Deep learning based monocular visual-inertial odometry

Mahdi Abolfazli Esfahani

2020

Mahdi Abolfazli Esfahani (2020). Deep learning based monocular visual-inertial odometry. Doctoral thesis, Nanyang Technological University, Singapore.
<https://hdl.handle.net/10356/152280>

<https://hdl.handle.net/10356/152280>

<https://doi.org/10.32657/10356/152280>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 23:34:08 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

DEEP LEARNING BASED MONOCULAR VISUAL-INERTIAL ODOMETRY

MAHDI ABOLFAZLI ESFAHANI

SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING

2020

DEEP LEARNING BASED MONOCULAR VISUAL-INERTIAL ODOMETRY

MAHDI ABOLFAZLI ESFAHANI

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

30-12-2020

.....

Date

ncheli

.....

Mahdi Abolfazli Esfahani

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

30-12-2020

.....
Date

NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU

.....
Assoc Prof. Wang Han

Authorship Attribution Statement

This thesis contains material from seven papers published and one paper under review in the following peer-reviewed journals / conferences in which I am listed as an author.

Chapter 3 is published as

- Mahdi Abolfazli Esfahani, Han Wang, Keyu Wu, Shenghai Yuan, “From local understanding to global regression in monocular visual odometry”, *International Journal of Pattern Recognition and Artificial Intelligence*, 2020. DOI: 10.1142/S0218001420550022
- Mahdi Abolfazli Esfahani, Keyu Wu, Shenghai Yuan, Han Wang, “A New Approach to Train Convolutional Neural Networks for Real-Time 6-DOF Camera Relocalization”, *IEEE 14th International Conference on Control and Automation (ICCA)*, 2018. DOI: 10.1109/ICCA.2018.8444299
- Mahdi Abolfazli Esfahani, Han Wang, Keyu Wu, Shenghai Yuan, “Unsupervised Scene Categorization, Path Segmentation and Landmark Extraction while Traveling Path”, *International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2020.

The contributions of authors are as follows:

- Associate Prof Wang provided the initial direction and edited the manuscript.
- I designed the study and implemented the idea, which was used for generating and testing results. I also analyzed the data and prepared the manuscript drafts.
- Mrs. Wu and Mr. Yuan helped in revising the manuscript drafts.

Chapter 4 is published as

- Mahdi Abolfazli Esfahani, Han Wang, Keyu Wu, Shenghai Yuan, “DeepD-SAIR: Deep 6-DOF camera relocalization using deblurred semantic-aware

image representation for large-scale outdoor environments”, *Image and Vision Computing*, 2019. DOI: 10.1016/j.imavis.2019.06.014

- Mahdi Abolfazli Esfahani, Keyu Wu, Shenghai Yuan, Han Wang, “Towards Utilizing Deep Uncertainty In Traditional SLAM”, *IEEE 15th International Conference on Control and Automation (ICCA)*, 2019.
DOI: 10.1109/ICCA.2019.8899572

The contributions of authors are as follows:

- Associate Prof Wang provided the initial direction and edited the manuscript.
- I designed the study and implemented the idea, which was used for generating and testing results. I also analyzed the data and prepared the manuscript drafts.
- Mrs. Wu and Mr. Yuan helped in revising the manuscript drafts.

Chapter 5 is published as

- Mahdi Abolfazli Esfahani, Han Wang, Benyamin Bashari, Keyu Wu, Shenghai Yuan, “Learning to Extract Robust Handcrafted Features with a Single Observation via Evolutionary Neurogenesis”, *Applied Soft Computing*, 2021.
DOI: 10.1016/j.asoc.2021.107424

The contributions of authors are as follows:

- Associate Prof Wang provided the initial direction and edited the manuscript.
- I designed the study and implemented the idea, which was used for generating and testing results. I also analyzed the data and prepared the manuscript drafts.
- Mr. Bashari helped in validating the method and data collection.
- Mrs. Wu and Mr. Yuan helped in revising the manuscript drafts.

Chapter 6 is published as

- Mahdi Abolfazli Esfahani, Han Wang, Keyu Wu, Shenghai Yuan, “OriNet: Robust 3D Orientation Estimation with a Single Particular IMU”, *IEEE*

Robotics and Automation Letters (RA-L), 2019.

DOI: 10.1109/LRA.2019.2959507

- Mahdi Abolfazli Esfahani, Han Wang, Keyu Wu, Shenghai Yuan, “OriNet: Robust 3D Orientation Estimation with a Single Particular IMU”, *International Conference on Robotics and Automation (ICRA)*, 2020
- Mahdi Abolfazli Esfahani, Han Wang, Keyu Wu, Shenghai Yuan, “AbolD-deepIO: A novel deep inertial odometry network for autonomous vehicles”, *IEEE Transactions on Intelligent Transportation Systems*, 2019.

DOI: 10.1109/TITS.2019.2909064

The contributions of authors are as follows:

- Associate Prof Wang provided the initial direction and edited the manuscript.
- I designed the study and implemented the idea, which was used for generating and testing results. I also analyzed the data and prepared the manuscript drafts.
- Mrs. Wu and Mr. Yuan helped in revising the manuscript drafts.

26-06-2021

.....
Date

NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU

.....
Mahdi Abolfazli Esfahani

Acknowledgement

My special thanks go to my dear supervisor, Associate Professor Wang Han, for his assistance, guidance, and motivation during my Ph.D. studies. He gave me lots of advice and energies during my Ph.D. studies to find the best direction. Besides, I am equally appreciative of all the Thesis Advisory Committee (TAC) members and the other examiners to review my thesis and be present at my oral examination.

Furthermore, it was a great honor for me to work with Wu Keyu and Yuan Shenghai on the drone project; I appreciate all of their support. Besides, I would like to thank my fellow Hasith Karunasekera, Zhang Handou, Guiju Ping, and Zhou Hui for all the great discussions and time.

I would also thank my dear parents, Javad and Maryam, and my brother and sister, Messiah and Sara, for their inexhaustible grace and love.

Contents

Statement of Originality	ii
Supervisor Declaration Statement	iii
Authorship Attribution Statement	iv
Acknowledgement	vii
Table of Contents	viii
Summary	xii
List of Figures	xiv
List of Tables	xxi
1 Introduction	1
1.1 Overview	1
1.2 Motivation	4
1.3 Objectives	5
1.4 Major Contributions of the Thesis	6
1.5 Organization of the Thesis	8
2 Literature Review	10
2.1 Overview	10
2.2 Visual Odometry	10
2.2.1 Traditional Visual Odometry	11
2.2.2 Deep Learning based Visual Odometry	15
2.3 6-DoF Camera Relocalization	16
2.4 Handcrafted Features Extraction	21
2.4.1 Traditional Feature Detection Techniques	23
2.4.2 Deep Learning based Feature Detection Techniques	25
2.5 Inertial Odometry	29

2.5.1	Traditional Inertial Odometry Methods	29
2.5.2	Deep Learning based Inertial Odometry Methods	31
2.6	Visual-Inertial Odometry	33
2.7	Concluding Remarks	35
3	Robust Training of the Regression Networks by Considering Local Boundaries	37
3.1	Overview	37
3.2	Unsupervised Scene Categorization, Path Segmentation, and Landmark Extraction while Traveling Path	38
3.2.1	Experiments and Results	42
3.3	Proposed Multi-Step Training Approach Applied on 6-DoF Camera Relocalization	46
3.3.1	Experiments and Results	49
3.4	From Local Understanding to Global Regression in Visual Odometry	54
3.4.1	CNN Structure	55
3.4.2	Proposed Training Procedure in Visual Odometry	57
3.4.3	Experiments and Results	60
3.5	Concluding Remarks	67
4	Dealing with Moving Objects	70
4.1	Overview	70
4.2	Preliminaries	70
4.2.1	Image Deblurring	70
4.2.2	Semantic Segmentation	71
4.3	DeepDSAIR: Deep 6-DoF Camera Relocalization using Deblurred Semantic-Aware Image Representation for Large-Scale Outdoor Environments	72
4.3.1	Deblurred Semantic-Aware Image Representation	72
4.3.2	Network Architecture	76
4.3.3	Training Procedure	78
4.3.4	Experiments and Results	79

4.4	Removing the Effect of Moving Objects in Traditional Visual Odometry Systems	84
4.4.1	Experiments and Results	88
4.5	Concluding Remarks	90
5	Learning to Extract Robust Handcrafted Features with a Single Observation via Evolutionary Neurogenesis	93
5.1	Overview	93
5.2	Proposed Method	97
5.2.1	CNNs Filter Visualization and Understanding	99
5.2.2	Newborn Convolutional Neuron Learning Procedure	100
5.2.3	Problem Formulation	106
5.2.4	Fitness Function for The Problem of Corner Detection	107
5.2.5	Evolutionary Optimization	109
5.2.6	Newborn Neuron Execution in Test Time	110
5.3	Experiments and Results	112
5.3.1	Implementation and Experimental Setup	112
5.3.2	Discussion	115
5.4	Concluding Remarks	127
6	Deep Learning based Inertial Odometry	130
6.1	Overview	130
6.2	Preliminaries	131
6.3	Estimating the Length of Rotation and Position Changes with a Single Particular IMU	132
6.3.1	Proposed Method	132
6.3.2	Experiments and Results	137
6.4	Robust 3-D Orientation Estimation with a Single Particular IMU	143
6.4.1	Proposed Architecture	144
6.4.2	From Large to Small Dropout Training Approach	148
6.4.3	OriNet-IMU Bias Calibration	149
6.4.4	Experiments and Results	150
6.5	Concluding Remarks	157

7 Conclusion and Future Work	160
7.1 Conclusion	160
7.2 Future Work	162
List of Publications	166
References	169
Appendix A	191

Summary

Autonomous vehicles require knowing their state in the environment to make a decision and achieve their desired goal. The Visual-Inertial Odometry (VIO) system is one of the most significant modules of autonomous vehicles. This module enables robots to know their position and orientation, relative to the starting point, in the environment. Despite decades of research using different sensors, current odometry systems suffer from various kinds of problems, resulting in odometry failure.

Different types of sensors and their combination are studied and examined by researchers in the past decades, and novel methods are proposed. In between all sensor combinations, using a monocular camera simultaneously with an Inertial Measurement Unit (IMU) is in researchers' focus in the past three years because of its lightweight, low-cost, and small combinations of sensors. This combination can be used in many applications, from smart mobile phones to micro aerial vehicles.

Traditional methods attempt to extract features from visual information, track them in consecutive frames, and tightly optimize the relation between feature points in consecutive camera frames with the secured IMU information to solve the problem. However, they cannot perform well in textureless regions and fail when they cannot track features. Moreover, they cannot handle some challenging scenarios where an object moves in front of the camera. Deep learning based VIO systems are investigated to tackle this challenge. While deep learning based VIO systems achieved satisfactory results, they suffer from wrong motion estimation and scale extraction problems in different scenarios; due to their low generalization capability.

This research investigates the pros and cons of existing deep learning based monocular VIO systems. It then proposes a new approach to train Convolutional Neural Networks (CNNs) for the regression problem of 6-DOF camera re-localization, which makes CNNs more robust. The proposed approach is further extended to solve the problem of Visual Odometry (VO) in an end-to-end manner. The effect of moving objects and motion blur are also studied, and approaches to make visual odometry and camera re-localization modules robust to them are

proposed.

Furthermore, since VO's primary step is to extract robust features that can be tracked in consecutive frames, a bio-inspired approach that learns to extract distinctive handcrafted features, from a single training image, with high generalization ability is proposed. The proposed method executes in real-time and surpasses traditional and deep learning based methods in terms of robust handcrafted feature extraction.

IMU suffers from bias and measurement noise, which makes the problem of Inertial Odometry (IO) much more complicated. Due to the error propagation over time (during robot position estimation), an inaccurate estimate or a small error will cause the odometry and localization system unreliable and unusable in a split of seconds. This research presents a novel triple channel deep IO network based on IMU's physical and mathematical model to solve IO's problem. The proposed method is capable of outputting the length of orientation and position changes over time. The proposed model simulates the noise model in the training phase and becomes robust to noise in testing. Besides, the proposed network architecture includes time interval between two consecutive IMU readings as input, which makes it robust to change of IMU frame rate and miss of IMU information. The proposed network architecture outperforms all the existing solutions on the IMU readings of challenging Micro Aerial Vehicle (MAV) dataset and improves the accuracy for about 25 percent. Furthermore, the approach is further extended to extract the full 3D orientation of MAV accurately.

