 \tag{2.2}$$

$$\mathbf{H}_2 = (\mathbf{H}_1 - \mathbf{F})/\mathbf{S} + 1 \tag{2.3}$$

$$\mathbf{D}_2 = \mathbf{D}_1 \tag{2.4}$$

After each convolutional layer, it is a convention to apply an activation layer (or nonlinear layer) immediately afterward. The purpose of this layer is to introduce nonlinearity to a system that basically has just been computing linear operations during the convolutional layers, the operations include element-wise multiplications and summations. Some popular activation functions are Sigmoid, TanH, ReLU etc..

The fully connected layer (also usually referred to as the InnerProduct layer) treats the input as a simple vector and produces an output in the form of a single vector. Neurons in a fully connected layer have full connections to all activations in the previous layer. And their activations can hence be computed with a matrix multiplication followed by a bias offset.

The loss function is a vital part of training DNNs. With a carefully designed loss function, the network can be trained to solve different kinds of problems. For supervised learning, we want to get to a point where the predicted label (network output) is the same as the training label (target output). In order to get there, the loss function itself is computed by the forward pass and the gradient with respect to the loss is computed by the backward pass by minimizing the loss function. Cross-entropy loss is usually used for classification problems, while the Euclidean loss is mainly employed for regression ones.

2.2.2 Deep Recurrent Neural Network

Recurrent Neural Network (RNN) is mainly designed to capture the temporal dynamics in video clips. It maintains the memory of its hidden states over time via feedback loops, and models the dependencies between current input and previous states. The RNN and its unfolded version are shown in Fig. 2.2(b). Given an input $\mathbf{x}_k$ at time k, a simple RNN updates

at time k by:

$$\mathbf{h}_k = H(\mathbf{W}_{xh}\mathbf{x}_k + \mathbf{W}_{hh}\mathbf{h}_{k-1} + \mathbf{b}_h) \tag{2.5}$$

$$\mathbf{y}_k = \mathbf{W}_{hy}\mathbf{h}_k + \mathbf{b}_y \tag{2.6}$$

where $\mathbf{h}_k$ and $\mathbf{y}_k$ are the hidden state and output at time k, respectively. $\mathbf{W}_{xh}$, $\mathbf{W}_{hh}$, $\mathbf{W}_{hy}$ are the weight matrices, $\mathbf{b}_h$, $\mathbf{b}_y$ are the bias terms, and $H()$ is a non-linear function. However, simple RNN suffers from the vanishing gradient problem in practice [29]. The problem is that in some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value. In the worst case, this may completely stop the neural network from training. In order to solve this problem, Long Short-Term Memory (LSTM) is widely used. Specifically, a LSTM has several gates to control when to keep or forget the memory. In deep RNNs related to visual SLAM, the RNNs are usually connected to the features of CNNs. This forms a paradigm termed Recurrent Convolutional Neural Network (RCNN), in which CNNs and RNNs capture the spatial and temporal representations from video clips respectively.

2.2.3 Auto-Encoder

An auto-encoder is a special kind of DNNs derived from CNN. As shown in Fig. 2.2(c), it consists of an encoder part and a decoder part. Specifically, an auto-encoder maps its input $\mathbf{x}$ into a hidden code $\mathbf{y}$ through the encoder part:

$$\mathbf{y} = e(\mathbf{x}) \tag{2.7}$$

where $e()$ is a non-linear function representing the encoder. Then the decoder part maps the hidden code $\mathbf{y}$ into a reconstruction $\mathbf{z}$ that usually represents the same main features with input $\mathbf{x}$. Its map function is:

$$\mathbf{z} = d(\mathbf{y}) \tag{2.8}$$

where $d()$ is a non-linear function (a neural network) denoting the decoder network. For decoder part, deconvolution layers, dilated convolution layers, upsampling and convolution

layers are often used for feature decoding. Auto-encoders have been widely used for depth estimation and semantic segmentation. We will discuss the details in the following sections.

2.2.4 Dataset

Deep learning methods require a large amount of data for training. In this part, we review the existing datasets which could be adopted for deep learning related visual SLAM tasks.

The KITTI benchmark [30] was collected in outdoor environments with a driving car. It provides stereo images with ground-truth 6-DoF poses derived from the fusion of multiple sensor data. Depth data is also provided with the calibrated laser. Some images in KITTI are labeled manually for image segmentation. Similar to KITTI, Cityscapes dataset [31] also has stereo images, depth images, semantic labeled images and 6-DoF poses. The RobotCar dataset [32] was collected with a car driving in Oxford for a year, which means it contains different views of the same place. EuRoc MAV dataset [33] was gathered by using a flying robot and could be used for VO and SLAM problems. TUM dataset [34] and NYU dataset [35] were collected with a hand-held RGB-D camera in indoor environments. They can provide color and depth images. In addition, NYU dataset [35] also provides some labeled images for semantic segmentation. PASACL VOC [36] and COCO [37] datasets with labeled images aim at image segmentation problems. ADE20K [38] contains more than 20K pixel-wise semantic annotated images.

To summarize, KITTI [30], Cityscapes [31], TUM [34], NYU [35] datasets can be used for depth estimation. For relocalization problem, one can use 7-scenes dataset [39] and Cambridge landmarks [40]. Meanwhile, KITTI [30], Robotcar [32], M'alaga [41], EuRoc MAV [33], NYU [35] and TUM [34] datasets are applicable to ego-motion estimation. For scene segmentation, PASCAL VOC [36], NYU [35], Cityscapes [31], KITTI [30], ADK20 [38] datasets can be utilized.

2.3 Depth Estimation with Deep Learning

Depth estimation is fundamental in a SLAM system. Model-based SLAM methods usually take advantage of camera parallax from multiple images to estimate the depth. With the development of deep learning, data-driven methods provide an alternative to depth estimation. Depth estimation with deep learning can be divided into supervised methods and

Table 2.2: Some important depth estimation methods with deep learning.

Methods	Year	Reference
Supervised methods	2014	Eigen et al. [42]
	2016	Liu et al. [43]
	2017	CNN-SLAM [44]
Unsupervised methods	2016	Garg et al. [45]
	2017	Godard et al. [12]
	2017	SfMLearner [1]

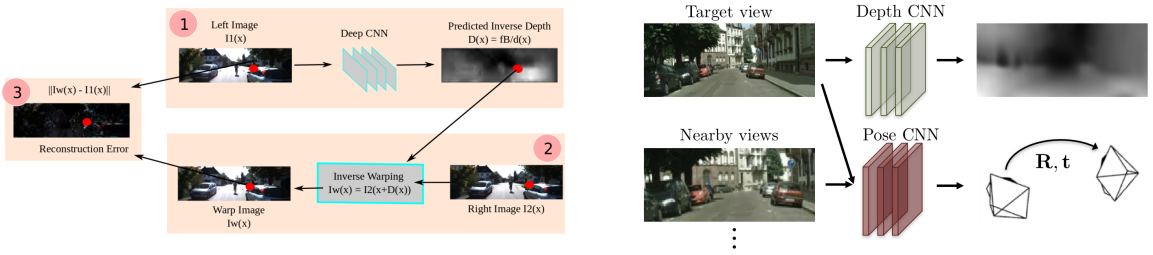


Figure 2.3: Depth estimation with unsupervised deep learning. (a) Garg's method (image from [45]) (b) SfMLearner (image from [1])

unsupervised methods.

2.3.1 Supervised Methods

Eigen et al. [42] designed a deep neural network to perform depth estimation with a single image. It is a supervised method where ground-truth depth map is required for network training. The network consists of two components: one for global structure prediction and one for local prediction refinement. A scale-invariant error is defined as the cost function for learning. The real scale of depth is recovered without any postprocessing. The proposed method produced good results on both NYU Depth [35] and KITTI [30] datasets.

Ladicky et al. [46] made use of the property of perspective geometry that the size of objects scales inversely with the depth to transform the image to the canonical depth for training. They also proposed to combine semantic segmentation and depth estimation together to improve the performance. The proposed method is also a supervised depth estimation method with single images.

Liu et al. [43] also presented a depth estimation method with single images using the

so-called deep convolutional neural field (DCNF), which integrates continuous Conditional Random Field (CRF) into a unified deep CNN framework. Further, a superpixel pooling method and fully convolutional networks (FCN) were proposed in [47] to improve the real-time performance. A similar approach was also presented by Li et al. [48].

The attempt to combine depth estimation with visual SLAM was made in [44], called CNN-SLAM. It is a monocular SLAM system in which the predicted depth map from CNN is dense and has the absolute scale. Comparing with model-based methods, only the depth is estimated from CNN while other parts, such as pose estimation, graph optimization, etc. are the same as feature-based SLAM. The proposed method demonstrated robust and accurate performance in pose estimation and map construction.

Ma et al. proposed a so-called "Sparse-to-Dense" [49] method to predict dense depth images, which could be used as a plug-in module to model-based SLAM methods to create an accurate, dense point cloud. They constructed two CNNs to fuse a RGB image with a sparse depth image. Their sparse depth image could be a model based SLAM or a low-cost LiDAR.

Supervised methods require a large amount of labeled data to train the networks. Since it is costly to collect labeled datasets, their applications are limited.

2.3.2 Unsupervised Methods

Recently depth estimation methods using unsupervised deep learning have emerged. These methods adopted stereo images for training. Stereo images are captured by using two cameras which have a fixed baseline. The main idea came from the representation capability of auto-encoders where the encoder is a CNN which represents left input image with a predicted depth map while the decoder is a warp function which synthesizes a reconstructed image from the right input image and the predicted depth map. The reconstructed error is used as the cost function to train the CNN [45].

Specifically, for the overlapped area between two stereo images, every pixel in one image can find its correspondence in the other with horizontal distance H in pixel [45]:

$$H = Bf/D \tag{2.9}$$

where B is the baseline of the stereo camera, f is the focal length and D is the depth value

of the corresponding pixel. By using the geometric constraint D map, the left image can be synthesized from the right and vice versa. Then the photometric loss function E is defined as below:

$$E = \sum \left\| I - I^{'} \right\|_2 \tag{2.10}$$

where I is the original RGB image matrix, and $I^{'}$ is the synthesized RGB image matrix which has the same size with I. By minimizing the photometric loss function E between the original left image and the synthesized left image, the network is trained fully unsupervised in an end-to-end manner. The proposed method can be viewed as a monocular depth estimation system for the reason that it only needs monocular images during testing. It even outperformed some supervised methods in terms of accuracy of depth estimation. Xie et al. [50] proposed to use a deep neural network to predict a disparity map from the left input image with unsupervised learning and then renders a novel right image for 2D-to-3D video conversion applications.

Godard et al. [12] improved the Garg's method [45] by wrapping left and right images across each other to synthesize corresponding images. In this way, the accuracy of depth prediction could be enhanced by penalizing both left and right photometric losses. Then Zhong et al. [51] presented a very similar unsupervised depth estimation system with stereo images as network inputs.

Zhou et al. [1] proposed SfMLearner, which uses a monocular image sequence for image alignment in order to estimate the depth and ego-motion simultaneously with unsupervised learning. The geometric constraint between temporal image pairs is used for synthesizing corresponding images. After the training of the networks, the depth images and camera poses can be simultaneously predicted by the networks in an end-to-end manner. However, the estimated depth map and ego-motion are lack of the scale. Based on SfMLearner [1], Yang et al. [52] proposed to use a CNN to represent the surface normal map. Both predicted depth map and normal map are used to construct the loss function. Vijayanarasimhan et al. [fm] presented SfM-Net which adds motion masks to the photometric loss. It can estimate optical flow, depth map, and ego-motion simultaneously.

Table 2.3: Some novel pose estimation methods with deep learning.

Methods	Year	Reference
Relocalization	2015	PoseNet [40]
	2017	Li et al. [53]
	2017	Clark et al. [54]
Ego-motion estimation (Supervised)	2017	DeepVO [13]
	2017	DeMoN [55]
Ego-motion estimation (Unsupervised)	2017	SfMLearner [1]
	2017	UnDeepVO [56]
Feature learning	2017	DeTone et al. [57]

2.3.3 Summary

The move from supervised learning towards unsupervised learning in depth estimation with the view of spatial geometric constraints is significant as no labeled data is required and life-long learning is made feasible [45]. Unsupervised learning depth estimation is also important in building up dense maps for SLAM systems. With the view of temporal constraints in an image sequence, it is possible to estimate the ego-motion with unsupervised learning. This will be reviewed in next section.

2.4 Pose Estimation with Deep Learning

Data-driven pose estimation with deep learning learns camera motion model and estimates poses directly without explicitly modeling. The relocalization problem was targeted with supervised deep learning first as it is possible to collect labeled data in pre-visited places. Building on the success of depth estimation with unsupervised learning, more attentions have been paid to ego-motion estimation by using unsupervised learning.

2.4.1 Relocalization with Deep Learning

Most deep learning networks are used for classification problems. Less are found for regression problems. Kendall et al . [40] first used a CNN to solve the pose regression problem. Their PoseNet was trained with supervised learning with the requirement of ground-truth 6-DoF poses available for training (see Fig. 3a). The loss function L is designed as below:

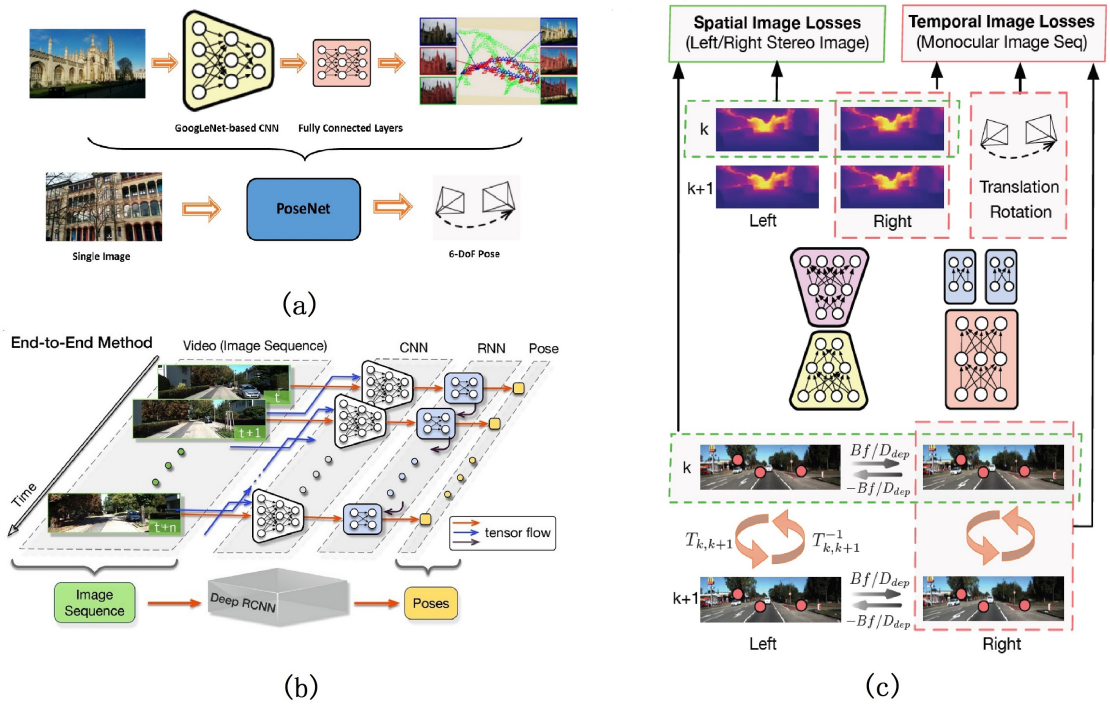


Figure 2.4: Pose estimation with deep Learning. (a) PoseNet (image from [40]) (b) DeepVO (image from [13]) (c) UnDeepVO [56]

$$L = \|\hat{\mathbf{x}} - \mathbf{x}\|_2 + \lambda \|\hat{\mathbf{q}} - \mathbf{q}\|_2 \tag{2.11}$$

where $\mathbf{x}$ is the representation of camera position, $\mathbf{q}$ is unit quaternion representation of camera orientation, and λ is the balance weight to normalize position and orientation losses. After supervised training with labeled data, the PoseNet could perform relocalization in pre-visited places with more robust performance than model-based methods.

In order to estimate the uncertainty of pose estimation, Kendall et al. [58] further proposed Bayesian PoseNet by using Dropout layers , Dropout layer a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. in the network as a means of sampling. The selection of balance weight is determined through a trial and error method. Kendall improved PoseNet and proposed a fusion model to train the network and automatically adjust the balance weights in [59]. The performance is improved, and the uncertainty can be estimated by the network without sampling.

Li et al. [53] then extended PoseNet from single CNN to dual-stream CNNs to accommodate color and depth inputs from RGB-D cameras. The proposed system showed very

robust performance when faced with challenging situations. They also applied PoseNet in night-time environment with a depth sensor [60].

Deep RCNN architecture is adopted to explore the temporal dynamics of camera motion in pose estimation. Clark et al. [54] proposed to use an RCNN to implement the pose regression with video clips. By taking image sequences as network inputs, the uncertainty of pose estimation was reduced and relocalization performance was improved. Hazirbas et al. [61] incorporated a spatial LSTM module into PoseNet to improve the relocalization performance. The proposed system takes a single color image as input. Naseer et al. [62] adopted PoseNet [40] as the basic network and used data augmentation technology to improve the relocalization performance. They applied the transformation to synthesize additional input images. By using multiple synthesized images as inputs to perform relocalization, the system achieved better performance.

2.4.2 Ego-Motion Estimation with Deep Learning

Apart from absolute pose regression, the ego-motion between two image frames can also be estimated by using deep learning inspired by stereo geometric models. Ego-motion estimation methods with deep learning can also be divided into supervised and unsupervised methods.

DeTone et al. proposed HomographNet [63] and used CNN to predict the homography parameters between image pairs. The proposed deep homography estimator outperformed the traditional homography estimation method based on ORB features.

Costante et al. [64] developed a CNN to estimate the ego-motion with supervised training. Wang et al. proposed monocular visual odometry system called DeepVO [13], which trains a RCNN to estimate the camera motion in an end-to-end manner. The temporal image sequence is introduced into RCNN with LSTM module (see Fig. 3b). The experiment results demonstrated an impressive performance on visual odometry. They then extended DeepVO to incorporate uncertainty estimation into the system [3]. Melekhov et al. [65] also presented a relative camera pose estimation system with CNN. Turan et al. proposed Deep endovo [66] which is similar with DeepVO [13], and applied it to the area of soft robotics [67].

Oliveira et al. [68] constructed a metric network for ego-motion estimation and a topological network for topological location estimation. The topological network discretizes the

trajectory into a finite set of locations and uses the CNN to learn the topological relationship. By successfully combining this network with ego-motion estimation network, the system demonstrated good performance in localization.

Ummenhofer et al. [55] proposed a system called DeMoN which consists of a chain of encoder-decoder networks. An iterative network is specifically designed for the system. DeMoN can estimate ego-motion, image depth, surface normal and optical flow simultaneously but needs labeled data for training.

Different from the methods using CNNs to estimate camera motions directly, DeTone et al. [57] developed one network to estimate the location of feature points and another network to match the extracted features and compute the homography. The network needs manually synthesized data for training.

Instead of predicting camera poses with a deep neural network directly, Peretroukhin et al. [69] proposed to use a model-based geometric estimator for pose prediction and a CNN for predicted pose correction. In detail, the proposed CNN is trained to learn the errors between the ground-truth poses and the predicted poses from a model-based estimator. The proposed system is called DPC-Net and could also be used for mitigating the effect of bad camera calibration parameters.

Aiming at the estimation of camera ego-motion, Costante et al. presented a novel CNN architecture which is called LS-VO [70]. LS-VO consists of an auto-encoder network to learn optical flow representations and is followed by a pose estimation network to predict camera poses. The networks take temporal image pairs as inputs and are trained jointly end-to-end.

In order to tackle the scale drift problem in monocular SLAM, Frost et al. [71] proposed to adopt a CNN to perform the speed regression from successive video frames. By further integrating the estimated speed into bundle adjustment, they successfully realized the scale-drift correction.

All the above-mentioned methods use supervised learning schemes which require labeled datasets. Labeling large amounts of data is difficult and expensive. It is very demanding for a visual SLAM system to learn under an unsupervised scheme so that the performance could be continuously improved by the increased size of unlabeled datasets.

Recently, [1] presented an ego-motion and depth estimation system with unsupervised

deep learning, but the system can not recover the absolute scale. Inspired by the unsupervised depth estimation methods [1, 12], Li et al. proposed UnDeepVO [56] which is a monocular visual odometry system with unsupervised learning. By using stereo pairs for training (see Fig. 3c). UnDeepVO demonstrated an extraordinary performance in pose prediction and depth estimation. Further, it can also recover the absolute scale of 6-DoF poses and depth maps. Nguyen et al. [72] also introduced the similar unsupervised deep learning method into homography estimation.

2.4.3 Sensor Fusion with Deep Learning

Clark et al. [73] proposed a sensor fusion network called VINet, which fuses the estimated pose from DeepVO [13] and the inertial sensor reading with a RCNN. The prediction network and the fusion network are trained jointly end-to-end, and the proposed fusion system demonstrated comparable performance with traditional sensor fusion methods. Turan et al. [74, 75] adopted the same method and presented a fusion system to fuse the 6-DoF poses from cameras and magnetic sensors.

Pillai [76] proposed an ego-motion estimation system that fuses the information from a camera with other sensors such as GPS, INS and wheel odometry. They also adopted a Mixture Density Network to use optical flow vectors from different kinds of camera optics.

Byravan et al. [77] proposed a CNN architecture called SE3-Net, which takes raw point cloud data as input and predicts SE3 rigid transformation.

2.4.4 Summary

Pose regression with CNN is a bold attempt to apply supervised deep learning for relocalization problems [40], while the ego-motion estimation [13] is a result of the capability of deep learning to capture the temporal motion dynamics. However, labeling data in large-scale hinders the application of supervised deep learning in visual SLAM systems. Unsupervised deep learning methods are powerful and promising for pose and depth estimation [1, 56].

2.5 Semantic Mapping with Deep Learning

For most autonomous robotic applications, the semantic perception of the environment is extremely important. In the computer vision community, semantic segmentation has been a long researched and established topic. When combining semantic segmentation with visual

Table 2.4: The milestone semantic perception methods with deep learning.

Methods	Year	Reference
Semantic segmentation	2015	FCN [47]
	2015	SegNet [78]
	2016	Deeplab [79]
	2016	Wu et al. [80]
	2016	PSPNet [81]
Semantic mapping	2017	Semanticfusion [11]
	2017	Li et al. [82]
	2017	Zhao et al. [83]

SLAM, it is possible to estimate semantic 3D map and camera motions simultaneously for robotic applications.

2.5.1 Semantic Segmentation

Long et al. first proposed a FCN [47] for pixel-wise semantic segmentation. The proposed system consists of convolutional layers but no fully connected layer in the network (see Fig. 4a). It achieved the state-of-the-art pixel-wise segmentation performance at that time. Afterwards, various CNN architectures were derived from the FCN [47]. Liu et al. [84] took advantage of the global context information and introduced the global pooling into the FCN [47]. The proposed system is called ParseNet, which outperformed the FCN in scene segmentation with the wider view of the network.

Badrinarayanan et al. [78] presented a novel network architecture called SegNet for scene segmentation. The SegNet is based on the FCN and has an encoder-decoder architecture (see Fig. 4b). The decoder performs the upsampling for low-resolution features and can recover the resolution of raw input. Afterwards, Kendall et al. [85] proposed Bayesian SegNet which uses dropout layers in the SegNet as sampling. The proposed Bayesian SegNet can estimate the probability for pixel-level segmentation and achieved better performance than SegNet.

CRFs have proved its powerful capability in image segmentation and was adopted as a post-processing method to refine the image segmentation. Zheng et al. [86] proposed to formulate the probabilistic mean field inference with CRFs as RNNs. By embedding CRFs

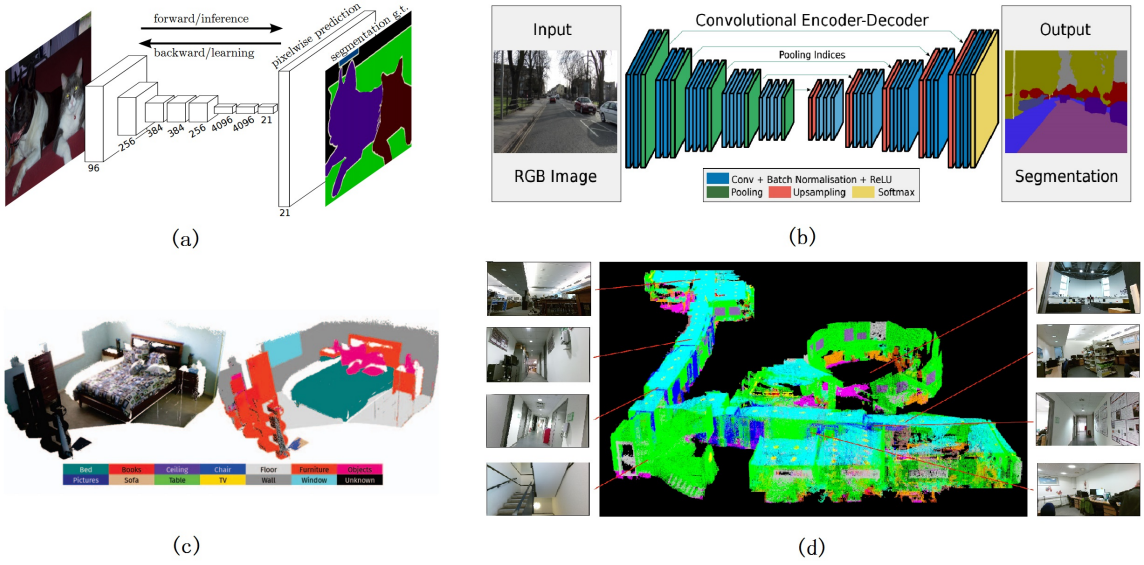


Figure 2.5: Semantic mapping with deep learning. (a) FCN (image from [47]) (b) SegNet (image from [78]) (c) Semanticfusion (image from [11]) (d) Semantic mapping [82]

into CNNs, they presented a novel network architecture called CRF-RNN, which combines the strength of both CNNs and CRFs. Afterwards, Arnab et al. [87] designed two high order potentials based on object detection and superpixels, and integrated them into the CRF-RNN. However, CRFs are especially computational intensive and not suitable for real-time applications.

The networks mentioned above all used the VGG [88] as their base network architecture. After He et al. proposed a very deep ResNet [89], most researchers began to use the ResNet as the basic network architecture. The ResNet demonstrated an astonishing performance in the ImageNet classification challenge [90] and has been widely applied for many tasks. Chen et al. [79, 91, 92] proposed to use the very deep ResNet, dilated convolution, and fully connected CRFs to perform image segmentation. By using dilated convolution, the field-of-view of filters could be enlarged effectively without increasing the computation. Atrous Spatial Pyramid Pooling (ASPP) and multiple scale technologies were also introduced in Deeplab [79], which performed extremely well in PASCAL VOC-2012 [93] semantic image segmentation dataset.

Wu et al. [94] explored variations of the ResNet in order to find the best network configuration, such as the number of layers, the size of field-of-view and the resolution of feature maps. An online bootstrapping method is also used during training to improve the seg-

mentation performance. The proposed network was evaluated on both PASCAL VOC-2012 benchmark and Cityscapes [31] benchmark. The results show that the proposed network was very competitive when compared with other methods.

Afterwards, Wu et al. [80] further studied the relationship between the depth of residual networks and the performance, and proved that some relatively shallow residual networks could outperform much deeper networks, particularly within some limitations. This performance is not only applied to the recognition task but also suitable for the semantic segmentation task.

Zhao et al. [81] proposed the Pyramid Scene Parsing Network (PSPNet) which won the ImageNet scene parsing challenge 2016 [38]. Different from the global pooling method proposed in [84], the global spatial context information in images was exploited by different-region-based aggregation with the proposed pyramid pooling model in [81].

Based on the SegNet [78], Hazirbas et al. [95] extracted features from color images and depth images respectively, and fused them together to perform upsampling. Both color features and depth features are exploited for segmentation with this FuseNet [95]. AdapNet was proposed by Valada et al. [96,97] for semantic segmentation in adverse environments. A novel fusion technology called Convoluted Mixture of Deep Experts (CMoDE) was presented to enable a multi-stream network to learn features from different modalities.

2.5.2 Semantic Mapping

Semantic information is particularly valued in robot-human and robot-environment interaction. With the progress in semantic segmentation using deep learning, semantic SLAM research grows rapidly. Li et al. [82] combined model-based SLAM methods with spatio-temporal CNN-based semantic segmentation (see Fig. 4d). The proposed system can perform 3D semantic scene mapping and 6-DoF localization simultaneously. The system could perform in large indoor environments. A similar semantic mapping system with the pixel-voxel network was proposed by Zhao et al. [83]. McCormac et al. [11] proposed Semanticfusion which integrates CNN-based semantic segmentation with the dense SLAM technology ElasticFusion. Semanticfusion can perform in indoor scenes and produce a dense 3D semantic map (see Fig. 2.5c).

2.5.3 Summary

Semantic SLAM is very challenging without the help of deep learning. The success in semantic segmentation, such as FCN [47] SegNet [78] Deeplab [80, 86], boosts the research in semantic SLAM [11, 82, 83]. It is expected more fruitful results will be generated in coming years. However, the most significant challenge in semantic SLAM is the supervised learning which requires a large amount of labeled dataset to train the networks. Unsupervised learning semantic SLAM is still not made available due to the lack of the breakthrough in unsupervised semantic segmentation.

Chapter 3

A Novel SLAM Algorithm based on Points and Plane-Patches

In this chapter, we propose a novel model-based RGB-D SLAM algorithm. The proposed system is based on the usage of feature points and plane patches for ego-motion estimation. The feature points are classified into plane points, smooth points and structural points according to the curvature. A plane patch is defined as a small-sized patch constructed by using a plane point. The feature points and plane patches are weighted with different values and combined together for pose estimation. We evaluate the proposed algorithm on multiple benchmark datasets to prove its accuracy and robustness.

3.1 Introduction

SLAM technique [98], [99] enables robots to localize themselves and construct the map in an unknown environment.

Nowadays, commercial RGB-D cameras are preferred in indoor robotic applications considering their low-price, good technical support, real-time performance and excellent sensing capability. They can provide not only color images which contain texture and appearance information, but also the depth of images containing range information and structure of objects which are robust and invariant when lighting condition varies. What is more, Visual SLAM using commercial RGB-D cameras [100] can also help the system obtain a dense 3D map of the environment.

Nowadays RGB-D SLAM algorithms can be divided into feature-based [100], [6] and

direct-based [101], [23], [102]. Feature-based algorithms are the main focus of this work, which extract and use invariant features as primitives to perform geometrical registration. Most current algorithms operate on low-level feature points, such as SIFT [103], SURF [104] and ORB [18]. These low-level primitives are generally noisy, redundant and require mass storage. Therefore, high-level primitives are introduced into the SLAM [105], [106], such as macroscopic-visible planes which are large planar regions (walls and floors etc.) They provide more stable parametric representation and compact interpretation of the scene. However the number of macroscopic-visible planes is usually very small and sometimes less than three in an image. And macroscopic-visible plane extraction is time-consuming due to the fact of plane normal calculations for all points. Furthermore, parameters of macroscopic-visible planes are sometimes different from those of constituted microscopic plane patches due to the noise from depth sensor and real geometry structure.

In this chapter, we introduce a novel RGB-D SLAM algorithm that uses both feature points and microscopic plane patches as primitives. We first extract feature points and use Principal Component Analysis (PCA) method to calculate the curvature and normal of each feature point. Then feature points are classified into plane points, smooth points and structural points. Next a feature matching process is carried out by combining feature point matching and plane patch matching. Feature points together with plane patches (called point-plane patch method in this thesis) are used to estimate the egomotion.

In the next section, a review of related works are provided. In section 3.3, the problem is stated and an overview of our SLAM system is given. Section 3.4 presents the method of plane patch extraction and matching. Section 3.5 addresses the issue of camera pose estimation using point-plane patch method. Section 3.6 demonstrates the experimental results and analyses the performance of our system. In section 3.7, our work is concluded.

3.2 Related Work

In this section, we give a review of the related works. As discussed in chapter 2, MonoSLAM was the earliest system to show SLAM performance using a single camera [98]. PTAM implemented a real-time SLAM algorithm by using parallel tracking and mapping threads [107]. DTAM improved the robust performance in textureless regions and fast motion situations with direct-based method [108]. As to RGB-D SLAM, Endres et al. [100] demon-

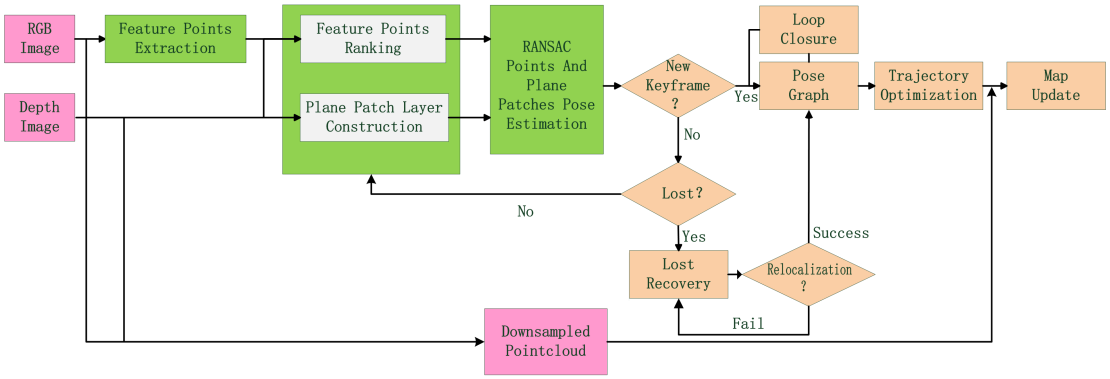


Figure 3.1: System overview of the proposed SLAM algorithm.

strated an RGB-D SLAM system which includes feature based pose estimation and pose graph optimization. Mur-Artal et al. [6] proposed an ORB-SLAM system which adopted ORB features and demonstrated real-time performance in tracking and mapping for monocular cameras and RGB-D cameras.

As to the related SLAM methods with planes as primitives, it can be traced back to the work done by Weingarten et al. [109]. They used 3D laser to detect planes and performed plane-to-plane registration with an ICP algorithm. Trevor et al. [110] combined RGB-D camera and 2D laser to detect the planar surfaces for pose estimation. Taguchi et al. [106] introduced a novel SLAM algorithm which treated both points and planes as primitives. Ataer-Cansizoglu et al. [111] improved Taguchi's SLAM system by predicting planes with a pose prediction model so that the time spent on plane extraction was decreased. Salas-Moreno et al. [22] presented a novel dense planar SLAM algorithm which perceived, merged and compressed the arbitrary planar regions and used them for registration. Gao [112] proposed to use planar points in pose estimation by extracting all planes first and then testing if feature points are on the planes. Kaess [105] proposed a minimum representation of plane parameters in order to produce a solution to the least-square estimation problem using Gauss-Newton methods. Cupec [113] used both planar surfaces and line segments to recognize places and relocalize the camera. Li et al. [114] presented to use points and surface normals to realize pose estimation which is similar with our algorithm and almost proposed at the same time compared with our method. The difference is that normals can only estimate rotation and are powerless to estimate translation.

In summary, macroscopic-visible planes potentially are able to produce the compact rep-

resentation, but consume more time on plane extraction.

3.3 Problem Description and System Overview

As with most SLAM algorithms discussed in chapter 1, the proposed SLAM algorithm is also separated into a front-end and a back-end. The difference is that two layers—feature point layer and plane patch layer—are used in the front-end of the proposed algorithm. The purpose of using both of them is to ensure the robust and real time performance in texture-less and large-scale environment. Our algorithm first extracts feature points in an image, and judges whether or not they are in plane patches based on feature points and their neighbourhood area. Then the plane patch's parameters are calculated. This is the work done in the plane patch layer construction (see Fig. 3.1). A plane patch consists of many robust plane points, leading to more compact representations and more accurate plane parameters. Meanwhile, the feature points are ranked according to their geometrical information. If a point is in a plane or in a smooth surface, it has less uncertainty in depth and contains more reliable information. If a point is on the edge of some objects and its depth is very different from neighbour points (its depth gradient is very large in one direction), it could have large noise in depth and may produce some bad effects in the process of pose estimation. Different weights are assigned to the points according to their uncertainties. The weighted features are able to improve the robustness and accuracy of the algorithm.

In the back-end, the algorithm selects keyframes and adds them into a pose graph. Then loop closures are detected by comparing current keyframe with previous nearby keyframes and keyframes having similar attitude. The general framework g2o [115] is used for graph optimization. The trajectory will be optimized and the map will be updated. If the algorithm loses the trajectory under some circumstances, it will perform a relocalization process from the lost frame. The overview of our approach is shown in Fig. 3.1.

3.4 Plane Patch Layer Construction

3.4.1 Plane Patches Extraction

For each points in pointcloud, we can calculate the curvature of these points. A plane patch is defined as a small-sized patch constructed by using 3D feature points with small curvature.

Feature points with small curvature are called plane points. Each plane point is treated as center of a potential plane patch, and the patch size is defined as 21×21 in pixel dimension through a trial and error method, PCA (Principle Component Analysis) technique [116] is adopted in extraction. For each patch, we have a set of points $\{\mathbf{p}_i = (x_i, y_i, z_i)^{\mathbf{T}}\}$, i = 1, ..., N. Let $\overline{\mathbf{p}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{p}_i$, then the 3×3 covariance matrix Σ can be calculated as below:

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{p}_i - \overline{\mathbf{p}})(\mathbf{p}_i - \overline{\mathbf{p}})^{\mathbf{T}} \tag{3.1}$$

Three eigenvalues $\lambda_1, \lambda_2, \lambda_3$ and their corresponding eigenvectors are computed if $rank(\Sigma) = 3$ [116]. The eigenvalues and the eigenvectors are the principle component of the pointcloud. If one eigenvalue is big and the other two are small, the pointcloud should be a line. If two eigenvalues are big and one is small, then the pointcloud should be a plane. Let $\lambda_1 > \lambda_2 > \lambda_3$, the eigenvector with minimum eigenvalue λ_3 represents the normal of this patch. In order to judge whether the selected patch is a plane or not, the criterion shown below is used:

$$\varepsilon = \frac{\lambda_3}{\sum_{i=1}^{3}\lambda_i} < th_p \tag{3.2}$$

Where ε is the curvature of the selected patch. If the eigenvalues of a covariance matrix meet the condition (2), the corresponding patch is treated as a plane and added into the plane patch layer. A plane is usually represented by a vector $\pi = (\pi_1, \pi_2, \pi_3, \pi_4)^{\mathrm{T}}$. For any point $\mathbf{p} = (x, y, z)^{\mathbf{T}}$ on a plane, the following equation is true.

$$\pi_1 x + \pi_2 y + \pi_3 z + \pi_4 = 0 \tag{3.3}$$

Plane normal $\mathbf{n} = (\pi_1, \pi_2, \pi_3)^{\mathrm{T}}$ is equal to the eigenvector with minimum eigenvalue λ_3, and $\pi_4 = -\mathbf{n}^{\mathrm{T}}\overline{\mathbf{p}}$ represents the distance from origin to the plane. If $\|\mathbf{n}\| = 1$ and $\pi_4 \geqslant 0$, the plane representation is unique.

Assume the transformation from global frame to local frame is T, then we have:

$$\pi_l = \mathbf{T}^{-\mathrm{T}}\pi_g \tag{3.4}$$

where π_l and π_g are both vectors which represent planes.

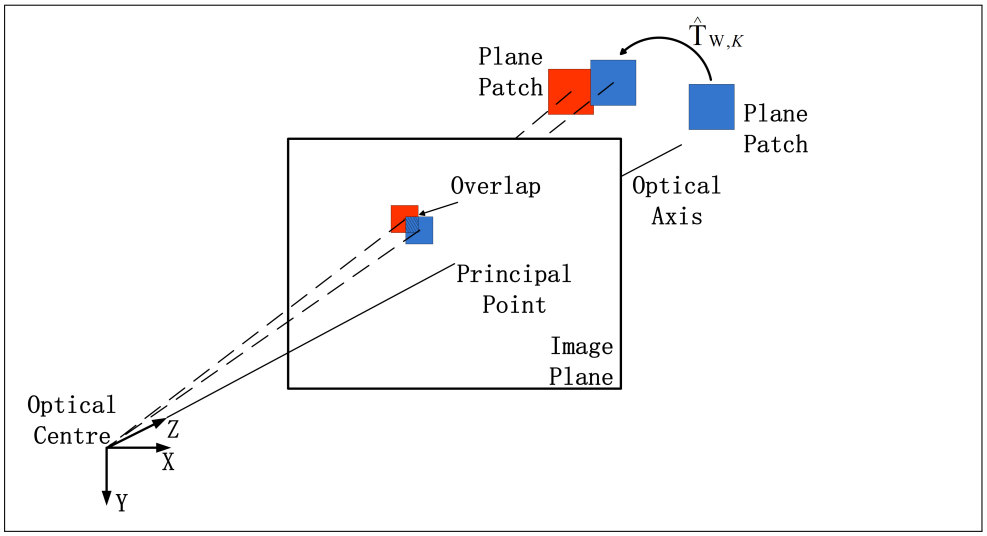


Figure 3.2: Plane patches matching approach. Plane patch in previous keyframe is transformed from global coordinate to current frame using pose prediction $\hat{\mathbf{T}}_{W,K}$, and the coplanarity criterion is checked first. Then both of the plane patch in current frame and the one predicted from previous keyframe are projected to the image plane in current frame to check the overlap criterion.

3.4.2 Plane Patches Matching

To match plane patches, two criteria—coplanarity criterion and overlap criterion—are used. The coplanarity criterion is to check if the normals and the distances from origin to plane between the pair of planes are the same. Here we propose to use the minimum representation of a plane for coplanarity criterion instead of the normal and the distance to reduce the computational complexity. The minimum representation of a plane is $\rho(\pi) = (\pi_1\pi_4, \pi_2\pi_4, \pi_3\pi_4)^{\mathrm{T}}$, which represents the intersection of a plane with the normal which passes through origin. The coplanarity can be evaluated by the Mahalanobis distance [113]:

$$d_c = (\rho(\pi_l) - \rho(\hat{\mathbf{T}}_k^{-\mathrm{T}}\pi_g))^{\mathrm{T}}\mathbf{Q}_c^{-1}(\rho(\pi_l) - \rho(\hat{\mathbf{T}}_k^{-\mathrm{T}}\pi_g)) \tag{3.5}$$

Where $\rho(\pi_l)$ is the minimum representation of plane π_l in local kth frame, $\rho(\pi_g)$ is the minimum representation of plane π_g in global frame. $\hat{\mathbf{T}}_k = \mathbf{T}_{k-1}\mathbf{T}_{k-2}^{-1}\mathbf{T}_{k-1}$ is the pose prediction transformed from the global coordinate to the kth frame, where $\mathbf{T}_{k-1}$ and $\mathbf{T}_{k-2}$ are the pose estimation of $(k-1)$th frame and $(k-2)$th frame, respectively. $\mathbf{Q}_c$ is related to the covariance matrix Σ_ρ representing the uncertainty of measured plane and the covariance matrix Σ_t representing the uncertainty of pose prediction. We take it as a constant here. So

the coplanarity criterion can be expressed by:

$$d_c \leqslant th_{co} \tag{3.6}$$

Assume d_c obeys χ^2 distribution, then the threshold th_{co} can be determined by a desired probability.

The overlap criterion is to check if there exists overlaps between two plane patches. As illustrated in Fig. 3.2, a plane patch in previous keyframe is first re-projected back into current frame. Then the center and size of plane patch are used to check the overlap based on the equation below:

$$d_o = (\frac{\mathbf{c}_l - \hat{\mathbf{c}}_{gl}}{a_l + \hat{a}_{gl}})^{\mathrm{T}} \mathbf{Q}_o^{-1} (\frac{\mathbf{c}_l - \hat{\mathbf{c}}_{gl}}{a_l + \hat{a}_{gl}}) \tag{3.7}$$

Where $\mathbf{c}_l$ and a_l are the center and size of the plane patch re-projected into the 2D image from local frame, and $\hat{\mathbf{c}}_{gl}$ and $\hat{a}_{gl}$ are the center and size of the plane transformed by the pose prediction $\hat{T}_k$ from global coordinate to local frame. $\mathbf{Q}_o$ is the covariance matrix related to Σ_c representing the uncertainty of center points and the covariance matrix Σ_t representing the uncertainty of pose prediction. We take it as a constant here. The overlap criterion is expressed as:

$$d_o \leqslant th_{ov} \tag{3.8}$$

Assuming d_o obeys χ^2 distribution, then the threshold th_{ov} can be determined by a desired probability.

3.5 Camera Pose Estimation with Points and Plane-patches

3.5.1 Feature Points Ranking

The feature points (SURF [104] [117] used here) are separated into three subsets. The components of the first subset are the called plane points which have small curvature. The remaining feature points are separated by their patch curvatures into smooth surface points indicating they are on smooth surfaces, but not on plane patches, and structural points representing edges or corners of some objects. They are shown in Fig. 3.3 with different colors. Following from (3.2), the following two criteria are used for smooth surface points (3.9) and structural points (3.10), respectively,

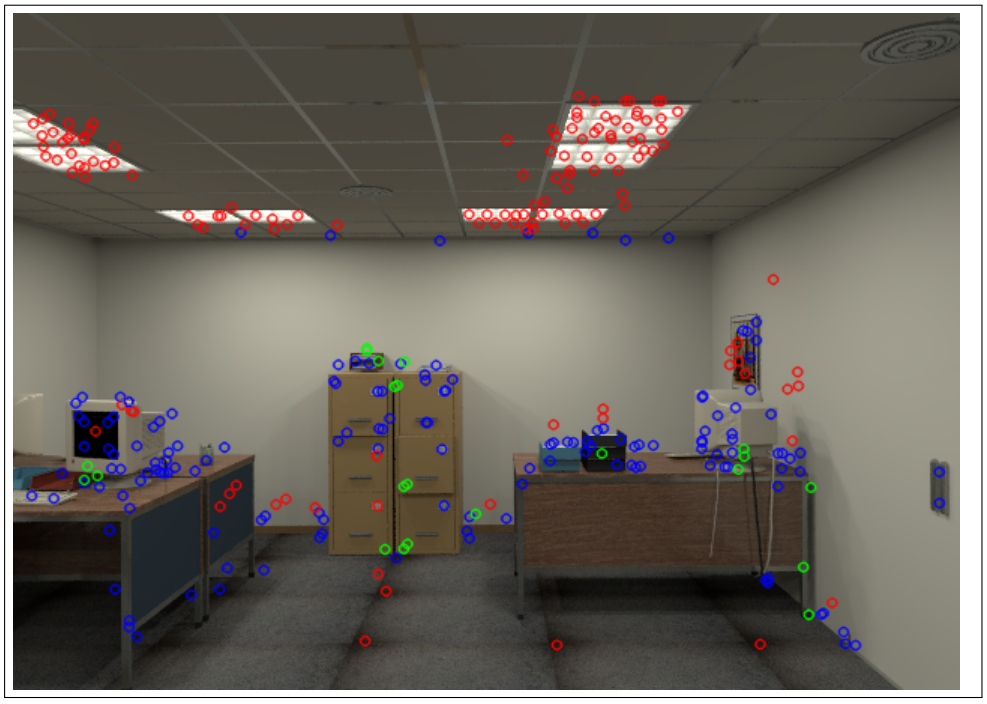


Figure 3.3: Different types of feature points. Red points, blue points and green points represent plane points, smooth surface points, and structural points respectively.

$$th_p < \varepsilon < th_s \tag{3.9}$$

$$\varepsilon > th_s \tag{3.10}$$

Each feature point is assigned with a weight. The weight parameter for a point should be inversely proportional to the uncertainty of the point or the curvature ε. Let the maximum curvature be ε_{max} and minimum curvature ε_{min}, the weight of feature points is defined below:

$$\begin{cases} \omega_i = \dfrac{\varepsilon_{min}(\varepsilon_{max} - \varepsilon)}{\varepsilon(\varepsilon_{max} - \varepsilon_{min})}, & plane\ point\ or\ surface\ point \\ \omega_i = \dfrac{1}{2}, & structual\ point \end{cases} \tag{3.11}$$

Each plane patch has a weight parameter ω_j which is equal to the weight parameter of its corresponding plane point.

3.5.2 RANSAC Registration using Points and Plane Patches

All feature points and plane patches are used to estimate transformation, with different weight parameters to enhance the robustness and accuracy of registration. Assume two matched sets of feature points using the Brute-Force matching method from OpenCV [117] from two frames are $\{\mathbf{p}_{gi}\}$ and $\{\mathbf{p}_{li}\}$, i = 1, ..., N. Let $\overline{\mathbf{p}}_g = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_{gi}$ and $\overline{\mathbf{p}}_l = \frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_{li}$. The matched plane patches are $\{\pi_{gj}\}$ and $\{\pi_{lj}\}$, j = 1, ..., M. To estimate the rotation R, we minimize the cost below:

$$\min(\sum_{i=1}^{N}\omega_i\|(\mathbf{p}_{li}-\overline{\mathbf{p}}_l)-\mathrm{R}(\mathbf{p}_{gi}-\overline{\mathbf{p}}_g)\|^2+\sum_{j=1}^{M}\omega_j\|\mathbf{n}_{lj}-\mathrm{R}\mathbf{n}_{gj}\|^2) \tag{3.12}$$

Where $\mathbf{n}_{gj}$ and $\mathbf{n}_{lj}$ are unit normal vector of π_{gj} and π_{lj}. To estimate the translation $\hat{\mathbf{t}}$, we minimize the cost below:

$$\min(\sum_{i=1}^{N}\omega_i\|\mathbf{t}-(\overline{\mathbf{p}}_l-\mathrm{R}\overline{\mathbf{p}}_g)\|^2+\sum_{j=1}^{M}\omega_j\|\mathbf{n}_{lj}^{\mathrm{T}}\mathbf{t}-(d_{lj}-d_{gj})\|^2) \tag{3.13}$$

Where d_{lj} and d_{gj} are π_4 component of plane patches π_{lj} and π_{gj}.

Random sample consensus (RANSAC) is very effective in distinguishing inliers and outliers. For point only SLAM, three non-collinear points have to be used as initial points. For our proposed approach, two plane points are randomly selected as initial features, then totally four features - two plane points and two plane patches are used to estimate transformation. The requirement for them is that the line passing through two feature points can not be parallel to both normals of two plane patches. If there is no enough features, one plane point and one feature point without plane patch can be used for pose estimation.

The penalty function to be minimised by RANSAC is

$$C=\sum_{i=1}^{N}D(\mathbf{p}_l,\mathbf{p}_g,\hat{\mathbf{T}})+\sum_{j=1}^{M}D(\rho_l,\rho_g,\hat{\mathbf{T}}) \tag{3.14}$$

where

$$D(\mathbf{p}_l,\mathbf{p}_g,\hat{\mathbf{T}})=\begin{cases}\|\mathbf{p}_l-\hat{\mathbf{T}}\mathbf{p}_g\|^2,\ \|\mathbf{p}_l-\hat{\mathbf{T}}\mathbf{p}_g\|^2<th_{ra1}\\ \mathrm{th}_{ra1}, \qquad \|\mathbf{p}_l-\hat{\mathbf{T}}\mathbf{p}_g\|^2\geq th_{ra1}\end{cases} \tag{3.15}$$

$$D(\rho_l, \rho_g, \hat{\mathbf{T}}) = \begin{cases} \|\rho_l - \hat{\mathbf{T}}\rho_g\|^2, \|\rho_l - \hat{\mathbf{T}}\rho_g\|^2 < th_{ra2} \\ \text{th}_{ra2}, \quad \|\rho_l - \hat{\mathbf{T}}\rho_g\|^2 \geq th_{ra2} \end{cases} \tag{3.16}$$

where ρ_l and ρ_g are the minimum representations of plane, th_{ra} is a threshold for the distance of matched features. If the distance is small than the threshold, they will be treated as inliers, otherwise, outliers.

3.6 Experimental Evaluation

The evaluation of proposed SLAM algorithm is conducted on benchmark datasets first. Then tests in our lab are presented, where the ground truth trajectory is provided from a motion capture system. Finally a large-scale indoor environment test is demonstrated. All the experiments are run on a laptop equipped with Intel Core i7-3610QM 2.3Hz CPU and 8G RAM. No GPU is used for computing.

3.6.1 Tests on Benchmark Datasets and in Our Lab

In this part, we first use the widely used open datasets from TUM [34] and ICL-NUIM [118] which provide color images and depth images along with ground truth all associated by timestamp. The ground truth is obtained by a motion capture system. Then we run our point-plane patch SLAM to further verify its effectiveness in our Robotic Arena equipped with a motion tracking system. Eventually we test our method on 16 benchmark datasets including different scenes such as office room, living room, large lab, desk and so on.

Absolute trajectory error (ATE) introduced in [34] measuring the distance between a estimated trajectory and a ground truth trajectory is used to evaluate our system. We compared our point-plane patch method with point-only method and another feature-based state-of-the-art RGB-D SLAM [100]. The ATE RMSE results using three methods are shown in Table 3.1. It can be seen from Table 3.1 that the RMSE using our point-plane patch method is much less than that using point-only method in most situations.

The plots of some trajectory differences are shown in Fig. 3.4 and the 3D point cloud maps constructed by our method are shown in Fig. 3.5. It can be seen that our SLAM algorithm can track camera positions in most scenes. The radius of our lab is about 5.5m and the height is about 8 m.

Table 3.1: Comparison of ATE RMSE between our method, point-only method and RGB-D SLAM on different kinds of benchmark datasets. The unit of RMSE is in meters.

Dataset	Point-plane patch	Point-only	RGB-D SLAM
fr1/desk	0.025	0.030	**0.023**
fr1/desk2	0.053	0.059	**0.043**
fr1/xyz	**0.014**	0.015	0.014
fr1/rpy	**0.022**	0.024	0.026
fr1/plant	**0.060**	0.074	0.091
fr2/desk	**0.047**	0.061	0.057
fr2/xyz	0.015	0.015	**0.008**
fr3/long	**0.031**	0.044	0.032
ICL-NUIM/living0	**0.021**	0.033	0.045
ICL-NUIM/living1	**0.011**	0.019	0.032
ICL-NUIM/office0	**0.022**	0.029	0.027
ICL-NUIM/office2	**0.018**	0.030	0.026
Our Arena lab/local	**0.038**	0.043	0.048
Our Arena lab/circle	**0.050**	0.058	0.056
Our Arena lab/wall	**0.037**	0.049	0.053
Our Arena lab/long	**0.062**	0.068	0.074

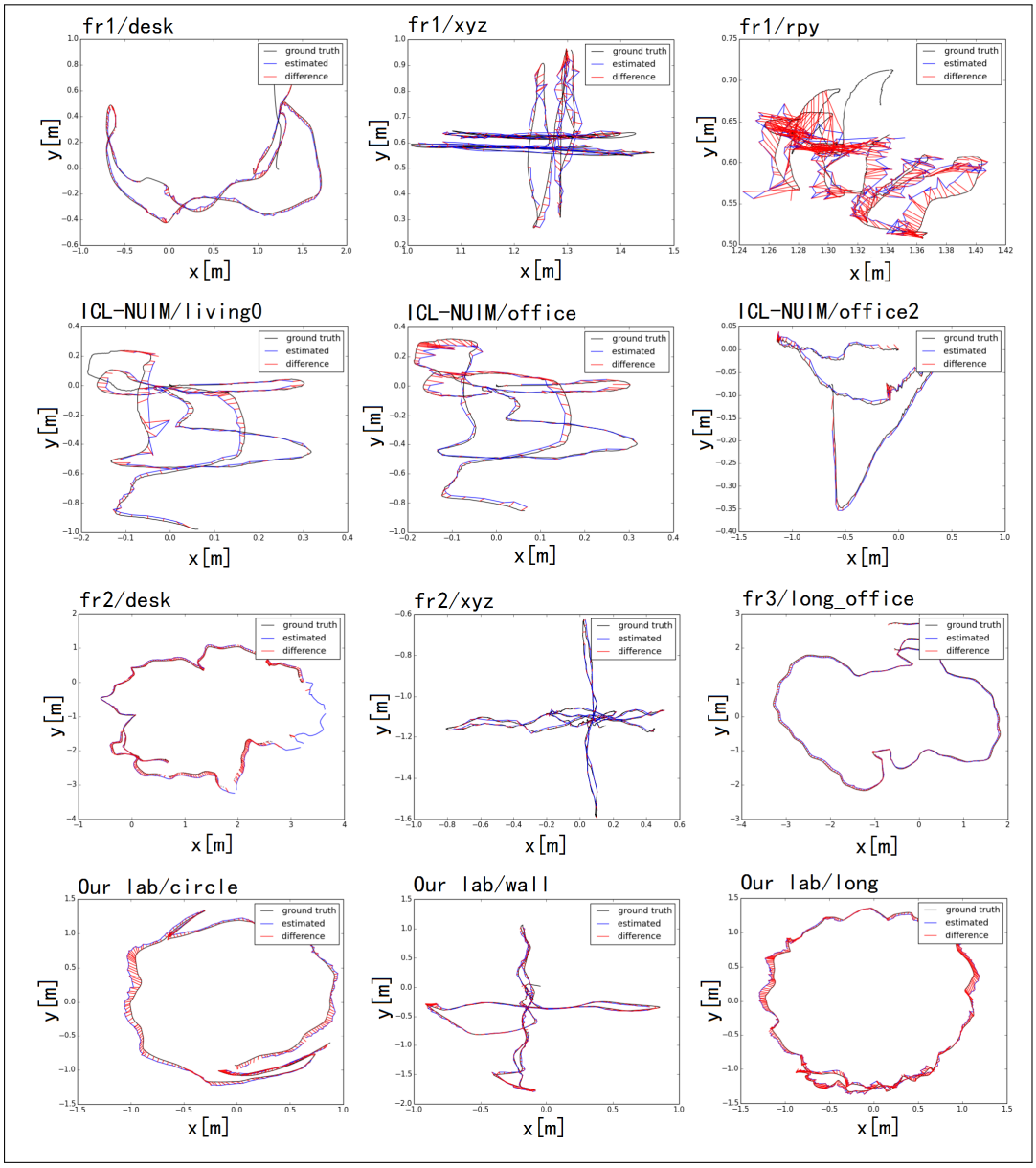


Figure 3.4: Comparison between estimated trajectory using our method and ground truth trajectory.

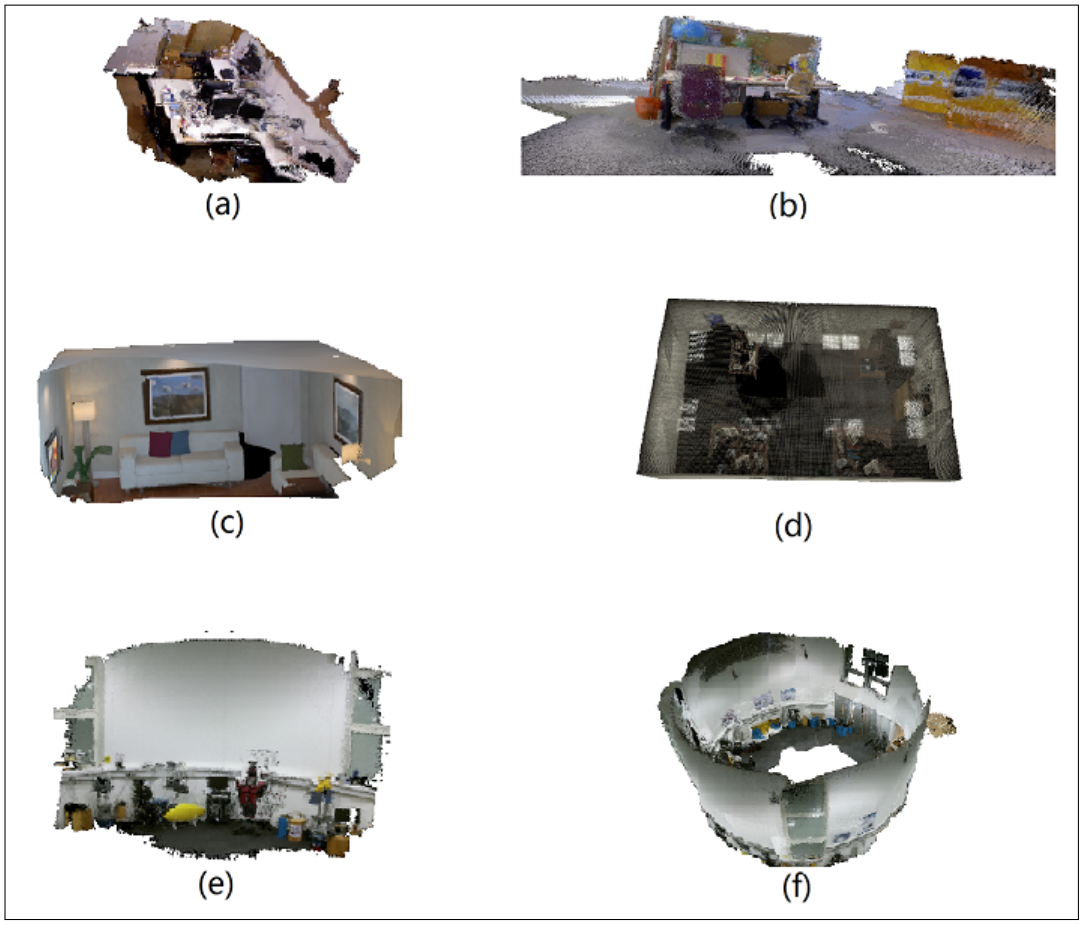


Figure 3.5: 3D map reconstructed from benchmark datasets using our method. (a) fr1/desk dataset. (b) fr3/long office dataset. (c) ICL-NUIM/living room 0 dataset. (d) ICL-NUIM/office room 2 dataset. (e) Our lab/wall. (f) Our lab/long.

Table 3.2: Average processing time for different procedures from 100 pairs of frames using ICL-NUIM office dataset.

Procedure	Time (ms)
Feature Points Extraction	35.03
Plane Patches Extraction	3.05
RANSAC Point-Plane Registration	7.45

Table 3.3: Feature and RANSAC feature inlier composition from 100 pairs of frames using ICL-NUIM office dataset.

Feature Type	Number	Inlier Number
Plane Point	157	50
Plane Patch	157	50
Curvature Point	176	42
Structural Point	33	5
Total	523	147

3.6.2 Computational Cost Analysis

Most SLAM applications have a high requirement for real-time performance. The time spent on macroscopic-visible plane extraction took about 50ms to 100 ms per frame. However, the time spent on plane patch extraction saved much time. In order to analyse the computational cost of our method, we randomly chose 100 pairs of frames from ICL-NUIM office dataset to record the processing time for different procedures. The average processing time computed from these pairs of frames is shown in Table 3.2. Here we used SURF as the point feature. The plane point curvature threshold th_p is 0.001, the curvature threshold th_s is 0.06, the RANSAC point inlier threshold th_{ra1} is 0.02^2 and the RANSAC plane inlier threshold th_{ra2} is 0.03^3. It can be seen from Table 3.2 that the average plane patch extraction time is 3.05 ms with about 366 feature points. Our SLAM system runs at about 10-15 Hz which depends

on the number of feature points and of loop closures.

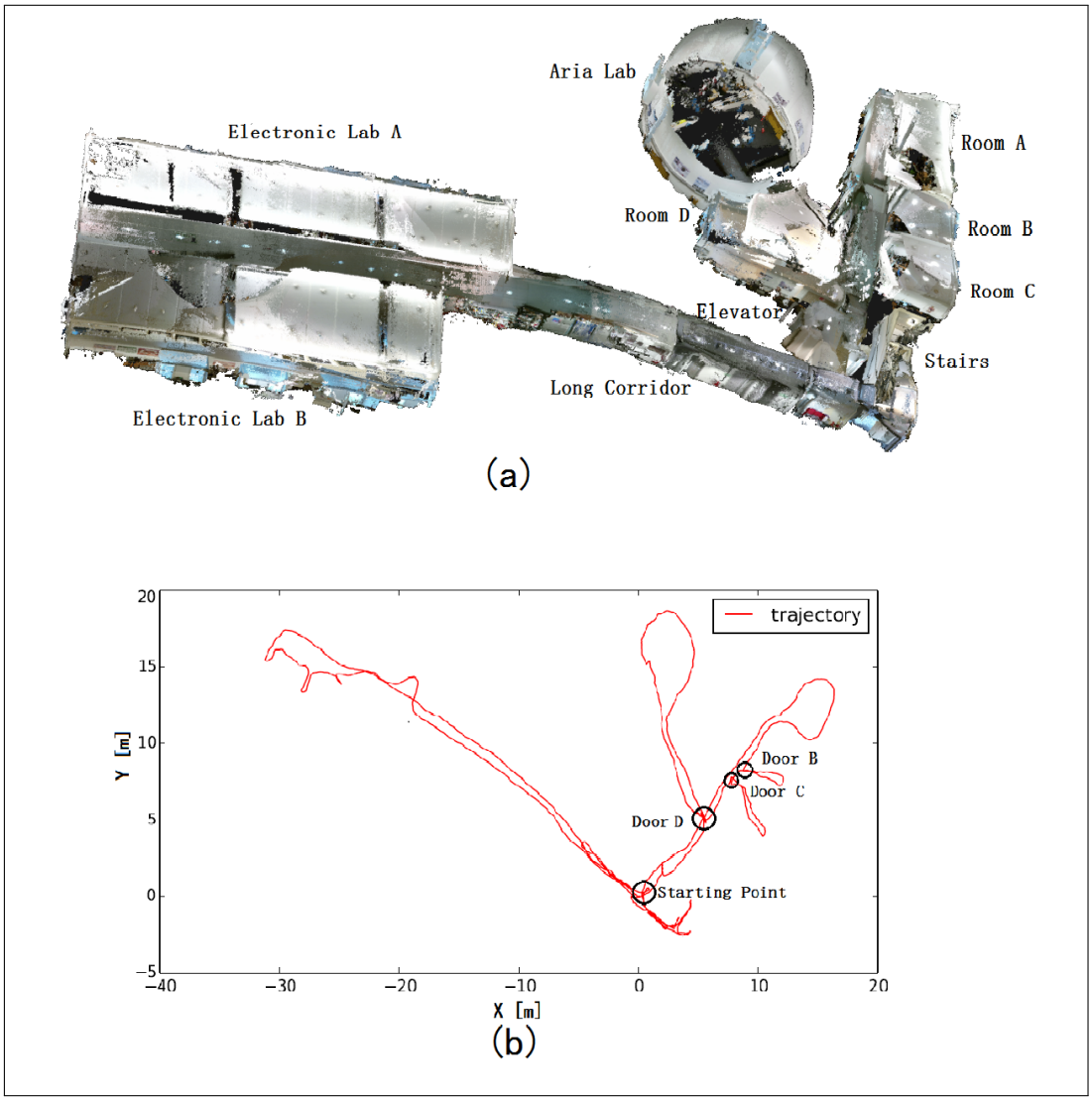


Figure 3.6: Trajectory and 3D map with the proposed SLAM algorithm in our building. (a) The 3D reconstructed dense map of one whole floor in our building. (b) The 2D trajectory while tracking.

From Table 3.3, it can be seen that the plane points with plane patches are more likely to be inliers while structural points are less likely to be inliers. It can mainly be explained by the depth image noise from RGB-D camera. The structural points have much more noise and larger covariance.