List of Figures

1.1	Different types of sensors generally utilized by autonomous vehicles	2
1.2	Problems that VIO modules should handle	5
2.1	Different approaches researchers use to solve the problem of visual odometry. An example approach, for each technique, is mentioned by green color. (methods: PTAM [78], DenseVLAD [164], ORB-SLAM [115], DTAM [117], seqSLAM [108], PoseNet [75], NetVLAD [11], DeepVO [170], X-View [52], Attentive Deep Local Features [121])	12
2.2	Categorizing popular state of the art traditional VO methods based on the type of features they are utilizing (methods: PTAM[78], MonoSLAM[34], ORB-SLAM[113], SVO[49], DSO[169], LSD-SLAM[42], DTAM[117])	13
2.3	Comparison of the Pipeline of Deep Learning based and Traditional SLAM Methods	14
2.4	Challenges to solve in the problem of 6-DoF camera relocalization. (a) shows the presence of dynamic objects in the environment which prevents modeling the static scene. Windows of the shop in (b) reflect the front building which tricks the relocalization module. Motion Blur blurs the sharp parts of the building in images (c) and (d) where robust informative features cannot extract and match. (e) and (f) are demonstrating the effect of the change in the light of the scene in which different correspondent pixels in two images have different values	20
2.5	Deep Learning versus Conventional Methods in the Problem of 6-DoF Camera Re-Localization	21
2.6	Deep Neural Networks Architecture (W_i^j represents the weight for the neuron j in the layer number i)	25

2.7	CNNs architecture is illustrated in this figure. W_i^j represents the weight for the neuron j in the layer number i and $*$ stands as convolution operator. The feature detection module is using convolutional and pooling layers to extract robust feature representations from the image, and the description module is an FCN that learns to map input to desired output. The learning policy for the FCN module is presented in equation 2.4.	27
2.8	Comparison of SINS and PDR workflow	30
2.9	IONet [24] Workflow	32
2.10	Workflow of Traditional Visual-Inertial Odometry Methods	34
2.11	VINet [30] Workflow for Robots Odometry	34
3.1	The green line shows the captured path in NTU. Boxes show the extracted clusters.	42
3.2	Computed Z_{rev} and $\ddot{Z}_{rev}$ on the feature representations of the captured image sequence at NTU.	43
3.3	Distance between global feature representation of clusters ($\bar{F}_c$) on the captured scenario in NTU	44
3.4	Extracted clusters for the sequence captured at NTU	44
3.5	Clustering King's College scene by the proposed approach. Points are places an image is captured, and points with similar colors are in one cluster.	45
3.6	Two images from different clusters. The right image is labeled to be in the boundary (cluster) of the left image wrongly.	46
3.7	Distances between feature representation of each cluster on the whole data of King's College Scene	46
3.8	Flowchart of the proposed training approach	50
3.9	Visualizing King's College training data in 3D position coordinate	51
3.10	Visualizing Old Hospital training data in 3D position coordinate	52
3.11	Visualizing Shop Facade training data in 3D position coordinate	52
3.12	Visualizing St Mary's Church training data in 3D position coordinate	53
3.13	Histogram of number of data points in each cluster after applying proposed clustering method on labels of training data	53

3.14	Different Inputs of the P-CNN Network Architecture	57
3.15	P-CNN Network Architecture	58
3.16	The Procedure of the Proposed Training Approach applied to VO .	61
3.17	Clustered Displacement of Camera Rotation Labels in 4 Classes . .	61
3.18	Clustered Displacement of Camera Position Labels in 4 Classes . . .	62
3.19	Number of Observations per Class After Applying the Proposed Clustering Technique	62
3.20	Average Rotation Error Across Test Sequence Speeds	63
3.21	Average Rotation Error Across Test Sequence Path Length	63
3.22	Average Translation Error Across Test Sequence Speeds	64
3.23	Average Translation Error Across Test Sequence Path Length	64
3.24	Comparing trajectories computed from trained P-CNN network us- ing proposed method and training it as regressor from beginning for sequence 08 of KITTI dataset	65
3.25	Comparing trajectories computed from trained P-CNN network us- ing proposed method and training it as regressor from beginning for sequence 09 of KITTI dataset	65
3.26	Comparing trajectories computed from trained P-CNN network us- ing proposed method and training it as regressor from beginning for sequence 10 of KITTI dataset	66
3.27	Effect of Number of Clusters on The Final Model Accuracy	67
4.1	Proposed Deep Semantic-Aware Image Representation (DSAIR) Mod- ule that applies before Deep CNN	73
4.2	Deep CNN Module	73
4.3	Sample image of the scene before and after applying DS AIR module	75
4.4	Effect of random cropping augmentation layer used by previous methods on the camera orientation parameters	77
4.5	Multi-Step Training Procedure	79

4.6	Illustrating the output of different stages of the DSAIR module. The motion blur of the raw input image (a) is handled efficiently by the Deblurring module and in (b) the edges and corners are sharp enough to extract meaningful feature representations. However, the semantic segmentation module wrongly detected the front view of the building as a dynamic object and DSAIR module makes it blur (c). The blurred part of the building has the most information to recover the position and orientation of the captured image.	82
4.7	Workflow of the Proposed Deep Uncertainty Aware Feature Extrac- tion Method	84
4.8	Dynamic Objects Detected After Merging Deep Uncertainties	85
4.9	Average Translation Error Across Different Speeds (Blue Line Shows The Result of The Proposed Method and The Red Line is The Orig- inal ORBSLAM2)	89
4.10	Average Rotation Error Across Different Speeds (Blue Line Shows The Result of The Proposed Method and The Red Line is The Orig- inal ORBSLAM2)	89
4.11	Average Translation Error Across Different Path Length (Blue Line Shows The Result of The Proposed Method and The Red Line is The Original ORBSLAM2)	89
4.12	Average Rotation Error Across Different Path Length (Blue Line Shows The Result of The Proposed Method and The Red Line is The Original ORBSLAM2)	90
5.1	Workflow of different feature extraction techniques (Mentioned meth- ods can be referenced as Harris [63], FAST [133] , SIFT [100], SURF [15], SuperPoint [38], and LIFT [176])	96
5.2	(a) shows the first layer's filters of the GoogLeNet, and (b) is the result of convolving those filters respectively on Lenna image.	100
5.3	This figure illustrates the learning procedure of the proposed ECNN module which models the effect of the brain's neurogenesis and as- trocyte (* stands as the convolution operation).	101

5.4	Procedure of executing ECNN newborn neuron with Rotation and Scale layers that applies on the input of the evolved newborn neuron in test time	111
5.5	The training image (a) and its corresponding ground truth corners (b) which are used to train the corner detection module are illustrated in this figure. The ground truth image shows the probability map of (a) where corner pixels have the value of one and are the interest points required to be detected by the neuron	112
5.6	(a) shows the optimization of the fitness function over different iterations for each set of experiments; the plotted value for each iteration is the average of the best fitness value in all 150 runs conducted in each experiment set. (b),(c),(d) and (e) are showing the learned corners for different steps by red dots. In these figures, more corners than actual corners are asked to extract by trained corner detector for visualization. Having two red points on top and down left of (e) means the actual corners are detected, and the detector is moved to other low probability parts and detected points as corners are the most significant ones and are similar to a corner; hence the corner detector is well trained.	118
5.7	(a) is demonstrating a sample test data, and (b),(c), and (d) are showing the 200 corner points detected in this image, by a newborn neuron which is trained to detect corners.	119
5.8	Comparison of the robustness of the proposed method with others in terms of illumination change. Each point of the figures shows the average ratio of features repeatedly detected after changing the illumination of each image of the dataset by adding a specific value shown on the horizontal axis.	120
5.9	Comparison of the robustness of the proposed method with others in the presence of Gaussian Noise with different variances (σ^2). Each point of the figures shows the average ratio of features repeatedly detected after adding Gaussian noise with a variance of σ^2 to each image of the dataset.	122

5.10	Comparison of the robustness of the proposed method with others in terms of change in the rotation of the image for θ degrees. Each point of the figures shows the average ratio of features repeatedly detected after rotating images in the dataset for θ degrees.	123
5.11	Comparison of the robustness of the proposed method with others in terms of change in the scale. Each point of the figures shows the average ratio of features repeatedly detected after scaling dataset images for a specific rate.	124
5.12	(a) shows the training data, and (b) is the normalized and scaled output probability map of convolving $\mathbf{g}_{newborn}^R$, $\mathbf{g}_{newborn}^G$ and $\mathbf{g}_{newborn}^B$ with the training image in the RGB color space. As it demonstrates, the trained neuron not only can work as a corner detector (as its primary output with high score) but also can be used to detect vertical and horizontal edges. In each black square, $\mathbf{g}_{newborn}^R$ extracts down right and top left, $\mathbf{g}_{newborn}^B$ extracts top right and down left, and $\mathbf{g}_{newborn}^G$ extracts all corners but have more influence on top and down left corners. Simultaneously, $\mathbf{g}_{newborn}^R$ have a low effect on extracting right and down side edges, $\mathbf{g}_{newborn}^B$ can help to detect the top and right edges, and $\mathbf{g}_{newborn}^G$ can help to extract and understand vertical and horizontal edges by giving different probabilities to different types of edges.	126
6.1	Proposed AbolDeepIO Network Architecture ($N1$, $N2$, and $N3$ are the number of LSTM layers used for accelerometer, gyroscope and sampling time inputs respectively. $N4$ is the number of LSTM layers used after concatenation of features.)	134
6.2	Windowing Approach used by AbolDeepIO	134
6.3	Different Approaches to Train Proposed Network Architecture According to The Window Observations	138
6.4	Average of ΔL and $\Delta\theta$ error propagation over time for all datasets	142
6.5	Challenges and proposed solutions for estimating 3D orientation with a single particular IMU	144

6.6	The procedure of the proposed <i>OriNet</i> module (b_1 , b_2 and b_3 are the gyroscope bias, b_w)	146
6.7	Importance of ζ which should be handled by ΔQ_{Net}	148
6.8	Brown line shows the best validation loss obtained by training the network from scratch and using the dropout value indicated on the horizontal axis. The blue line shows the best validation loss achieved by training the network with the proposed training approach; The proposed training method reduces the dropout during training, and so the utilized dropout at each training epoch is indicated on the horizontal axis.	154
6.9	Weights of most active neurons in ΔQ_{Net} input space	154
6.10	Best and mean fitness value for the test dataset #4	155
6.11	Estimated and exact quaternion of test dataset #4	157

List of Tables

3.1	Details of <i>Cambridge Landmarks</i> Dataset	50
3.2	Comparing the median re-localization error when the network is trained with and without the proposed training approach	55
4.1	Dataset Details	80
4.2	Comparing the median error of the proposed method with the ex- isting approaches to solve the problem of 6-DoF camera relocalization	81
5.1	Definition of parameters used in modeling newborn neurons	102
5.2	This table compares the network parameters, learning and execution performance of the proposed ECNN module with other learning- based feature detection techniques. The reported execution time is for processing an image with a size of 777×1280 pixels.	116
5.3	Mean, Minimum, Maximum, Variance and Median of best fitness value in each of the experiment sets. 150 runs are conducted in each of the experiment sets.	116
5.4	p -value among experiments	116
5.5	The average ratio of matched features for viewpoint datasets. Top three results in each row are colored.	120
5.6	The average ratio of repeated features after applying different trans- forms. Top three results in each row are colored.	120
5.7	p -value of Wilcoxon test among different methods (cells get brighter as the value gets to zero and the null hypothesis rejects and alter- native hypothesis which states row $\mathbf{A} >$ column $\mathbf{B}$ (based on their distribution median) satisfies; cells regarding values less than the significance level of 5% are purely white.)	121
6.1	Train and Test Splits of ASL EuRoC MAV Dataset [18]	138
6.2	Properties of Different Network Architectures of the Proposed Method To Train and Evaluate	139
6.3	MAE/RMSE of $\Delta\theta$ for Windowing Size of 20	141

6.4	MAE/RMSE of ΔL for Windowing Size of 20	141
6.5	MAE/RMSE of $\Delta\theta$ for Windowing Size of 10	141
6.6	MAE/RMSE of ΔL for Windowing Size of 10	142
6.7	Number of units in each layer of the <i>OriNet</i>	151
6.8	Considered variants of the proposed method	151
6.9	Comparing variants of the proposed method with others in terms of MAE/RMSE of $ \Delta Q $ estimation	152
6.10	Comparing OriNet full quaternion estimation capability with IO and VIO frameworks. q_w , q_x , q_y , and q_z columns refer to the MAE in each coordinate, $ Q $ is the mean length of the error, and $\bar{\theta}_D$ is the mean distance between the estimated and ground-truth quaternions in degrees.	156

Chapter 1

Introduction

1.1 Overview

Autonomous vehicles have gained much attention in the past few years due to the significant application of autonomous vehicles on people's daily life; to name some Unmanned Ground Vehicles (UGVs) can help to transport peoples and goods among locations, and Unmanned Aerial Vehicles (UAVs) can help in firefighting or faster delivery of goods, in long distances.

In the early ages, researchers tried to make intelligent vehicles by utilizing traditional algorithms. However, recently by the presence of small and compact Graphical Processing Units (GPUs), it is possible to benefit from GPUs' parallel processing power on autonomous vehicles and execute novel methods with more computational costs fast. Hence, the researcher's direction has moved to Artificial Intelligence (AI) and Machine Learning (ML) methods to develop a fully intelligent autonomous system. Furthermore, recent advances show that Deep Learning approaches can solve problems more robustly and accurately, which has gained lots of attentions.