3.6.3 Test on a Large-scale Environment

In order to test the robustness of our method, we also ran our system in one large scale floor in our building that includes three large labs, four rooms, a long corridor, stairs and

an elevator. The frames were over 16000, the area of the floor was about 400 m^2 and the trajectory distance our platform moved was over 150 m. There were many challenge scenes, such as long corridor, pedestrians, featureless wall, high angular velocity movement, entering and exiting doors and so on. The reconstructed map is shown in Fig. 3.6(a). The trajectory accuracy can be evaluated when it passed the doors marked in Fig. 3.6(b). As the width of these doors is less than one meter, less than one third meter accuracy can be observed when passing the centers of the doors.

3.7 Conclusions

In this chapter, we present a novel point-plane patch SLAM algorithm which shows better performance in localization accuracy than point-only SLAM algorithms. This could be attributed to the usage of plane patches and different types of feature points. These "new features" are able to enhance the matching results and accordingly improve the accuracy of pose estimation. The time spent on extracting plane patches is not significant when comparing with macroscopic visible planes. The real time performance is still maintained at the same level with point only SLAM algorithms.

However, when facing with some challenging scenes, the proposed model-based method can hardly extract efficient features and work robustly. Deep learning can learn features automatically in an end-to-end manner and may provide some solutions. Next step, we would like to combine SLAM with Deep Learning to realize localization improve the robustness and accuracy performance in challenge situations.

Chapter 4

Indoor Relocalization in Challenging Environments with Dual-stream Convolutional Neural Networks

This chapter introduces supervised deep learning to solve the pose regression problem. An indoor relocalization system based on dual-stream CNN is proposed. A novel encoding method of depth images is presented for the proposed system. A training mechanism by stage for the dual-stream CNN is introduced. The proposed indoor relocalization system is evaluated on multiple public benchmark datasets and self-collected datasets. The results show that not only the relocalization accuracy is greatly enhanced, but also the system robustness is improved in challenging scenes such as large scale, dynamic, fast movement and night-time environments.

4.1 Introduction

Indoor relocalization is a challenging task which is widely studied in the areas of mobile robot navigation and computer vision. It enables robots the capacity to infer where they are in a previously visited place.

Appearance based methods are most popular for the relocalization task. Hand-crafted features such as Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF) or Oriented FAST and Rotated BRIEF (ORB) are extracted and stored first. Then im-

ages with similar appearance are retrieved using Bag of Words (BoWs) technology. Frame-to-frame feature correspondence and pose estimation [119] are implemented for localization at last. This method can achieve satisfactory precise and real-time performance when the view is consistent. Appearance based relocalization and loop closure detection are also widely used in visual Simultaneous Localization And Mapping (SLAM) [120] [121]. However, they have strict requirements for the environment that limit their applications in practice. They would no longer be useful when the appearance changes. Both moving objects and lighting variation could lead to appearance changes. [122] Motion blur caused by fast movements of the camera could also result in failures as manually designed features are fragile when encountering motion blur. In addition, accurate frame-to-frame pose estimation needs small viewpoint angle which limits the widespread usage of appearance based methods.

Deep Convolutional Neural Networks (CNNs) designed for image processing have achieved astonishing success in computer vision. Object recognition and detection capabilities are greatly improved due to the wide usage of CNNs [28] [123]. CNNs can learn different features according to various targets and provide an end-to-end solution to perception problems. As mentioned in section 2.4, recently 6-DOF camera pose regression method with CNNs (PoseNet) [40] was proposed. Unlike SLAM or appearance based relocalization, the pose regression with CNNs does not need to store keyframes, match features between frames and perform pose optimization. And the storage memory and computing time using CNNs do not increase when exploring large scale area. Therefore, it can implement large scale relocalization without area limitation. Furthermore, its performance in challenging situations could be improved as well. However, we find that PoseNet with only color images as the input could degrade the performance when working in some extremely challenging indoor environments.

In this chapter, we present a novel dual-stream CNN architecture with RGB-D camera which can implement indoor relocalization even in extremely challenging environments. Depth images are introduced in a separate stream to learn reliable range features. Range features from depth stream and appearance features from color stream are jointly optimized to implement the 6-DOF pose regression by learning the proposed CNN. Our main contributions in this chapter are summarized as follows:

- We present a dual-stream CNN to achieve indoor relocalization in challenging environments with an end-to-end manner. The CNN can learn localization features from both color and depth images, and estimate camera poses from these learned features. A network training strategy that divides the training into three stages is proposed. Approximately 20% improvement on localization accuracy is achieved compared with PoseNet.

- We study the encoding methods of depth images and propose a novel method called minimized normal + depth (MND) encoding to solve the 6-DOF pose regression problem. The MND encoding images contain both pixel orientation information and absolute depth information, and maintain the ability to leverage the transfer learning at the same time. Different network architectures with color and depth images as the input are discussed. By taking the depth information into the CNN, not only the relocalization accuracy is improved, but the system robustness in some challenging environments is enhanced.

- Robustness evaluation experiments are implemented in dynamic, night-time and fast movement environments. Experiments on a large scale indoor dataset and public datasets are also presented.

In the following section, we provide a review of the related work. In section 4.3, we give an introduction to the architecture of the proposed CNN, present the method for pose regression and the encoding method of depth images. Section 4.4 addresses three stages for network training with the dual-stream CNN. Section 4.5 demonstrates the experimental results on different datasets using the dual-stream CNN. In section 4.6, we give a summary conclusion and the future work we would like to investigate.

4.2 Related Work

Relocalization is a significant technology in robotics which could be used for search and rescue, navigation, intelligent services and so on. Place recognition is a problem related to relocalization. When compared with place recognition, relocalization needs to solve an additional geometric transformation problem. Relocalization could also be transferred to a loop closure detector in SLAM [124]. Although some work has been introduced in chapter 2, we

give a more detailed review of related works focusing on visual geometry-based relocalization, CNN-based relocalization and encoding methods of depth images in this section.

4.2.1 Visual Geometry Relocalization

In the early years, Iterative Closest Point (ICP) [125] algorithm was usually adopted for relocalization by registering local point clouds from frames to global point clouds from global map. Steder at el. [126] firstly extracted 3D features of images captured from RGB-D cameras, then place recognition and relocalization are implemented with 3D feature points registration using ICP. Except for feature points, high level features such as planes [127] and lines are also used for localization. Cupec at el. [113] extracted robust line segments and planar surface segments as primitive features instead of feature points to recognize places. The system performed robustly even in some appearance changed environments. A novel place recognition and scene registration method using multi-planes was addressed by Fernández-Moralc at el. [128]. High-level surface normal semantic planes, color information and other features are used. However, ICP algorithm and high-level primitives extraction are time consuming, especially in large scale environments.

In order to carry out scalable recognition within a short time, Nister at el. [129] introduced Bag of visual Words (BoWs) technology into object recognition and retrieval. Williams at el. [130] compared different loop closure detection (namely place recognition) approaches and found that image-to-image method with visual words scales best in large environment. Cummnins at el. [131] presented FAB-MAP with SURF point features as visual words to implement place recognition and relocalization in large scale environments. Afterwards they improved FAB-MAP and introduced it into SLAM [132]. Nevertheless, SURF features extraction is time consuming and limits their wide application in robotics.

Gálvez-López at el. [133] implemented faster place recognition using bag of binary words technology with Features from Accelerated Segment Test (FAST) and Binary Robust Independent Elementary Features (BRIEF). By encoding images into binary feature words and using fast extraction features, the system achieved satisfactory real-time performance in large scale environment. But both point features adopted are not rotation and scale invariant, leading to invalidation of the system in obvious appearance and viewpoint changed scenes. Based on [133], Mur-Artal et al. [17] proposed to use binary words consisting of ORB fea-

tures which can be computed in 10ms with rotation and scale invariant. They then introduced the proposed relocalization and loop closing technology into ORB-SLAM [6]. However, all features used are hand-crafted and extracted from color images which limits their application when they are applied to extremely challenging environments.

4.2.2 CNNs based Relocalization

Chen et al. [134] introduced CNNs combined with spatial and sequential checking into place recognition for the first time. By adopting robust features learned from CNNs, the system can conduct large scale place recognition and improve precise performance significantly compared with the state-of-the-art methods via hand-crafted features. In order to enhance the system robustness in challenging environments, such as severe appearance and viewpoint changes, Sunderhauf et al. [135] [136] analysed several widely used CNNs and proposed to leverage mid-layer features to cope with appearance variation and top-layer features to handle viewpoint variation. At the same time, by integrating locality-invariance hashing and semantic search space partitioning, the system achieves the real-time performance based on CNNs for the first time.

Regression forest method was introduced into indoor relocalization with Kinect by Shotton at el. [39]. The forest needs to be trained first, then 3D points in local frame could be matched to points within a global map with pre-trained forest which limits its applications in a small area. Precise 6-DOF camera position is calculated with geometry optimization at last.

Kendall et al. [40] proposed a novel convolutional neural network–PoseNet to perform relocalization in large outdoor environments with an end-to-end manner. PoseNet takes color images as the network input and achieves spectacular performance in relocalization. It particularly performs better in large scale and dynamic environments compared with traditional methods. This is the first time to solve the 6-DOF pose regression problem with CNN. The authors improved PoseNet and proposed Bayesian PoseNet [58] afterwards. By adding a Dropout layer into PoseNet and multiple randomized cropping input images, each generated position was modelled with uncertainty and the relocalization precision was improved. Hazirbas et al. [61] incorporated a spatial LSTM module into PoseNet to improve the performance. In order to estimate the uncertainty of pose estimation, Kendall et al. [58]

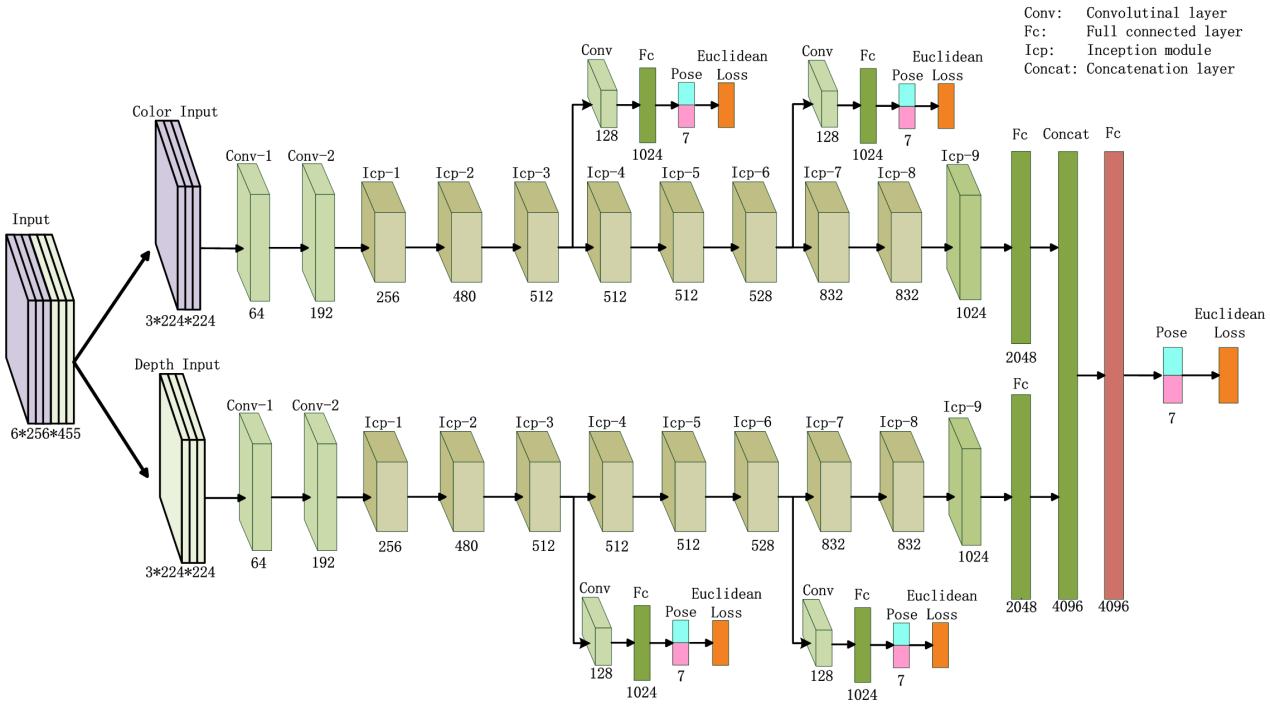


Figure 4.1: Architecture overview of the proposed dual-stream CNN for indoor relocalization. Color images and depth images are fed to the network separately as shown above. Each stream of the network outputs a place feature vector with the size of 2048. A concatenation layer and a fully connected layer are added to generate an information-rich feature vector with the size of 4096 which, finally converges to camera pose including position and attitude represented by a quaternion. The input for the color stream is a 3*224*224 RGB image, the input for the depth stream is a 3*244*244 post-processed depth image, the output of the network is a vector with dimension 7 which represents a camera pose.

proposed Bayesian PoseNet by adding a Dropout layer into the network as a means of sampling. He also proposed a fusion model to train the network and automatically adjust the weights of rotational and translational parts in [59]. Clark et al. [54] proposed to use a RCNN to implement the pose regression with video clips by taking the temporal information into consideration.

However all of the above methods have not paid attention to indoor complex environments and demonstrated satisfactory performance when encountering night-time, fast movement and dynamic environments.

4.2.3 Encoding Method of Depth Images for CNNs

CNNs which adopt the power of convolution to learn features from color images have demonstrated spectacular capability on object recognition and detection problems. For depth

images used in neural networks, preprocessing is necessary in order to achieve satisfying performance. Couprie at el. [137] introduced original depth information into CNNs to perform semantic segmentation for indoor scenes. Compared with color only images as the network input, they found it is better to use both color and depth images as the network input for segmentation problems. Aiming at object detection problems with RGB-D inputs, Gupta at el. [138] addressed a novel depth image encoding approach which takes horizontal disparity, height above ground and angle with gravity (HHA) as three image channels. Notice that HHA is computed from the pixels of object in image, which means that this method is not suitable for entire depth image encoding in 6-DOF pose regression. Normalized depth image which encodes scaled surface normal as three image channels is introduced by Lenz at el. [139] and Hinterstoisser at el. [140] to solve the detection problems. Nevertheless, normalized depth image encoding approach pays much attention to object structural contrast and loses the sight of original absolute range information. In order to achieve better performance in recognition problems with RGB-D cameras, a simple and efficient encoding method that colorizes depth images was proposed. Eitel at el. [141] transformed depth images from single channel to three channels by applying jet colormap. Schwarz at el. [142] rendered depth images with color palette and achieved the colorization of depth images. Experimental results demonstrate that normalized and colorized depth images seem to carry more information than original depth images and are more suitable as the network input. In this chapter, we propose a novel encoding approach that takes the advantage of normalized depth images and maintains original range information at the same time for the pose regression problem.

4.3 Preliminaries

In this part we will introduce the network architecture we used in this chapter first. Then we present the working mechanism of our dual-stream CNN for indoor relocalization. At last, a novel effective depth encoding approach for our dual-stream CNN is proposed.

4.3.1 Network Architecture

The proposed dual-stream CNN is designed to perform robust indoor relocalization in challenging environments such as motion blur, dynamic environments with mobile objects or pedestrians and dark scenes. As shown in Fig. 4.1, each stream is a separate CNN revised

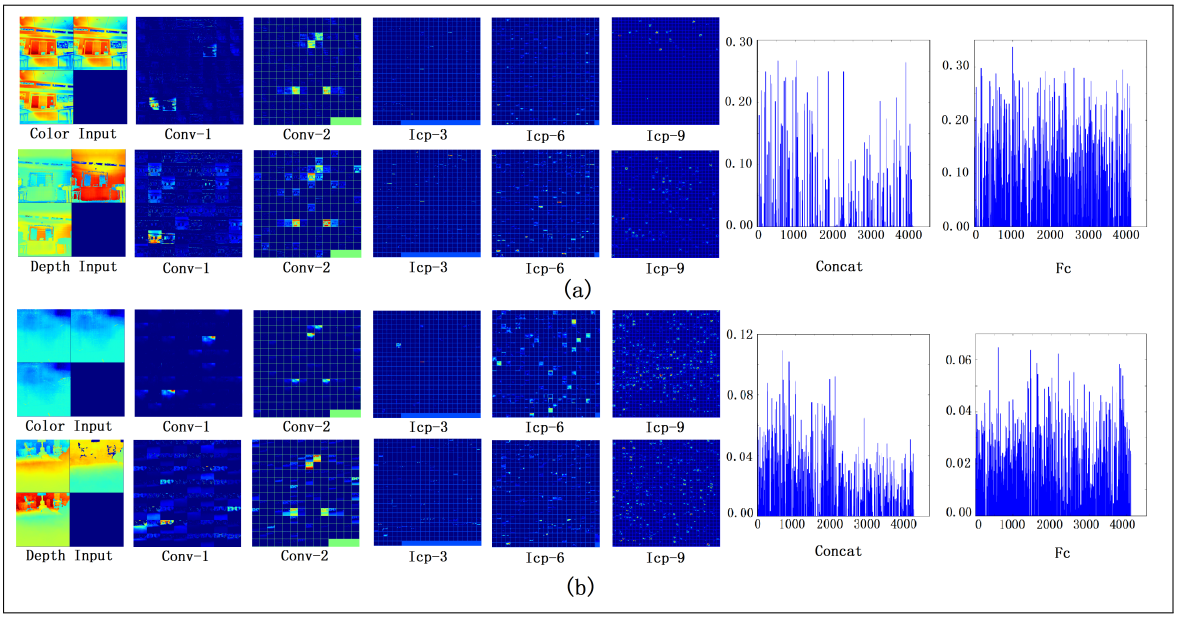


Figure 4.2: Selected network layers for different network inputs. (a) Images captured in normal situation as network inputs. Both color images and depth images are in good quality. (b) Images captured at night-time as network inputs. Color images are mostly black and can hardly be recognized even by our humans. Notice that all network weights we used here are the same and pre-trained from images in normal situations.

from GoogLeNet [123], which achieved the state-of-the-art performance in ImageNet [90] Large-Scale Visual Recognition Challenge 2014 (ILSVRC14) for object recognition and detection. We replace the Softmax layer in GooLeNet with a Euclidean layer.

In the dual-stream CNN, there are some inception modules which are improved from the modules introduced in Network in network [143]. By using inception modules, one can increase the width of the network without increasing the computational complexity. Compared with similar performance networks without inception modules, the networks with inception modules can obtain faster speed, or they outperform other networks with same depth [143]. Similar with the structure of GoogleNet [123], except for convolutional layers, fully connected layers and inception modules shown in schematics Fig. 4.1, there are also pooling layers, Rectified Linear Unit (ReLU) layers, Local Response Normalization (LRN) layers, dropout layers and Euclidean loss layers which we do not shown in the figure.

The input data composed of color images and depth images are sliced and fed to each stream respectively. In this way, dual-stream CNN can not only learn localization features from color images, but also can learn them from depth images which contain geometrical and structural information. A concatenation layer is used to put all features together and

another fully connected layer is added for better representations of localization features.

4.3.2 Dual-stream CNN for Pose Regression

For traditional image-to-map [125] or image-to-image [130] pose registration in relocalization, hand-crafted point features should be extracted and matched first, then the transformation will be estimated by minimizing the cost function below along with Random Sample Consensus (RANSAC) to remove feature outliers.

$$min \sum_{i=1}^{n} W_i \left\| \mathbf{X}_i - \mathbf{T}\mathbf{X}_i{}' \right\|^2 \tag{4.1}$$

Where $\mathbf{T}$ is the 4×4 transformation matrix containing rotation $\mathbf{R}$ and translation $\mathbf{t}$. $\mathbf{X}_i = (x_i, y_i, z_i, 1)^\mathsf{T}$ is the homogeneous position representation of feature point. $\mathbf{X}_i{}' = (x_i', y_i', z_i', 1)^\mathsf{T}$ is the matched point of $\mathbf{X}_i$ in appearance similar image or global map. W_i is the weight of corresponding point. Feature extraction and matching play a crucial role in place recognition and pose estimation especially in challenging situations. Both mismatching and failures in robust feature extraction can result in system failure.

Compared with CNNs designed for object recognition problems, the CNNs for 6-DOF pose regression use a Euclidean loss layer as the top layer instead of softmax classifier layers. Euclidean loss drives the position learning by comparing the network output to the labelled 6-DOF pose and minimizing the least-squared cost. The Euclidean loss E is computed in the network as shown below:

$$E = \frac{1}{2N} \sum_{n=1}^{N} \|\hat{x}_n - x_n\|_2^2 \tag{4.2}$$

Where x_n is the labelled (ground truth) vector corresponding to the input image, $\hat{x}_n$ is the estimated vector produced by the CNN in an end-to-end manner, and N is the number of vectors that the CNN produced.

In the dual-stream CNN for pose regression, camera pose is a labelled vector composed of position $\mathbf{p} = (p_x, p_y, p_z)^\mathsf{T}$ and orientation represented by unit quaternion $\mathbf{q} = (q_a, q_b, q_c, q_d)^\mathsf{T}$. Therefore the Euclidean loss $E_\mathbf{p}$ is computed as below:

$$E = E_{\mathbf{p}} + \lambda E_{\mathbf{q}} \tag{4.3}$$

Where $E_{\mathbf{p}}$ is the positional Euclidean loss, $E_{\mathbf{q}}$ is the orientational Euclidean loss and λ is the balancing weight between these two parts. For the reason that orientation represented by unit quaternion and position measured in meters have obvious different measurement units, a weight λ is introduced to balance the costs $E_{\mathbf{p}}$ and $E_{\mathbf{q}}$ in order to achieve favorable results. [40]

Assume that each image has real robust feature representations $\{y_1, y_2, ..., y_n\}$ to be learned. $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$ are feature representations our CNN has learned. With an inner product layer $x = \sum_{i=1}^{4096}(\mathbf{W}_i * y_i + b_i)$ as the input of Euclidean layer, where the $*$ is the matrix multiplication operation. The feature representation and corresponding weights can be learned by minimizing the cost below:

$$min \sum_{i=1}^{4096} W_i \|\hat{y}_i - y_i\|^2 \tag{4.4}$$

From the cost above, we can see that the dual-stream CNN will try to learn the real localization features and automatically weight them. This demonstrates a great advantage over the methods using hand-crafted features.

The fully connected layer we added after the concatenation layer in our dual-stream CNN plays an active role in balancing the weights of learned features between color inputs and range inputs. As shown in Fig. 4.2, the concatenation layer connects 2048 color features learned from color images and 2048 range features from depth images separately. When we use the images captured from normal situations, the mean value of texture features from color images is about 0.20, and the mean value of range features is about 0.10. By adding fully connected layers and re-weighting all features, the mean value of range features is increased from 0.10 to 0.20. This improves the role of features from depth image as shown in Fig. 4.2(a). For night-time images, all the color images are mostly black and fully of noise. However, their depth images still contain abundant structural range information. As shown in Icp-6 and Icp-9 in Fig. 4.2(b), the number of bright areas in the color stream are obvious more than that in the depth stream. This represents that the features learned from the color images are more noticeable than the features learned from the depth images. In this way,

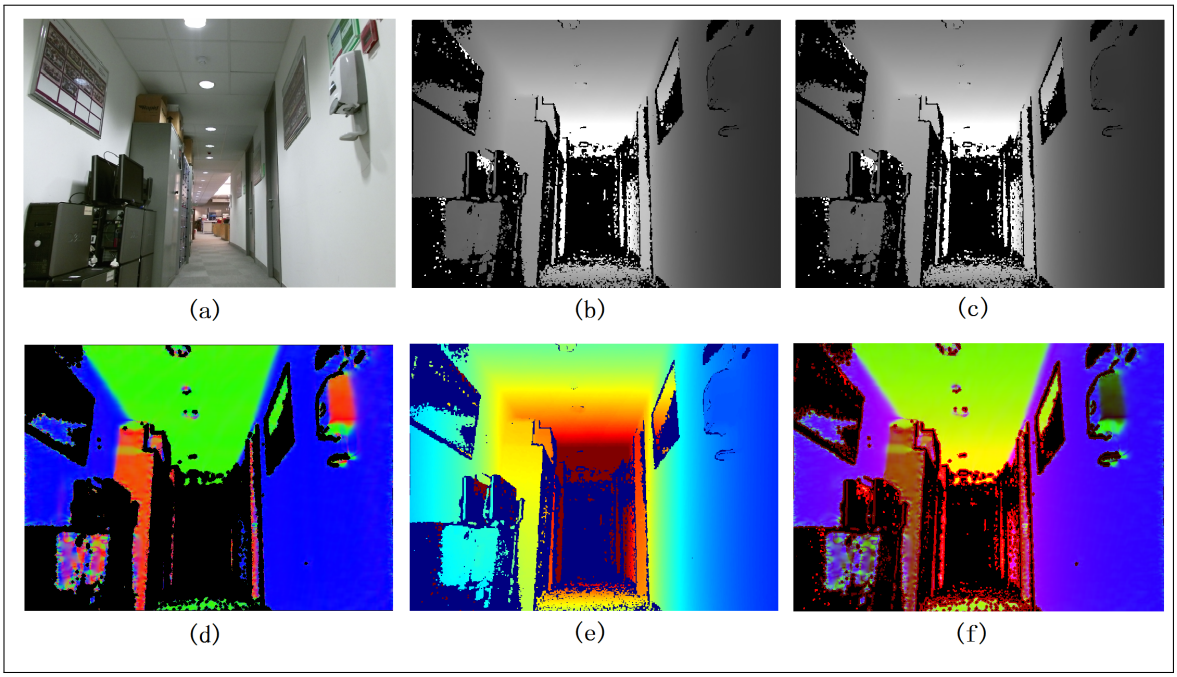


Figure 4.3: Different encoding methods of depth images. (a) Color image. (b) Single layer scaled depth image. (c) Triple-layer scaled depth image. (d) Normalized depth image with computed normal parameters as three image channels. (e) Colorized image using jet color map. (f) Minimized normal+depth(MND) image.

as shown in the concatenation layer in Fig. 4.2(b), the mean value of texture features from color images is about 0.08 and that from depth images is about 0.04, which has bad impact for relocalization performance and makes the results uncertain. Fortunately, the fully connected layer can re-balance features and decrease the mean value of color texture features from 0.08 to 0.04 which reduces their weights. The network is smart enough to weight different features according to different situations, i.e. it strengthens the weights of texture features in normal situations and weakens the weights of structural features in challenging situations such as night-time. In this way, by combining the texture features from color images and the structural range features from depth images and weighting them automatically, the dual-stream CNN can cope with many extremely challenging environments.

4.3.3 Encoding Methods for Depth Images

In the dual-stream CNN, depth images from RGB-D cameras are introduced for a separate stream CNN to learn structural range features in order to enhance the system performance. However, the convolutional layers in CNNs are particularly designed for color images in which pixel channels mainly represent light intensity. In contrast, the pixel value of depth

images represents scaled distance between the optic centre of camera and objects in the environment. We can hardly achieve satisfactory performance with raw depth images used as the inputs of CNNs directly. Necessary preprocessing for depth images ought to be taken before feeding them into CNNs.

The simplest preprocessing method for depth images is to rescale pixel values from 0–10000 to 0–255. The processing result of this method is the single layer grayscale image (one channel) that is shown in Fig. 5.3(b). To leverage the transfer learning from the network weights pretrained on ImageNet or Places [144], we can also duplicate the grayscale image shown in Fig. 5.3(b) and create three copies of the depth layer. Then triple-layer scaled depth image (three channels) could be obtained that is shown in Fig. 5.3(c). As shown above, the appearance of processed images between these two methods has no significant difference. Another famous encoding method is to use surface normals as three channels of images [139] [140]. The unit surface normals $[n_x, n_y, n_z]$ should be computed from depth image first, then they are rescaled from 0–1 to 0–255. The normalized image is shown in Fig. 5.3(d). Colorization of depth images is also an easy but efficient way for depth encoding [141] [142]. Each pixel intensity value in depth image corresponds to three channel values (red, green and blue separately). Fig. 5.3(e) is a colorized jet map, as the intensity value increases from 0 to 255, the color changes from blue to green, then yellow to red.

From Fig. 5.3(d), we can see that all pixels on the wall have the same color even though they have different depth values. On the contrary, the wall pixels with discriminative depth hierarchies in colorized image are labelled with different colors as shown in Fig. 5.3(e). However, the box on the right part of Fig. 5.3(e) shows little difference on structure and the box in Fig. 5.3(d) is easy to distinguish. In a word, the normalized images pay more attention to relative structural information, so the objects in the normalized images could be easily distinguished. Nevertheless, the normalized images do not contain absolute range information, which is contained in original depth images and colorized images.

In this chapter, we propose to use the minimized normal and depth images (MND) to encode depth images. The MND uses rescaled $n_x{}'$, $n_y{}'$ and depth d as three channels of image separately. For scaled surface normal $[n_x{}', n_y{}', n_z{}']$, we have $n_x{}'^2 + n_y{}'^2 + n_z{}'^2 = 255^2$. Therefore normalized depth images can be represented by $n_x{}'$ and $n_y{}'$ which is called minimized normal representation. The third channel we use in our approach is the scaled original depth

value. As shown in Fig. 5.3(f), the pixels of wall with similar normal and the pixels of box with similar depth both have more obvious local contrast when using our MND encoding method. The advantage of this method is that processed depth images retain both relative structural information and absolute range information and the networks maintain the ability to leverage the transfer learning at the same time.

Another problem we need to consider when using depth images for CNNs is the input size. The dual-stream CNN we use in this chapter requires an input image with a size of 224×224. The raw depth image captured from Kinect One is 960×540 and that captured from Kinect 360 or Xtion is 640×480. In our system, original images are resized to 455×256 or 341×256, and then randomly cropped to a size of 224×224. We have also tried to resize original depth images to 224×224 directly and maintain all information from depth, but found that the relocalization performance was not as good as expected.

In addition, invalid depth data is inevitable when using RGB-D sensors (Kinect v1, Kinect v2, Xtion and et al.). We briefly treat invalid depth data with zeros and let the network learn to handle this data. There could be some more elegant ways to explore for invalid depth data, such as using spatial or temporal interpolation techniques. This is left for our future research.

4.4 Training Mechanism for Dual-stream CNN

The selection of training mechanism is very important for system performance. For traditional CNNs, we can train them in an end-to-end way and optimize the network weights directly. For our dual-stream CNN for relocalization, traditional training methods can hardly ensure the convergence of the dual-stream CNN. In this work, the dual-stream CNN training for relocalization is divided into three stages.

4.4.1 Training Separate Streams

The first training stage for our dual-stream CNN is a separate stream training stage. Different from the network architecture shown in Fig. 4.1, the network in this stage does not have the concatenation layer and the additional fully connected layer. The stream with color images as inputs is called color stream, and the stream with depth images as inputs is called depth stream. The Euclidean Loss layer is appended to the end of both color stream and depth

stream. Therefore, the network in this stage has two pose outputs, one is produced by 2048 textural features from the stream with color images as inputs, another one is generated by 2048 range features from the stream with depth images as inputs. The network here has one input and two outputs and the two streams can be trained simultaneously. Both streams take transfer learning from the same network weights pretrained on the dataset Places [144] used for training place classifiers. The weights of all the layers are trained. Transfer learning focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. Although the purpose of two channels are different, both streams in our network can converge in a short time with desirable performance by using transfer learning. Different Euclidean balancing weights are adopted for the color stream and the depth stream while training. For color stream, we use 1 for positional Euclidean loss weight and λ for orientational Euclidean loss weight. Weights for depth stream are 1.2 times the weights for color stream, i.e. the positional Euclidean weight is 1.2 and the orientational Euclidean weight is 1.2λ. The depth stream is endowed with heavier weights for the reason that depth images are not affected by light intensity or motion blur and are more robust than color images to some extent.

4.4.2 Fine-tuning Full Connected Layer

After the separate stream training stage, the network can learn many preliminary position features from depth images and color images respectively. In this stage, we recovery the network architecture as shown in Fig. 4.1. The concatenation layer and fully connected layer are brought back to the network to re-weight preliminary features learned from two streams.

The network weights obtained from stage one are used for transfer learning. We only perform fine tuning operation on the fully connected layer while the weights of other layers are kept invariant. This is done by setting all learning rates to zeros in the network except for the last fully connected layer. By fine tuning the fully connected layer, the relocalization performance of the system is improved greatly compared with single color stream or single depth stream. Additionally, for the reason that only the fully connected layer is fine tuned, the number of iteration in this stage is only about 10000 to 15000 which means a short time is required.

Table 4.1: Relocalization performance comparison with different depth image encoding methods using PoseNet. The unit for position error is meter (m), and the unit for orientation error is degree (°).

Input	Median position error	Median orientation error
RGB	0.55m	5.13°
Single depth	0.57m	3.83°
Triple-depth	0.52m	3.36°
Normalized depth	0.49m	3.02°
Colorized depth	0.48m	3.10°
MND	**0.46m**	**2.85**°

4.4.3 Fine tuning overall Dual-stream CNN

In order to achieve the best performance for indoor relocalization, we fine tune the overall dual-stream CNN in the final stage. The weights of all network layers are trained in this stage. The network pretrained in the second stage is taken for transfer learning in the third stage. All learning rates set to zeros in the second training stage are set back to the same value in the first training stage. Base learning rate should be smaller than that in the first two stages, which is 0.8 times in this chapter. This stage needs about 20000 iterations to converge. Position features and their weights are adjusted slightly. The system performance is further improved compared with that in the second stage.

4.5 Experimental Evaluation

In this section, we evaluate the relocalization performance of our proposed dual-stream CNN. We first compare different depth encoding methods and select the best one for 6-DOF pose regression with CNN. Then the architectures to take the advantage of both color images and depth images are discussed. A large scale relocalization experiment is presented. Following that, the quantitative experiments based on Microsoft 7-scenes benchmark dataset is given by comparing with PoseNet [40]. At last, the experiments on extremely challenging situations

with our system is presented. The dual-stream CNN is designed using Caffe [145], and all experiments are performed on a desktop equipped with Nvidia GeForce Titan X GPU card and Intel Core i7-4790 4.0GHz CPU.

4.5.1 Range Images Encoding Methods

In this part, we compare the proposed MND depth encoding method with other popular encoding methods. The dataset called Second Floor here is collected in the second floor of our network building using Kinect One. The scale of the second floor is about $40m \times 30m$. For the reason that we do not have motion capture system covering the whole floor, we use the state-of-the-art SLAM algorithm–ORB SLAM [6] to label collected images. The labelled result is also used as the groundtruth. The training dataset contains about 4300 frames and the test dataset contains about 5460 frames.

For the network training, we only perform the separate stream training in the process and take the relocalization results from color stream and depth stream, respectively. The Stochastic Gradient Descent (SGD) is adopted as the training solver. Learning rate policy is STEP with 80 epochs. The base learning rate is 0.00001, the gamma is 0.94 and the momentum is 0.9. Both streams take transfer learning and the number of training iteration is about 30000. For color stream, the positional weight is 1 and the orientational weight λ is 250. For depth stream, the positional weight is 1.2 and the orientational weight λ is 300. We found that it is important to give the depth stream a little bigger weight to make our system robust.

The relocalization results from the depth stream with different encoding depth images and color stream are shown in Table. 4.1. We can see from the table that the depth stream performs better than the color stream in orientation. Normalized depth images and colorized depth images have no significant difference in relocalization as the network inputs, but both are better than single depth and triple-depth. From our experiments, the normalized encoding performs better in position and the colorized encoding performs better in orientation. Our proposed MND method which includes both structural and range information achieves the best performance in indoor relocalization since the MND has the most significant local contrast among all the depth encoding methods.

Table 4.2: Results of different network architectures with RGB-D images as inputs. The first row in the table shows the result of PoseNet [40] with color image (3 layers) as input. The second row shows the result of PoseNet with color image and single scaled depth image (4 layers) as input. The third one shows the result of PoseNet with color image and colorized depth image (6 layers) as input. The forth row and the fifth row both use the dual-stream CNN for indoor relocalization. The difference is that last two fully connected layers are fine tuned for the third row and the whole network is fine tuned for the forth row.

Network architecture	Scale	Train/ Test	Input	Median position error	Median orientation error
PoseNet	40×30×5m	4300/ 5460	RGB (3)	0.55m	5.13°
PoseNet	40×30×5m	4300/ 5460	RGB+single-depth (4)	0.42m	2.77°
PoseNet	40×30×5m	4300/ 5460	RGB+MND depth (6)	0.68m	3.51°
Dual-stream CNN (Fine tuning fully connected layers)	40×30×5m	4300/ 5460	RGB-MND depth (3+3)	0.33m	1.83°
Dual-stream CNN (Fine tuning whole network)	40×30×5m	4300/ 5460	RGB-MND depth (3+3)	**0.27m**	**1.65°**

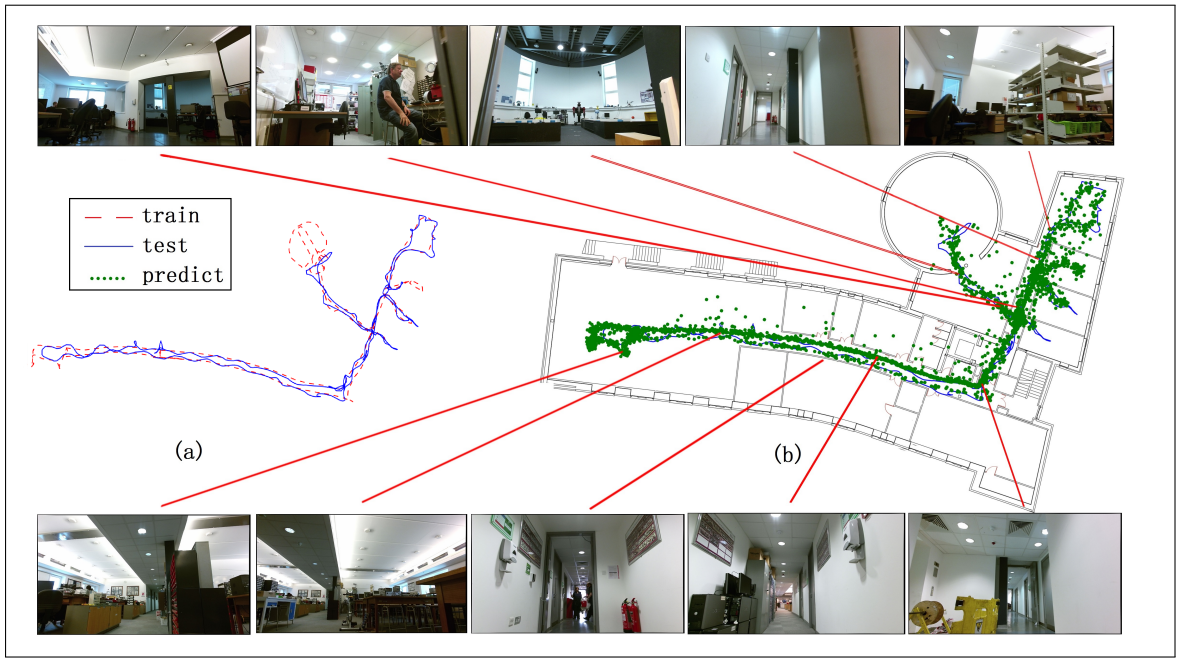


Figure 4.4: Predicted trajectory using our method in our network building. The red trajectory (dashed line) in (a) is used for training, the blue one (solid line) in (a) and (b) is the ground truth of test dataset. The green one (dots) in (b) is the pose predicted by the dual-stream CNN.

4.5.2 Network Architecture for RGB-D Images

After selecting the MND encoding as our depth preprocessing method, we will further discuss the network architecture taking both color images and depth images as input. The same dataset (Second Floor) as used above, is used here. As shown in Table 4.2, four different CNNs are presented with RGB-D images as inputs. We first feed color+single-depth images (4 layers as inputs) and color+MND depth (6 layers as inputs) to PoseNet, respectively. The results show that PoseNet with color+single-depth as inputs performs better in relocalozation than that with color+MND depth. So we know that the localization performance of CNNs can not be improved with the network width if we can not find the right way. Then the dual-stream CNN taking color and MND depth as inputs is used to perform indoor relocalization. The performance of the dual-stream CNN is much better than PoseNet due to the mix of color and depth images. As mentioned above, we divide the network training into three stages. After fine tuning the fully connected layer, we also fine tune the whole network in the third training stage. This step further improves the relocalization performance in both position and orientation. In addition, we also try to concatenate the fully connected layers after Icp-3

Table 4.3: Quantitative analysis. Comparison with PoseNet and Bayesian PoseNet based on the public 7-Scenes dataset downloaded from Microsoft Research. Here the median relocalization error is used to represent the system performance. The scale for Fire, Heads, Chess, Pumpkin, Office, Redktichen, Stairs dataset is 2.5×1×1m, 2×1×0.5m, 3×2×1m, 2.5×2×1m, 2.5×2×1.5m, 4×3×1.5m, 2.5×2×1.5m and 2.7×1.8×1.1m respectively. The training/test numbers for above 7-scenes dataset are 2000/2000, 1000/1000, 4000/2000, 4000/2000, 6000/4000,7000/5000 and 2000/1000 respectively.

Dataset	PoseNet (RGB-D)	PoseNet (RGB+MND)	PoseNet (RGB)	PoseNet (MND)	Dual-stream (RGB-MND)	Bayesian PoseNet (RGB)	Bayesian PoseNet (MND)	Bayesian Dual-stream (RGB-MND)
Fire	0.56m, 13.57°	0.58m, 17.06°	0.52m, **12.54**°	0.55m, 16.76°	**0.51m**, 12.88°	**0.42m**, 12.65°	0.46m, 16.83°	0.43m, **12.52**°
Heads	0.39m, 15.25°	0.33m, 15.10°	0.38m, 13.46°	0.34m, 14.77°	**0.30m, 12.73**°	0.27m, 13.06°	0.26m, 14.77°	**0.25m, 12.72**°
Chess	**0.36m, 6.91**°	0.38m, 7.65°	0.39m, 8.05°	0.45m, 9.68°	**0.36m**, , 7.79°	0.29m, 7.33°	0.32m, 9.02°	**0.28m, 7.05**°
Pumpkin	0.51m, 8.43°	0.48m, **8.16**°	0.58m, 9.20°	0.52m, 8.68°	**0.45m**, 8.30°	0.49m, 8.57°	0.37m, 8.19°	**0.36m, 7.53**°
Office	0.63m, 12.56°	0.61m, 13.22°	0.56m, **9.39**°	0.54m, 12.54°	**0.48m**, 9.68°	0.38m, **8.73**°	0.36m, 11.72°	**0.30m**, 8.92°
Redkitchen	0.94m, 18.21°	0.73m, 13.30°	0.87m, 11.40°	0.63m, 12.46°	**0.58m, 10.49**°	0.75m, 10.62°	0.51m, 11.65°	**0.45m, 9.80**°
Stairs	0.53m, **11.94**°	0.63m, 13.79°	0.54m, 13.71°	0.63m, 14.40°	**0.48m**, 13.21°	0.47m, 13.71°	0.59m, 14.01°	**0.42m, 13.06**°
Average	0.56m, 12.41°	0.53m, 12.61°	0.55m, 11.10°	0.52m, 12.75°	**0.45m, 10.72**°	0.44m, 10.67°	0.41m, 12.31°	**0.35m, 10.22**°

(regression 1) and the fully connected layers after Icp-6 (regression 2) respectively, and fine tune the new concatenated networks. The relocalization precision for these two concatenated networks is 1.19m, 5.76°and 1.07m, 4.04°, respectively. So concatenating the final features is optimal in our network.

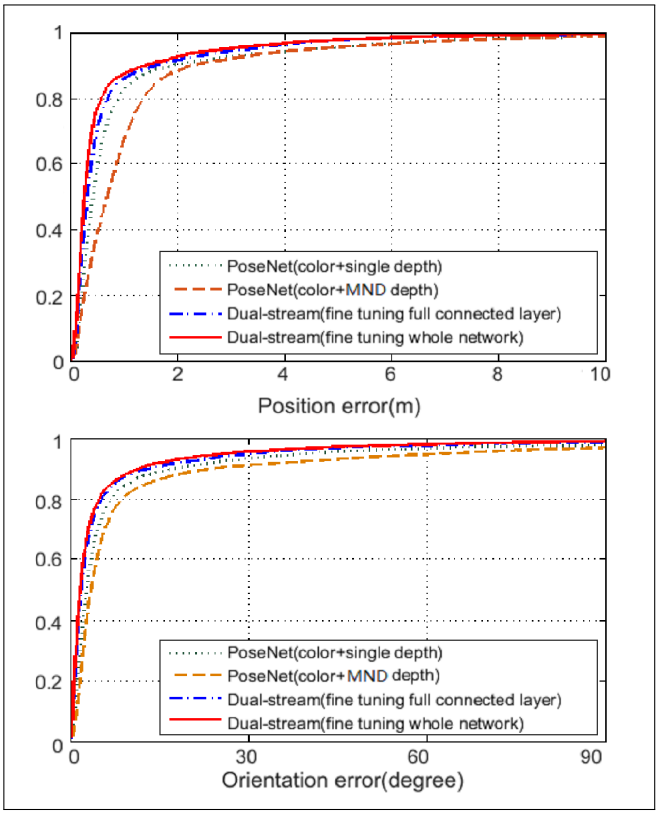


Figure 4.5: Cumulative histograms of relocalization error with different network architectures for RGB-D images as input.

The cumulative histograms of indoor relocalization error produced by the above four approaches are plotted in Fig. 4.5. The dual-stream CNN after fine tuning the whole network performs best among them in all the aspects.

2D relocalization trajectory with Second Floor dataset is shown in Fig. 4.4. Fig. 4.4(a) shows the groundtruth trajectory of the training dataset and the test dataset. The dashed line represents the trajectory of training dataset. The solid line represents the goundtruth trajectory of test dataset. Fig. 4.4(b) shows the groundtruth trajectory and predicted poses of the test dataset, and we put them in the 2D floor plan of our building in order to show them clearly. Some images of the scenes are given. The dots represent the predicted positions from our dual-stream CNN. As shown in Fig. 4.4, most predicted camera poses are almost

the same with the groundtruth. However, there are also a few predicted poses which have considerable error. This could be improved by extending our network to a Bayesian dual-stream CNN, which can model the uncertainty of poses and remove noisy data points, but the Bayesian dual-stream will cost more time.

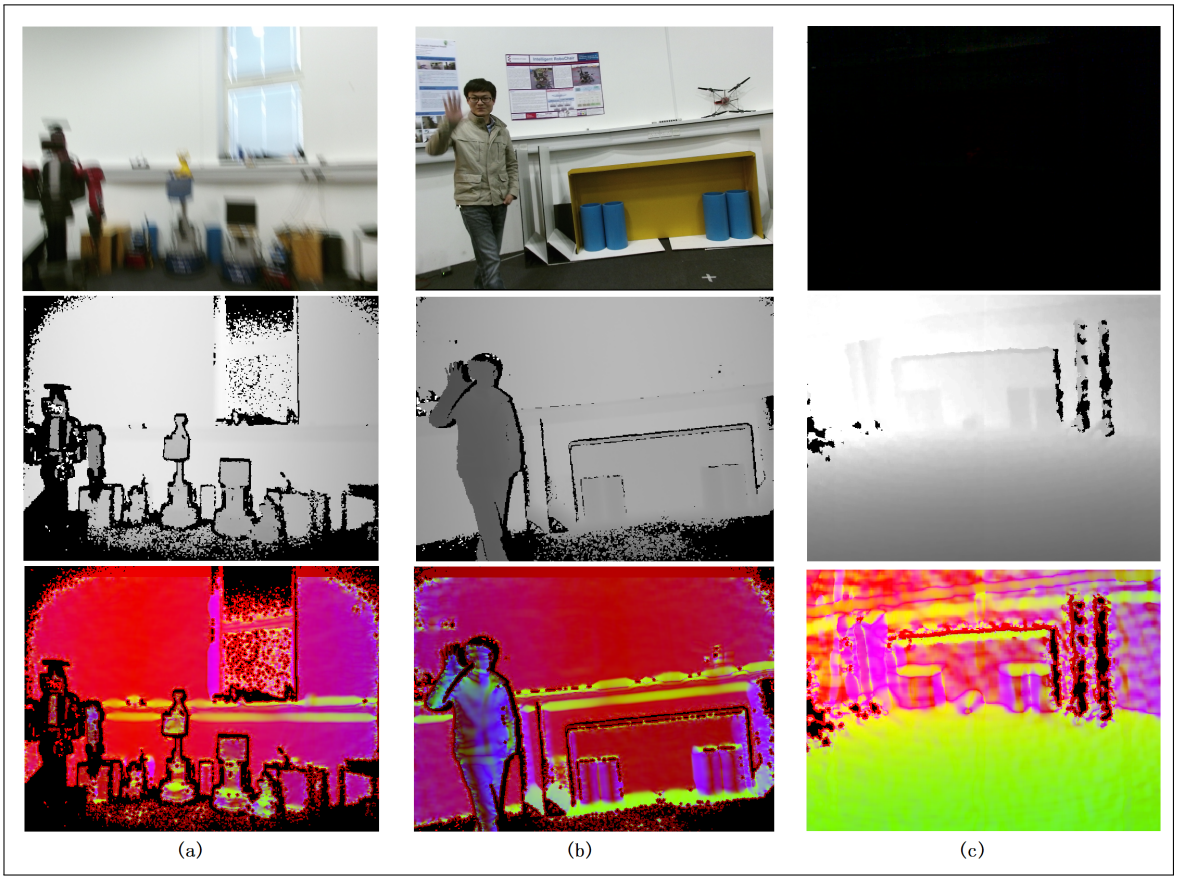


Figure 4.6: Challenging scenes for indoor relocalization with the dual-stream CNN. The images in the first row are color images. The second row is the scaled depth image and the third row is the MND depth image. (a) Motion blur produced by fast movement. (b) Dynamic scenes with pedestrian or other moving objects. (c) Night-time or dimly indoor environments.

4.5.3 Quantitative Analysis

In this part, we compare our dual-stream CNN with PoseNet. We also transform the dual-stream CNN to a Bayesian dual-stream CNN and compare it with the Bayesian PoseNet [58]. The transformation is very similar with that from PoseNet to Bayesian PoseNet. The Dropout layers are added to color stream and depth stream separately, the network inputs are randomly cropped from preprocessed images for 30 times, and the final result takes the average of all 30 network outputs.

Table 4.4: Indoor relocalization results in challenging scenes with different methods. Our system shows about 30%–70% improvement in precise compared with PoseNet in these challenging but everyday situations.

Dataset	Scale	Train/ Test	Network architecture	Input	Median position error	Median orientation error
Normal	4×4×1m	5540/ 766	PoseNet	RGB	0.36m	6.08°
Normal	4×4×1m	5540/ 766	Dual-stream CNN	RGB-MND depth (3+3)	**0.26m**	**4.32°**
Fast movement	4×4×1m	5540/ 357	PoseNet	RGB	0.52m	13.03°
Fast movement	4×4×1m	5540/ 357	Dual-stream CNN	RGB-MND depth (3+3)	**0.31m**	**9.00°**
Dynamic	4×4×1m	5540/ 964	PoseNet	RGB	0.44m	9.82°
Dynamic	4×4×1m	5540/ 964	Dual-stream CNN	RGB-MND depth (3+3)	**0.32m**	**8.62°**
Night-time	4×4×1m	3473/ 656	PoseNet	RGB	1.24m	29.10°
Night-time	4×4×1m	3473/ 656	Dual-stream CNN	RGB-MND depth (3+3)	**0.39m**	**7.48°**

We take the public Microsoft 7-Scenes dataset as benchmark to compare the system performance in indoor relocalization. As shown in Table 4.3, PoseNet with color images and Posenet with depth images have no significant difference in indoor relocalization, sometimes the former with texture features performs better, sometimes the latter with range features performs better. Our proposed dual-stream CNN which takes both color images and depth images together achieves better performance than PoseNet. So does Bayesian dual-stream CNN when compared with Bayesian PoseNet. When comparing our dual stream CNN with PoseNet, both position accuracy and orientation accuracy have gained more than 15% improvement as shown in table II and table III.

4.5.4 Qualitative Analysis based on Challenging Datasets including Fast Movement, Night-time, Dynamic Scenes

This part introduces the advantage of our dual-stream CNN in indoor relocalization when encountering challenging situations which other methods can hardly deal with.

All the datasets are collected in our Robot Arena lab equipped with a motion capture system which can provide the groundtruth. For night-time scenes, we use Asus Xtion to collect RGB-D images. For other datasets, we use Kinect Xbox one. Notice that all the training datasets are collected in normal environments which means there are no fast movement of camera, moving objects and dark scenes when collecting the training dataset. The dual-stream CNN will infer the original lighting, shape, texture from images when appearance changes, and then estimate the camera poses.

Fig. 4.6 shows the color images, depth images and MND depth images in three challenging scenes. Fast movement of the camera, dynamic scenes including pedestrians or other moving objects, night-time scenes are really challenging for visual localization, but they happen everyday and everywhere in our life. From the figure we can see that the color images in these situations are in particularly bad quality. On the contrary, the depth images maintain all range features without much information loss. Our dual-stream CNN combining textural, structural and range features outperforms PoseNet significantly when encountering these challenging situations. From Table 4.4, we can see that there are approximately 27%–68% and 12%–64% improvements to position precision and orientation precision, respectively, with our system when compared with PoseNet. The cumulative histograms of indoor relo-

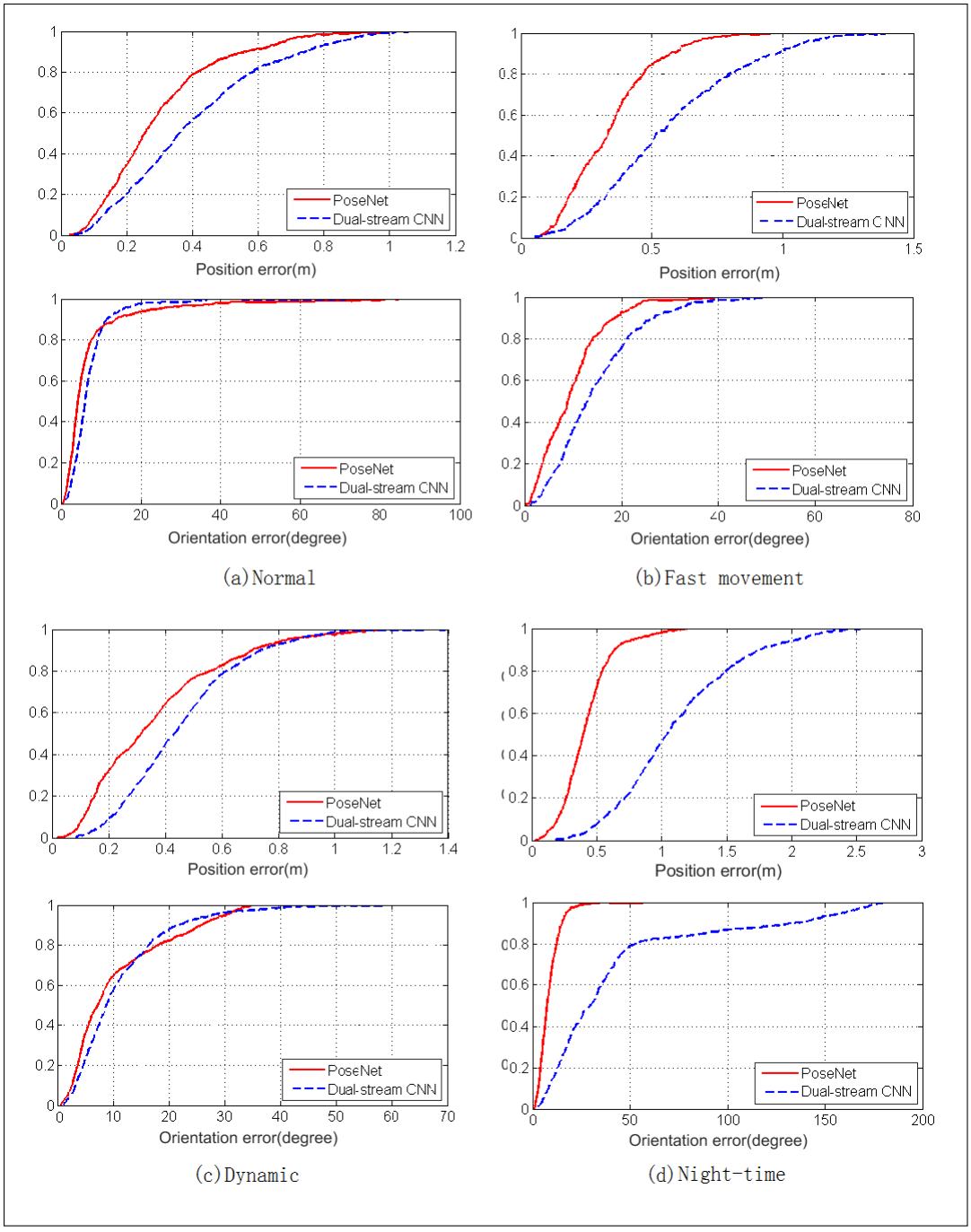


Figure 4.7: Cumulative histogram of relocalization error for challenging indoor test datasets with the dual-stream CNN. The dual-stream CNN shows significant advantages over PoseNet especially in the challenging situations.

calization error in these situations are plotted in Fig. 4.7. Our system performs especially well when faced with fast movement of the camera and night-time scenes on account of introducing the depth stream.

We have also tried to use SLAM technology with hand-crafted features to perform localization in these challenging environments, but found that most camera poses will be lost in dynamic and motion blur situations, and the algorithm will completely lose efficacy in night-time environments.

4.6 Conclusions

In this chapter, we have presented a novel dual-stream CNN for 6-DOF pose regression which shows spectacular performance in large scale indoor relocalization. A novel depth encoding method that takes the minimized normal and depth (MND) as processed depth image is addressed. Compared with other encoding methods such as normalized depth or colorized depth, our MND encoding approach presents comparable performance in relocalization. Moreover, by introducing depth information with a separate stream, our network could learn both texture features from color images and structural range features from depth images. In this way, the relocalization precision is improved compared with PoseNet, and the system robustness is greatly enhanced when faced with challenging environments such as fast movement, dynamic objects and night-time in which other algorithms demonstrate poor performance.

However, our network needs labelled dataset to train the network. Obtaining ImageNet-scale labelled dataset is very difficult and there is still no such dataset available for pose regression problems. Without this kind of dataset, people can hardly see the extremely potential of deep learning for pose estimation. Therefore, our next mission is to develop some unsupervised deep learning methods in order to tackle the problem.

Chapter 5

DeepSLAM: A Robust Monocular SLAM System with Unsupervised Deep Learning

Aiming at the ego-motion estimation and the 3D dense mapping, we propose DeepSLAM, a novel unsupervised deep learning based visual SLAM system. By introducing fully unsupervised deep learning, the problem of lacking of labelled datasets can be solved. Our DeepSLAM is composed of Mapping-Net, Tracking-Net, Loop-Net, and a graph optimization unit. The Mapping-Net is used for depth estimation, the Tracking-Net is used for pose estimation, the Loop-Net is used for loop closure detection, and the graph optimization unit is used for pose graph construction and optimization. Spatial losses, temporal losses and an outlier rejection mechanism are designed to drive networks learning. The evaluation experiments are performed on multiple benchmark datasets and the system demonstrates the state-of-the-art performance on both accuracy and robustness.

5.1 Introduction

Tremendous efforts have been made to visual SLAM in the robotics and computer vision communities. Especially, over the past decade several state-of-the-art visual SLAM systems have been designed based on sparse feature points [4, 6, 14] or photometric consistency of dense pixels [15]. Since most of these methods are geometric model based, they can not

learn automatically from raw images or benefit from continuously increased datasets. Some of them are also fragile under challenging scenes. There increasingly arises a question, particularly when encountering large-scale dataset, that whether it is possible to understand and tackle the visual SLAM problem from a data-driven perspective and whether data-driven approaches are beneficial.

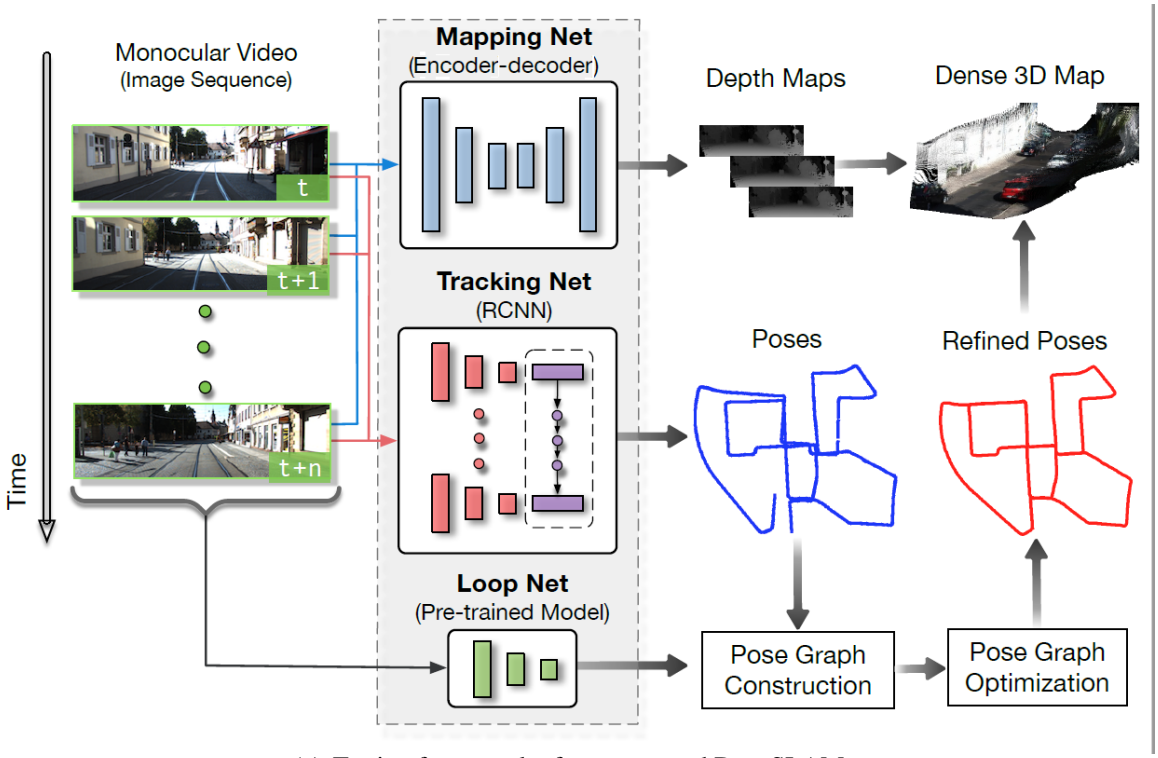
Recently, Deep Learning (DL) based methods have demonstrated a promising performance on pose and depth estimation [40] [42]. However, most of them learn from raw images with limited consideration of geometric models (triangularity geometry) which have been well understood and recognized as the fundamentals of visual SLAM systems. It has been demonstrated that the learning representation for depth estimation is more efficient if geometric models are respected [45]. Therefore, it is interesting to see how the learning representation could be effectively exploited for visual SLAM by seamlessly incorporating the wide knowledge accumulated over decades on geometric models.

Most DL based methods are based on supervised learning schemes which require labeled datasets. However, labeling large amounts of data is difficult and expensive. This requirement limits the potential application scenarios of DL based methods. This is particularly true in the context of visual SLAM because robots typically operate in completely unknown environments. It is very demanding for a visual SLAM system to learn under an unsupervised scheme so that the performance could be continuously improved by the increased size of unlabeled datasets.

In this chapter, we propose DeepSLAM, an unsupervised DL based monocular SLAM system. It takes monocular color images as its input and outputs pose trajectory, depth map, and 3D point cloud simultaneously (see Fig. 5.1). It learns from unlabeled stereo image pairs with an unsupervised scheme. Because we do not need the ground truth (target) to supervised train our networks, so our method is unsupervised learning. We exploit the combination of DL and geometric constraints in DeepSLAM to improve the performance. Our main contributions are summarized as follows:

- A novel visual SLAM framework based on unsupervised DL is proposed. It is a result from the combination of DL and geometric constraints.

- Deep Recurrent Convolutional Neural Network (RCNN) is designed to model the ego-



(a) Testing framework of our proposed DeepSLAM

(b) Challenging scenes our DeepSLAM can face with.

Figure 5.1: Testing framework of our proposed DeepSLAM. (a) Testing framework of our proposed DeepSLAM. It is composed of Tracking-Net, Mapping-Net and Loop-Net. Tracking-Net is a RCNN based architecture to perform pose estimation. Mapping-Net is a auto-encoder architecture to perform depth estimation. Loop-Net is a DNN architecture to perform loop closure detection. Pose graph construction and optimization are implemented as the back-end. Our proposed DeepSLAM system takes monocular color images as the input, and produces depth maps, object masks, trajectories, point clouds as the outputs. (b) DeepSLAM also demonstrates a robust performance in some challenging scenes, such as image distortion, bad white balance, night-time, raining, etc.

motion by leveraging both spatial and temporal properties of a sequence of stereo images during training.

- DeepSLAM integrates the DL based tracking result and loop closure detection with a graph based optimization mechanism, resulting in a superior performance in terms of accuracy and robustness.

- Outliers rejection is handled by using the uncertainty derived from error maps of both geometric and photometric consistencies. This improves the robustness performance of DeepSLAM in challenging scenes.

Note that although DeepSLAM uses stereo images (stereo images come from two cameras with a fixed baseline) for training, only monocular vision is required for testing. Therefore, DeepSLAM is a monocular visual SLAM system.

The rest of this chapter is organized as follows. Section 5.2 reviews related work. The system architecture of the proposed DeepSLAM is provided in Section 5.3, followed by the presentation of training losses and outliers rejection in Section 5.4. Section 5.5 introduces the construction and optimization of pose graph in our DeepSLAM. Section 5.6 provides our experimental results. Finally, our conclusion is drawn in Section 5.7.

5.2 Related Work

In this section, aiming at the related works of our proposed DeepSLAM, we give a brief overview of related works including supervised deep learning methods and unsupervised deep learning methods.

5.2.1 Supervised Deep Learning for Pose Estimation

Apart from pose regression, the ego-motion between two image frames could be estimated by using DL inspired by stereo geometric models. Costante [64] developed a CNN to estimate the ego-motion with supervised training. Wang et al. [13] trained a RCNN to estimate the camera motion (DeepVO). Melekhov [65] presented a relative camera pose estimation system with CNN. Oliveira et al. [68] constructed a metric net for ego-motion estimation and a topological net for topological location estimation. Clark et al. [73] also proposed a sensor fusion network called VINet, which fuses the estimated pose and the inertial sensor

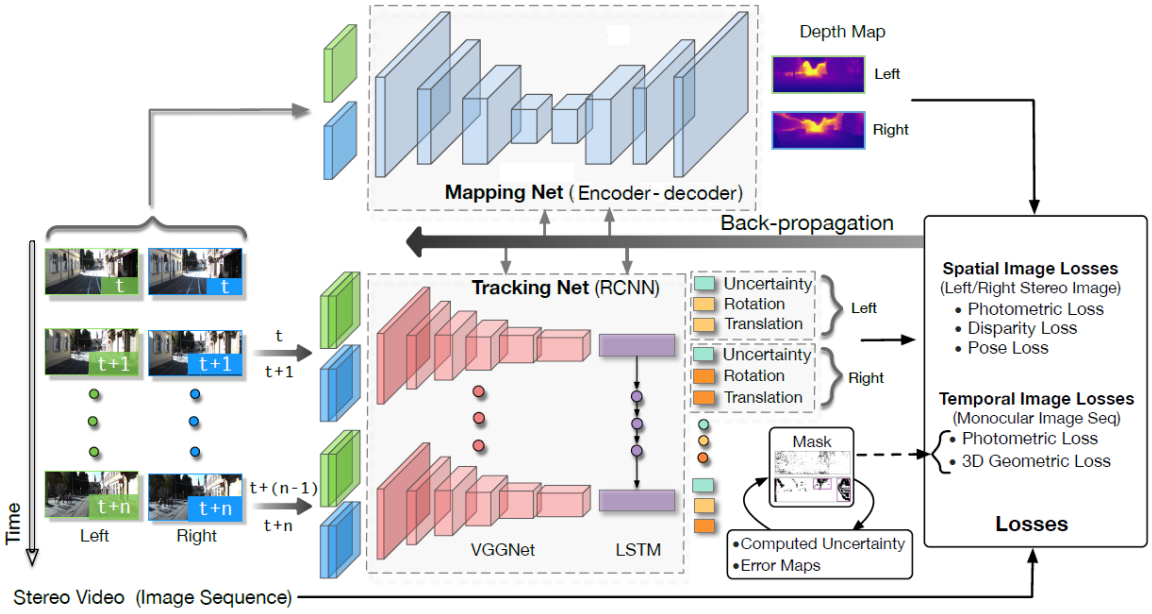


Figure 5.2: Training scheme of the proposed DeepSLAM system. We use stereo images to train the system in order to recover the scale of pose and depth estimation. The spatial image losses between a stereo image pair and the temporal image losses between a sequence image pair are formulated to train the networks. The error map produced from the system is used as the loss masks for outliers rejection. The uncertainty produced from the system is also used to train the networks. The input of the Mapping-Net is a 3*416*128 RGB image, and the output of the Mapping-Net is a 1*416*128 depth image. The input of the Tracking-Net is a 6*416*128 RGB image with 5 consecutive frames, and the output of the Tracking-Net is a 5*7 matrix which represents the translation, rotation and the estimated uncertainty of 5 input consecutive frames, respectively.

reading with a RCNN. Pillai [76] proposed an ego-motion estimation system that fused the information from camera with other sensors such as GPS, INS and wheel odometry. Ummenhofer [55] proposed "DeMoN" to estimate ego-motion, image depth, surface normal and optical flow simultaneously. DeTone [57] developed one network to predict the location of feature points and another network to compute the homography with manually synthesized data. Tateno et al. [44] employed a CNN to estimate the depth map, then used a geometric model-based SLAM system to perform the pose estimation.