In order to make intelligent vehicles, various types of sensors can be utilized, and each of them has its own pros and cons [10]. Different types of sensors and their advantages and disadvantages can be seen in the Figure 1.1. Industries always look for a small, low-cost, and accurate solution. Among various types of sensor combinations, Inertial-Measurement Units (IMUs) can perform well when it is utilized with a vision sensor, and this combination has attracted researchers and industries.

Vision-based sensors can provide rich information from the environment, which can solve most autonomous vehicles' problems. To name some, vision-based sensors can be used to solve problems such as obstacle and object detection, object tracking, and simultaneous localization and mapping (SLAM). However, processing the sensor's data to extract the required information accurately is challenging.

	Sensor	Advantages	Disadvantages
GPS/GNSS	<ul style="list-style-type: none"> - Provides absolute position with known value of error - No error accumulation over time 	<ul style="list-style-type: none"> - Unavailable in indoor, underwater, and closed areas - Affected by RF interference
	IMU	<ul style="list-style-type: none"> - Provides both position and orientation using 3-axis accelerometer and gyroscope - Not subject to interference outages 	<ul style="list-style-type: none"> - Position drift (position estimation requires second-order integral) - Have long-term drift errors
	Ultrasonic sensor	<ul style="list-style-type: none"> - Provides a scalar distance measurement from sensor to object - Inexpensive solution 	<ul style="list-style-type: none"> - Reflection of signal wave is dependent on material or orientation of obstacle surface - Suffer from interference if multiple sensors are used - Low angular resolution and scan rate
	Laser sensor	<ul style="list-style-type: none"> - Similar to sonar sensors but has higher accuracy and scan rate - Return the distance to a single point (rangefinder) or an array of distances (scanner) 	<ul style="list-style-type: none"> - Reflection of signal wave is dependent on material or orientation of obstacle surface - Expensive solution
	Stereo	<ul style="list-style-type: none"> - Image scale and depth information are easy to be retrieved - Provide 3D vision 	<ul style="list-style-type: none"> - More expensive and needs more calibration effort than monocular cameras - It is degraded to the monocular case when the stereo baseline is much smaller than the distances to the scene from the camera - Difficult interfacing and synchronization.
	Monocular	<ul style="list-style-type: none"> - Low cost and easy deployment - Light weight: good for small robots - Simple calibration 	<ul style="list-style-type: none"> - Suffer from image scale uncertainty

Figure 1.1: Different types of sensors generally utilized by autonomous vehicles

Current vision-based systems cannot reach the reliability level reachable by utilizing other types of sensors (bigger and more expensive). Processing time is another key feature everyone is looking to, and some of the vision-based approaches that can overcome other sensors have required more processing time. Hence, it is essential to investigate a robust method that can work in real-time with vision-based sensors.

This study mainly focuses on solving the problem of monocular visual-inertial odometry. In this study, the best approach to solve visual and inertial odometry problems separately investigates, and this study is not focusing on the fusion of these sensors. Odometry refers to tracking the scale of position and orientation changes of the robot as it moves along the path; so the position of the robot relative to the initial position is the outcome. It is an important module for autonomous robots in urban areas and indoor environments in which the Global Positioning System (GPS) signal is not available; it is also a useful module to reduce GPS error in scenarios that include GPS signal. Theoretically, it is not possible to extract the scale of movement with a single camera, so an IMU sensor is utilized. While IMU can help monocular cameras to extract the scale of movement, it suffers from bias and measurement noise, which makes the problem challenging. It is also important to mention that 6-DoF camera relocalization is a problem that tries to understand

the position and orientation that an image is captured in the environment; considering previously recorded images with the 6-DoF (position and orientation) labels. This problem is a sub-problem of visual odometry, and throughout this thesis, the proposed approaches are analyzed under this problem mainly, and then the approach is modified and tested as a visual odometry module. Knowing the revisited states and their position via the camera relocalization module helps the odometry module recover the full SLAM system error. At the same time, it is a critical module alone to understand the location in an environment. To make problem definitions clear, the problems this thesis is focusing on are as follow:

- **6-DoF Camera Relocalization:** In this problem, the training data consists of images with the corresponding position and orientation label that demonstrate the 6-DoF position that each image is captured from in the environment. The position and orientation labels are relative to a reference point in each scenario. For a given test query (image), the 6-DoF camera relocalization module should estimate the capturing point's position and orientation.
- **Visual Odometry:** In this problem, the training data consists of consecutive frames with their relative position and orientation changes. The visual odometry module should return the relative position and orientation change for a given consecutive test frames in the execution time.
- **Inertial Odometry:** The training data consists of consecutive IMU readings and the position and orientation of the robot. The IMU frequency is more than position and orientation labels, so for each subset of IMU readings, one position and orientation label exists. For the given consecutive IMU readings, as test queries, the inertial odometry module should return the relative position and orientation changes between the IMU readings subset.

The rest of this chapter gives an overview of this research work and contributions.

1.2 Motivation

Odometry refers to the robot's relative localization ability. In this manner, the robot uses its sensors (e.g. visual and inertial) to localize itself relative to the start point. This module is the core module of any autonomous robot where the robot wants to behave intelligently to achieve a goal in the environment.

Autonomous navigation systems require knowing their local or global position in the environment in order to do path planning and create a map of the desired scene. Although different odometry systems are proposed, they suffer from various problems depending on the type of sensor they use. For instance, stereo systems require accurate calibration parameters (the relation between multiple cameras and intrinsic parameters of the camera), RGB-D sensors like Microsoft Kinect suffer from measurement noise, and Light Detection and Ranging (LIDAR) systems are bulky and expensive. Among all different sensor combinations, using a monocular camera simultaneously with an Inertial Measurement Unit (IMU) is in researchers' focus in the past three years. This is the smallest low-cost combination of sensors that can be used in different applications, and more importantly, smart mobile phones also have this sensor combination.

As an essential requirement for autonomous navigation, Monocular Visual-Inertial Odometry (VIO) has been one of the most extensively studied topics over the past three years. Although traditional VIO methods have been widely utilized in various applications, their performance depends on the scene structure. Traditional methods are based on tracking feature points in consecutive frames (from visual sensor). Failure to track feature points in consecutive frames results from moving objects and the existence of poor visual information in the scene. Moreover, the IMU suffers from measurement noise, and the propagation of IMU measurements over time leads to large errors. Problems that a VIO module needs to tackle can be seen in the Figure 1.2 ¹.

Deep Neural Network (DNN) architectures, such as Convolutional Neural Networks (CNNs), are promising to overcome the bottlenecks and failures of traditional approaches and learn to extract and estimate camera motion in consecutive frames. In recent years, different network architectures have been proposed to solve VIO's

¹Part of the figure is from <http://www.autonomousroad.com/?p=20>

Figure 1.2: Problems that VIO modules should handle

problem, and such networks have achieved satisfactory results in places with low texture information. However, as a new research domain, the performance of the proposed methods is not promising. Existing works are different in the type of input (optical flow, raw RGB, etc.), network architecture, and sometimes the type of output and loss function.

Based on all the mentioned issues, the best network architecture that can solve the problem of Visual Odometry and Inertial Odometry (as separate problems) needs to be investigated, which is the focus of this research. Having the best approach for each of the aforementioned tasks helps the overall system and fusion module to be robust and accurate.

1.3 Objectives

The main objectives of this research are as the followings:

- Investigating a training approach and network architecture that assists in solving the regression problem of visual odometry robustly.
- Investigating an approach that can make visual odometry systems robust to moving objects and motion blur.
- Since extracting features and tracking them in the consecutive frames is the

main module of a visual odometry system, a robust feature detector plays a critical role and needs to be investigated.

- The best deep network architecture to solve the problem of inertial odometry with the presence of a single and particular IMU needs to be investigated.

1.4 Major Contributions of the Thesis

The main contributions of this thesis can be summarize as the following.

- Convolutional Neural Networks (CNNs) are mainly introduced to solve classification problems, in which there exists a huge number of observations for each of the desired labels. In the regression problems, such as visual odometry, the output space contains a wide range of variables, and the number of observations per each of the values in output space is very limited. Hence, the network cannot learn to solve the problem robustly. In this research, a new approach to train CNNs for regression problems is investigated in which the network trains as a classifier and learns to output a boundary (cluster), for each of the input observations, in the first step. After that, the network fine-tunes to solve the regression problem. Hence, the network solves the regression problem in the network's local boundaries for each observation learned in the first step. In this manner, the network will know each input's local boundary during execution, solve the regression problem inside that boundary, and perform robustly. The proposed approach is applied to solve the 6-DoF camera relocalization and visual odometry problems. The experiments and results show how it can help deep networks to be trained and perform robustly.
- Recent advances in Simultaneous Localization and Mapping (SLAM) systems make robots behave intelligently in different situations. However, performing well in dynamic environments requires the extraction of robust features from the static context of consecutive frames. Deep Learning, nowadays, helps to extract meaningful and robust feature representations. Using various Deep Learning architectures and retrieving the uncertainty information concerning

dynamic objects of the scene and the environment's motion flow helps traditional SLAM systems extract robust features in dynamic environments. This research uses deep semantic segmentation and deblurring modules to implement a robust platform to solve camera relocalization and visual odometry problems. The results show the significant influence of the proposed approach in solving the aforementioned issues when a moving object presents in the scene.

- The extraction of distinct and robust feature points is the main step of traditional camera relocalization and visual odometry modules. While CNNs can assist in improving machines' performance, they require a vast amount of annotated training data. This research has worked and introduced a new bio-inspired approach that can learn to extract robust handcrafted features fast with a single training data. Evolutionary algorithms are combined with the CNNs to model the brain's neurogenesis behavior. The proposed approach benefits from the knowledge of a pre-trained CNN that contains mature neurons and evolves a newborn convolutional neuron, via Particle Swarm Optimization, to detect corners robustly with the presence of a single training data in a short time. The results demonstrate that the proposed corner detection module outperforms existing techniques in terms of robustness in various conditions.
- Inertial Measurement Units (IMU) suffer from bias and measurement noise, which makes it much more complicated to tackle the problem of Inertial Odometry (IO). Due to the error propagation over time, an inaccurate estimation or a small error will cause the odometry and localization system unreliable and unusable in a split of seconds while estimating robot position. This research investigates novel deep Inertial Odometry network architectures that can assist in extracting the scale of camera movement and the full 3D orientation of IMU in the world frame. Furthermore, a method that can calibrate IMU bias in each execution is proposed. The experiments and results on a challenging Micro Aerial Vehicle dataset show that the proposed network architectures can perform robustly and outperform existing tech-

niques. The proposed full 3D orientation estimation module, which is based on a single particular IMU, can outperform existing methods which are fusing IMU with a monocular camera, which is a great achievement.

1.5 Organization of the Thesis

The organization of the thesis is as follows.

- Chapter 2, reviews the existing techniques for camera relocalization, visual odometry, and inertial odometry. Besides, feature detection techniques are also reviewed as a new feature detection module is proposed in this thesis.
- Chapter 3 presents a new approach to train CNNs for regression problems. The proposed approach is discussed and evaluated in solving 6-DoF camera relocalization and visual odometry problems. Furthermore, an approach that helps in scene categorization, path segmentation, and landmark extraction while traveling path is proposed. This module helps to divide the environment into sub-boundaries, which is the pre-requirement of the proposed training approach.
- Chapter 4 investigates a solution that assists in dealing with moving objects and helps traditional and deep learning based approaches to perform robustly with the presence of the moving objects.
- Chapter 5 proposes and evaluates a bio-inspired training approach that helps to learn and extract handcrafted features fast with the presence of a single training data.
- Chapter 6 proposes a deep network architecture that helps in solving the problem of inertial odometry. This chapter includes two novel approaches, where one is for estimating the scale of orientation and position changes, and the other one is for the estimation of the full 3D orientation of the IMU over time in the world frame.
- Finally, Chapter 7 concludes the proposed approaches and discusses the future of the proposed techniques.

Chapter 2

Literature Review

2.1 Overview

Localization, mapping, and planning are essential modules of an autonomous robot. Each of these modules requires different types of inputs in order to output various rich information. The odometry module's input can be (individually or cooperatively) stereo or monocular images from the visual sensor, depth information from sensing sensors, Inertial Measurement Unit (IMU) sensor measurements, etc.

This report's focus is mainly on the Monocular Visual-Inertial Odometry (VIO) for autonomous vehicles. Therefore, in this chapter, the previous works in this field and the research area of 6-DoF camera relocalization, the most significant sub-module of the Visual Odometry (VO) problem, which is considered in this research to perform experiments, are reviewed.