5.2.2 Unsupervised Deep Learning for Depth and Ego-motion Estimation

The main problem in the supervised pose estimation systems is the requirement of a large amount of labeled data to train the networks. Currently, the size of labeled datasets is limited

and they are costly to collect. This restrains the performance of the supervised learning systems from further improvement. Recently unsupervised DL methods were successfully applied for depth estimation, inspired by the image wrap technique "spatial transformer" [146]. Garg et al. [45] proposed an unsupervised depth estimation method by exploiting left-right photometric constraint in stereo image pairs. The network training is fully unsupervised in an end-to-end manner. Xie et al. [50] proposed a 2D-to-3D video conversion method with unsupervised learning. Godard et al. [12] improved the Garg's method by wrapping left and right images across each other. Zhou et al. [1] proposed SfMLearner, which used a monocular image sequence for image alignment in order to estimate the depth and ego-motion simultaneously with unsupervised learning. Vijayanarasimhan et al. [147] proposed SfM-Net which added motion masks to the photometric loss. It could estimate optical flow, depth map and ego-motion simultaneously.

In summary, unsupervised DL techniques are promising a new research trend within the visual SLAM research field. The data-driven approach has already demonstrated a versatile capability and has the potential to improve the robust performance for SLAM applications. The integration of DL techniques with model-based SLAM methods could make a success in improving the performance of both estimation accuracy and robustness.

5.3 System Overview of DeepSLAM

The backbone of DeepSLAM system includes three deep neural networks: Tracking-Net, Mapping-Net, and Loop-Net. When testing, all the networks take a monocular image sequence as the inputs. Tracking-Net estimates camera poses and the corresponding uncertainties, Mapping-Net predicts depth maps, and Loop-Net produces sparse feature vectors for loop closure detection.

The DeepSLAM testing framework is shown in Fig. 5.1. The DeepSLAM training framework is shown in Fig. 5.2. The trained Tracking-Net, Mapping-Net and Loop-Net can be viewed as the front-end of our DeepSLAM system to yield a pose graph, while the pose graph optimization is employed to refine the poses as the back-end. The integration of the trained networks with the graph optimization process leads to the improvement in both accuracy and robustness. This section focuses on Tracking-Net and Mapping-Net. The details on Loop-Net and pose graph optimization will be given in Section V.

5.3.1 Tracking-Net with RCNN Architecture

Tracking-Net is a RCNN architecture constructed from a CNN and a Recurrent Neural Network (RNN) to estimate poses and uncertainties. Since visual SLAM problem needs to model camera dynamics and deal with sequential data (image sequence and video), the CNN that processes single or several stacked images are not sufficient. Therefore, Tracking-Net is equipped with a RNN to learn sequential representation from the CNN features. This enables DeepSLAM not only to exploit the temporal information for current pose estimation, but also to build the local connections among poses similar to local bundle adjustment. As shown in Fig. 5.3, the correlation between two consecutive images of a monocular sequence is captured through the RCNN architecture. Note DeepSLAM uses Long Short-Term Memory (LSTM) [29] as the RNN since simple RNNs suffer from the vanishing gradient problem in practice. We will provide further details of the RCNN on the construction of local pose graph in Section 5.5.1.

Tracking-Net takes a sequence of n consecutive monocular images and produces $(n-1)$ vectors with 7 elements (3 elements are the change in position, 3 elements are the change in orientation, 1 element is the estimation confidence). The rotation and translation of a pose are estimated through two independent fully-connected layers after the LSTM layers. The uncertainty here is also produced through two fully-connected layers at the end of the network, and it is a confidence scalar which couples both depth and poses estimation. The changes in position and orientation are used to calculate a transformation matrix $\widehat{T}_{k,k+1}$. Then the transformation is used to synthesis the input image I_k to produce $I^{'}_{k+1}$ which is an estimation of what next frame should look like (see equation 5.8). Next we define a loss function which compares the estimated image $I^{'}_{k+1}$ to the original image I_{k+1}. Finally, the network is trained to minimize these loss functions by gradient descent optimization with respect to the network weights in all layers. The mean of error maps are used to compute the uncertainty loss, see more details in IV.

5.3.2 Mapping-Net with Encoder-Decoder Architecture

Mapping-Net is an encoder-decoder architecture to produce dense depth maps. [78] It encodes the color image into a hidden state, which is further decoded to a predicted depth map

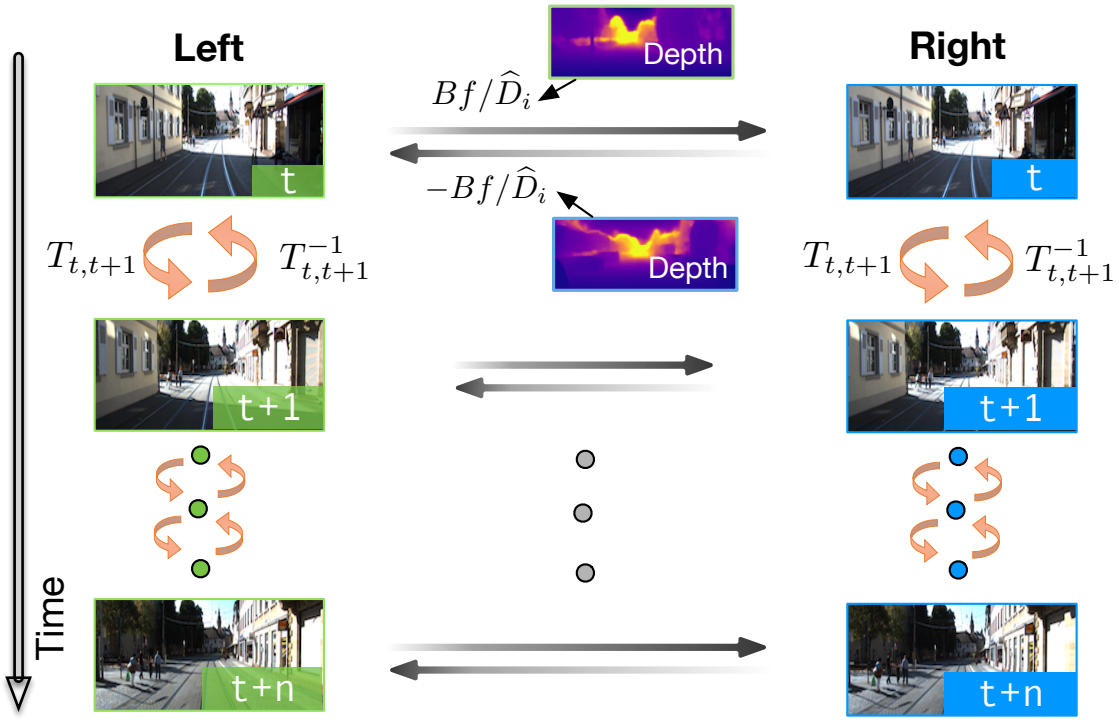


Figure 5.3: Spatial and temporal constraints used to formulate the spatial and temporal losses.

with the deconvolutional operation. The size of the network input (color image) and the size of the network output (predicted depth image) are same. The predicted depth images are then used to construct loss functions (see more details in IV). Finally, the network is trained to minimize these loss functions with respect to network weights in all layers. Since it only has convolutional layers, it is flexible to the size of input images. Different from other depth estimation methods [12] [1] which produce disparity images (inverse of the depth) from the network, Mapping-Net is designed to directly estimate depth maps. Our training trails show that the whole system is easier to converge when trained in this way.

For most monocular SLAM methods, a pre-defined scale has to be applied. One feature of our DeepSLAM is to recover the absolute scale of depth and pose. It is credited to our training scheme shown in Fig. 5.2. During training, we feed both left and right images into the Mapping-Net, which produces the depths of left and right sequences. The estimated depth images of stereo images are used for training with the consideration of geometric models.

5.3.3 Training of DeepSLAM

The training scheme of the DeepSLAM is shown in Fig. 5.2. Tracking-Net and Mapping-Net are trained together using stereo image pairs in an unsupervised manner. The purpose of using stereo image pairs instead of monocular ones for training is to recover the scales of ego-motion and depth estimation. Loop-Net is a pre-trained CNN for identifying loop closures.

As shown in Fig. 5.2, we utilize both spatial and temporal geometric consistencies of stereo image sequences to formulate the loss function. The spatial geometric consistency represents the geometric projective constraint between the corresponding points in left-right image pairs, while the temporal geometric consistency represents the geometric projective constraint between the corresponding points in two consecutive monocular images. By using these constraints to construct the loss function and minimizing them all together, the networks learn to estimate scaled 6-DoF poses and depth maps in an end-to-end unsupervised manner. We are now in the position to discuss the losses used for training.

5.4 Unsupervised Training based on Spatial and Temporal Geometric Consistencies

This section describes the losses designed for training Mapping-Net and Tracking-Net in an unsupervised manner. In general, there are two kinds of losses to be minimized for training DeepSLAM models: spatial image loss and temporal image loss. The construction of these two losses is shown in Fig. 5.3.

5.4.1 Spatial Image Loss of a Pair of Stereo Images

Spatial image loss is defined by the geometric constraints between a pair of left and right stereo images. Based on these constraints and the known stereo baseline, it can recover the absolute scales for both poses and depth maps. It consists of photometric consistency loss, disparity consistency loss and pose consistency loss computed from a pair of stereo images.

Photometric Consistency Loss

For a pair of stereo images, every overlapped pixel i in one image can find its correspondence in the other image with a horizontal distance H_i [45]. Given its depth value D_i, the distance H_i can be calculated by

$$H_i = Bf/D_i \tag{5.1}$$

where B is the baseline of stereo camera and f is the focal length. Therefore, by using the predicted depth map $\widehat{D}_i$ from the Mapping-Net, a distance map H can be generated for the whole image. Based on H, we can synthesize a new image by warping an image from the other through spatial transformer [146]. Assume I_l' and I_r' are the synthesized left and right images from original right image I_r and left image I_l, respectively. The left-right photometric consistency losses can be constructed as

$$L_{l,r}^p = \sum \lambda_s f_s(I_l, I_l') + (1-\lambda_s)\left\|I_l - I_l'\right\|_1 \tag{5.2}$$

$$L_{r,l}^p = \sum \lambda_s f_s(I_r, I_r') + (1-\lambda_s)\left\|I_r - I_r'\right\|_1 \tag{5.3}$$

where $\|\cdot\|_1$ is the L1 norm, λ_s is a weight scalar, $f_s(\cdot) = (1 - SSIM(\cdot))/2$ and $SSIM(\cdot)$ is the Structural SIMilarity (SSIM) metric [148] [149] to evaluate the quality of a synthesized image.

Disparity Consistency Loss

The disparity map [45] is defined by

$$Q = H \times w \tag{5.4}$$

where w is the image width which is the scalar, H is the distance map matrix introduced before, $\times$ is multiplication operation. Denote Q_l and Q_r as the left and right disparity maps, respectively. Therefore, the estimated left and right disparity maps can also be computed from H. Similar to the photometric consistency loss(equation 5.1), we can use H to synthesize Q_l', Q_r' from Q_r, Q_l, respectively. By using these disparity maps, the disparity consistency losses

can be constructed as

$$L^d_{l,r} = \sum \left\| Q_l - Q'_l \right\|_1 \tag{5.5}$$

$$L^d_{r,l} = \sum \left\| Q_r - Q'_r \right\|_1 \tag{5.6}$$

Pose Consistency Loss

If left and right image sequences are employed to separately estimate the 6-DoF transformations of camera motion through the Tracking-Net, ideally these relative transformations should be exactly the same. Therefore, the differences between these two groups of pose estimates can be introduced as a left-right pose consistency loss as

$$L^o = \lambda_p \left\| \widehat{\mathbf{x}}_l - \widehat{\mathbf{x}}_r \right\|_1 + \lambda_r \left\| \widehat{\varphi}_l - \widehat{\varphi}_r \right\|_1 \tag{5.7}$$

where x is a vector representing translation, φ is a vector representing rotation, $[\widehat{\mathbf{x}}_l, \widehat{\varphi}_l]$ and $[\widehat{\mathbf{x}}_r, \widehat{\varphi}_r]$ are the estimated poses from left and right image sequences by the Tracking-Net, respectively. λ_p and λ_r are the position and rotation weights. Note that the length of image sequence can be variable thanks to the recurrent network in the Tracking-Net.

5.4.2 Temporal Image Loss of a Sequence of Monocular Imagery

Temporal image loss exploits geometric constraints among multiple views of a monocular image sequence to enable the Mapping-Net to produce meaningful depth maps and the Tracking-Net to estimate camera motion. As shown in Fig. 5.3, the RCNN architecture enables correlation between two consecutive monocular images. It includes photometric consistency loss and 3D geometric registration loss.

Photometric Consistency Loss

Different from the previous photometric consistency loss of a pair of stereo images, the photometric loss here focuses on the temporal information among an image sequence. For each image pair I_k, I_{k+1} with some scene overlaps, we can obtain their synthesized images I'_k and I'_{k+1} by using spatial transformer [146]. Specifically, for an overlapped pixel $p_k = (u_k, v_k, 1)^T$ in the kth frame, we can derive its corresponding pixel $p'_{k+1} = (u'_{k+1}, v'_{k+1}, 1)^T$ in

the $(k+1)$th frame through

$$p'_{k+1} = K\widehat{T}_{k,k+1}\widehat{D}_k K^{-1} p_k \tag{5.8}$$

where K is the camera intrinsic matrix [117], $\widehat{D}_k$ is the pixel's depth estimated from the Mapping-Net, $\widehat{T}_{k,k+1}$ is the camera coordinate transformation matrix from the kth frame to the $(k+1)$th frame predicted by the Tracking-Net. Based on this, I'_k and I'_{k+1} can be constructed from I_{k+1} and I_k, respectively. Define a temporal photometric error map between an image I_k and its synthesized image I'_k as $E_p^k = I_k - I'_k$. Then, the photometric error maps for the k-to-$(k+1)$ and $(k+1)$-to-k consistencies are

$$E_p^k = I_k - I'_k, \quad E_p^{k+1} = I_{k+1} - I'_{k+1} \tag{5.9}$$

Then, the photometric losses of image pair I_k, I_{k+1} from the monocular image sequence are

$$L^p_{k,k+1} = \sum M_p^k \left(\lambda_s f_s(I_k, I'_k) + (1-\lambda_s)\left\|E_p^k\right\|_1\right) \tag{5.10}$$

$$L^p_{k+1,k} = \sum M_p^{k+1} \left(\lambda_s f_s(I_{k+1}, I'_{k+1}) + (1-\lambda_s)\left\|E_p^{k+1}\right\|_1\right) \tag{5.11}$$

where M_p^k and M_p^{k+1} denote the bitwise mask of the corresponding photometric error map. We will discuss the mask in Section 5.4.3. The definition of the photometric loss is similar to the one used in the DTAM [7]. Note that frames k and $k+1$ are not necessarily consecutive but they should have overlapped pixels. Since DeepSLAM has a recurrent neural network architecture, the photometric losses are determined by several pairs of images in the image sequence, which facilities the construction of local graph. See more details for local graph in next section.

3D Geometric Registration Loss

Geometric loss is used to constrain and estimate the transformations by considering 3D point clouds. It is similar to Iterative Closest Point (ICP) [125], a well-known method to align point clouds. In our DeepSLAM system, we also use this loss for pose estimation.

Assuming P_k and P_{k+1} are the 3D point clouds in the kth and $(k+1)$th camera coordinations, and P'_k and P'_{k+1} are the transformed point clouds in these two coordinate frames. Then

we construct the geometric losses in the monocular image sequence as

$$L^g_{k,k+1} = \sum M^k_g \left\| E^k_g \right\|_1 \tag{5.12}$$

$$L^g_{k+1,k} = \sum M^{k+1}_g \left\| E^{k+1}_g \right\|_1 \tag{5.13}$$

where $E^k_g = P_k - P'_k$ and $E^{k+1}_g = P_{k+1} - P'_{k+1}$ are the temporal geometric error maps, and M^k_g and M^{k+1}_g are the mask of the corresponding geometric error map.

With regards to existing works, [45] and [12] used left-right photometric loss to estimating depth map, [12] used left-right disparity consistency loss. [1] tried to use the photometric loss of image sequence to recover ego-motion and depth. However, no work has gathered all these losses together to recover both scaled camera pose and depth map.

5.4.3 Uncertainty Estimation and Outliers Rejection

Uncertain estimation and outliers rejection are essential in SLAM systems. For the uncertainty estimation of supervised pose regression methods with DL, they either adopt a sampling method by using Dropout [58] or add a balance factor to the network as a mixture model [59] [54] to obtain the uncertainty of pose estimation. Different from those supervised methods, DeepSLAM can produce the projective photometric error maps E^k_p, E^{k+1}_p and the projective geometric error maps E^k_g, E^{k+1}_g for two consecutive images I_k and I_{k+1}. Assuming μ^k_p, μ^{k+1}_p, μ^k_g, μ^{k+1}_g are the mean of E^k_p, E^{k+1}_p, E^k_g, E^{k+1}_g, respectively. Then, the uncertainty scalar of pose estimation and depth estimation with the kth frame and $(k+1)$th frame can be represented as

$$\sigma_{k,k+1} = 2 \times S(\mu^k_p + \mu^{k+1}_p + \lambda_e(\mu^k_g + \mu^{k+1}_g)) - 1 \tag{5.14}$$

where $S(\cdot)$ is the Sigmoid function and λ_e is the normalizing factor between the geometric and photometric errors. Sigmoid is the function normalizing the uncertainty between 0 and 1 to represent the belief on the accuracy of pose estimate. We use $\sigma_{k,k+1}$ to train the uncertainty estimation $\widehat{\sigma}_{k,k+1}$ of the Tracking-Net:

$$L^u_{k,k+1} = \left\| \sigma_{k,k+1} - \widehat{\sigma}_{k,k+1} \right\|_1 \tag{5.15}$$

$\widehat{\sigma}_{k,k+1}$ is estimated by the Tracking-Net (network output) and represents the uncertainties of estimated poses and depth maps. Intuitively, $\widehat{\sigma}_{k,k+1}$ is small when the estimated poses and depth maps are accurate enough to reduce the photometric and geometric errors.

In real-world environments, the photometric and geometric losses could be corrupted by dynamic objects. Therefore, we introduce the masks for the error maps in the previous temporal losses. We propose a novel method to construct bitwise masks to reject the outliers during training. According to the error values in the error maps, masks are constructed with a percentile q_{th} of pixels as value 1 and a percentile $(100 - q_{th})$ of pixels as value 0. Specifically, based on the uncertainty $\sigma_{k,k+1}$, the percentile q_{th} of the pixels is determined by

$$q_{th} = q_0 + (100 - q_0)(1 - \sigma_{k,k+1}) \tag{5.16}$$

where $q_0 \in (0, 100)$ is the basic constant percentile. Then, we can construct the bitwise masks M_p and M_g to filter out $(100 - q_{th})\%$ of the big errors (as outliers) in the error maps. The generated masks not only automatically adapt to the different percentage of outliers, but also can be used to infer dynamic objects in the scene. This will be discussed in detail in Section 5.6.4.

5.5 Pose Graph Construction and Optimization

Pose graph optimization plays an important role in geometric model based SLAM systems due to its ability to reduce the cumulative pose drifts. In our system, we also perform a pose graph optimization with both local and global pose connections. The local pose graph is built upon a short sequence of consecutive images as a direct result of the recurrent model of the Tracking-Net considering an image sequence, i.e. the local pose graph is constructed from consecutive image frames. The global loop closures are detected by the Loop-Net from the historical images, which are usually non-consecutive.

5.5.1 RCNN based Local Pose Graph

The RCNN architecture of the Tracking-Net is able to learn the relationship between CNN features over time as the camera moves, modeling the motion dynamics of the camera from image sequence. Therefore, based on the structure of Tracking-Net, we can construct the

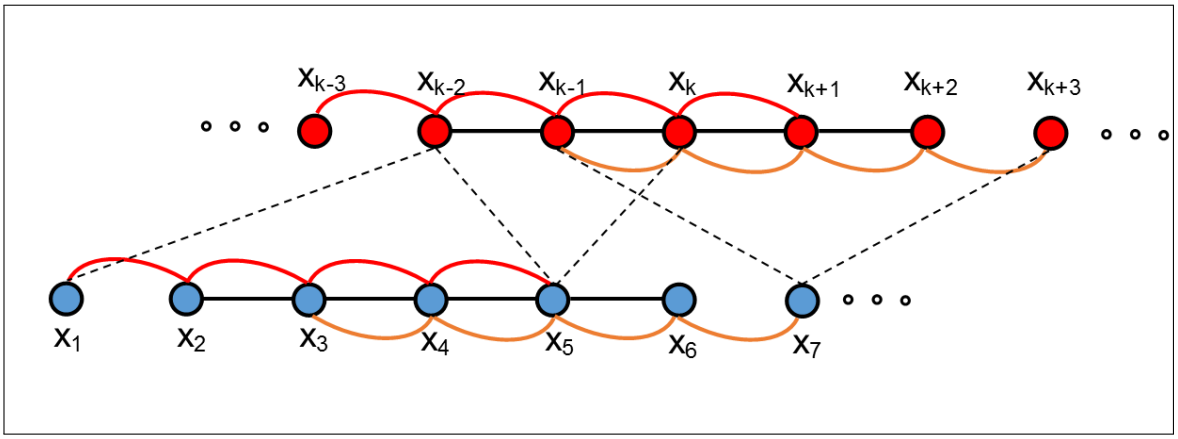


Figure 5.4: Pose Graph with local and global connections. The dotted lines represent the global loops detected from Loop-Net, while the solid lines represent the local loops generated by Tracking-Net. The local graph with image sequence length 5 is shown here as an example.

local pose graph directly. Assuming the length of an image sequence is n, each time the Tracking-Net can estimate $(n-1)$ relative poses to build the local pose graph. An example of the pose graph built with sequence length 5 is shown in Fig. 5.4. As the camera moves over time, our system can gradually construct the pose graph with the local loops.

5.5.2 CNN based Global Loop Detection

For global pose graph, we use the Loop-Net to perform the place recognition and loop detection between non-neighboring frames. The Loop-Net in our DeepSLAM system is a DNN model pre-trained on Image-Net dataset for object recognition. Note that there is no training involved for the Loop-Net. The Inception ResNet V2 architecture [150] is adopted here. The weights of Inception ResNet V2 trained on ImageNet is borrowed here. The Loop-Net maps images into feature vectors. We can then compute the cosine distance of two feature vectors from an image pair to detect loop closures [135]:

$$d_{cos} = cos(\mathbf{v_1}, \mathbf{v_2}) \tag{5.17}$$

where $\mathbf{v_1}$, $\mathbf{v_2}$ are the feature vector representations of an image pair. When d_{cos} is smaller than a threshold d_{cos}^{th}, the image pair is treated as a loop. To be more specific, the principle is that if two scenes contain the same set of recognisable objects within them, then there is a high likelihood that the two views were taken from the same location. v1 is the feature

vector of probabilities corresponding to objects recognized in view 1. v2 is the feature vector of probabilities corresponding to objects recognized in view 2. Then Eq. 5.17 indicates the similarity between two scenes.

After the Loop-Net detects global loops, we will use our Tracking-Net to calculate the transformation between detected image pairs. Since the recurrent structure makes the Tracking-Net flexible to the length of image sequence, we can use the sequence length 2 to compute the pose transformation. Once a global loop is inserted into the pose graph, G2o [151] is used for pose graph optimization as the back-end to reduce the cumulative drifts.

5.6 Experimental Evaluation

In this section, we demonstrate the tracking and mapping performance of our proposed DeepSLAM system. We conducted the evaluation on pose and depth accuracy separately in order to see how each network performs. Firstly, the Tracking-Net was tested as a standalone monocular visual odometry system without loop closure detection. We compared it against one unsupervised deep learning method (SfMLearner) [1], and two model based methods: ORB-SLAM without loop closure detection, and VISO2. Secondly, the full system of DeepSLAM (including loop closure detection) was tested against ORB-SLAM with loop closure detection. Thirdly, the Mapping-Net was tested against other unsupervised deep learning methods to evaluate the depth estimation performance. The above evaluations were conducted on KITTI dataset [30].

Then we conducted the evaluation on the robustness of our DeepSLAM against other model based SLAM methods using some challenging datasets in RobotCar [32]. We also tested our system on a self-collected dataset with a low-cost camera.

The proposed DeepSLAM was designed with the DL framework TensorFlow and trained on NVIDIA DGX-1 with Tesla P100. For testing, a laptop equipped with NVIDIA GeForce GTX 980M and Intel Core i7-6820HK 2.7GHz CPU was used. The GPU memory required for the Tracking-Net is less than 400MB with 40Hz real-time performance. The Adam optimizer was employed to train the network for up to 20-30 epochs. The starting learning rate was 0.001 and decreased by half for every 1/5 of total iterations. The parameter β_1 is 0.9 and

β_2 was 0.99. The sequence length of images feeding to the Tracking-Net was 5. The image size was 416×128.We also resized the output images to a higher resolution to compute the losses and fine-tuned the networks in the end.

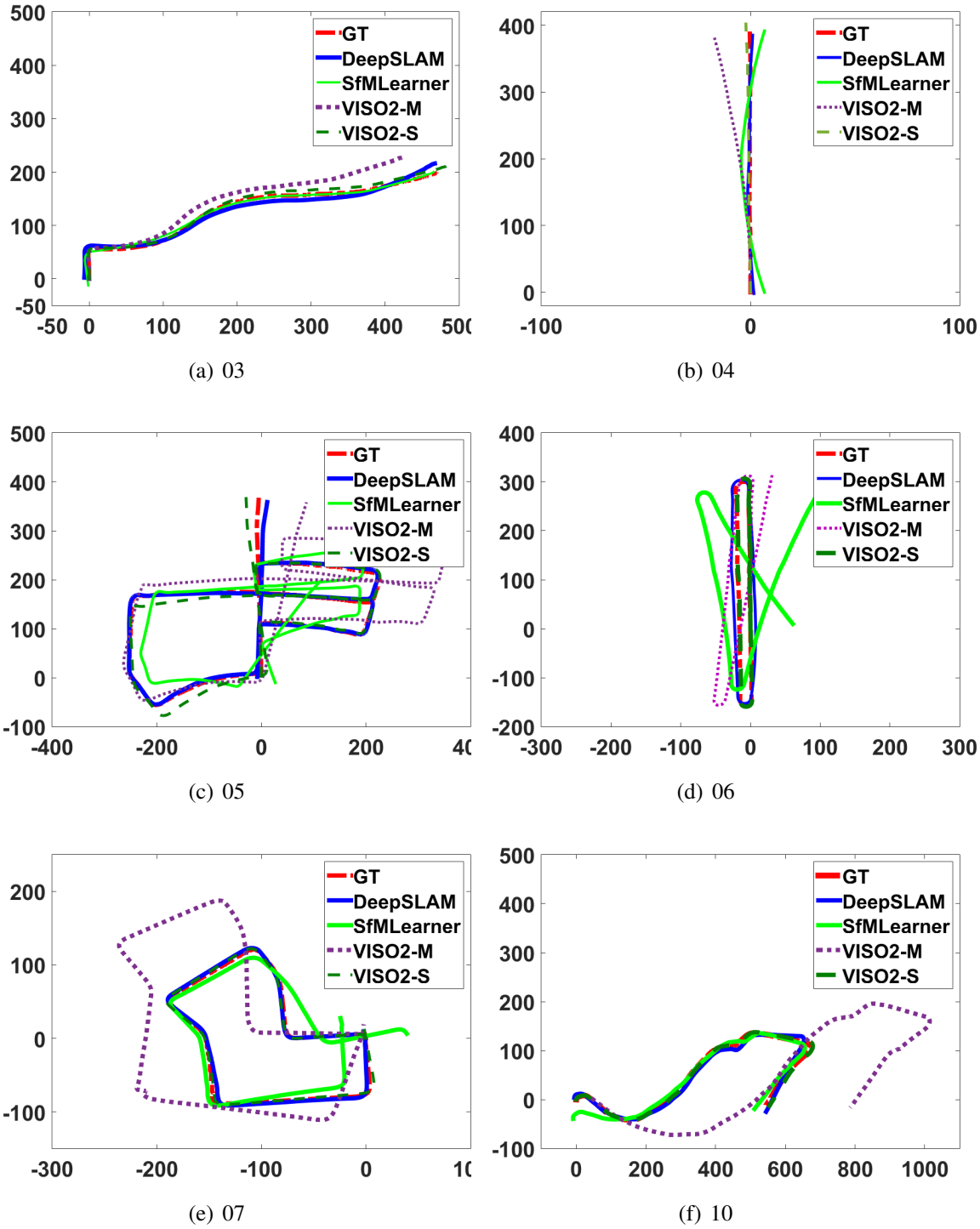
To achieve a better training performance, some measures on data augmentation were taken, such as left-right image augmentation, rotational data augmentation and image color augmentation. For left-right image augmentation, we used the left-right camera motion consistency as mentioned in section IV-A-3. For rotational data augmentation, we increased the proportion of rotational data. This is because the portion of rotational data is significantly smaller than that of the translation. For image color augmentation, we performed random brightness in the range [0.9, 1.1], random gamma in the range [0.9, 1.1], and random color shifts for three color channels in the range [0.9, 1.1] to mitigate possible overfitting.

5.6.1 Pose Accuracy Performance on KITTI

We first evaluated the accuracy performance of our DeepSLAM system on KITTI Odometry dataset [30]. KITTI dataset includes the ground-truth for 11 video sequences (00-10). The full system of DeepSLAM includes the Tracking-Net with loop closure detection and graph optimization. We first use KITTI sequences 00-02, 08-09, 11-21 as the training datasets, and use KITTI sequences 03-07, 10 as the testing datasets.

The detailed quantitative results are listed in Table 5.1. We used the standard evaluation method provided along with KITTI dataset: average translational root-mean-square error (RMSE) drift (%) and average rotational RMSE drift ($^\circ/100$m) on length of 100m-800m. We also added two data-driven learning methods (ESP-VO and SfMLearner) and three model-based methods (monocular ORB-SLAM, monocular VISO2-M and stereo VISO2-S) into the table for comparison. VISO2-M and monocular ORB-SLAM did not work with resolution 416×128, and we used input images with size 1241×376. For stereo methods, VISO2-S also used input images with size 1241×376. ESP-VO is a supervised learning method, and it used KITTI sequences 00, 02, 08, 09 for network training. SfMlearner is an unsupervised method, and it used sequences 00-08 for network training. The best tracking results among learning methods are made in bold.

As shown in the table, our DeepSLAM outperforms ESP-VO and SfMLearner in terms of tracking accuracy. When compared with ESP-VO, we can use more datasets for network



(a) 03 (b) 04

(c) 05 (d) 06

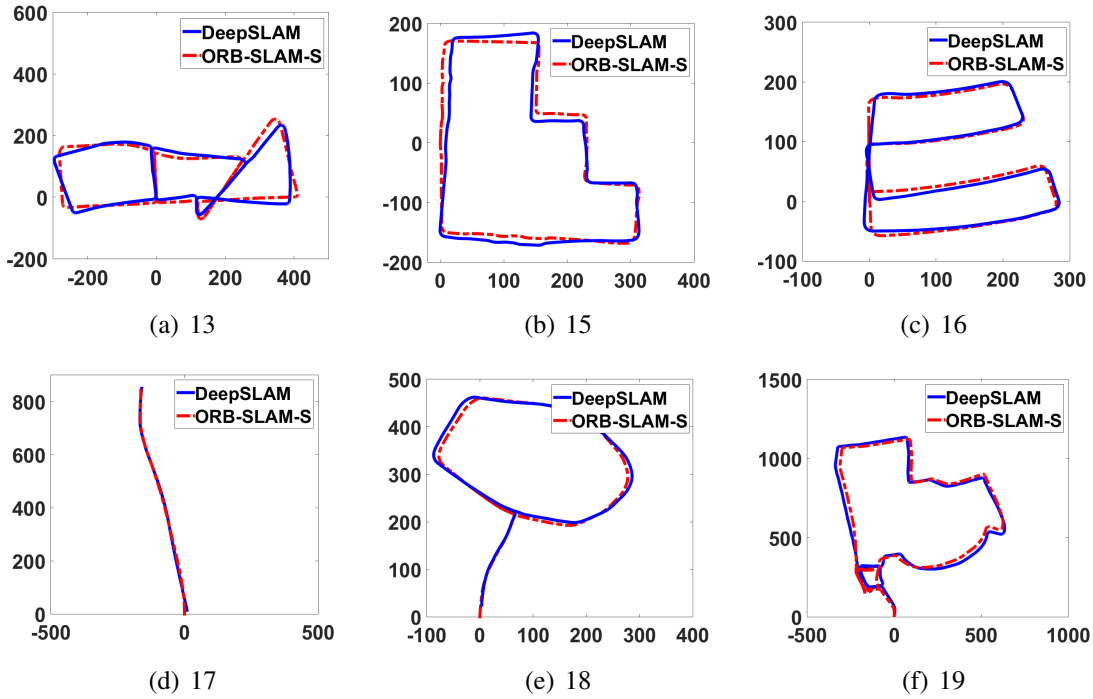
(e) 07 (f) 10

Figure 5.5: Trajectories on KITTI sequences 03-07, 10 using our DeepSLAM system (best viewed in color). KITTI sequences 00-02, 08, 09, 11-21 are used for our network training. Trajectories with SfMLearner [1], VISO2-M [2] and VISO2-S [2] are also plotted for comparison.