2.2 Visual Odometry

VO, as the base of localization, mapping and planning modules, outputs information about frame-to-frame camera movement, and lots of researches have been done in this field in the past few years [87, 95]. VO methods perform differently according to the type of input; for instance, methods based on stereo camera or depth sensors mostly try to use information from the scene structure, while methods based on a monocular camera use the change of image intensity. Hence, in one category, VO methods can be classified according to their employed visual sensor to Stereo and Monocular VO methods. Stereo methods extract the depth map of the scene using geometric methods. Converting this valuable information into point cloud helps to estimate camera movement accurately [7]. Scaramuzza and Friedrich [141] also show that minimizing pixel intensity errors and the error between projected 3D points among consecutive frames increases the odometry accuracy. However, depth information suffers from measurement noise, and lots of work is done to reduce it [43].

Monocular VO methods only have information about the intensity of each pixel. Such methods try to extract distinctive sparse or dense features from two consecutive frames, track and match them, and estimate the camera movement according to such features' change over time; Feature points are informative pixels in the image such as edges, corners, etc.. The significant difference between stereo and monocular VO methods is that monocular VO does not have any information about the depth of the field of view, and so lacks in estimating the scale of movement. Hence, such methods cannot estimate the scale of movement accurately. Recent methods suggest using Inertial Measurement Unit (IMU) as another input source to estimate the scale of camera movement [87, 95], which is also considered in this research.

VO methods can be classified as geometric and learning based methods. To estimate camera movement, feature points are extracted, described, and tracked in consecutive frames in the first category. In contrast, in the second category, learning-based VO, a model that can fit input information to desired output learns; besides, some techniques are learning to extract robust feature points via deep network architectures, and they do the next steps required by a VO module (tracking and matching features, and movement estimation) the same as the traditional techniques.

As a summary, Figure 2.1 illustrates different solutions and approaches researchers are using to solve the problem of VO. In next subsections, each of these approaches are reviewed.

2.2.1 Traditional Visual Odometry

Traditional vision-based Simultaneous Localization and Mapping (SLAM) methods solve the VO problem in a multi-step manner [115]. In the first step, they extract feature points from the two consecutive frames. Afterward, the extracted features describe and the corresponding feature points in two frames detects according to the description of different features. Using the position information of the matched feature points, the camera translation and orientation displacement parameters retrieve. Hence, the most critical module in this pipeline is robust feature extraction. Based on the number of features a method considers to solve the problem of visual

Figure 2.1: Different approaches researchers use to solve the problem of visual odometry. An example approach, for each technique, is mentioned by green color. (methods: PTAM [78], DenseVLAD [164], ORB-SLAM [115], DTAM [117], seqSLAM [108], PoseNet [75], NetVLAD [11], DeepVO [170], X-View [52], Attentive Deep Local Features [121])

odometry, it can be classified as a sparse or dense method. Sparse methods use only a small selected subset of pixels (feature points) in an image (frame). On the other hand, dense approaches use most or all of the pixels in each frame. The computational cost of dense approaches is more than sparse approaches, but they are more robust and accurate in texture-less (feature-less) scenarios. In sparse approaches selecting the best subset of pixels in a frame plays a critical role; the subset of pixels should be consistent and be matched in consecutive frames.

In another way, the VO methods can be classified as direct and indirect methods. Direct methods use pixel intensity directly to match features and do optimization on map and camera parameters. On the other hand, indirect approaches try to extract high-level feature representations for feature points, which help them get robust to change in illumination while matching features. Figure 2.2¹ categorizes traditional VO techniques based on the type of features they have utilized.

ORB-SLAM2, as an example, introduced by Mur-Artal et al. [115] is one of the most popular and state of the art traditional SLAM methods. ORB-SLAM2 consists of tracking, mapping, and loop closing modules, which run in parallel. A Bundle Adjustment(BA) module runs once a loop detects by the loop closing module; BA minimizes the 3D reprojection error across different views in the map. The mapping

¹Part of the figure is from <https://blog.cometlabs.io/teaching-robots-presence-what-you-need-to-know-about-slam-9bf0ca037553>

Figure 2.2: Categorizing popular state of the art traditional VO methods based on the type of features they are utilizing (methods: PTAM[78], MonoSLAM[34], ORB-SLAM[113], SVO[49], DSO[169], LSD-SLAM[42], DTAM[117])

module optimizes the local map by applying the local bundle adjustment, and the loop closing module detects the loop to correct the drift of motion estimation. ORBSLAM2 uses ORB features [134] to extract feature points, and matches them in consecutive frames in order to estimate motion. Figure 2.3(a) shows the workflow of the traditional SLAM methods.

As stated, conventional VO methods extract and track features in consecutive frames, which can be the scene's structure in stereo images and corners or edges in monocular images. Hence, such methods' performance is high in places with the existence of a high number of feature points, and they may lose track in challenging low-texture environments. Various types of feature extraction and description techniques have been introduced in the last decay. The well-known SIFT [99] feature extraction and description method is widely used for applications such as object recognition. However, the computation time of SIFT is quite high, which makes it unusable in real-time applications such as VO, SLAM, etc.. SURF [15] features have been introduced afterward to speed up feature extraction. More investigates applied to find faster versions based on the GPU [147]. Traditional and deep learning based handcrafted feature detection techniques will be reviewed in

(a) The Work-flow of Traditional SLAM Methods such as ORBSLAM2

(b) The Work-flow of Deep Learning based SLAM Methods

Figure 2.3: Comparison of the Pipeline of Deep Learning based and Traditional SLAM Methods

the next sections. It is also important to mention that conventional methods also require an accurate camera calibration (intrinsic/extrinsic camera parameters) to give a robust, reliable, and stable output.

2.2.2 Deep Learning based Visual Odometry

Deep Learning, mainly deep Convolutional Neural Networks (CNN), achieved satisfactory results to solve classification problems in the field of Computer Vision and became in the focus of researchers since 2012 [80]. CNN learns to extract various types of features and use their union to describe the whole or part of an input image. Since a deep CNN is learning to extract a meaningful representation for perceptions based on the mixture of the abstract features, learned description is robust to noise and handles specified challenges. Robust high-level feature representations, learned by deep CNN, has been used to solve different computer vision problems [129, 91].

Researchers' focus in robotics was also moved to CNN and Deep Learning in the past five years when the first work was published in this area in 2015 [79]. The only limitation of such machine learning techniques is the requirement of training data, which should be similar to the testing environment to get an accurate and robust result. The learning procedure learns to extract feature points according to input training images' appearance. Hence, if the environment changes too much, the model needs to fine-tune by providing some training data.

Deep Learning based Visual Odometry techniques can be classified into two categories. One category is solving the problem in an end-to-end manner [90]. Researches in this category are feeding two or multi consecutive frames into their deep network architecture, and their output is the position or orientation changes. They train the network either in a supervised manner [170, 109, 112, 79], by utilizing the position and orientation change labels, or in an unsupervised manner by training the network based on stereo vision. The unsupervised approaches utilize frames captured from parallel cameras at the same time step to compute depth information; then, the depth information is utilized in the loss function at training time to minimize the re-projection error or feature reconstruction loss based on the 3D information [8, 181, 90]; they execute the trained model based on a monocu-

lar camera in test time. On the other hand, since robust feature point extraction is the main key for a robust odometry system, some methods tried to learn and extract robust feature points from input images [14]. Both categories have their own pros and cons. The first category gets more sensitive to camera parameters and the environment, and when it is going to be tested with a new type of camera or new environment, the network needs to be fine-tuned. However, they can perform robustly in the presence of moving objects, illumination changes, motion blur, and so on. On the other hand, methods that are extracting robust feature points by deep learning techniques are fast and accurate, but their performance lacks in the presence of moving objects or in the textureless scenarios. However, new techniques, like one proposed in this thesis, benefit from semantic information and other deep uncertainties, and combine them with traditional techniques, to solve mentioned issues and deploy fast and reliable approaches.

2.3 6-DoF Camera Relocalization

Outdoor 6-Degree of Freedom (DoF) camera relocalization is one of the challenging tasks for autonomous robots and intelligent systems. This module enables them to determine revisited states for navigation systems [94] and reduces the odometry error of Simultaneous Localization and Mapping (SLAM) systems using a loop closure module [9]. Place recognition [25] is also one of the subproblems which enables intelligent systems to recognize the nature of the environment including the location of the captured image. It is also an essential module for virtual reality systems where a 3D model needs to join the scene and identifying the camera position is critical [103].

It is also important to mention that, while recent advances in the problem of Visual Odometry achieved satisfactory results using unsupervised learning [177, 128, 175], they still require to solve the problem of camera relocalization or loop closure to reduce the error of odometry estimation. Having a robust solution for camera relocalization leads to a robust Visual Odometry solution, and they are not separate.

Existing conventional appearance-based approaches [12, 137, 165, 179] are saving descriptor of extracted handcrafted feature points, such as scale-invariant fea-

ture transform (SIFT) [100], speeded up robust features (SURF) [15], oriented features from accelerated segment test (FAST) [132] and rotated binary robust independent elementary features (BRIF) [134], from training images in a database. Afterward, using different techniques such as Bag of Words (BoW) or modeling descriptors using Gaussian Mixture Models (GMM) the position and orientation of a new query retrieves.

While such database matching methods act fast, their performance drops significantly by increasing the scale of the problem. They also require big storage to save feature descriptor and label of the training data to match with the test query. On the other hand, methods based on Structure from Motion (SfM) [93, 138] create a 3D model of the scene in the beginning and associate a descriptor to each of the 3D points. For a test query, they extract feature points and describe them to match the 2D descriptors with the ones in the 3D map. n -point [81] solver also uses as an alternative solution to match feature points using RANSAC [47]. Such methods complexity also increases exponentially by the size of the problem. To resolve this issue, correspondence search approaches terminate searching after finding a fixed number of correspondences [93], or they only use a subset of 3D points and their descriptor to reduce the memory usage [21].

Both mentioned approaches, still cannot perform well in large-scale environments where there is a chance to have two or multiple points with similar properties. This characteristic result in failure of the feature matching module and grows the RANSAC run-time. co-visibility information uses by researchers to adapt the RANSAC sampling method and also remove outliers [92, 136, 139, 161].

To increase the accuracy and generalization ability of the conventional methods researchers combined machine learning techniques with the conventional ones. Gronat et al. [60], and Cao and Snavely [20] trained a linear classifier on top of the Bag of Words (BoW) to understand the change of the scene. Williams et al. [173] examined different loop closure modules and proved that methods using BoW are the best in modeling the large-scale environments. FAB-MAP proposed by Cummins and Newman [32] used SURF feature points and BoW and achieved satisfactory results. However, SURF feature extraction is computationally expensive, and researchers are not able to use it directly in robotic applications. To speed

up, Gálvez-López and Tardos [51] used a bag of binary words on top of FAST and BRIEF feature descriptors. While their method performs in real-time, their feature representation is not scale and rotation invariant and so fails in large-scale environments. Mur-Artal and Tardós [114] introduced the binary bag of words based on ORB features to benefit from scale and rotation invariant features in binary BoW, and they have used it in robotics applications [113] as a SLAM module.

Other types of conventional methods benefit from machine learning techniques to model the behavior of the feature descriptors. Donoser and Schmalstieg [39] consider each feature as one cluster and solve the feature matching as a classification problem by ensembling of random ferns. Random Forest learning model used by Guzman-Rivera et al. [62] and Shotton et al. [145] to predict the position of each pixel in 3D and solve camera relocalization on RGB-D data. Other researches model uncertainty and predict the probability map for an input image and manage the problem on top of the probability map [127]. While the mentioned methods are based on extracting feature points and matching them in 2D-3D, some works in the domain of RGB-D camera relocalization, use high-level features such as line segments or planes to solve the problem. Alehdaghi et al. [7] introduced the parallel RANSAC to extract planes of the scene quickly and understand the motion of camera by matching correspondence planes in consecutive frames. Cupec et al. [33] further selected informational lines and planes from input frames to solve the problem of place recognition. However, such RGB-D methods require a depth map which obtains using stereo vision or lidar sensors. Such sensors are expensive, sensitive to noise, have a complex structure, require calibration and industrial people do not like to use such methods.

The introduced methods perform well when the visual content and light are consistent, and there is no moving object in the scene. Besides illumination change and moving objects which change the appearance of the image, Motion Blur is one of the most significant difficulties which occur widely when an image captures with a moving camera or significant motion exist in the scene. Moreover, hand-craft features cannot extract informative features in the existence of motion blur, and relocalization module fails during execution. Facing a mirror is another major challenge in an outdoor environment where everything in the captured image is

swapped. Getting a decision in such scenarios is hard and risky in real-time applications especially micro aerial vehicles. Figure 2.4 is demonstrating the mentioned problems.

Despite the conventional methods, where an extensive database of descriptors of the scene requires to be saved and match with the test image descriptors, in CNN based approaches a test image regresses to the expected position and orientation according to the learned weights of the network on the training data. Since only the weights of the network are critical, the size of memory and execution time is constant by changing the scale of the scene. Hence, this solution is useful to solve large-scale problems. In theory, deep CNN methods should be better than traditional 6-DoF camera relocalization methods. However, finding an optimum solution is complicated, and the existing deep CNN solutions cannot outperform traditional ones and researchers are still working on it. The main drawback of deep CNN in such problems is their requirement of high generalization ability with the presence of a low number of training data. While deep CNN approaches can not outperform traditional methods, they recently can outperform them in case of facing a large training data, especially in the problem of visual odometry [170].

Chen et al. [26] firstly used CNN in the problem of place recognition, and by learning robust representation for different places they improved the accuracy of place recognition significantly. Sunderhauf et al. [159, 160] advanced the system robustness by breaking CNN architecture into two stages of learning appearance change, which manages by middle layers of the network, and getting robust to the change of viewpoints by top layers. Arandjelović et al. [11] also trained a CNN to extract a single descriptor for each image in the database and solve the problem of the place recognition using its rich descriptor.

The first deep CNN module to solve the 6-DoF camera relocalization problem, called PoseNet, is proposed by Kendel et al. [75] which trains the GoogLeNet network architecture [162] end-to-end as a regressor. This method achieved satisfactory results in large-scale outdoor environments where conventional methods require big storage, and their performance drops significantly. Kendel et al. [73] enhanced the performance of their proposed approach by modeling uncertainty and using the Bayesian model. Walch et al. [168] added a Long Short-Term Memory

(a) Moving pedestrians and cars in the scene

(b) The reflection of the building inside the mirror of the shop

(c) Effect of Motion Blur

(d) Part of the image (c) to show the effect of Motion Blur

(e) Effect of illumination change

(f) Effect of illumination change

Figure 2.4: Challenges to solve in the problem of 6-DoF camera relocalization. (a) shows the presence of dynamic objects in the environment which prevents modeling the static scene. Windows of the shop in (b) reflect the front building which tricks the relocalization module. Motion Blur blurs the sharp parts of the building in images (c) and (d) where robust informative features cannot extract and match. (e) and (f) are demonstrating the effect of the change in the light of the scene in which different correspondent pixels in two images have different values

Figure 2.5: Deep Learning versus Conventional Methods in the Problem of 6-DoF Camera Re-Localization

(LSTM) at the end of Kendel et al. network architecture. This layer helps to learn the correlation of the feature representation, and reduce feature dimension to have a dense meaningful representation of the image.

Other methods feed sequence of consecutive frames into a Long Short-Term Memory (LSTM) unit [29]. While such methods perform well, they require a flow of images and cannot work with a single shot in real-time. Li et al. [89] added a channel to PoseNet to learn from both RGB and Depth data in indoor environments and benefit from the depth map in RGB-D cameras.

As a summary, Figure 2.5 illustrates the workflow of conventional and deep learning based 6-DoF camera re-localization techniques.

2.4 Handcrafted Features Extraction

Handcrafted feature or interest point extraction refers to extracting distinctive informative key points from images. These feature points could be edges and corners which are robust to noise and occlusion and can describe the image properly. Such features are the primary requirement used in various computer vision applications

such as place recognition [51], visual odometry [126] and object detection [57]. Due to the different applications of handcrafted features, many kinds of researches have been devoted to this area [63, 86, 122, 38], and researchers are working on extracting unique feature points that are reliable to scale, occlusion and illumination change. However, finding a final accurate and computationally efficient solution for this purpose is impractical.

Handcrafted feature points are referred to pre-defined features such as corners or edges, while interest points are the points with specific behavior which are trackable in corresponding images globally. Initial methods regarding the handcrafted keypoint extraction mostly concentrate on finding pixels with a specific characteristic, such as being local maxima in the derivative of the image intensity. More advanced methods such as Scale Invariant Feature Transform (SIFT) [100] and Speeded Up Robust Features (SURF) [15] are not focusing on extracting a specific type of handcrafted features, and they investigate image to extract interest points. They benefit from the Difference of Gaussian (DoG) and try to find regions that are distinctive from their neighbors. SIFT [100] is robust to scale, but its excellent performance achieves by high computation cost. An ensemble of different methods is used to propose SURF [15] to benefit from the robustness of the SIFT feature with a lower computational cost. However, recent researches mostly attempt to use machine learning and neural network methods to extract various features from an image [122].

The trade-off between computational cost and accuracy has always been challenging in the field of handcrafted feature extraction. While deep learning based methods can achieve satisfactory results, they cannot execute in real-time on low-power embedded systems and CPU; besides, they need a massive amount of annotated training data to obtain a robust result. On the other hand, traditional methods can perform in real-time, but they lack in accuracy and robustness. This section is reviewing existing traditional and learning based feature detection approaches.

2.4.1 Traditional Feature Detection Techniques

The algorithm developed by Moravec et al. [110] is one of the earliest corner detection techniques. It measures the change of intensity in a rectangular patch at every 45° and finds the local maxima. Due to several shortcomings of this method, Harris, and Stephan [63] developed an improved version of the Moravec algorithm. This method applies the Gaussian function instead of the rectangular patch and analyzes all neighbors rather than just considering neighbors in the directions of 45° . Harris corner detector evaluates the change of the intensity for the shift $[u, v]$ as

$$E(u, v) = \sum_{x, y} \mathbf{w}(x, y) [I(x + u, y + v) - I(x, y)]^2 \quad (2.1)$$

where $\mathbf{I}$ is the grayscale image and presents the intensity of pixels, (x, y) are pixels indices in the image coordinate, and $\mathbf{I}(x, y)$ demonstrate the value of pixel in the (x, y) position of the image. $\mathbf{w}$ is the window function which could be Gaussian to give more weight to the center pixel of the window or pulse function to give the same weight to all pixels in its window. Considering that, corners have a significant change of intensity in all directions, the desired goal is extracting patches with large $E(u, v)$, which demonstrate the large difference in neighboring pixels. Harris has simplified the solution by applying the Taylor series and proposed a corner response as

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 \quad (2.2)$$

where k is an empirically determined constant, and λ_1 and λ_2 are eigenvalues in

$$\mathbf{M} = \sum_{x, y} \mathbf{w}(x, y) \begin{bmatrix} \mathbf{I}_x^2 & \mathbf{I}_x \mathbf{I}_y \\ \mathbf{I}_x \mathbf{I}_y & \mathbf{I}_y^2 \end{bmatrix} \quad (2.3)$$

where $\mathbf{I}_x$ and $\mathbf{I}_y$ refer to gradient of the image $\mathbf{I}$ in x (horizontal) and y (vertical) directions respectively. R completely relies on the eigenvalues of gradient of image $\mathbf{I}$ and it is large for corners. Shi-Tomasi [143] corner detector uses $\min(\lambda_1, \lambda_2)$ instead of using Eq. 2.2 as the corner response, and according to this measurement, they select more robust features.

Other groups of algorithms do not use the derivative of images. Smallest Uni-

value Segment Assimilating Nucleus (SUSAN) method introduced by Smith and Brady [149], calculates the number of pixels in a circle mask, which their intensity is similar to the center pixel; they name this number as USAN. Small USANs with a value below a certain threshold has more probability of being a corner. The Feature from Accelerated Segment Test (FAST) method [133] is a variation of SUSAN, which considers a point as a corner if n contiguous pixels in the circle are all brighter or darker from the center pixel with a certain threshold. This method significantly improved the time efficiency of SUSAN's approach by sampling pixels from neighbors of the candidate pixel.

All methods are considering a boundary around pixels, and by comparing the gradient or intensity of the center pixel with the surrounding, they measure a corner response which helps to classify a pixel as a corner. Hence, they apply a pixel level filtering. This approach is not limited to corner detection methods, and it is also used for extraction of various kinds of features such as edges. The similar behavior between an edge and corner detection which exists in different kinds of handcrafted feature extraction modules is applying a window on a local boundary and convolving this window with a filter to extract knowledge. This concept is repeatedly used by CNNs to extract meaningful abstract feature representations.

One of the fundamental problems with the handcrafted feature extraction techniques is their problem with the scale and rotation transforms. Since the relation between pixels in an image changes with the scale and rotation transforms, researchers moved towards proposing interest point detectors such as Scale Invariant Feature Transform (SIFT) [100], Speeded Up Robust Features (SURF) [15], Binary Robust Invariant Scalable Keypoints (BRISK) [86], and Oriented FAST and Rotated BRIEF (ORB) [134]; BRIEF stands for Binary Robust Independent Elementary Features [19] technique. Such methods are performing feature extraction in multiple scales and rotations. For instance, for an input image, they defined interest points as maxima and minima of the Difference of Gaussian (DoG) in scale space. However, their performance still lacks in different scenarios, and researchers recently moved towards benefiting from CNNs to extract more robust features.

2.4.2 Deep Learning based Feature Detection Techniques

2.4.2.1 Overview of Learning Procedure in Deep Learning

Deep Neural Networks (DNNs) achieved satisfactory results to solve various problems in recent years by applying higher dimension nonlinear transforms on the input. For instance, given input $\mathbf{X}$, DNNs output representation $\mathbf{R}_n$ for the n^{th} layer by computing

$$\mathbf{R}_n = f_n(\mathbf{W}_n^T \mathbf{R}_{n-1} + b) \quad (2.4)$$

where

$$\mathbf{R}_1 = f_1(\mathbf{W}_1^T \mathbf{X} + b) \quad (2.5)$$

and $\mathbf{W}$ and f are learnable weights of the network in each layer and the non-linear activation functions respectively; and b is the bias. DNNs have more than 2 layers in their structure. Figure 2.6 is illustrating the deep neural networks architecture.

Figure 2.6: Deep Neural Networks Architecture (W_i^j represents the weight for the neuron j in the layer number i)

DNNs include Convolutional Neural Networks (CNNs), Auto-Encoders (AEs), and Deep Belief Networks (DBNs). Among these, CNNs have achieved satisfactory results in recent years because of their hierarchical multi-step learning procedure. CNNs solve problems in two main steps: a) extracting features in the early stages,

and b) describing features to classify or regress input in the later ones. Different feature representations can be found for the input $\mathbf{X}$ by changing the number of layers or the type of activation function used in each layer. Simonyan and Zisserman [146] have shown that increasing the number of layers helps to improve the overall accuracy of the network with the presence of a massive amount of training data. A static non-linear activation function usually applies to each neuron to change the dimensionality of features, and some works have shown that learning activation function in the training phase can help to increase the accuracy [3]. While increasing the number of layers in a deep CNN, and learning the activation function helps to improve the accuracy, they also increase the training time as well.

In order to update the weights of neural networks and find the optimum weights for a specific problem, Back Propagation (BP) algorithm is used [83]. Its performance depends on the hyper-parameters, such as the learning rate and type of activation functions. Besides, grid search [101] and evolutionary algorithms [40] are used to find the optimum hyper-parameters in the literature.

The first stage of CNNs consists of various convolution and pooling layers which extract various features over different scales. Different network architectures have been proposed in recent years, and researchers have improved the efficiency of architectures by introducing and using residual gates [66] or inception modules [162]. Each neuron of the convolutional layer learns weights of a filter with a specific size to extract features from the input image (the filter made by weights convolves with the input), and pooling layer is downsampling the input image. The max operator usually uses in the pooling layer to downsample the input image. The pooling layer aims at resizing the image to investigate robust features in multiple scales.

After the feature extraction step, in convolutional neural networks, the learned feature representation is flattened and moves towards Fully Connected (FC) layers, which are a simple Multi-Layer Perceptron (MLP). This MLP is learning to map the feature representations to the output domain, and so can describe feature representation in higher dimensions. The output of the network is compared with the label of the training data in supervised learning, to measure the error using the defined loss function. Then, using the BP algorithm, the weights of the network are updated to minimize the error. The stages of a deep CNN are shown in Figure 2.7.

Figure 2.7: CNNs architecture is illustrated in this figure. W_i^j represents the weight for the neuron j in the layer number i and $*$ stands as convolution operator. The feature detection module is using convolutional and pooling layers to extract robust feature representations from the image, and the description module is an FCN that learns to map input to desired output. The learning policy for the FCN module is presented in equation 2.4.

Recent advances in CNNs have shown its satisfactory result in solving various kinds of problems such as image recognition [146], camera relocation [45], and monocular depth estimation [58]. However, the way CNN learns and how it performs is still a black box for researchers, and lots of researches have been done to understand different layers. Zeiler and Fergus [180] have done the most significant study to understand the CNNs. They have visualized weights of different layers and applied the deconvolutional operator to the result of the network to understand various features extracted by the network in different layers. Their result shows that lower layers of the network are extracting basic features while the later layers are learning more abstract features and can extract a semantic from images.

The deep CNN by itself is not scale and rotation invariant, and further mechanisms require to make network robust to such transforms. Observing different images of an object with different scales and rotations helps the network to become robust to such transforms. In addition, each neuron learns to extract features for different rotations and scales, and once an image, with a specific rotation, inputs to the network, the neuron with similar rotation with the input activates and network performs robustly on such observations. Hence, having many neurons in each layer helps the network to become robust to different transforms and extract various kinds of features.