Table 5.1: Tracking results on KITTI dataset with our proposed DeepSLAM system. Except for our DeepSLAM, two data-driven learning methods (ESP-VO [3] and SfMlearner [1]) and three model-based methods (ORB-SLAM, VISO2-M and VISO2-S) are added into the table for comparison. Note that monocular VISO2-M and ORB-SLAM did not work with image size 416 $\times$ 128, so their results with image size 1242$\times$376 are shown here. Our DeepSLAM and SfMLearner use images with 416$\times$128. The best results among data-driven learning methods are made in bold.

Seq.	Monocular										Stereo	
	DeepSLAM (416×128)		ESP-VO [3] (1242×376)		SfMLearner [1] (416×128)		ORB-SLAM [6] (1242×376)		VISO2-M [2] (1242×376)		VISO2-S [2] (1242×376)	
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
03	7.66	4.30	**6.72**	6.46	13.08	**3.79**	0.67	0.18	8.47	8.82	3.21	3.25
04	**4.56**	**1.90**	6.33	6.08	10.68	5.13	0.65	0.18	4.69	4.49	2.12	2.12
05	**3.25**	**1.31**	3.35	4.93	16.76	4.06	3.28	0.46	19.22	17.58	1.53	1.60
06	**4.97**	**1.53**	7.24	7.29	23.53	4.80	6.14	0.17	7.30	6.14	1.48	1.58
07	4.71	**1.84**	**3.52**	5.02	17.52	5.38	1.23	0.22	23.61	29.11	1.85	1.91
10	**8.35**	**3.93**	9.77	10.2	14.36	3.98	7.27	1.00	41.56	32.99	1.17	1.30
mean	**5.58**	**2.47**	6.15	6.66	15.99	4.52	3.21	0.37	17.48	16.52	1.89	1.96

- t_{rel}: average translational RMSE drift $(\%)$ on length of 100m-800m.
- r_{rel}: average rotational RMSE drift $(^\circ/100m)$ on length of 100m-800m.



(a) 13 (b) 15 (c) 16

(d) 17 (e) 18 (f) 19

Figure 5.6: Trajectories on KITTI dataset using our DeepSLAM system. There is no ground-truth for these trajectories. We plotted trajectories with stereo ORB-SLAM (ORB-SLAM-S) for reference. ORB-SLAM-S uses images with 1242 $\times$ 376. KITTI sequences 00-08 are used for network training.

training as our DeepSLAM does not need datasets with annotated ground truth. This may explain the reason that our DeepSLAM outperforms ESP-VO. When compared with SfM-Learner, our DeepSLAM system adopts more carefully designed spatial and temporal losses functions, takes RCNN as Tracking-Net architecture and introduces Loop-Net as a graph based optimization mechanism. This may explain why our DeepSLAM outperforms SfM-Learner. Our DeepSLAM also outperforms monocular VISO2-M, but its performance is not as good as ORB-SLAM and stereo VISO2-S as DeepSLAM can not maintain the local map and global map like ORB-SLAM. The estimated trajectories on sequences 03-07, 10 with above methods are plotted in Fig. 5.5. As shown in the figure, the trajectories from our DeepSLAM achieves good performance in terms of pose estimation.

We also used KITTI sequences 00-08 for network training and the rest sequences for testing. There are no ground-truth provided for KITTI sequences 11-21, thus we plotted trajectories with stereo ORB-SLAM (ORB-SLAM-S) for reference. The trajectories of sequences 13, 15-19 are plotted in the figure, as shown in Fig. 5.6, the performance of our DeepSLAM is comparable with ORB-SLAM-S.

5.6.2 Robustness Performance on Challenging Scenes

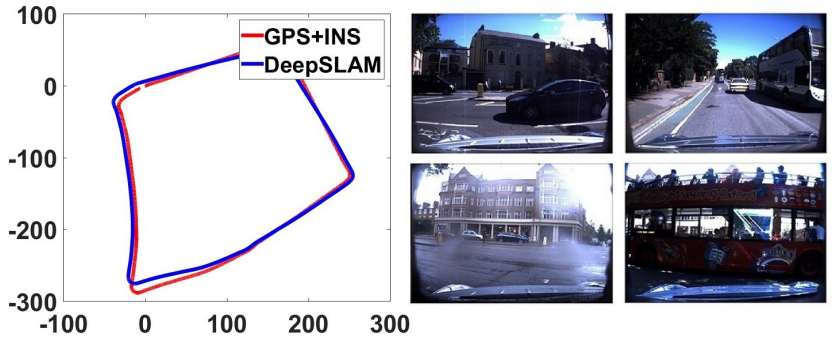
Robustness is a significant factor for wider applications of visual SLAM. We used the public RobotCar [32] dataset to test the robustness of our proposed DeepSLAM system. The RobotCar dataset was collected in different environments over 1 year. We chose some datasets collected in challenging environments to test our system. Dataset (b) in Fig. 5.7 was used for training the Tracking-Net and Mapping-Net.

As shown in Fig. 5.7, the challenging scenes include image distortion, excessive exposure, night-time, bad white balance, and raining. These scenes are difficult for model based visual SLAM to perform accurately and robustly. Model-based SLAM methods are sensitive to camera parameters and are fragile when facing with challenging scenes. Fig. 5.7(a) is the situation with image distortion. Fig. 5.7(b) is the situation that there is excessive exposure in parts of the trajectory. Fig. 5.7(c) is the situation that the data is collected at night while raining. Fig. 5.7(d) is the situation that the images are collected while raining. No ground-truth is provided. We compared our results with the trajectories collected by GPS/INS. For Fig. 5.7(c), the GPS signal was poor due to the rain. For Fig. 5.7(d), the GPS/INS almost did not work due to the terrible weather, and we plotted the trajectory with our DeepSLAM in Google map for reference. As shown in the figure, our DeepSLAM demonstrated a resilient behavior when encountering these challenging scenes.

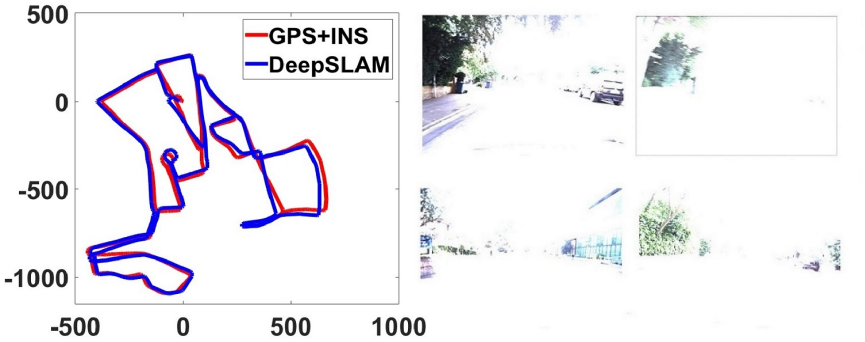
Table 5.2 shows the results from DeepSLAM, LSD-SLAM and stereo ORB-SLAM on challenging scenes of RobotCar dataset. The results from LSD-SLAM and ORB-SLAM are similar with the results reported by Pascoe et al. [26]. When encountering the challenging

Table 5.2: Robustness performance on challenging scenes of RobotCar dataset. ✓ represents working well, × represents losing tracking.

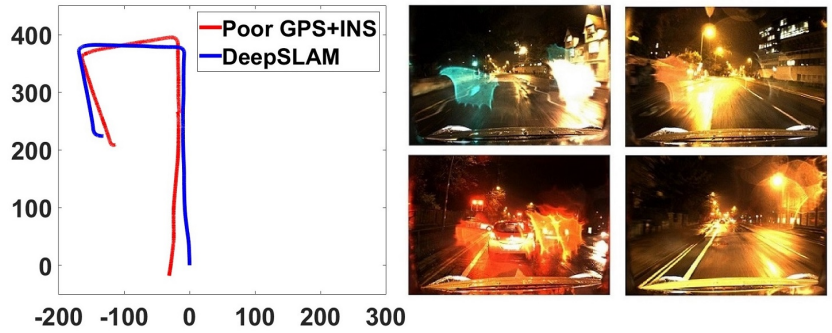
Methods	Dataset	Environments			
		Overcast	Sun	Rain	Night
LSD-SLAM [15]	RobotCar	×/✓	×	×	×
ORB-SLAM-S [6]	RobotCar	✓	×	×	×
DeepSLAM	RobotCar	✓	✓	✓	✓



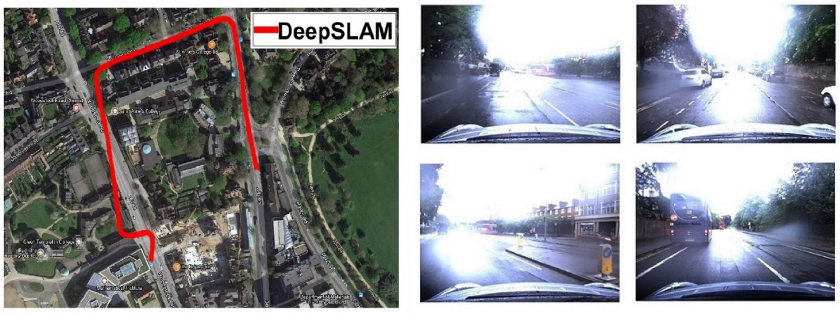
(a) 2014-06-26-08-53-56



(b) 2015-08-12-15-04-18



(c) 2014-11-21-16-07-03 (seq. 10)



(d) 2015-05-29-09-36-29 (seq. 06)

Figure 5.7: Robustness performance of our DeepSLAM on some challenging environments in RobotCar dataset. The left part of each subfigure shows the trajectory produced from our DeepSLAM, and the right part is the testing images taken in different locations. (a) Images with distortion, (b) Images with excessive exposure, (c) Images collected at night while raining, (d) Images collected while raining.

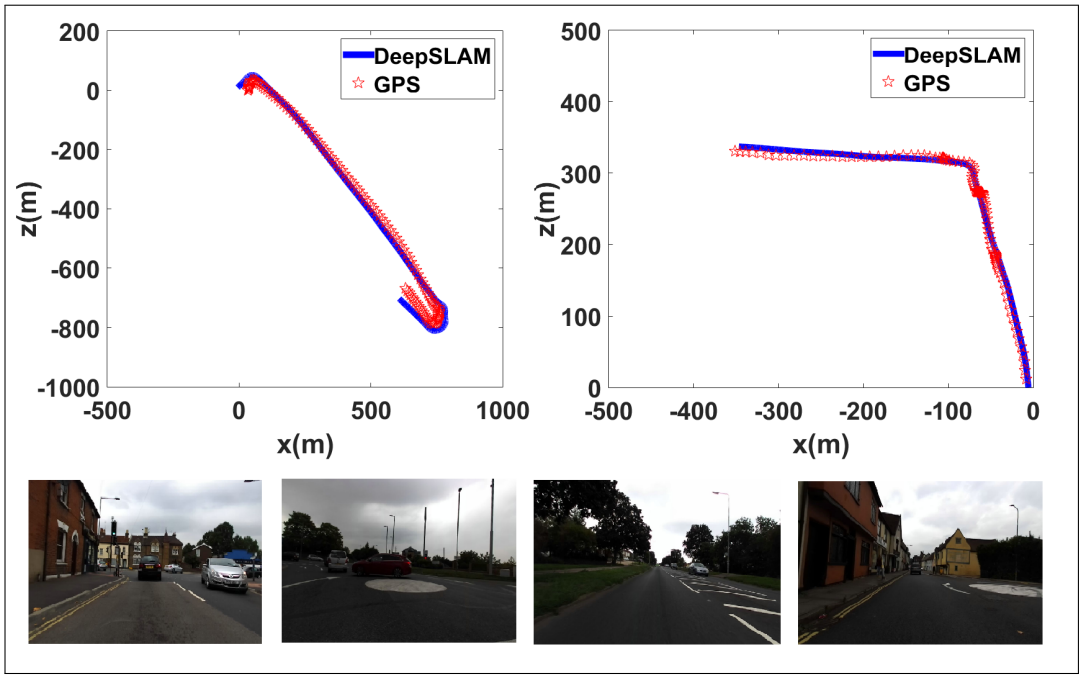


Figure 5.8: Testing on our self-collected dataset using a low-cost ZED camera.

scenes such as raining, night-time and bad white balance, LSD-SLAM and ORB-SLAM can hardly work, but our DeepSLAM works well and performs more robustly than others.

5.6.3 Testing with a Low-cost Camera

We also used a low-cost stereo camera–ZED to collect some data ourselves and tested our system. We used a laptop, a cheap GPS and a ZED camera to collect outdoor data. No other types of equipment were used, and we did not have the ground-truth. We used the GPS data to provide the reference. The trajectories from our DeepSLAM and GPS are plotted in Fig. 5.8. The weather was cloudy when we collected the data, so the images are quite dim. As shown in the figure, our DeepSLAM still works well.

5.6.4 Outliers Rejection

As introduced above, We used the photometric error maps and geometric error maps (3D point-cloud registration error maps) from monocular image sequences to generate the loss mask and uncertainty. The loss mask can reject the outliers and refine the estimated poses. This is due to the fact that the 3D registration error map includes depth and pose estimation information, and the photometric error map includes photometric information, depth and pose estimation information. The uncertainty is related to the mean of the error map, which

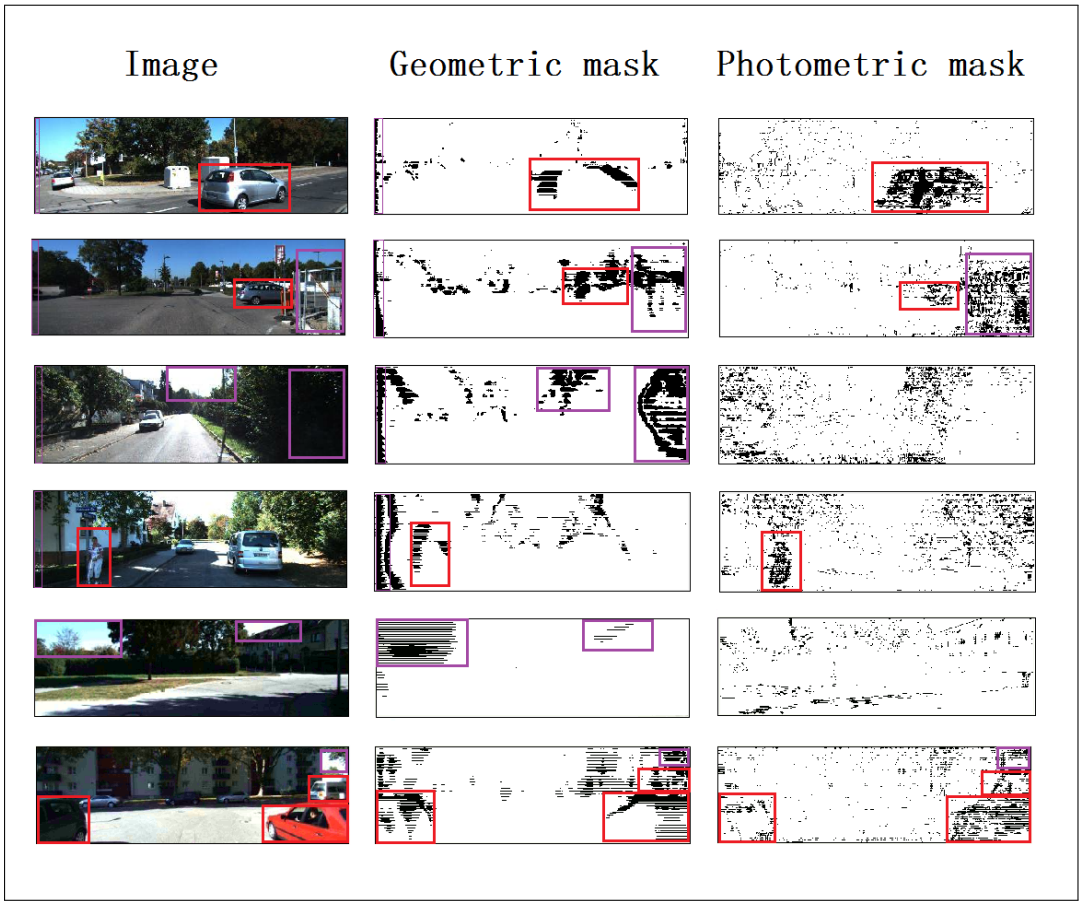


Figure 5.9: Geometric mask and photometric mask for outliers rejection. The red boxes represent moving objects and the purple boxes represent depth values with high uncertainty.

is used to automatically select the size of the mask.

Fig. 5.9 shows the 3D registration error mask and the photometric error mask. The red boxes in the figure are moving objects in the scenes, such as pedestrians and vehicles. The purple boxes in the figure are depth values with high uncertainty. These values tend to be sky, extreme dark places, edges of objects or non-overlap areas of left-right images. It is very hard for the network to recover the real depth value of these places, and therefore the estimated depth value has a high uncertainty. In our system, the photometric mask is directly related to moving objects, and the geometric mask is related to estimated depth values with high uncertainty.

5.6.5 Depth Estimation and 3D Reconstruction

The Mapping-Net of our DeepSLAM can also produce the scaled dense depth map. Fig. 5.10 shows some raw RGB images and the dense depth maps generated by our DeepSLAM.

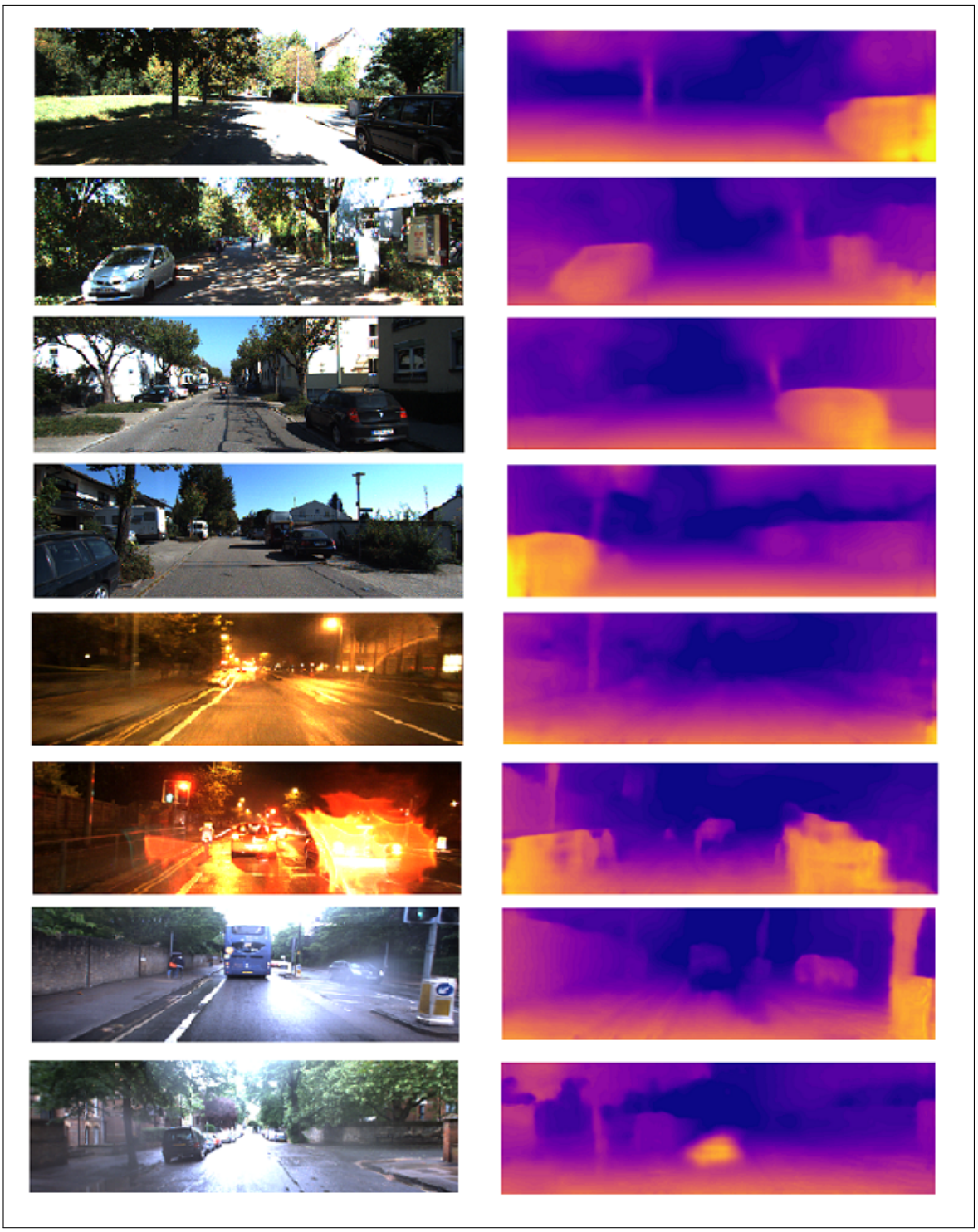


Figure 5.10: The estimated depth maps with our Mapping-Net. The above four rows are the images from KITTI dataset. The below four rows are the images from challenging scenes in RobotCar dataset.

Table 5.3: Depth estimation results on KITTI using the split of Eigen et al. [42].

Methods	Dataset	Input size	Supervision	Scale	Network	Capped depth	Error metric			
							Abs Rel	Sq Rel	RMSE	RMSE log
Eigen [42]	K (raw)	612×184	✓	✓	VGG	80m	0.214	1.605	6.563	0.292
Monodepth [12]	K (raw)	512×256	×	✓	ResNet-50	80m	0.148	1.344	5.927	0.247
SfMLearner [1]	K (raw)	416×128	×	×	VGG	80m	0.208	1.768	6.856	0.283
SfMLearner [1]	K (raw)	416×128	×	×	VGG	50m	0.201	1.391	5.181	0.264
DeepSLAM	K (odo)	416×128	×	✓	VGG	80m	0.1724	1.659	6.362	0.259
DeepSLAM	K (odo)	416×128	×	✓	VGG	50m	0.164	1.288	4.782	0.204
DeepSLAM	K (odo)	416×128	×	✓	VGG	30m	0.147	0.834	3.031	0.207

Figure 5.11: Reconstructed 3D map with our DeepSLAM system.

The above four rows are the estimated depth maps on KITTI dataset, and the below four rows are the estimated depth maps of the challenging scenes on RobotCar dataset. As shown in the figure, the depths of cars, trees and trunks are clearly visible. Further our Mapping-net demonstrates a robust performance on depth estimation even facing with challenging scenes, such as night-time and raining. Note that there is a car in the right part of both the sixth and the seventh images. Although the quality of the images is poor, and there are some apertures and excessive exposures, our Mapping-Net can successfully estimate the depth of these cars.

The detailed quantitative depth estimation results are listed in Table 5.3. RobotCar dataset does not provide the ground-truth for depth maps. We used KITTI dataset to evaluate the performance of our Mapping-Net quantitatively. As shown in the table, our method outperforms the supervised one [42] and the unsupervised one without scale [1], but performs not as good as [12]. This could be caused by a few reasons. First, we only used parts of KITTI dataset (KITTI odometry dataset) for training while all other methods use full KITTI dataset to train their networks. Second, [12] used higher resolution (512×256) input and

a different net (ResNet-based architecture). Third, the temporal image loss we used could introduce some noise (such as moving objects) for depth estimation.

Benefit from the powerful ability of pose estimation and depth estimation with our DeepSLAM system, we can also reconstruct the dense 3D point-cloud of the scenes with a monocular camera. Fig. 5.11 shows some 3D dense maps with our DeepSLAM system.

5.7 Conclusions

Most state-of-the-art SLAM algorithms rely on geometric models and optimization engines to estimate the structure of environment and the motion of camera. This chapter approaches to the problem from a data driven perspective, i.e. training deep networks with existing data sets. Our evaluation results show the data driven approach DeepSLAM outperforms the monocular model based SLAMs. Further we demonstrated that DeepSLAM is versatile and robust.

The system architecture of DeepSLAM mimics that of model based SLAMs. The important geometric models and constraints are respected and embedded into the network architecture and the loss function. This provides a guarantee for estimation accuracy.

DeepSLAM falls within an unsupervised learning framework. This distinguishes itself from supervised deep leaning approaches to SLAM. There are more unlabeled data sets available than labeled data sets, which results to the superior performance of DeepSLAM over supervised learning SLAM.

In this chapter, we can construct dense 3D map with deep learning methods. However, we still can not perceive the semantic information from the 3D map. In the next chapter, we plan to study the semantic mapping that combines deep learning methods with SLAM.

Chapter 6

Semantic Scene Mapping with Spatio-Temporal Deep Neural Network for Robotic Applications

In order to perceive the semantic information of the environments, in this chapter, we combine deep learning with SLAM and propose a semantic pixel-wise mapping system for potential robotic applications. A novel spatio-temporal deep neural network for semantic segmentation is proposed. The network is a dual-stream architecture which includes a spatial stream with images as input and a temporal stream with image difference as input. Then based on the semantic segmentation results, a 3D semantic map can be built up by using the 3D point cloud from a SLAM algorithm. The proposed system is evaluated on both public benchmark dataset and self-collected-labelled dataset. The results show better performance on semantic perception than previous spatial only neural networks.

6.1 Introduction

Semantic scene mapping is a challenge and significant task for autonomous navigation, localisation, robot-environment interaction, etc. As it can provide semantic information and understanding of the environments, it is widely investigated in robotics, computer vision, Augment Reality (AR), and Virtual Reality (VR). Semantic pixel-wise segmentation is the basis of semantic scene mapping, and has gained a great success due to the spectacular de-

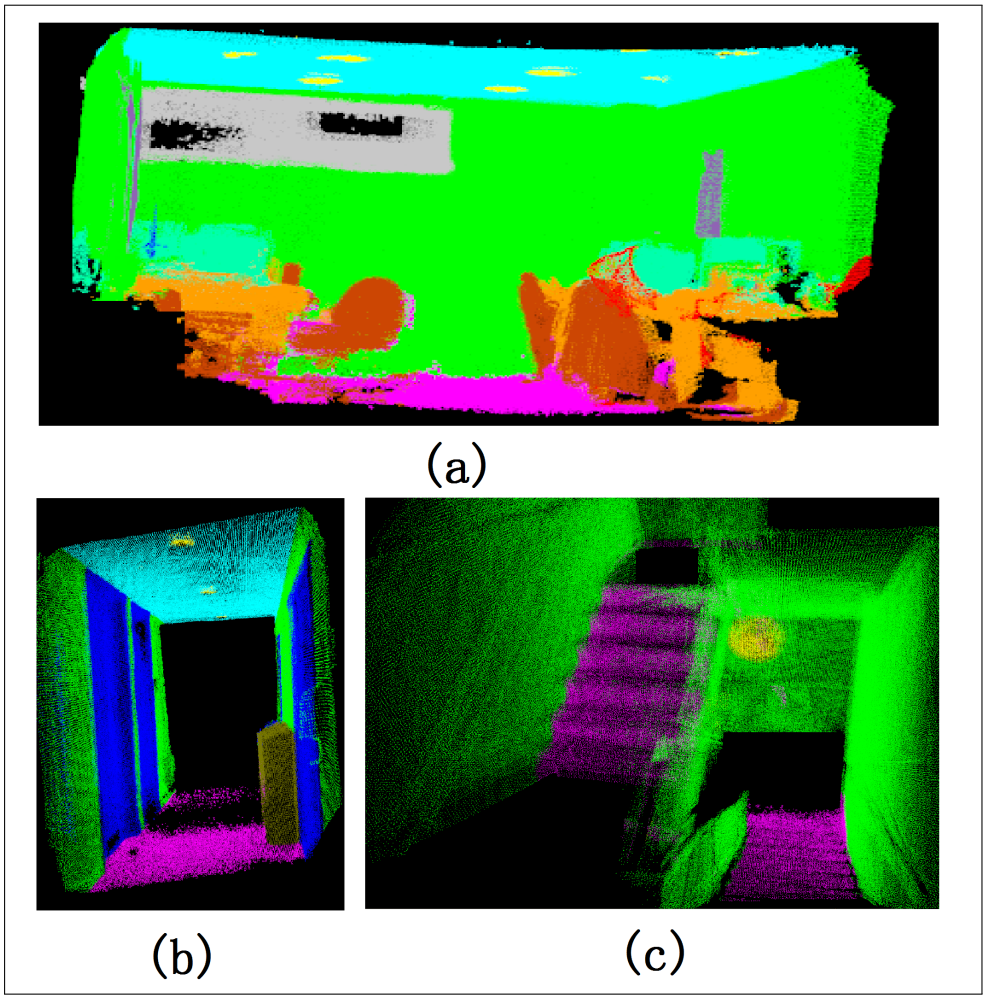


Figure 6.1: 3D semantic maps. (a) 3D semantic map of a room (b) 3D semantic map of a corridor (c) 3D semantic map of a staircase.

velopment of deep Convolutional Neural Networks (CNNs) in the past few years.

Since a Fully Convolutional Network (FCN) [47] was proposed for scene segmentation and achieved the state-of-the-art pixel-wise segmentation performance, researchers have proposed many different kinds of deep convolutional neural network architectures derived from the FCN. These CNNs can learn the spatial information from images and obtain pixel-wise understanding of the environments. But most of them are for static images and time-consuming. They can hardly have a real-time performance for the reason that pixel-wise image segmentation with FCN usually takes about 200ms. This work aims to produce a semantic pixel-wise segmentation from video streams, which not only contain the spatial information but also the temporal information for potential robotic applications. The temporal information is valuable in the pixel-wise segmentation as pixels in adjacent images

have some forms of corresponding geometry constraints. In this chapter, we present a novel spatio-temporal neural network for semantic segmentation. In addition to the images as spatial information, the image difference between two consecutive images is used as the temporal information for the network. And the computational burden is reduced by selecting only keyframes for segmentation while the non-keyframe segmentation is predicted by the results from keyframes.

It is known that 3D point cloud maps produced by visual SLAM [120] [121] algorithms only represent the occupation information and are less meaningful. However, 3D semantic maps can provide more meaningful information for robotic applications. Semantic pixel-wise segmentation can be combined with a 3D point cloud map to yield a 3D semantic map. In this chapter, we also present how to yield a 3D semantic map through this combination. This is achieved by using a SLAM algorithm to construct a 3D point cloud map and then labelling each point in the point cloud by using the semantic segmentation result. Some 3D semantic maps are shown in Fig. 6.1.

Our main contributions in this chapter are summarised as follows:

- We propose a novel spatio-temporal neural network for semantic scene segmentation. Both images and image differences between two consecutive images are taken as the inputs for the network. In this way, both the spatial and temporal information are considered for semantic segmentation.

- We propose to select the keyframes for semantic segmentation while the non-keyframe segmentation is established by the prediction result from keyframes. In this way, the computational burden when handling with video streams can be reduced and the real-time performance can be improved.

- We develop a practical semantic scene mapping system by using a visual SLAM algorithm combined with the result from semantic segmentation. Qualitative and quantitative experiments based on the public dataset and our own dataset are presented. The 3D semantic map of our dataset is built up for robotic applications.

In the following section, we will give a review of the related work. In section 6.3, we will provide an introduction to the architecture of the proposed CNN for semantic segmen-

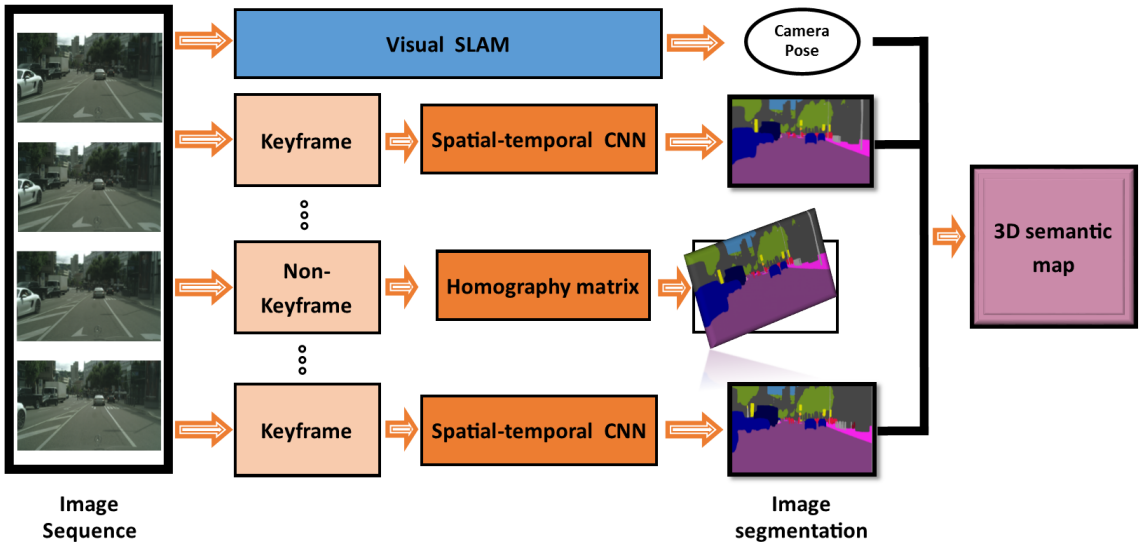


Figure 6.2: System overview of the proposed semantic mapping system. For images from a video steam, we classify them as keyframes and non-keyframes according to the geometric constraints and time interval. For keyframes, we adopt the proposed spatio-temporal neural network to perform the semantic segmentation. For non-keyframes, we estimate the homograghy matrix (2D projective transformation) between the last keyframe and current frame. Then the non-keyframe segmentation is predicted by the results from keyframes. At the same time, a visual SLAM algorithm is used for camera pose estimation. In the end, the 3D semantic map is constructed by our system. The input of the system is RGB images and the output of the system is 3D semantic pointcloud.

tation, and present the fusion method for spatial and temporal information. The procedures of keyframe selection, segmentation prediction for non-keyframes, and semantic map construction will be also described in this section. Section 6.4 will demonstrate the experimental results on different datasets using the proposed system. In section 6.5, we will give a summary conclusion and the future work we would like to investigate.