2.4.2.2 Review of Deep Learning based Interest Point and Handcrafted Feature Detection Techniques

Recently, some studies have utilized CNNs for detecting interest points. DeTone et al. [38] have proposed a two-step method to find robust interest points using CNN. In the first step, a CNN with a massive amount of data is trained to extract interest points; then in order to make this method robust to different transforms, another CNN is trained with the learned knowledge to find similar features between two images (feature matching). Yi et al. [176] have also proposed another approach to learn invariant features. Their method consists of three main components: a) feature detector, b) orientation estimator, and c) feature descriptor. Unlike other researches, the technique proposed by Yi et al. first learns the descriptor of the image pixels and then moves towards detecting the features in the image. LF-Net [122] creates a score map on the scale pyramid of the images to discover interest points. They also use another copy of the same network to detect features from the warped image. Then, they use the extracted features to estimate the transform between images, to minimize the error of transform estimation. Dusmanu et al. [41] have also considered a similar approach and detected feature points after finding a description for input image pixels; they have considered a single network for detection and description of feature points. While their method can learn rich feature representations, it significantly lacks robust feature point detection and cannot extract the same pixel, with a distance of less than 4 pixels, over images captured from the same place in different conditions. To benefit from the advantages of different techniques, Barroso-Laguna et al. [14] used a combination of traditional and learning-based interest point detection modules. They demonstrated that combining different approaches helps to benefit from the advantages of different methods to propose robust feature extraction techniques. While proposed methods can outperform traditional ones, they require a massive amount of training data, and they mostly lack in the presence of extreme in-plane rotation transform. Furthermore, most of these methods are interest point extractors and are not forced to extract handcrafted features such as corners.

2.5 Inertial Odometry

Among widely used sensors, the Inertial Measurement Unit (IMU) is a small size and low-cost sensor which is in focus of industries. However, this sensor suffers from measurement noise and bias, which makes the problem of odometry much more complicated to be solved by a single IMU. While recent researches [126] have achieved satisfactory results by fusing IMU accelerometer and gyroscope information with other sensors, solving the problem of odometry with single IMU requires more focus in order to find the optimal - robust and accurate - solution.

IMU measures linear acceleration, with an accelerometer, and angular velocity, with a gyroscope, in the IMU body frame at each time step, which can be used to estimate robot orientation and scale of movement in the world (robot) frame. In the early stage, IMU was fused with GPS in order to implement an accurate odometry system [67], and recently the monocular Visual-Inertial Navigation Systems (VINS) are in the focus of researchers [126]. The existing Inertial Odometry (IO) systems can be classified into traditional pre-integration methods based on the IMU Strapdown Inertial Navigation System (SINS) [140], and learning-based methods [44, 30] which train a network to extract rich feature representation from IMU measurements.

The early versions of the inertial system were massive, big and expensive but they were accurate. These properties make applications of previous inertial systems to be on ships, submarines, etc.. Microelectromechanical systems (MEMS) researchers enabled small and cheap IMU to be available on all kind of robots [16] and smartphones [102]. The accuracy of recent IMUs is not comparable with prior systems, and IMU measurements need to be fused with other informative sensors such as GPS, camera, LIDAR, etc. to get a reliable result. In this section, the traditional, and learning-based IO approaches are reviewed.

2.5.1 Traditional Inertial Odometry Methods

There are longtime focus and research on Strapdown Inertial Navigation System (SINS) [140], to find the optimum inertial odometry solutions. SINS workflow, shown on Figure 2.8, follows the mathematical model of IMU. In this manner, IMU accelerometer and gyroscope need to be transformed to world navigation

Figure 2.8: Comparison of SINS and PDR workflow

frame first, and after applying single and double integral to measurements the orientation and position of the sensor are retrieved. Variable G in Figure 2.8 refers to gravity which should remove from the measurement of accelerometer before any calculations.

Pedestrian Dead Reckoning (PDR), on the other hand, first determines the steps of movement and its orientation in the application of human position tracking; and after reading measurements from IMU for each step, the length of movement measures. However, this system also suffers from the accumulation of errors, because of wrong step detection, inaccurate length measurement and also a high number of parameters which needs to be tuned. Fusing PDR with other sensors such as floor plan [174], WiFi fingerprinting [69], etc. helps PDR work better, but the drift of position tracking not resolves. This model works for human movement tracking and cannot be mounted to robots directly. Step and Heading Systems (SHS) also use the inertial measurements to handle the PDR. SHS works well in short range movement but needs to apply in a fixed orientation of the inertial sen-

sor. OpenShoe is an INS project [119, 118] that relies on several inertial hardware mounted on foot.

Recent advances in Inertial Odometry (IO) are following SINS workflow by benefiting from Extended Kalman Filters (EKF). EKF encourages modeling series of noisy observations over time and applies to Accelerometer and Gyroscope information to model uncertainty and output reliable information [4]. However, error estimation requires in EKF, and most EKF based odometry systems are combining IMU information with other sensors such as GPS and Magnetometer [125].

EKF systems which use IMU as the only source of information, follow the zero-velocity update algorithm [148] because the only source of auxiliary information in IMU is zero-velocity updates [151, 119]. In this manner, IMU needs to have zero-velocity in a particular time to measure the IMU orientation from the accelerometer [50] and estimate bias and measurement noise [151]. Zero-velocity based EKF systems are detecting periods where the IMU is stationary; afterward, they are using measurements in stationary periods to update the IMU errors and filter the measurements [50]. Such an EKF model cannot apply to intelligent robots and autonomous vehicles which are not stationary for an extended period, such as micro aerial vehicles.

2.5.2 Deep Learning based Inertial Odometry Methods

Deep Learning approaches, especially Sequential Deep Learning methods like Long Short Term Memories (LSTMs), can extract essential feature representations from input space to model the output. LSTMs apply on a frame or sequence of data points, and finding a distribution that matches a sequence of feature representations can output a more accurate result. While IONet [24] is working on solving the problem of odometry based on IMU information of smartphones for pedestrian tracking, recent works are combining IMU and visual information to solve the drift problem of IMU in the problem of robots odometry [30]. Robots have lots of sudden motions, and small noise in the IMU measurements makes the problem of robots odometry much more complicated. Hence, there are not much literature on deep learning based Inertial Odometry for autonomous vehicles, and they are mostly in the area of Deep Learning based Visual-Inertial systems.

Figure 2.9: IONet [24] Workflow

Chen et al. [24] introduced a method based on LSTMs to learn the odometry of smartphones using their IMU measurements. While their method achieved a good result in the area of human pose estimation, their method is not robust to noise and change of IMU frequency. Their network architecture consists of two connected LSTM layers, each with 96 hidden cells. Their model learns to output the movement of IMU for each set of IMU measurements in a group of continuous-time measurements in a window size of N measurements. Their method performs well for IMU motion estimation in smartphones but cannot perform well on real robots because of not modeling uncertainty and noise in their architecture, which makes their network have low generalization ability. Besides, they have considered a fixed place for the smartphone in different scenarios. Figure 2.9 demonstrates the workflow of their method. It is important to mention that for real robots, especially micro aerial vehicles (MAVs), sudden motions exist, causing position drift and resulting in high noise variations; it makes modeling noise in the architecture significant.

Wagstaff and Kelly [167] proposed a zero-velocity EKF model which updates measurements using EKF at certation time that the LSTM model detects a zero-

velocity state. Similar to traditional methods, this model is based on zero-velocity EKF and requires knowledge about stationary states of the IMU. Their model outperforms conventional methods in the area of pedestrian inertial navigation. However, their model is still not usable in the field of autonomous vehicles where the vehicle is not stationary for a long time.

Other works are fusing visual and inertial sensors, and concatenate visual and inertial feature representations to output a robust measurement which discusses in the next section.

2.6 Visual-Inertial Odometry

Since IMU itself is faulty, modern methods are fusing information from IMU with other types of sensors such as GPS, Lidar, etc. [22]. To fuse information from different sources EKF or graph optimization usually uses [88]. For instance, Alatiser and Hancke [5] proposed a visual-inertial technique based on EKF to fuse information from visual and inertial sensors. Besides, OKVIS [87] and VINS [88] are following a structure similar to Figure 2.10. Such methods extract sparse or dense features from two consecutive camera frames and try to match them using different types of descriptors, and apply Structure From Motion (SFM) [172] in the first step on the visual information. Afterward, information extracted from visual information is formulated and tightly optimized with the information received from the IMU to estimate the robot odometry. These methods also require accurate calibration between camera and IMU. It is also important to mention that, calibration of the IMU also requires to remove the large bias.

On the other hand, deep learning approaches have also considered the same workflow via deep models. Clark et al. in [170] and [29] worked on extracting distinctive features from images to learn motion between two consecutive frames. Using deep Convolutional Neural Network, their method learns to extract distinctive features from images in the first steps. In the next step, they use LSTMs to learn relationships and correlation between extracted feature vectors in a time domain to estimate the 6-DoF camera position. While this monocular model performs well, it suffers from the wrong scale of movement estimation. Clark et al. in [30] moved toward using IMU measurements to solve the problem of wrong scale ex-

Figure 2.10: Workflow of Traditional Visual-Inertial Odometry Methods

Figure 2.11: VINet [30] Workflow for Robots Odometry

traction. The first step is the same as their earlier work, and they learn to extract distinctive features from continuous frames. However, for the second step, they

concatenate the feature representations obtained from visual information in the first step with IMU feature representations extracted from a two-layered LSTM with 1000 hidden neurons each (the input of IMU network is the concatenation of accelerometer and gyroscope measurements). Using the concatenation of feature vectors, their model learns to estimate the 6-DoF camera position. Existing approaches on deep learning based Inertial Odometry are limited, and all are concatenating measurements in the input space and extract the feature representation to model the desired output. Figure 2.11 illustrates the workflow of deep learning based Visual-Inertial Odometry systems, specifically VINET [30], for solving the problem of odometry.

2.7 Concluding Remarks

This chapter reviewed the existing approaches in the area of Visual-Inertial Odometry. The advantages of each of the visual and inertial approaches are explained, and drawbacks are pointed out. Based on each module's limitation, the next chapters are looking for optimal - robust and accurate - solutions to solve each of the visual and inertial odometry problems. Finding the optimal solution for each module helps fusion techniques deploy a robust system for autonomous vehicles.

Chapter 3

Robust Training of the Regression Networks by Considering Local Boundaries

3.1 Overview

Recently proposed Machine Learning methods to solve the regression problem of Camera Relocalization and Visual Odometry (VO) have achieved satisfactory results by training Convolutional Neural Networks (CNNs) as regressors end-to-end. However, the significant point is that CNNs are mainly proposed to solve classification problems, and they perform well when there exists sufficient data from each label. The existence of a high number of training data for each label encourages the network to learn better (more robust and distinctive) sparse or dense features. In the regression problems such as camera relocalization, the floating-point output space has a wide range, and the number of training data for each label of the output space is low. Hence, this chapter focuses on this issue and proposes a new two-step approach for training a regression network for 6-DoF camera relocalization and VO problems. The proposed approach converts the problem into a classification one and tries to learn a local boundary for each observation in the first step. The number of observations for local boundaries is high, and the network can learn rich and robust feature representations. After learning the local boundary related to different observations, transfer learning utilizes, and the learned knowledge generalizes to solve the global regression problem. Since understanding the number of local boundaries plays a critical role in the proposed approach, a novel approach that assists in detecting different segments in the environment in an unsupervised manner presents in the next section. This is a pre-requirement of the proposed multi-step training technique in the problem of 6-DoF camera relocalization. Then, the next sections, introduce and apply the proposed multi-step training approach on VO and 6-DoF camera relocalization problems, and the experiment and results come to discuss and demonstrate how this method works and performs.

3.2 Unsupervised Scene Categorization, Path Segmentation, and Landmark Extraction while Traveling Path

Discretizing the environment based on the visual context of the scene is an essential task of intelligent robots. This module is critical because it assists intelligent robots in understanding the scene and figuring out re-visited places. Furthermore, it can help to extract some landmarks in the environment and cluster the environment into several sub-spaces. Such sub-spaces represent a boundary that a similar visual context can be achieved by capturing an image at each point inside them. All points inside a clustered region are within a meaningful distance of their center called the landmark or anchor point.

Considering that the robot captures RGB image I_t at time t , the proposed system is required to detect transition points along the sequence and divide the environment into various portions, where each has a distinct view, in an unsupervised manner.

This research aims to perform the mentioned task in an unsupervised manner. Hence, the pre-trained GoogLeNet [162] network architecture (G) trained on Places 365 database [183] to solve the task of place recognition is utilized to extract robust feature representations for new observations. It is important to mention that for each new observation passed to the pre-trained network G , the feature representation at the place of the first fully connected layer is considered; the descriptor (feature representation) size is 1024. Hence, having the pre-trained network G and images I_t , the feature representation of images are extracted over time and stored in F_t (equation 3.1).

$$F_t = G(I_t) \tag{3.1}$$

Considering F_t as the feature representation of each image, the required task is to extract transition points based on changes in feature representations. In the first step, Ward's Linkage [46] clustering algorithm is applied. Linkage is a hierarchical clustering technique that combines data points and clusters hierarchically based on a merging cost. It tries to keep the merging cost in its optimum in each step of the hierarchical procedure. Ward's metric analyzes the variance of clusters and

measures how much the sum of the square will increase if two clusters A and B merged. The merging cost can be defined mathematically as

$$\bar{Z}(A, B) = \sum_{i \in A \cup B} \|x_i - m_{A \cup B}\|^2 - \sum_{i \in A} \|x_i - m_A\|^2 - \sum_{i \in B} \|x_i - m_B\|^2 \quad (3.2)$$

where m_j is the center in cluster j , and x_j is a data point in that cluster (each data point is a feature representation F_t). Since each point represents one cluster at the beginning, the merging cost $\bar{Z}$ will be zero, and it will increase over time. Linkage tries to combine data points and clusters so that merging cost have the minimum value in each hierarchical step.

After applying the Linkage clustering with Ward's merging cost, a hierarchical table that demonstrates the merged clusters or data points and their merging cost at each step can be achieved. The merging cost is available from the time it is 0 where each data point is considered as a single cluster, till its maximum value, which is the time all data points are placed in a single cluster.

The merging cost vector has rich information that can be used to identify transition points in the scene and distinguish viewpoints. We consider the hierarchical merging cost vector as Z , which moves from the minimum value of merging cost to the maximum value (it is important to mention that places that Z has the value of zero are removed from the vector).

In order to find the number of clusters or unique boundaries, it is possible to benefit from the acceleration of distance growth. In this manner, the second-order derivative of Z is computed and named $\ddot{Z}$, and its reverse is considered as $\ddot{Z}_{rev}$.

Moving on $\ddot{Z}_{rev}$ shows the acceleration of distance growth as the number of clusters increases; since it moves from the time all data points are in one cluster to the time each data point is a single cluster by itself. Hence, it will have the maximum value at its beginning. The variance of $\ddot{Z}_{rev}$ from the value at index idx to the end of the vector demonstrates how acceleration is changed relative to its mean value in that part; it is named as $\sigma^2(\ddot{Z}_{rev}^{idx})$. Practically, it demonstrates the variance between clusters as the number of clusters increases, and when it has a small value it means that clusters have separable data points and data points in

each cluster is similar; because merging data points and clusters did not lead to a big jump in the merging cost.

It is expected that the required number of clusters, C , to break down the environment occurs when

$$\min_{idx} \sigma^2(\ddot{Z}_{rev}^{idx}) \leq \lambda \quad (3.3)$$

satisfies, where λ is a control parameter to be tuned. The unique number of clusters in the scene can be known by finding the minimum index (idx) that satisfies Eq. 3.3. Since idx starts from 0, and it is based on the second derivatives of Z , the number of clusters in the environment (C) is equal to $idx+2$; the first value in Z_{rev} is for the time that two clusters are merged into one. The proposed condition analysis the variance of data points in clusters when $idx+2$ clusters exist, and we expect low variance when data points are demonstrating a similar viewpoint in each cluster. When we have a jump in second derivative which results in an increase in variance, it means that two clusters which are not similar are merged. Finding the minimum idx that satisfies Eq. 3.3 results in a reasonable number of clusters, because there is no jump in merging cost, clusters are distinct and data points are similar inside each cluster.

Afterwards, the Linkage clustering algorithm with Ward's merging cost applies to cluster feature representations to C number of clusters. Each of these clusters is representing a sub-domain in the environment, and based on position of the data points in each cluster, the center of the cluster can be defined. For each cluster c , the global feature representation can be represented as the average of feature representations in that cluster; which can be defined as

$$\bar{F}_c = \frac{1}{N_c} \sum_{i \in c} F_i \quad (3.4)$$

where $\bar{F}_c$ is the average of feature representations in cluster c (named as global feature representation of the cluster); N_c represents the number of data points in the cluster c , and F_i is feature representation of a data point in cluster c . The pseudocode of the proposed approach can be seen in the Algorithm 3.1. In the next sub-section, the proposed method is evaluated and analyzed in various environments.

Algorithm 3.1: Pseudocode of the proposed approach for extracting sub-spaces of the environment

```

Input : Pre-trained Network ( $G$ )
          Captured images over time ( $I_t$ )
Output: Extracted sub-spaces (clusters)
// Extract feature representations for the captured images
1  $F_t = G(I_t)$ 
// Apply Linkage clustering with Ward's merging cost metric
2  $Z = \text{Linkage}(F_t, \text{'ward'})$ 
// Compute second order derivatives of  $Z$ 
3  $\ddot{Z} = \text{second-order-derivatives}(Z)$ 
// Reverse  $\ddot{Z}$ 
4  $\ddot{Z}_{rev} = \text{reverse}(\ddot{Z})$ 
// Move over  $\ddot{Z}_{rev}$  and find the place that satisfies equation
   3.3
5  $idx = 0$ 
6 while  $\sigma^2(\ddot{Z}_{rev}^{idx}) > \lambda$  do
7   |  $idx = idx + 1$ 
8 end while
// Compute distinct number of clusters  $C$  in the environment,
   and extract clusters
9  $C = idx + 2$ 
10  $\mathbf{S} = \text{Extract-Clusters}(Z, C)$ 
11 return  $\mathbf{S}$ 

```

3.2.1 Experiments and Results

The performance of the proposed approach is analyzed in a scenario captured in Nanyang Technological University (NTU), Singapore, and a publicly available database that is designed to solve the problem of camera relocalization generally.

A handheld smartphone is utilized to capture the scenario at NTU. The image sequences are started from location of an office and moving out from its back-door. Then, the camera is moved towards the end of the block through the garden view and is moved into a corridor. Later, the camera is moved into the car-park and is returned to the office. The captured scene is a loop, and the path is shown in Figure 3.1 in a satellite view.

Figure 3.1: The green line shows the captured path in NTU. Boxes show the extracted clusters.

The I-frames of the captured scenario are extracted, and the feature representations (F_t) are computed. Computed $\ddot{Z}_{rev}$, and Z_{rev} , after applying Linkage on F_t , can be seen in the Figure 3.2. There are 63 I-frames, and as can be seen, the Z_{rev} has its maximum when all data points are in one cluster, and it moves to 0

Figure 3.2: Computed Z_{rev} and $\ddot{Z}_{rev}$ on the feature representations of the captured image sequence at NTU.

as the number of clusters is increased, and each image is considered as a cluster.

Applying Algorithm 3.1 with $\lambda = 1.5$ results in 6 clusters as follow: 1. inside office, 2. the garden view, 3. corridor, 4. turning from corridor to the car park, 5. car park, and 6. front of the office. The clusters can be seen in the Figure 3.1, on the satellite view.

By considering the average of feature representations of all images in one cluster as the cluster feature vector ($\bar{F}_c$), it is possible to compute the pairwise distance between clusters. The distance between clusters can be visualized and displayed in a confusion matrix as Figure 3.3. As it illustrates, the distance between the mean feature representation of each cluster with others is reasonable. It is important to mention that there is no wrong clustering of input images in the final output, and images within clusters are tagged correctly. Sample images of each cluster can be seen in the Figure 3.4.

In another experiment, the proposed technique is applied on the King's College scenario of the Cambridge outdoor dataset [75]. There are eight sequences for King's College, and after extracting feature representations from the 1220 training images of the scene, the proposed technique is applied. The experiment is applied to the whole data and a single sequence separately. The results show that the 5600 m² scene has been divided into 4 clusters based on the whole data, and 5 clusters based on a single sequence (on the sequence number 6). The result is shown in the XY coordinate on Figure 3.5.

Figure 3.3: Distance between global feature representation of clusters ($\bar{F}_c$) on the captured scenario in NTU

(a) Inside Office

(b) Garden View - Office Back-door

(c) Corridor

(d) Turn to Car Park

(e) Car Park

(f) Office Front-door

Figure 3.4: Extracted clusters for the sequence captured at NTU

(a) Result for sequence number 6

(b) Result by considering all sequences at once

Figure 3.5: Clustering King's College scene by the proposed approach. Points are places an image is captured, and points with similar colors are in one cluster.

As can be seen in Figure 3.5, most locations with a captured image are well clustered, but there exist some positions which are clustered wrongly. It is due to the fact that the scene contains some periodic structures which result in obtaining very similar images in different locations, which makes it difficult to distinguish their site. By the way, the vital goal for us in this research is to figure out anchor points and the number of clusters required to break the environment into subspaces, which is achieved well. It is also important to mention that the miss clustering rate is too low and satisfies this research requirement. It is also possible to benefit from the position of captured images in the XY coordinate (from the odometry module) to avoid such errors. Figure 3.6 shows two sample images that should be in different clusters but are considered in the same cluster due to the similar structure of the building the camera is viewing at different capturing locations.

Computing the average of feature representation vectors in each cluster ($\bar{F}_c$) when whole data is utilized, and computing pairwise distance among clusters results in Figure 3.7. As can be seen, the distance between Red and Green clusters is small, and it is much less than the minimum distance in our prior experiment on the data captured at NTU. Looking into Figure 3.5 (b), it can be seen that Red and Green clusters have points which are clustered (labeled) wrongly, and the small distance between features in Figure 3.7 can show the reason, which is viewing a repetitive and similar structure in different locations. Comparing Figures 3.7 and 3.3 shows

Figure 3.6: Two images from different clusters. The right image is labeled to be in the boundary (cluster) of the left image wrongly.

Figure 3.7: Distances between feature representation of each cluster on the whole data of King’s College Scene

that having a small distance between clusters results in having some data points with wrong cluster label. In contrast, larger distance can significantly reduce the probability of having wrong cluster label for data points. Hence, it is possible to look into distance among clusters and check the probability of having an issue in clustering data points, and see whether the environment has such repetitive and similar structures or not.

3.3 Proposed Multi-Step Training Approach Applied on 6-DoF Camera Relocalization

In this section, a two-step approach to train CNN for the problem of camera relocalization is introduced. This section uses PoseNet [75] as the main module and analyses the two-step training approach on its proposed network architecture. The PoseNet network architecture’s output is the pose vector for a given image that

contains three values showing a 3D camera position relative to a global reference frame and four values showing quaternion orientation. The CNN trains as regressor using Euclidean loss with the following objective function, equation 3.5, where β balances the error between position and orientation difference. This parameter is fine-tuned using a grid search approach to be optimized. In equation 3.5, x and q are the estimated position and orientation respectively and x' and q' are the ground truth position and orientation.

$$loss(I) = ||x - x'|| + \beta ||q' - \frac{q}{||q||}|| \quad (3.5)$$

The network architecture utilized by PoseNet is the modified version of GoogLeNet [162] with 23 trainable convolutional layers with two inception modules. PoseNet has replaced the softmax layer of GoogLeNet with an affine regressor to output position and orientation. PoseNet also contains another fully connected layer before the final regression layers of size 2048. It is important to mention that the output Quaternion is normalized. Besides, the training and testing data contains images with their corresponding capturing position and orientation information.

In the proposed system, despite training the network as a regressor from the beginning, the network learns to extract local boundaries in the first step. Hence, the number of segments ($nCluster$) in the environment is determined in the first step, by utilizing the approach proposed in the previous section. Then, training data is clustered to $nCluster$ number of clusters. Clustering is applied to position and orientation labels of training data to benefit from the rich label information and avoid miss-clustering.

To train a classifier network for the 6-DoF camera relocalization problem, training data is clustered according to the position and orientation values. By utilizing the training data point's label for identifying the clusters, the miss-clustered data points in the previous section's segmenting approach would be clustered correctly (since position gives rich information about the place). An important condition here is that number of data points in different clusters should be near each other. This condition helps the classification network have a balanced number of training data at each cluster and learn feature for each cluster fairly. This requirement is

achieved by proposing and minimizing

$$f = \sum_{i=1}^{nCluster} N_i \sigma_i^2 + \sum_{i=1}^{nCluster} \sum_{j=1}^{nCluster} (N_i - N_j)^2 \quad (3.6)$$

function where σ_i and N_i are the variance and number of data points of the cluster i ; $nCluster$ shows the number of required clusters. In equation 3.6, the first term minimizes the variance of data points in each cluster to keep similar data points in the same cluster. The second term makes different clusters to contain same number of data.