6.2 Related Work

In this part, we will review the research on semantic scene segmentation using deep Convolutional Neural Networks (CNNs). Chapter 2 has already provided some general overviews on this part. Here, more detailed techniques will be focused. We first review the semantic segmentation with single stream (spatial only) network. The research on multi-stream networks are followed. Then we review the research on semantic mapping with SLAM.

6.2.1 Spatial Image Segmentation with CNNs

Semantic segmentation is a traditional field in computer vision [152]. Long et al. [47] first proposed Fully Convolutional Network (FCN) which successfully applied CNN to pixel-wise segmentation, depth estimation, and optical flow estimation. Liu et al. [84] proposed ParseNet which enabled the wider view of the network. Badrinarayanan et al. [153] presented an encode-decoder architecture called SegNet based on VGG net [88] and FCN [47]. Afterwards, Chen et al. [91] [79] [92] proposed to use the very deep ResNet [89], dilated convolution, and fully connected CRFs to segment images. Zhao et al. [81] proposed the Pyramid Scene Parsing Network (PSPNet) which won the ImageNet scene parsing challenge 2016 [38]. Besides, Tu et al. [154] introduced optical flow as the temporal information. By combining this motion-based saliency method with a region-based image saliency method, they demonstrated a spatio-temporal system for object segmentation. Doborjeh [155] made use of spatio-temporal EEG data and used spiking neural networks to realize the classification of signal.

6.2.2 Multi-stream CNNs

With regard to temporal information, Wang et al. [156] [157] proposed a novel temporal segmentation network to exploit the optical flow along with color images to enhance the performance of action recognition in video streams. The proposed network demonstrated a high performance in the Large Scale Activity Recognition Challenge 2016. Eitel at el. [141] and Schwarz at el. [142] rendered disparity images with a color palette, and used a two-stream CNN to obtain a better performance with RGB-D cameras. Hazirbas et al. [95] proposed FuseNet [95] that exploited both color features and depth features for segmentation. Valada et al. [96] [97] presented AdapNet for semantic segmentation in adverse environments. The proposed network is a multi-stream architecture with multi modalities as inputs.

To the author's best knowledge, no one has used the temporal information for semantic pixel-wise segmentation along with CNNs so far. In this chapter, we will investigate how to combine both spatial and temporal information for semantic pixel-wise segmentation.

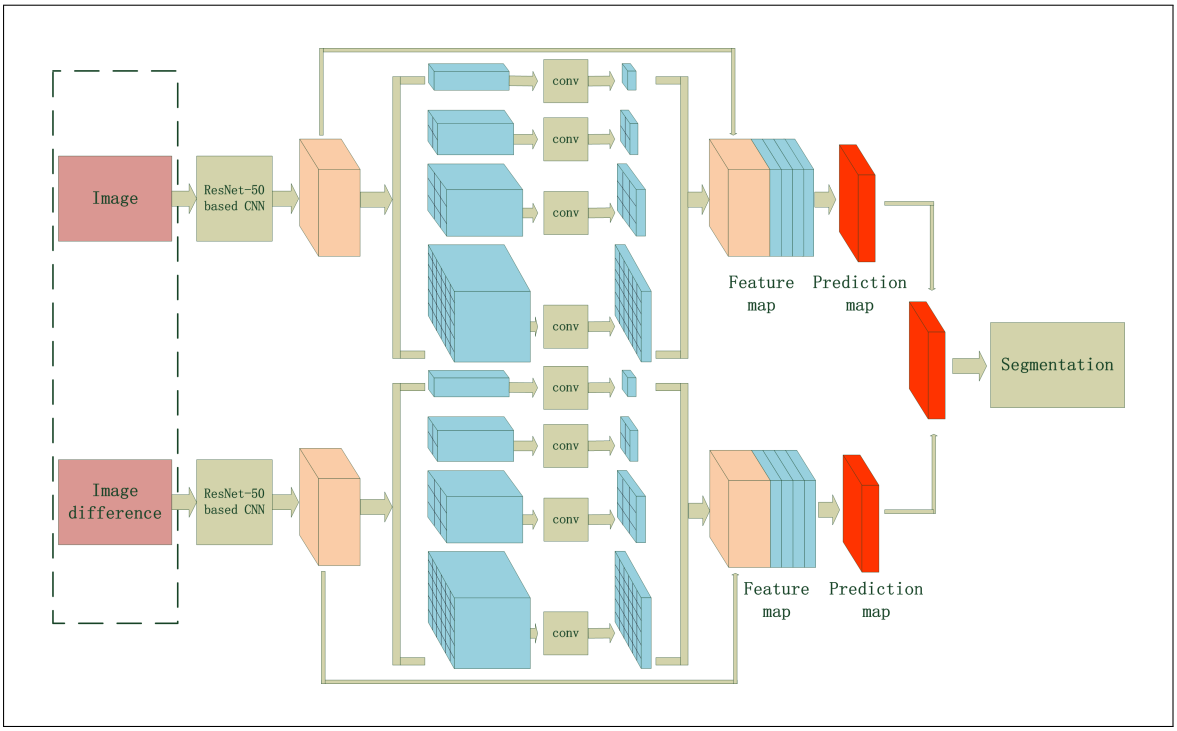


Figure 6.3: Spatio-temporal neural network architecture. The color image and the image difference are fed into the network. Then the prediction probability maps are learned through the separated networks and fused together for image segmentation. The size of network input for each stream is $3 * 521 * 521$, and the size of network output is $1 * 521 * 521$.

6.2.3 Semantic mapping with SLAM

For robotic applications, it is significant to locate the robot itself and perceive the semantic environments simultaneously [158]. Salas-Moreno et al. [22] presented a planar SLAM system which could detect the planar in the environment and yield a planar map. A demonstration which replaced a wall with a facebook web page was shown by the proposed system. They also proposed a SLAM system called SLAM++ [16]. The system detected objects such as chairs and desks and then utilised these objects for the localisation. However, only planes, desks and chairs were extracted and perceived by the above SLAM system. More extensive semantic map could be built up for robotic applications by combining visual SLAM algorithms with the semantic segmentation. This is the research objective of this chapter.

6.3 Approach

The proposed system is shown in Fig. 6.2. The details of the spatio-temporal neural network are discussed first. Then the geometry based segmentation prediction for non-keyframes

will be followed. At last, we will present our method for 3D semantic mapping with visual SLAM technology.

6.3.1 Spatial Segmentation Network Architecture

The basic spatial neural network architecture we use in this chapter is the PSPNet [81] which had an excellent performance in the ImageNet scene parsing challenge 2016. Its main advantage is the combination of very deep ResNet, dilated convolution and pyramid pooling module.

At the very beginning, researchers preferred to use the standard convolution followed by pooling to extract the features and then adopted the deconvolution to recover information from the feature maps. However, this method caused loss of original detail due to the use of pooling. It also needs to use intensive computational power and large memory. [91] proposed the dilated convolution which was also called atrous convolution. Its basic idea is to implement the convolution for features maps with holes. By using the dilated convolution, the kernel of convolution is widened to some extent, the field-of-view is effectively enlarged, and the features are extracted and maintained efficiently. Further, it does not require any additional computation and memory. In order to explain the dilated convolution explicitly, we take an one dimension signal for example. Let $x[i]$ be a 1-D input signal with length K, $y[i]$ is the output of the dilated convolution, so the dilated convolution [91] can be defined as:

$$y[i] = \sum_{k=1}^{K} w[k] * x[i + r \cdot k] \tag{6.1}$$

Where rate r represents the stride that is used by the dilated convolution to sample the input signal $x[k]$, $w[k]$ is the weight matrix, $*$ is the multiplication operation. In this way, the dilated convolution has the functions which combine the standard convolution, pooling, and deconvolution. Compared with the traditional methods, it also enlarges the resolution of resulting feature map while maintaining more information. In particular, the standard convolution is a special case of the dilated convolution where the rate r is 1.

The pyramid pooling module is another important factor that the PSPNet [81] outperforms other networks in semantic segmentation. As shown in Fig. 6.3, the pyramid pooling module is appended after the final feature map to better learn the contextual information.

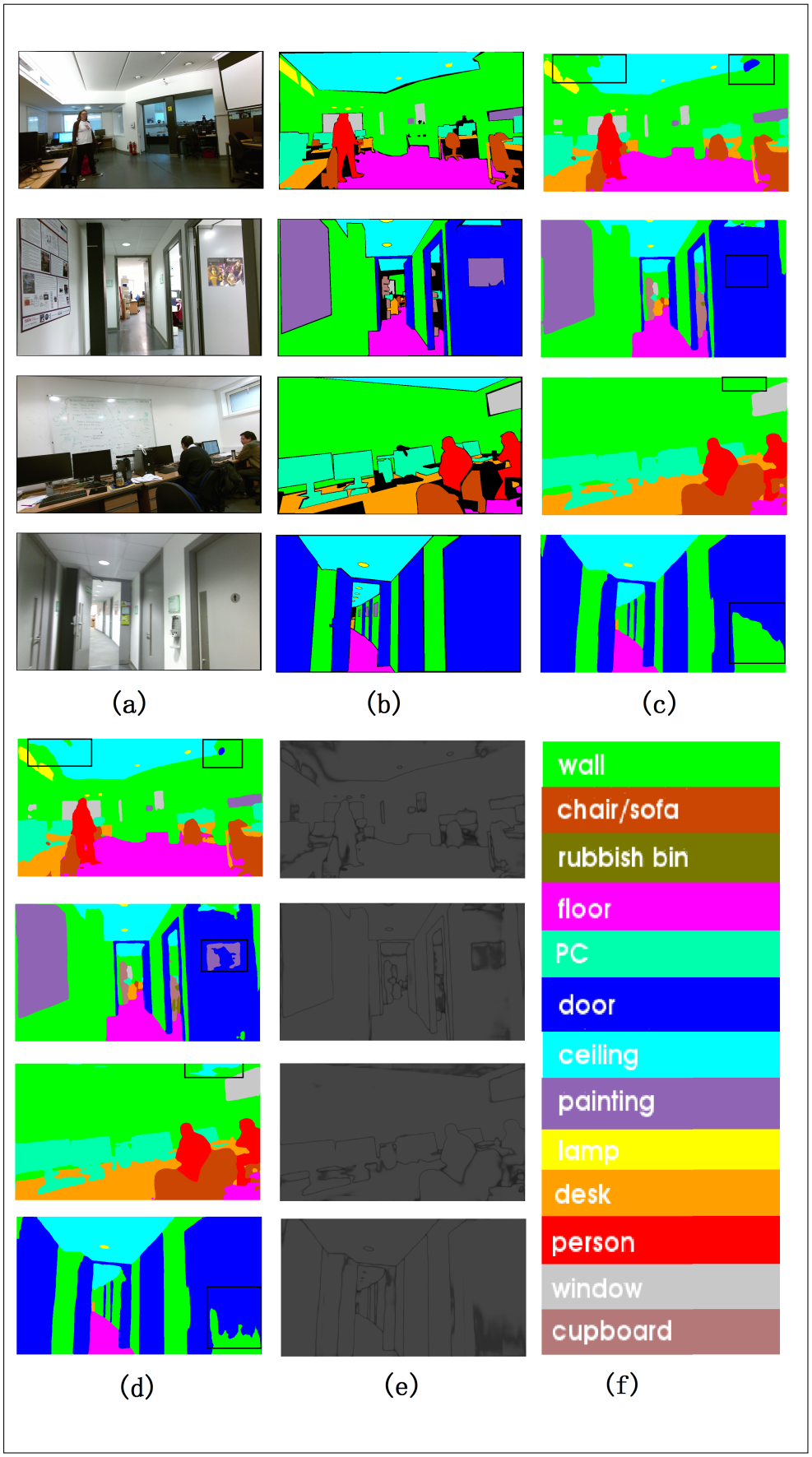


Figure 6.4: Visual comparison on our manually labelled dataset (Essex Indoor). (a) Image (b) Ground truth (c) Spatial PSPNet (d) Spatio-temporal CNN (e) Uncertainty map for segmentation (f) ColorMap.

Four different pooling scales (1, 1/2, 1/3, 1/6) followed by convolutions are applied in this module. Then four hierarchical feature maps are upsampled with bilinear interpolation. Finally, these learned features and the original feature map are concatenated into a new final global feature map to yield the segmentation result. In this way, the sub-region contextual information is better utilised along with the global contextual information.

6.3.2 Proposed Spatio-temporal CNN for Semantic Segmentation

Most existing neural networks take spatial images as input and yield the semantic segmentation as output. In this section, we discuss how to use both spatial and temporal information for semantic pixel-wise segmentation through supervised learning. PSPNet [81] is adopted as the basic single stream network architecture in our system. For dynamic video streams, long memory images (long image sequence) play a less important role in current segmentation. It is hard to use long memory images to improve the segmentation accuracy on account of the fact that there is few pixel correspondences between the images. In contrast, short memory images are valued for semantic segmentation from video streams. Here we use the image difference between two consecutive images as the temporal information. Then we propose a CNN architecture, which has two streams, one is the color image stream to capture the spatial features and another is the image difference stream to capture the temporal features, as shown in Fig 6.3.

By applying convolution and softmax to the final feature maps after the pyramid pooling module, both spatial stream and temporal stream can generate the category prediction $P_s(x)$ and $P_t(x)$ for each pixel separately. We introduce three strategies to fuse the two streams together. (1) Pixel-wise prediction fusion - This fusion method is to treat the pixel-wise segmentation prediction from each stream as an independent normal distribution. Then we can use the element-wise operation such as sum or max to fuse the prediction. (2) Feature map sum - This fusion method is to implement the element operation for the final feature maps from the pyramid pooling module. The feature maps are added together in element level, and then we let the network to learn from the fused feature map. (3) Feature map concatenation - This fusion method is to concatenate the final feature maps from the two separate streams. By stacking them together into multiple channels, the neural network is trained end-to-end and the the feature maps are fine-tuned. Detailed comparisons between

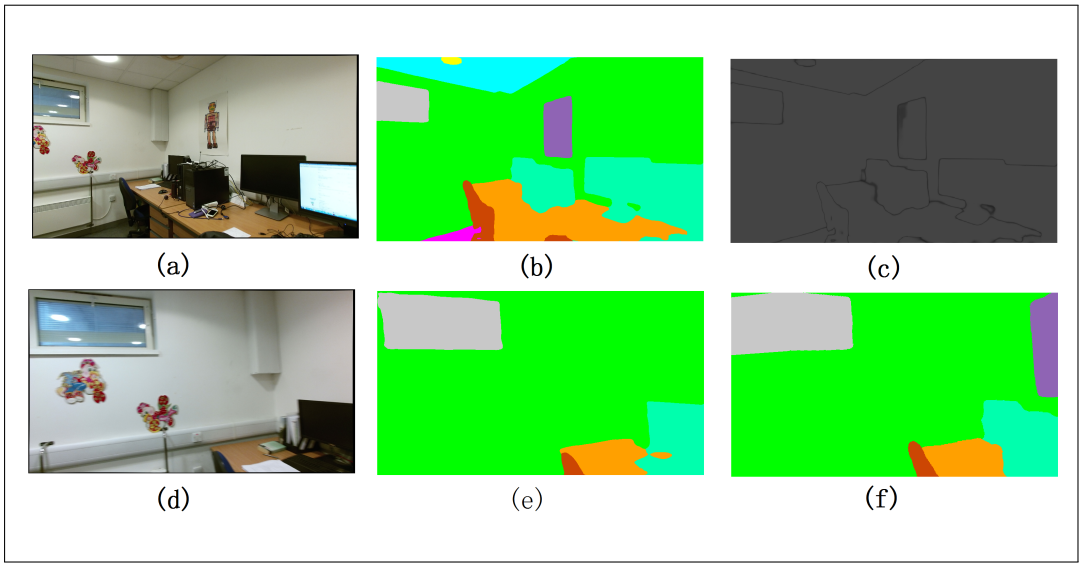


Figure 6.5: Image segmentation prediction with homography matrix. (a) Keyframe (b) Keyframe segmentation with CNN (c) Uncertainty map for keyframe (d) Non-keyframe (e) Non-keyframe segmentation with CNN (f) Segmentation prediction with homography matrix.

these methods are given in 6.4.3.

6.3.3 Image Segmentation Prediction with Homography Matrix

Pixel-level image segmentation with the network is time-consuming. We can not use the network to segment every image for real-time applications. Due to the overlaps between consecutive images, we can just select some keyframes from the image sequence to perform the semantic segmentation with the proposed network. As shown in Fig. 6.2, we classify the images as keyframes and non-keyframes. For keyframe images, the pixel-level image segmentation is performed to yield the segmentation result. For non-keyframe image, the pixel-level segmentation map is predicted from the result of neighbour keyframe images. The segmentation prediction is conducted by using the homography matrix to predict the segmentation map of overlap regions. A 3×3 homography matrix H is computed through OpenCV [117] first. It can map $[u,v]$, the 2D coordinate of a pixel in the keyframe, to $[u',v']$, the corresponding pixel in the non-keyframe. The matrix H is defined as below:

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{6.2}$$

The homography matrix can transform the segmentation map of a keyframe to the predicted segmentation map of the non-keyframe. The matrix $[H_{11}, H_{12}; H_{21}, H_{22}]$ is the rotation term, and the vector $[H_{13}, H_{23}]^T$ is the translation term. The rotation matrix, the translation vector, and the time interval jointly determine whether an image is a keyframe. Specifically, we compare the norm of the rotation matrix, the translation vector and the time interval with the corresponding thresholds. If one of these three is bigger than its corresponding threshold, we choose the frame as a keyframe. Only keyframes are processed to produce the semantic segmentation map. So the overall processing time is saved and the real time performance is improved. Fig. 6.5(f) shows the result of segmentation prediction for Fig. 6.5(d). Although the segmentation performance of Fig. 6.5(f) is not as good as Fig. 6.5(e), but it is still acceptable for robotic applications.

6.3.4 3D Semantic Scene Mapping with a SLAM Algorithm

The visual SLAM algorithm can simultaneously determine the robot pose and constrcut a 3D point cloud map for the environment. But the 3D point cloud map is less meaningful for robotic applications. If each point in the point cloud could be labelled with semantic meaning, a 3D semantic scene map is obtained, which is more meaningful for various robotic applications. That means we can simultaneously determine the robot pose and construct the 3D semantic scene map. In this chapter, we combine the network for semantic segmentation with a visual SLAM algorithm to do this challenging task.

The system include the spatial and temporal deep CNN proposed above and a standard visual SLAM algorithm. They run in parallel. The input to the SLAM algorithm is the images and the output are the robot pose and the 3D point cloud map. For each image, each point in the point cloud is labelled with the corresponding result in the semantic pixel-wise segmentation. Then next image is processed, and the labelled point cloud are merged together as a global 3D semantic map via the transformation of robot pose, i.e. the global semantic map can be obtained as below:

$$Global\ Map: \sum_{i=1}^{n} \mathbf{T}_{cw}\mathbf{X}_c = \sum_{i=1}^{n} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \mathbf{X}_c \tag{6.3}$$

Where $\mathbf{T}_{cw}$ is the 4×4 transformation matrix obtained from the visual SLAM algorithm.

Table 6.1: Segmentation performance comparison with different fusion methods. The unit for pixel-wise accuracy and mean IoU is percentage (%).

Method	Pixel-wise accuracy	Mean IoU
Spatial Stream	88.91	95.93
Temporal stream	82.34	94.12
Feature Map Concatenation	89.21	96.30
Feature Map SUM	89.85	96.46
Prediction MAX	89.46	96.46
Prediction SUM	**90.68**	**96.74**

$\mathbf{T}_{cw}$ is composed of rotation $\mathbf{R}$ and translation $\mathbf{t}$ that can transform points from camera coordinate to world coordinate. $\mathbf{X}_c = (x_i, y_i, z_i, 1)^{\mathbf{T}}$ is the homogeneous position representation of a point in the camera coordinate.

6.4 Experimental Evaluation

In this section, we will evaluate the segmentation performance of our proposed spatio-temporal neural network and present a 3D semantic map system. We will first introduce our manually labelled dataset that was collected from our office. Secondly, we will discuss the fusion methods of spatial stream and temporal stream for segmentation prediction. Following that, the quantitative evaluations based on Cityscapes benchmark dataset will be given by comparing with different segmentation networks. Then the qualitative evaluation will be presented. In the end, the 3D semantic map construction is demonstrated by our proposed system.

The proposed CNN is designed using the Caffe [145] platform, and all experiments are performed on a desktop equipped with one Nvidia GeForce Titan X GPU card and Intel Core i7-4790 4.0GHz CPU.

Table 6.2: Per-class segmentation IoU (%) on Cityscapes dataset. All methods are trained with the fine annotations of training dataset. Among these methods, CRF-RNN [86] and DeepLab [79] use CRFs as the post-processing.

Method	road	swalk	build.	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU
CRF-RNN [86]	96.3	73.9	88.2	47.6	41.3	35.2	49.5	59.7	90.6	66.1	93.5	70.4	34.7	90.1	39.2	57.5	55.4	43.9	54.6	62.5
FCN [47]	97.4	78.4	89.2	34.9	44.2	47.4	60.1	65.0	91.4	69.3	93.9	77.1	51.4	92.6	35.3	48.6	46.5	51.6	66.8	65.3
ParseNet [84]	97.5	78.5	89.5	40.4	45.9	51.1	56.8	65.3	91.5	69.4	94.5	77.5	54.2	92.5	44.5	53.4	49.9	52.1	64.8	66.8
DeepLab [79] (ResNet-101)	**97.9**	**81.3**	90.3	**48.8**	47.4	49.6	57.9	67.3	**91.9**	**69.4**	94.2	**79.8**	**59.8**	**93.7**	56.6	67.5	**57.5**	**57.7**	68.8	70.4
Spatial-PSPNet [81] (ResNet-50)	97.2	79.7	90.7	43.1	51.3	52.0	59.4	69.0	91.4	61.3	94.1	77.4	50.4	93.0	65.4	73.8	53.3	54.9	72.4	70.0
Spatio-temporal CNN	97.4	80.8	**91.2**	47.3	**52.9**	**56.2**	**60.8**	**71.5**	91.7	60.9	**94.2**	77.8	48.7	93.4	**66.8**	**75.1**	57.0	53.3	**73.4**	**71.1**

6.4.1 Indoor Dataset for Scene Segmentation

In this part, we introduce the dataset collected from our office environment in the network building of our university. In order to test the efficiency of temporal information for scene parsing, we need to have a dataset first. Most available datasets only contain discrete images and their ground truth labels. Video stream dataset for semantic segmentation are not readily available. So we have to use a Kinect One camera to build our own dataset for the second floor of our building. The scale of the second floor is about $40m \times 30m$. And the image size of our dataset is 960×540 in pixel level. Both disparity image sequence and color image sequence are provided. Then we manually segment the collected images into 13 categories, as shown in Fig. 6.4. They are wall, chair, rubbish bin, floor, PC, floor, ceiling, painting, lamp, desk, person, and window. The semantic information of these categories is very important for robot navigation and robot-environment interface, especially the semantic information of floor, wall, ceiling, door and person. In addition for training the network, our dataset is also used for 3D semantic scene map construction.

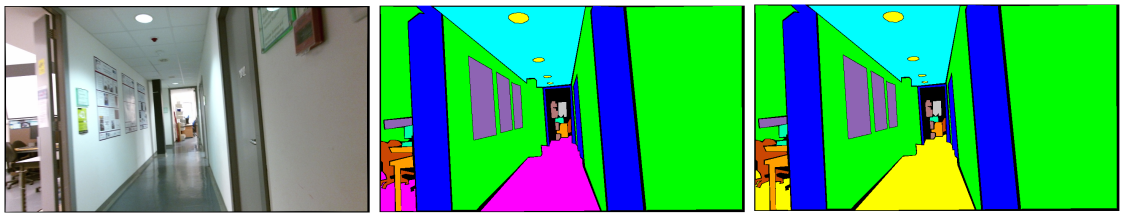
6.4.2 Training Details

We train the two streams separately first. The ResNet-50 [89] based PSPNet [81] architecture is used for two separated streams. The model weights which have already been trained from the ImageNet scene parsing challenge 2016 are used for transfer learning for our dataset. Due to the limitation of GPU memory in our experiments, we choose the ”cropsize” as 521 and the “batchsize” as 3. The “poly” learning rate policy is adopted for training. The base learning rate and the power are set to 0.00025 and 0.9, respectively. The weight decay and the momentum are set to 0.0001 and 0.9, respectively. The iteration number for training the two separated streams is 20000. An auxiliary loss during training is used and the weight of this additional loss is set to 0.4. We also use the data augmentation during training. The image is randomly resized to 0.5 to 2, and the random mirror is also adopted.

For the dual-stream neural network training, we change the base learning rate to 0.0001. The “batchsize” is set to 1 because of memory limitation.

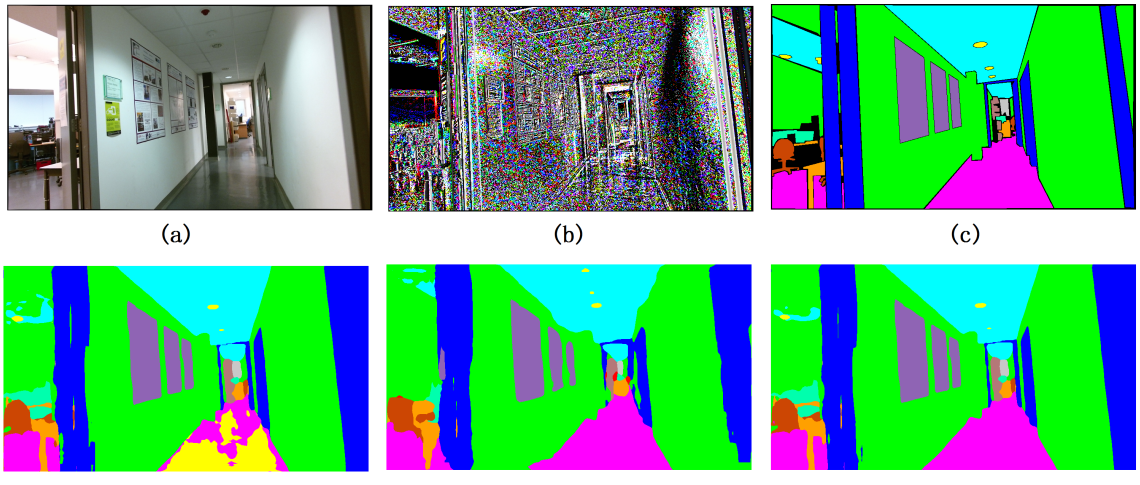
Table 6.3: Per-class segmentation IoU (%) on the Essex Indoor test dataset when trained with the "problem" dataset.

Method	wall	ceiling	floor	lamp	desk	person	chair	PC	door	painting	windows	cupboard	rubbish bin	mIoU
Spatial PSPNet	90.27	89.21	98.95	70.30	95.30	87.19	89.69	91.17	81.26	79.90	88.22	70.27	97.76	86.88
Temporal PSPNet	89.92	87.05	90.49	57.24	75.55	83.26	77.35	79.85	81.39	84.62	87.99	73.30	93.76	81.67
Spatio-temporal CNN	95.45	89.97	96.26	64.18	91.42	94.72	88.33	93.33	93.74	90.10	93.10	83.26	98.83	**90.21**



(a) (b) (c)

Figure 6.6: Train the network with the "problem" dataset. We deliberately labelled the floor in an image as the lamp in the ground truth image. (a) The image in the training dataset. (b) The ground truth with right label in the training dataset. (c) The ground truth with wrong label in "problem" dataset.



(a) (b) (c)

(d) (e) (f)

Figure 6.7: Segmentation results of the networks using "problem" dataset for training. (a) color image in the test dataset (b) difference image in the test dataset (c) ground truth (d) segmentation result with color image as inputs (e) segmentation result with image difference as inputs (f) segmentation result with both color image and image difference as inputs.

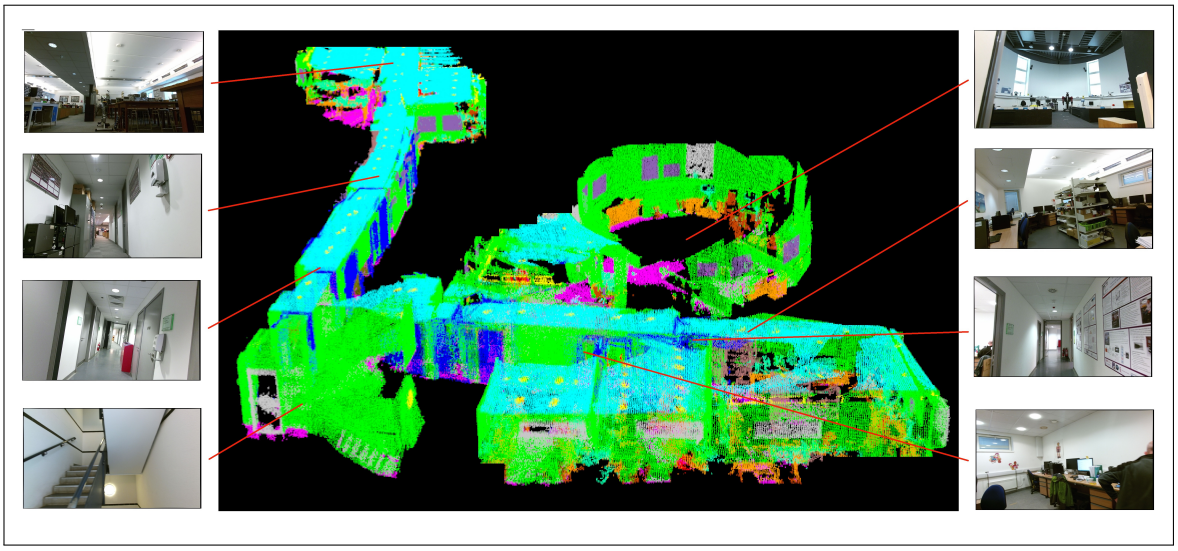


Figure 6.8: 3D semantic map of the second floor in the network building of our university.

6.4.3 Fusion Method

In our proposed network, the image difference is used as the input of temporal stream. The image difference is the subtraction of current keyframe from the previous image. It can maintain the temporal information between two frames. We also tried to use optical flow in the experiments, but found that optical flow only represents moving objects and other information was lost. This leads to the poor performance in semantic segmentation.

In this part, we use our manually labelled dataset (Essex Indoor dataset) to test and compare different fusion methods. As explained in part III-B, we mainly compare three fusion methods here. As shown in table 6.1, pixel-wise prediction sum is the best fusion method for semantic segmentation. The weight of different streams for fusion is an important parameter for segmentation accuracy. In the experiment, the weight of spatial stream is set to 0.7, and the weight for temporal stream is set to 0.3. The stage-by-stage network training mechanism is used for our spatio-temporal CNN. We first train the spatial stream and the temporal stream separately. Then we transfer the learned network weights from the separate single streams to the dual-stream CNN, and finally fine tune the dual-stream CNN.

6.4.4 Quantitative Analysis

In this part, we compare the proposed spatio-temporal neural network with other CNNs for semantic segmentation. The public dataset Cityscapes [31] is taken as the benchmark. The

images of Cityscapes dataset are collected in the urban environment from different cities, and the camera is fixed in the car.

Considering potential robotic applications, we only use the PC with one graphic card for processing. This means the memory is limited for training when compared with the PSPNet [81]. The PSPNet adopted Rsenet-101 [89] as its basic network architecture for Cityscapes dataset. They used four GPUs for training and have much more memory. So when training the spatial stream and temporal stream in our experiments, we adopt the ResNet-50 based PSPNet, and set the "cropsize" and "batchsize" to 617 and 2, respectively. For the dual stream CNN training, we set the "cropsize" and "batchsize" to 569 and 1, respectively. The spatio-temporal CNN is much bigger than the single streams and needs more memory for training. The transfer learning is used first, then we fine-tune the network weights from the model learned from ADK20K [38] dataset.

The results are showed in table 6.2. Deeplab [79] uses CRFs as a post-processing method to enhance the performance. The proposed spatio-temporal CNN does not use CRFs, but outperforms the Resnet-50 based spatial-PSPNet and other networks. Fig. 6.9 lists some results, and we can see that the segmentation performance is improved by using the temporal information.

6.4.5 Qualitative Analysis

This part evaluates the benefit of using the temporal information for CNN to process video streams. Among the training dataset, we deliberately label one category in one image wrongly while all other images are labelled correctly. If the network is robust to the problem, it should be able to distinguish the wrong label. As shown in Fig. 6.6, We deliberately label the floor (bright purple) as the lamp (yellow) in one training image. Then the "problem" dataset is fed to the network for training.

Table 6.3 shows the mIoU results on the Essex Indoor dataset. All the networks are trained with the "problem" dataset. As shown in the table, the spatio-temporal CNN demonstrates the best performance in semantic scene segmentation.

Fig. 6.7 shows the segmentation results for different neural networks. Fig. 6.7 (a) is selected from the test dataset. It is the neighbour image of Fig. 6.6(a). These two images have some similarityies but are totally different. Fig. 6.7(d) is the segmentation result from

the spatial stream with color images as the input. Fig. 6.7(e) is the segmentation result from the temporal stream with image difference as the input. Fig. 6.7(e) is the segmentation result from our proposed spatio-temporal CNN. As shown in the figure, the spatial stream segments the floor as the lamp, i.e. it fails to segments the floor, while the proposed fusion CNN segment the floor successfully. The floors in other test images are all segmented successfully with the spatial stream. This result means only using color images as the network input can not find the "problem" in the the dataset, while using both spatial and temporal information can make right segmentation decision when facing with the "problem" dataset.

6.4.6 3D Semantic Mapping

An 3D semantic map is very useful for robotic applications. By constructing the 3D semantic map, the robots can interact autonomously with the environment. For example, the robots can navigate themselves in an unknown environment by detecting the road and the robots can implement grasping tasks by detecting different objects in the 3D space.

We obtain the 3D semantic map of our second floor in the building by combining the spatio-temporal neural network with a SLAM algorithm. The state-of-the-art SLAM algorithm (ORB-SLAM [17] [6]) is used, which is able to obtain the camera pose and the point cloud map in real time. The keyframes are selected and then fed to the spatio-temporal C-NN for semantic segmentation. Compared with the system without keyframe structure, the real-time performance speeds up from 3Hz to 11Hz. By labelling the point cloud using the result of semantic pixel-wise segmentation, the 3D semantic map is constructed and shown in Fig. 6.8. Although there is some noise points in the map, it does provide the meaningful information for potential robotic applications. The main cause for the noise points is due to the measurement limitation of the depth camera. We plan to tackle the problem in our future work.

6.5 Conclusions

In this chapter, we have presented a novel spatio-temporal CNN for image segmentation which shows a better performance when compared with the CNNs using only spatial information. The image difference is taken as the temporal information for additional network input in the proposed network. Different fusion methods for spatial and temporal information

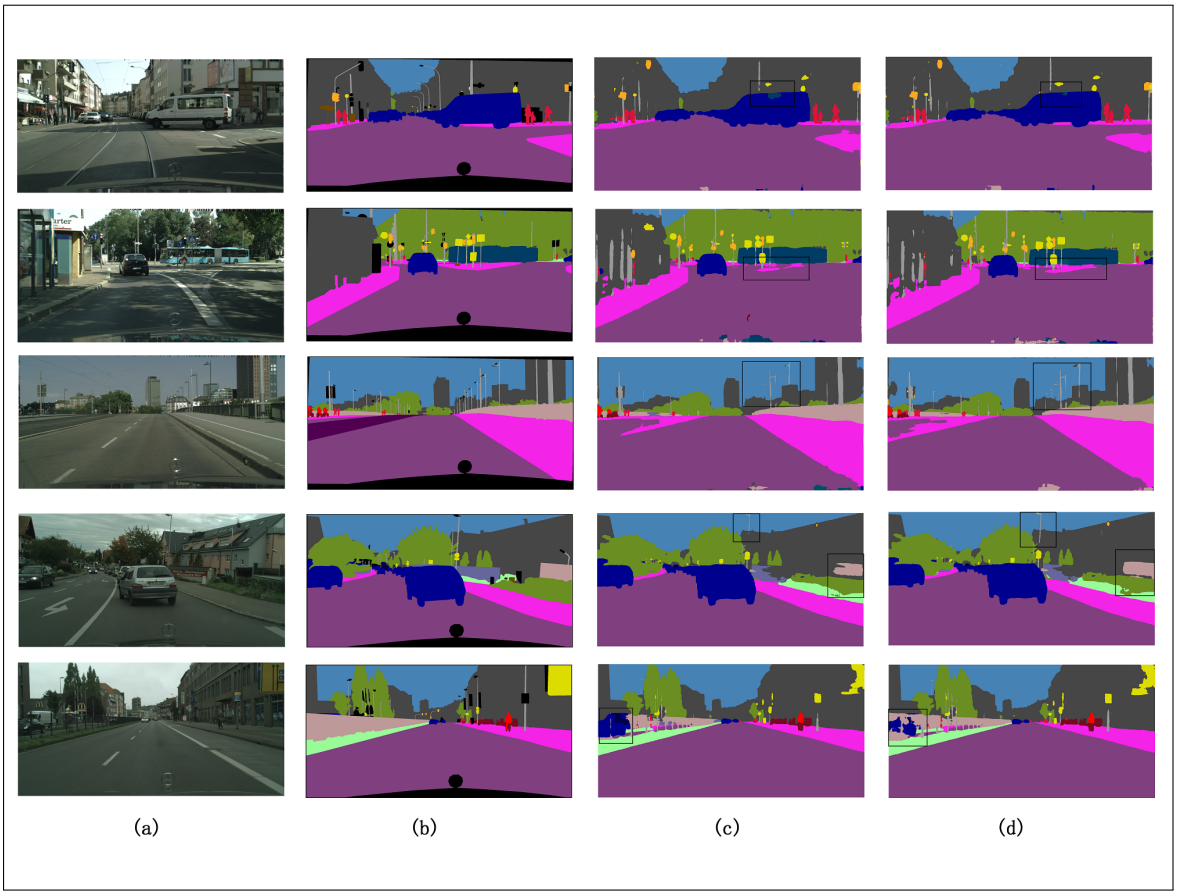


Figure 6.9: Visual comparison on Cityscapes dataset. (a) Image (b) Ground truth (c) Spatial PSPNet (d) Spatio-temporal CNN.

are discussed and compared. A global 3D semantic map is constructed with the proposed system which combines the spatio-temporal CNN with a SLAM algorithm. However, there are some noisy points in the constructed 3D map. This is caused by the limitation of depth camera and the wrong segmentation of the scene in some images. In the future, we would like to investigate how to improve the constructed 3D map.

Chapter 7

Conclusion

This chapter reviews this research, summarizes the major contributions achieved and future work to follow.

7.1 Research Overview

Aiming at solving robust localization and semantic mapping problems by only relying on low-cost cameras, this research goes from geometric vision to deep learning and studies both model-based SLAM methods and data-driven based SLAM methods. Firstly, model-based SLAM method based on points and plane-patches is proposed. Secondly, we combine SLAM and deep learning and present a robust relocalization system with supervised deep learning. Thirdly, we develop a data-driven SLAM system with totally unsupervised deep learning. In the end, a semantic mapping method with the convolutional neural network is proposed.

A model-based RGB-D SLAM algorithm is proposed firstly. We present a novel model-based RGB-D SLAM algorithm which is based on the usage of both feature points and plane patches for ego-motion estimation. In the proposed algorithm, the feature points are classified as plane points, smooth points and structural points. The plane points are on the planes with small curvature, the smooth points are on the smooth surfaces, and the structural points are on the edges or corners of some objects. A plane patch is defined as a small-sized patch constructed by using a plane point. The feature points and the plane patches are weighted according to the curvature, and then combined together to estimate the pose between two frames. A matching mechanism is introduced with the combination of both

feature points and plane patches. The coplanarity criteria and overlap criteria are proposed to match the plane patches.The proposed method can improve the accuracy and robustness of pose estimation, while maintaining the real-time performance. The evaluation experiments are performed based on multiple public benchmark datasets (TUM datasets and ICL-NUIM datasets) and self-collected datasets.

A data-driven based localization method with supervised deep learning is proposed secondly. In order to improve the robustness of localization system, we propose to adopt supervised deep learning to solve the pose regression problem. A dual-stream Convolutional Neural Network (CNN) is proposed to take both color images and depth images as network inputs. A novel depth image encoding method is presented. The proposed encoding method includes structural information while maintaining original depth information. Moreover, an stage-by-stage training strategy is introduced for the proposed relocalization system. Evaluation experiments are performed to prove the accuracy and robustness performance of the system. Based on the public Microsoft 7-Scenes benchmark, we show that the proposed system outperforms the state-of-the-art deep learning based system in terms of the accuracy. Based on the self-collected challenging dataset, we show that our relocalization system demonstrates very robust performance while facing with challenging scenes such as large scale, dynamic, fast movement and night-time environments. What is more, in a large-scale indoor experiment, our system shows accurate relocalization performance with 0.3m in position and 4° in orientation.

A novel data-driven based SLAM system with unsupervised deep learning is proposed thirdly. In order to tackle the problem of lacking of the labelled dataset and further improve the robustness performance, we proposed DeepSLAM, a novel visual SLAM system based on unsupervised deep learning. The DeepSLAM is fully unsupervised which means it does not need labelled dataset and can be trained with large amount of self-collected unlabelled dataset. Our DeepSLAM is composed of a Tracking-Net, a Mapping-Net, a Loop-Net and a pose graph optimization unit. The Tracking-Net is a Recurrent Convolutional Neural Network (RCNN) based architecture which is used for 6-DoF pose estimation. The Mapping-Net is an encoder-decoder based architecture which is used for depth estimation. The Loop-Net is pre-trained CNN which is used for place recognition and loop closure detection. The pose graph optimization unit is used for pose graph construction and optimization. Our pro-

posed DeepSLAM is evaluated based on multiple benchmarks. Based on the KITTI dataset, our DeepSLAM demonstrates comparable performance when compared with the state-of-the-art SLAM system in terms of accuracy. Based on the RobotCar dataset, our DeepSLAM shows more robust performance than other model-based SLAM methods, especially facing with challenging scenes.

A semantic pixel-wise mapping algorithm with deep learning is proposed for potential robotic applications in the end. In order to perceive the semantic information of the environments, we propose a novel semantic mapping system combining deep learning and SLAM. A novel spatio-temporal deep neural network is proposed for semantic segmentation. Dual-stream CNN architecture is adopted to take two streams at the same time. One stream takes images as inputs and is called spatio stream. One stream takes image differences as inputs and is called temporal stream. By taking both images and image differences, the proposed network can learn spatial information and temporal information at the same time. Moreover, in order to improve the real-time performance, a keyframe selection mechanism is also proposed. In this way, the computation burden is reduced efficiently. Based on the semantic segmentation results from the proposed network and the 3D point cloud from the SLAM algorithm, a semantic 3D map is built up with our system. The evaluation experiments are performed based on a public dataset (Cityscapes benchmark) and a self-collected dataset we labelled manually (Essex indoor benchmark). The results show improvement on both pixel-wise accuracy and Intersection over Union (IoU) for scene segmentation.

7.2 Contributions

The major contributions made in this thesis are briefly outlined as follows:

(1) **SLAM based on points and plane-patches (Chapter 3)**

- A novel RGB-D SLAM algorithm based on feature points and plane patches. The feature points are classified into three types according to the curvature. Then the feature points and plane patches are weighted and combined together to estimate the ego-motion between two frames.
- A feature matching process is proposed combining feature point matching and plane patch matching. Two criteria (coplanarity and overlap) are proposed to match

the plane patches from two frames.

(2) **Relocalization with Supervised Deep Learning (Chapter 4)**

- A novel indoor relocalization system based on dual-stream CNN. The network can take both color images and depth images as inputs and achieve robust relocalization. Moreover, a training strategy that divides the training into three stages is proposed for the system.
- A novel encoding method called minimized normal + depth (MND) encoding is proposed to solve the 6-DOF pose regression problem. The MND encoding images contain both pixel orientation information and absolute depth information, and maintain the ability to leverage the transfer learning at the same time.

(3) **SLAM with Unsupervised Deep Learning (Chapter 5)**

- A novel visual SLAM framework based on unsupervised deep learning. It is a result from the combination of deep learning and geometric constraints. The proposed DeepSLAM system consists of Tracking-Net, Mapping-Net, Loop-Net and a graph optimization unit.
- Deep Recurrent Convolutional Neural Network (RCNN) based Tracking-Net is designed to model the ego-motion by leveraging both spatial and temporal properties of a sequence of stereo images during training.
- DeepSLAM integrates the deep learning based tracking result and loop closure detection with a graph based optimization mechanism, resulting in a superior performance in terms of accuracy and robustness.
- Outliers rejection is handled by using the uncertainty derived from error maps of both geometric and photometric consistencies. This improves the robustness performance of DeepSLAM in challenging scenes.

(4) **Semantic mapping with Spatio-temporal Neural Network (Chapter 6)**

- A novel spatio-temporal neural network for semantic scene segmentation. Both images and image differences between two consecutive images are taken as the inputs for the network to enhance the performance.

- A keyframe selection mechanism for semantic segmantation. The non-keyframe segmentation is established by the prediction result from keyframes. In this way, the computational burden when handling with video streams can be reduced.

7.3 Publication List

Here are the academic publications that have been published during this PhD study.

Journals

- Ruihao Li, Qiang Liu, Jianjun Gui, Huosheng Hu, Dongbing Gu. Indoor Relocalization in Challenging Environments with Dual-stream Convolutional Neural Networks. IEEE Transactions on Automation Science and Engineering (T-ASE), 2017.

- Ruihao Li, Dongbing Gu, Qiang Liu, Zhiqiang Long, Huosheng Hu. Semantic Scene Mapping with Spatial-temporal Deep Neural Network for Robotic Applications. Cognitive Computation, 2017.

- Ruihao Li, Sen Wang, Dongbing Gu. DeepSLAM: A Robust Monocular SLAM System with Unsupervised Deep Learning. IEEE Transactions on Robotics (TRO), 2018. (Under Review)

- Ruihao Li, Sen Wang, Dongbing Gu. Ongoing Evolution of Visual SLAM from Geometry to Deep Learning: Challenges and Opportunities. IEEE Transactions on Automation Science and Engineering (T-ASE), 2018. (Under Review)

- Qiang Liu, Ruihao Li, Huosheng Hu, Dongbing Gu, Extracting Semantic Information from Visual Data: A Survey. Robotics 5. no. 1 (2016): 8.

- Qiang Liu, Ruihao Li, Huosheng Hu, Dongbing Gu, Using Deep Learning Technique to Build a Semantic Map for Visually Impaired People. IEEE Transactions on Cognitive and Developmental Systems (TCDS), 2018. (Under Review)

Conferences

- Ruihao Li, Qiang Liu, Jianjun Gui, Huosheng Hu, Dongbing Gu. A Novel RGB-D SLAM Algorithm Based on Points and Plane-Patches. In Automation Science and

Engineering (CASE), 2016 IEEE International Conference on, pp. 1348-1353. IEEE, 2016.

- Ruihao Li, Qiang Liu, Jianjun Gui, Huosheng Hu, Dongbing Gu. Night-time Indoor Relocalization Using Depth Image with Convolutional Neural Network. In Automation and Computing (ICAC), 2016 22nd International Conference on, pp. 261-266. IEEE, 2016.

- Ruihao Li, Sen Wang, Zhiqiang Long, Dongbing Gu. UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning. 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018. (Accepted)

- Qiang Liu, Ruihao Li, Huosheng Hu, Dongbing Gu. Using Semantic Maps for Room Recognition to Aid Visually Impaired People. In Automation and Computing (ICAC), 2016 22nd International Conference on, pp. 89-94, IEEE, 2016.

- Qiang Liu, Ruihao Li, Huosheng Hu, Dongbing Gu. Building semantic maps for blind people to navigate at home. In Computer Science and Electronic Engineering (CEEC), 2016 8th, pp. 12-17. IEEE, 2016.

7.4 Future Work

Aiming at the specific visual localization and semantic mapping for robotic scenarios, this thesis presents some novel and advanced solutions. Specifically, unsupervised deep learning has bridged visual SLAM and deep learning closer and sets up a new era for visual SLAM. There are many works and things to do in the future.

- **ImageNet-Scale Dataset for Deep Learning Visual SLAM.** Most of deep learning based methods are based on supervised learning schemes which require labeled datasets. However, labeling a large amount of data is time-consuming and labor-intensive. This requirement limits the potential application scenarios of deep learning based methods. This is particularly true in the context of visual SLAM because some robots or autonomous systems typically operate in completely unknown environments.

Current results show the robustness of supervised deep learning based methods could outperform model-based ones in some challenging scenes. However, large-scale labeled datasets are the bottleneck for further development. It is appealing yet hard to get ImageNet-scale dataset for all visual SLAM applications.

Therefore, it is very demanding for a visual SLAM system to learn under an unsupervised scheme. The performance could be continuously improved by the increased size of unlabeled datasets. As unsupervised deep learning methods [1, 56] has already shown some promising results, it will be very interesting to see how the performance of visual SLAM changes as the size of the training dataset increases. Unsupervised deep learning is expected to exploit a truly large-scale data, boosting the capability of visual SLAM in terms of robustness and semantic understanding.

- **Semantic SLAM with High-Level Understanding.** For intelligent robots or autonomous systems, understanding semantic information is essential and important. Fully convolutional neural networks have produced the state-of-the-art results on pixel-wise semantic segmentation in the last few years.

  Object-level semantic SLAM methods with deep learning will play a significant role in large-scale and complex environments. Objects can be extracted from geometric 3D map produced from visual SLAM systems. Further understanding object properties and mutual relations will enable a better interaction between robots and human or robots and environments. Moreover, object-level semantic information has the potential to improve the accuracy and robustness of pose estimation while pose estimation can do the same for semantic segmentation [11, 82, 83].

  With the aid of high-level understanding of the scenarios, task-driven SLAM which could provide high efficiency and wide generalization is also a promising area to explore.

- **Adaptive SLAM Methods for Different Sensing Modalities.** Different kinds of sensors bring in different features about the environments. How to take the most advantage of each of them in visual SLAM has always been a big question to answer. Apart from conventional optimal state estimation based multi-sensor fusion, sensor fusion and management in the framework of deep learning is being proved increasingly useful.

Learning based methods potentially generate new adaptive visual SLAM paradigms which can accommodate different sensing modalities to replace the calibration process.

- **Integration of Model based Methods with Deep Learning.** Model-based SLAM methods have already achieved great success. But they heavily depend on the successful detection of features. Most existing features, such as SIFT, SURF or ORB features, are still fragile when encountering featureless or challenging scenes. We can use the powerful representation capability in deep learning to extract more robust scale-invariant, lighting-invariant and rotation-invariant features. Extracting and matching features robustly by using supervised deep learning have already been reported in [57]. Exploring the use of unsupervised learning for extracting and matching more robust features is a means to improve model based methods.

  Maintaining global consistent map is a very important component of any SLAM system. For model-based methods, graph-based pose optimization and global bundle adjustment are the key to gain the high accuracy by maintaining a consistent map in back-end. Using deep learning methods to maintain global consistent map is still an open question to answer.

Bibliography

[1] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[2] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3D reconstruction in real-time," in *Intelligent Vehicles Symposium (IV)*, 2011.

[3] S. Wang, R. Clark, H. Wen, and N. Trigoni, "End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks," *The International Journal of Robotics Research*, 2017.

[4] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE, 2007, pp. 225–234.

[5] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

[6] R. Mur-Artal, J. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[7] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2320–2327.

[8] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 15–22.

[9] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[10] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[11] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4628–4635.

[12] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2043–2050.

[14] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[15] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 834–849.

[16] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1352–1359.

[17] R. Mur-Artal and J. D. Tardós, "Fast relocalisation and loop closing in keyframe-based SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 846–853.

[18] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE international conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2564–2571.

[19] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[20] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, "Low-latency visual odometry using event-based feature tracks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 16–23.

[21] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," in *European Conference on Computer Vision*. Springer, 2016, pp. 349–364.

[22] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison, "Dense planar SLAM," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2014, pp. 157–164.

[23] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE international symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2011, pp. 127–136.

[24] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[25] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *IEEE International Conference on Computer Vision*, 2013, pp. 1449–1456.

[26] G. Pascoe, W. Maddern, M. Tanner, P. Piniés, and P. Newman, "NID-SLAM: Robust monocular SLAM using normalised information distance," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[31] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.

[32] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017.

[33] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

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

[35] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," *European Conference on Computer Vision*, pp. 746–760, 2012.

[36] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[38] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ADE20K dataset," *arXiv preprint arXiv:1608.05442*, 2016.

[39] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in RGB-D images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2930–2937.

[40] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DoF camera relocalization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2938–2946.

[41] J.-L. Blanco-Claraco, F.-Á. Moreno-Dueñas, and J. González-Jiménez, "The málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 207–214, 2014.

[42] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in Neural Information Processing Systems*, 2014, pp. 2366–2374.

[43] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2024–2039, 2016.

[44] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[45] R. Garg, G. Carneiro, and I. Reid, "Unsupervised CNN for single view depth estimation: Geometry to the rescue," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 740–756.

[46] L. Ladicky, J. Shi, and M. Pollefeys, "Pulling things out of perspective," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 89–96.

[47] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[48] B. Li, C. Shen, Y. Dai, A. van den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1119–1127.

[49] F. Ma and S. Karaman, "Sparse-to-Dense: Depth prediction from sparse depth samples and a single image," *arXiv preprint arXiv:1709.07492*, 2017.

[50] J. Xie, R. Girshick, and A. Farhadi, "Deep3d: Fully automatic 2D-to-3D video conversion with deep convolutional neural networks," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 842–857.

[51] Y. Zhong, Y. Dai, and H. Li, "Self-supervised learning for stereo matching with self-improving ability," *arXiv preprint arXiv:1709.00930*, 2017.

[52] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia, "Unsupervised learning of geometry with edge-aware depth-normal consistency," *arXiv preprint arXiv:1711.03665*, 2017.

[53] R. Li, Q. Liu, J. Gui, D. Gu, and H. Hu, "Indoor relocalization in challenging environments with dual-stream convolutional neural networks," *IEEE Transactions on Automation Science and Engineering*, 2017.

[54] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen, "VidLoc: 6-DoF video-clip relocalization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[55] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "DeMoN: Depth and motion network for learning monocular stereo," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[56] R. Li, S. Wang, Z. Long, and D. Gu, "Undeepvo: Monocular visual odometry through unsupervised deep learning," *arXiv preprint arXiv:1709.06841*, 2017.

[57] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Toward geometric deep SLAM," *arXiv preprint arXiv:1707.07410*, 2017.

[58] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4762–4769.

[59] ——, "Geometric loss functions for camera pose regression with deep learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[60] R. Li, Q. Liu, J. Gui, D. Gu, and H. Hu, "Night-time indoor relocalization using depth image with convolutional neural networks," in *International Conference on Automation and Computing (ICAC)*. IEEE, 2016, pp. 261–266.

[61] F. W. C. Hazirbas, L. L.-T. T. Sattler, S. Hilsenbeck, and D. Cremers, "Image-based localization using LSTMs for structured feature correlation."

[62] T. Naseer and W. Burgard, "Deep regression for monocular camera-based 6-DoF global localization in outdoor environments."

[63] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep image homography estimation," *arXiv preprint arXiv:1606.03798*, 2016.

[64] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, "Exploring representation learning with CNNs for frame-to-frame ego-motion estimation," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 18–25, 2016.

[65] I. Melekhov, J. Kannala, and E. Rahtu, "Relative camera pose estimation using convolutional neural networks," *arXiv preprint arXiv:1702.01381*, 2017.

[66] M. Turan, Y. Almalioglu, H. Araujo, E. Konukoglu, and M. Sitti, "Deep endovo: A recurrent convolutional neural network (RCNN) based visual odometry approach for endoscopic capsule robots," *Neurocomputing*, 2017.

[67] H. Zhao, K. O?Brien, S. Li, and R. F. Shepherd, "Optoelectronically innervated soft prosthetic hand via stretchable optical waveguides," *Science Robotics*, vol. 1, no. 1, p. eaai7529, 2016.

[68] G. L. Oliveira, N. Radwan, W. Burgard, and T. Brox, "Topometric localization with deep learning," *arXiv preprint arXiv:1706.08775*, 2017.

[69] V. Peretroukhin and J. Kelly, "DPC-Net: Deep pose correction for visual localization," *arXiv preprint arXiv:1709.03128*, 2017.

[70] G. Costante and T. A. Ciarfuglia, "LS-VO: Learning dense optical subspace for robust visual odometry estimation," *arXiv preprint arXiv:1709.06019*, 2017.

[71] D. P. Frost, D. W. Murray, and V. A. Prisacariu, "Using learning of speed to stabilize scale in monocular localization and mapping."

[72] T. Nguyen, S. W. Chen, S. S. Shivakumar, C. J. Taylor, and V. Kumar, "Unsupervised deep homography: A fast and robust homography estimation model," *arXiv preprint arXiv:1709.03966*, 2017.

[73] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "VINet: Visual-Inertial odometry as a sequence-to-sequence learning problem." in *AAAI*, 2017, pp. 3995–4001.

[74] M. Turan, Y. Almalioglu, H. Gilbert, A. E. Sari, U. Soylu, and M. Sitti, "Endo-VMFuseNet: Deep visual-magnetic sensor fusion approach for uncalibrated, unsynchronized and asymmetric endoscopic capsule robot localization data," *arXiv preprint arXiv:1709.06041*, 2017.

[75] M. Turan, Y. Almalioglu, H. Araujo, T. Cemgil, and M. Sitti, "Endosensorfusion: Particle filtering-based multi-sensory data fusion with switching state-space model for endoscopic capsule robots using recurrent neural network kinematics," *arXiv preprint arXiv:1709.03401*, 2017.

[76] S. Pillai and J. J. Leonard, "Towards visual ego-motion learning in robots," *arXiv preprint arXiv:1705.10279*, 2017.

[77] A. Byravan and D. Fox, "SE3-Nets: Learning rigid body motion using deep neural networks," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 173–180.

[78] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for scene segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[79] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *arXiv preprint arXiv:1606.00915*, 2016.

[80] Z. Wu, C. Shen, and A. v. d. Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *arXiv preprint arXiv:1611.10080*, 2016.

[81] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *arXiv preprint arXiv:1612.01105*, 2016.

[82] R. Li, D. Gu, Q. Liu, Z. Long, and H. Hu, "Semantic scene mapping with spatio-temporal deep neural network for robotic applications," *Cognitive Computation*, Nov 2017. [Online]. Available: https://doi.org/10.1007/s12559-017-9526-9

[83] C. Zhao, L. Sun, B. Shuai, P. Purkait, and R. Stolkin, "Dense RGB-D semantic mapping with pixel-voxel neural network," *arXiv preprint arXiv:1710.00132*, 2017.

[84] W. Liu, A. Rabinovich, and A. C. Berg, "ParseNet: Looking wider to see better," *arXiv preprint arXiv:1506.04579*, 2015.

[85] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *arXiv preprint arXiv:1511.02680*, 2015.

[86] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *IEEE International Conference on Computer Vision*, 2015, pp. 1529–1537.

[87] A. Arnab, S. Jayasumana, S. Zheng, and P. H. Torr, "Higher order conditional random fields in deep neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 524–540.

[88] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015, pp. 1–14.

[89] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[90] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 248–255.

[91] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," *arXiv preprint arXiv:1412.7062*, 2014.

[92] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3640–3649.

[93] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[94] Z. Wu, C. Shen, and A. v. d. Hengel, "High-performance semantic segmentation using very deep fully convolutional networks," *arXiv preprint arXiv:1604.04339*, 2016.

[95] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers, "FuseNet: Incorporating depth into semantic segmentation via fusion-based CNN architecture," in *Asian Conference on Computer Vision*, vol. 2, 2016.

[96] A. Valada, G. Oliveira, T. Brox, and W. Burgard, "Towards robust semantic segmentation using deep fusion," in *Robotics: Science and Systems (RSS 2016) Workshop, Are the Sceptics Right? Limits and Potentials of Deep Learning in Robotics*, 2016.

[97] A. Valada, J. Vertens, A. Dhall, and W. Burgard, "Adapnet: Adaptive semantic segmentation in adverse environmental conditions," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

[98] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, Proceedings. IEEE International Conference on*, pp. 1403–1410, 2003.

[99] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira, "Large-scale 6-DOF SLAM with stereo-in-hand," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 946–957, 2008.

[100] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *Robotics, IEEE Transactions on*, vol. 30, no. 1, pp. 177–187, 2012.

[101] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 3748–3754, 2013.

[102] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense RGB-D SLAM with volumetric fusion," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015.

[103] P. C. Ng and S. Henikoff, "Sift: Predicting amino acid changes that affect protein function," *Nucleic acids research*, vol. 31, no. 13, pp. 3812–3814, 2003.

[104] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Computer vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[105] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 4605–4611, 2015.

[106] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-plane SLAM for hand-held 3D sensors," in *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 5182–5189, 2013.

[107] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Mixed and Augmented Reality (ISMAR), IEEE and ACM International Symposium on*, pp. 225–234, 2007.

[108] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), IEEE International Conference on*, pp. 2320–2327, 2011.

[109] J. Weingarten and R. Siegwart, "3D SLAM using planar segments," in *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, pp. 3062–3067, 2006.

[110] A. J. Trevor, J. G. Rogers III, H. Christensen *et al.*, "Planar surface SLAM with 3D and 2D sensors," in *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 3041–3048, 2012.

[111] E. Ataer-Cansizoglu, Y. Taguchi, S. Ramalingam, and T. Garaas, "Tracking an RGB-D camera using points and planes," in *Computer Vision Workshops (ICCVW), IEEE International Conference on*, pp. 51–58, 2013.

[112] X. Gao and T. Zhang, "Robust RGB-D simultaneous localization and mapping using planar point features," *Robotics and Autonomous Systems*, vol. 72, pp. 1–14, 2015.

[113] R. Cupec, E. K. Nyarko, D. Filko, A. Kitanov, and I. Petrović, "Place recognition based on matching of planar surfaces and line segments," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 674–704, 2015.

[114] S. Li and A. Calway, "Absolute pose estimation using multiple forms of correspondences from RGB-D frames," in *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 4756–4761, 2016.

[115] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Robotics and Automation (ICRA), IEEE International Conference on*, pp. 3607–3613, 2011.

[116] I. T. Jolliffe, "Principal component analysis and factor analysis," in *Principal component analysis*. Springer, 1986, pp. 115–128.

[117] G. Bradski and A. Kaehler, "OpenCV," *Dr. Dobb?s journal of software tools*, vol. 3, 2000.

[118] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *Robotics and automation (ICRA), IEEE international conference on*, pp. 1524–1531, 2014.

[119] C. Ye, S. Hong, and A. Tamjidi, "6-DOF pose estimation of a robotic navigation aid by tracking visual and geometric features," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1169–1180, 2015.

[120] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[121] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping: Part II," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[122] Q. Liu, R. Li, H. Hu, and D. Gu, "Extracting semantic information from visual data: A survey," *Robotics*, vol. 5, no. 1, p. 8, 2016.

[123] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (ICCV)*, 2015, pp. 1–9.

[124] S. Lowry, N. Sunderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *Robotics, IEEE Transactions on*, vol. 32, no. 1, pp. 1–19, 2016.

[125] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–607.

[126] B. Steder, G. Grisetti, and W. Burgard, "Robust place recognition for 3D range data based on point features," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1400–1405.

[127] R. Li, Q. Liu, J. Gui, D. Gu, and H. Hu, "A novel RGB-D SLAM algorithm based on points and plane-patches," in *IEEE International Conference on Automation Science and Engineering*. IEEE, 2016, pp. 1348–1353.

[128] E. Fernández-Moral, P. Rives, V. Arévalo, and J. González-Jiménez, "Scene structure registration for localization and mapping," *Robotics and Autonomous Systems*, vol. 75, pp. 649–660, 2016.

[129] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2. IEEE, 2006, pp. 2161–2168.

[130] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular SLAM," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009.

[131] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[132] ——, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.

[133] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *Robotics, IEEE Transactions on*, vol. 28, no. 5, pp. 1188–1197, 2012.

[134] Z. Chen, O. Lam, A. Jacobson, and M. Milford, "Convolutional neural network-based place recognition," *arXiv preprint arXiv:1411.1509*, 2014.

[135] N. Sunderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford, "On the performance of ConvNet features for place recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 4297–4304.

[136] N. Sunderhauf, S. Shirazi, A. Jacobson, F. Dayoub, E. Pepperell, B. Upcroft, and M. Milford, "Place recognition with ConvNet landmarks: Viewpoint-robust, condition-robust, training-free," *Proceedings of Robotics: Science and Systems XII*, 2015.

[137] C. Couprie, C. Farabet, L. Najman, and Y. Lecun, "Indoor semantic segmentation using depth information," in *International Conference on Learning Representations*, 2013, pp. 1–8.

[138] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," in *European Conference on Computer Vision*. Springer, 2014, pp. 345–360.

[139] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[140] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 858–865.

[141] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust RGB-D object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 681–687.

[142] M. Schwarz, H. Schulz, and S. Behnke, "RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1329–1335.

[143] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[144] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in Neural Information Processing Systems*, 2014, pp. 487–495.

[145] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[146] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[147] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki, "SfM-Net: Learning of structure and motion from video," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[148] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[149] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Is L2 a good loss function for neural networks for image processing?" *ArXiv e-prints*, vol. 1511, 2015.

[150] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning." in *AAAI*, 2017, pp. 4278–4284.

[151] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3607–3613.

[152] J. Xie, L. Yu, L. Zhu, and X. Chen, "Semantic image segmentation method with multiple adjacency trees and multiscale features," *Cognitive Computation*, vol. 9, no. 2, pp. 168–179, 2017.

[153] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv preprint arXiv:1511.00561*, 2015.

[154] Z. Tu, A. Abel, L. Zhang, B. Luo, and A. Hussain, "A new spatio-temporal saliency-based video object segmentation," *Cognitive Computation*, vol. 8, no. 4, pp. 629–647, 2016.

[155] Z. G. Doborjeh, M. G. Doborjeh, and N. Kasabov, "Attentional bias pattern recognition in spiking neural networks from spatio-temporal EEG data," *Cognitive Computation*, pp. 1–14, 2017.

[156] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao, "Towards good practices for very deep two-stream convnets," *arXiv preprint arXiv:1507.02159*, 2015.

[157] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: towards good practices for deep action recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 20–36.

[158] M. Hülse, S. McBride, and M. Lee, "Fast learning mapping schemes for robotic hand–eye coordination," *Cognitive Computation*, vol. 2, no. 1, pp. 1–16, 2010.