To minimize equation 3.6 and determine the best clusters, a set of vectors are defined to represent each clusters' center; here, we have $nCluster$ cluster centers and vectors. Since in the 6-DoF camera relocalization problem, each input image's label has three values for position and four values for orientation in the quaternion space, the vector representation for each cluster's center has seven values. So, $nCluster$ vectors with the size of seven show the clusters' center. Each data point in the training set is within the nearest cluster; distance computes by using Euclidean distance between the cluster's representation point and the data points label. By moving clusters representing points (clusters' center) in the floating-point output space (data points' label vector), and computing the cluster label for each of the observations in the training set, the fitness function (equation 3.6) can be calculated, and the clusters' center that minimizes equation 3.6 can be identified. Hence, equation 3.6 can be minimized by iteratively moving the clusters' center in the environment and finding their best position that minimizes the equation. To optimize in this manner, an evolutionary optimization method [77] is used. The optimization method initializes the clusters' representing points (clusters' center) for a fixed size population; each population has a set of clusters' representing points. Then, in each iteration, the fitness function (equation 3.6) computes for each population by finding out observations' cluster based on the generated clusters' center (each observation belongs to the nearest cluster center). The clusters' representing points are then moved by the utilized optimization technique [77] based on the whole population fitness value and their clusters' center to develop the next iteration's clusters' center. This procedure continues for a fixed number of generations

or reaching a steady state in which the fitness function is not changing for a period.

After determining the clusters in the training data, the PoseNet network architecture is trained as a classifier by replacing the network affine regressor layer to softmax to make it a classifier. This classifier is required to estimate the cluster number that each of the observations belongs to. The trained network will have rich knowledge about the local boundary of each observation.

After training such a classification network, the softmax layer is replaced once again with an affine regressor to output position and orientation. Then, the trained classification network fine-tunes to use the knowledge learned during the classification step to do the regression. In this manner, the network learns to classify data to a limited number of classes, during the classification step, where training data are similar and have a smaller range of position and orientation values in each segment; the learned knowledge assists the network during the fine-tuning procedure to regress data points inside each segment, and solve the regression problem in the next step. Hence, the network is regressing input to a small boundary output space based on the input's segment (using learned knowledge during the classification process).

Figure 3.8 illustrates the procedure of the proposed method in brief. The experiment and results are analyzed in the next section to show the performance of the clustering technique and the proposed training approach.

3.3.1 Experiments and Results

To evaluate the proposed method, four different scenes of Cambridge Landmarks dataset [74, 73, 75] are used. Pedestrians and moving vehicles also present in this dataset, and data is collected from different viewpoints with different weather and lighting conditions. To make it more challenging, train and test images are from separate paths and are not sampled from this dataset's training data trajectory. Table 3.1 shows information about the training and testing data. The implementation of this work is done under Caffe using Nvidia 1080 TI GPU.

First of all, the number of clusters (transition points) are extracted, and the proposed clustering technique is applied to the training data label to determine segments and cluster their labels. Figure 3.9-3.12 show position of data points in

Figure 3.8: Flowchart of the proposed training approach

Table 3.1: Details of *Cambridge Landmarks* Dataset

Scene	Number of Train Frames	Number of Test Frames	Spatial Extent (m)
King's College	1220	343	140 x 40 (m)
Old Hospital	895	182	50 x 40 (m)
Shop Facade	231	103	35 x 25 (m)
St Mary's Church	1487	530	80 x 60 (m)

3D world coordinate and data points with the same color are in the same cluster. It shows that the proposed clustering technique has clustered similar data points into one cluster. Furthermore, Figure 3.13 show that number of data points in different clusters is similar to each other, and the proposed clustering method can keep the number of data points in different clusters the same.

Figure 3.9: Visualizing King's College training data in 3D position coordinate

Figure 3.10: Visualizing Old Hospital training data in 3D position coordinate

Figure 3.11: Visualizing Shop Facade training data in 3D position coordinate

Figure 3.12: Visualizing St Mary's Church training data in 3D position coordinate

(a) King's College

(b) Old Hospital

(c) Shop Facade

(d) St Mary's Church

Figure 3.13: Histogram of number of data points in each cluster after applying proposed clustering method on labels of training data

In the next step, a classifier network is trained, based on the training images and their correspondent cluster label, to learn extraction of each observation's cluster label (segment) from images. The network is trained by fine-tuning from a pre-trained GoogLeNet model trained on Places 365 dataset [184]; using a pre-trained model on the places dataset helps to use rich features learned on that huge dataset for place recognition and obtain a robust network fast.

Training the classification network helps to classify data captured from the scene into clusters according to their positions. Since observations in each segment of the environment will be similar, fine-tuning the classification network as a regressor in the next step leads to using the network's knowledge regarding the image's segment to determine the position and orientation of the observation inside segments. Hence, the network knows in which partition an image is captured, and when it wants to learn regression, it needs to map it to the floating-point values existing in that boundary, which means that it maps input to a smaller output space, which is inside its cluster boundary.

A comparison of the proposed training approach with the baseline is reported in Table 3.2. As shown, applying a two-step training approach (the proposed technique) is better than training the network as a regressor from the beginning. Hence, training the network to learn the classification of data points into local boundaries in the first step, and fine-tuning the network in the next step to regress inputs inside their local boundaries improves the performance. This is mainly because of learning to do the regression of data points based on their local boundary, which is smaller than the global output space. This is similar to human behaviors. Humans try to memorize some local landmarks in the environment and localize their position relative to that local landmark, not the global point.

3.4 From Local Understanding to Global Regression in Visual Odometry

Recent Deep Learning based Visual Odometry (VO) methods train an end-to-end network to solve VO as a regression problem directly without visualizing and sensing the label of training data in the training procedure. In this section, the proposed multi-step training approach is utilized for training Convolutional Neural

Table 3.2: Comparing the median re-localization error when the network is trained with and without the proposed training approach

Scene	Training PoseNet From Beginning as Regressor	Training PoseNet with Proposed Method
King's College	1.92 m 5.40 degrees	1.55 m 2.18 degrees
Old Hospital	2.31 m 5.38 degrees	2.25 m 5.13 degrees
Shop Facade	1.46 m 8.08 degrees	1.45 m 7.21 degrees
St Mary's Church	2.65 m 8.48 degrees	2.33 m 7.44 degrees

Networks (CNN) for the regression problem of VO. The proposed method first changes the problem to a classification problem to learn different subspaces with similar output space observations. After solving the classification problem, the original regression problem solves using the knowledge achieved by solving the classification problem. This approach helps CNN solve the regression problem globally in a local domain learned in the classification step and improves the VO regression module's performance for approximately ten percent. It is important to mention that the VO module's output is the position and orientation changes between frames. Hence, despite the previous section, the direction of movement will construct the local boundaries in VO.

This section introduces a novel approach to train a robust CNN model for the regression problem of VO. The input of the VO CNN module is the Optical Flow of two consecutive input frames, and the outputs are three values showing displacement of camera translation and three other values showing displacement of camera rotation in two consecutive frames. In the next sub-section, the structure of Optical Flow based CNN is proposed, and afterward, the training procedure is introduced.

3.4.1 CNN Structure

Since Optical Flow images demonstrate motion in two consecutive frames, it is feasible to use it to solve VO; it allows the VO module to have a shallow network with a low number of layers because the optical flow extracts rich motion information.

Borx algorithm [17] minimizes equations 3.7-3.9 to achieve Optical Flow between frames. In these equations, $I : \Omega \subset \mathbb{R}^3 \rightarrow \mathbb{R}$ is the rectangular image sequence, and $w := (u, v, 1)^T$ is the searched displacement vector between images where u and v are location displacement indicators in image space and demonstrate the flow of each pixel (1 is the time difference between frames and refers to the next time step); ∇ is the spatial gradient, and ∇_3 is the spatio-temporal gradient (in case of having two frames, ∇_3 is also the spatial gradient). V_{data} assists in intensity value consistency and gradient consistency. It states that the pixel intensity and gradient should be matched by moving the pixel with w (among two frames); γ is a weight that demonstrates how much the flow should lie on the gradient consistency term. Besides, Φ is an increasing concave function that helps to control outliers. V_{data} only considers the displacement of a pixel locally without considering its neighbors. To consider neighbors, V_{smooth} utilizes to ensure smoothness of gradient and make sure no problem exists by the presence of a vanishing gradient in some part of the image. Finally, the two terms are combined and minimized in the equation 3.7 with the weight factor of α .

$$V(u, v) = V_{data} + \alpha V_{smooth} \quad (3.7)$$

$$V_{data}(u, v) = \int_{\Omega} \Phi(|I(x+w) - I(x)|^2 + \gamma |\nabla I(x+w) - \nabla I(x)|^2) dx \quad (3.8)$$

$$V_{smooth}(u, v) = \int_{\Omega} \Phi(|\nabla_3 u|^2 + |\nabla_3 v|^2) dx \quad (3.9)$$

The network architecture used to train Optical Flow-based VO is a part of Costante et al. [31] network architecture. In this network, called P-CNN and shown in Figure 3.15, the input is eight times down-sampled of the entire optical flow image and downsampled image of local parts after dividing the optical flow into four equal sub-images, shown in Figure 3.14. In Figure 3.14, blue lines are the input of the Convolutional Neural Network module which is shown on Figure 3.15. In this manner, by using downsampled sub-images and the original image, both local and global information of optical flow image extracts and uses to solve Visual Odometry. The benefit of four local parts is removing the effect of a moving object presented in local parts and making the network robust to moving objects.

Figure 3.14: Different Inputs of the P-CNN Network Architecture

The network structure consists of two convolutional layers that learn to extract features from the optical flow images. After two convolution layers, convolution layers' features concatenate together and forward to the fully-connected layer for regression. Results of P-CNN [31] show that two convolution layers learn different features, and concatenating those features improves VO's overall accuracy. This improvement is because simple features learn with the first convolution layer, while the second convolution layer learns more complex feature representations by using the first layer's outcome. Hence, the concatenation of these two features leads to a rich representation. The final output of this network is going to be six parameters showing the displacement of camera translation and rotation change.

3.4.2 Proposed Training Procedure in Visual Odometry

Having bounded output space and lots of input for each of the output labels helps the network to learn major features and become robust to noise. Regression prob-

Figure 3.15: P-CNN Network Architecture

lems, such as VO, require CNN to map input space to the large floating-point output space (for VO, the label is the relative camera translation and rotation change in two consecutive frames). Previous works on learning-based VO systems train a CNN as a regressor by giving raw or pre-processed images as input and getting the displacement of camera translation and rotation as output; Their regression models cannot efficiently learn features and the best weights due to the diverse output space.

This section utilized the proposed two-step training approach to train and achieve an efficient regression model. As described, in this method, CNN trains as a classifier to learn and extract similar output subspaces. Supervised learning and clustering output labels of the training data define similar subspaces in VO. Having near to the same amount of data points for different labels is an essential policy to train CNN as a classifier. This rule helps the classification network to have the same number of observations for different labels in training time and learn to extract significant features intelligently. Applying clustering techniques such as Linkage cannot satisfy this clause. This clause satisfies by minimizing the equation 3.6 proposed in the previous section.

Minimization of equation 3.6 results in a set of vectors representing the center of different clusters. In VO's problem, each cluster's center is a six-dimensional vector, according to the labels, which are six parameters showing displacement of camera translation and rotation. To determine each observation's cluster, the Euclidean distance of observations to the center of each cluster measures, and each observation belongs to the nearest cluster (cluster with the smallest distance). To minimize equation 3.6, an evolutionary optimization method [77] is used. In this manner, in the first step, a random population is generated. Each population has $nCluster$ vectors showing the center of each cluster. Then, the cluster each observation belongs to calculates in each population, and then the output score of the equation 3.6 measures. Afterward, the population moves in the label space, using the evolutionary method, to find the global minimum of the equation 3.6.

Despite the problem of 6-DoF camera relocalization, it is not possible to determine the number of clusters ($nCluster$) automatically in some problems such as VO. It is because of the complexity of detecting the number of subspaces from the

input (visual information in VO) based on the problem's nature. Here, the number of clusters ($nCluster$) can be defined by the problem complexity and visualizing training labels. In the problem of VO, rotation plays a critical rule. For instance, in most cases, when a vehicle turns, its speed is slower than the time it moves straight. Hence, by visualizing data points, the number of clusters ($nCluster$) is determined as four. After clustering data points to a specified number of clusters with the policy of having a flat histogram of data points over different clusters, CNN trains as a classifier. The trained classifier is then fine-tuned as a regression system by replacing the softmax layer with the Euclidean loss to regress feature space to specific floating-point output labels. The advantage of this approach is that the model first learns the domain that input data is related to, and after that, by training the regression model, each of the inputs will regress to the local output domain. It is like looking at the input data locally in a smaller output domain. Figure 3.16 illustrates the procedure of the proposed training approach.

3.4.3 Experiments and Results

In this section, the proposed method evaluates on the KITTI benchmark [54], which uses to evaluate different types of Computer Vision problems, especially problems related to autonomous vehicles. Provided images in this dataset are captured by Pointgrey Flea2 stereo cameras. Videos captured as ten frames per second undistorted images with a resolution of 1240×386 . Ten sequences of this dataset are provided with the ground truth, which uses in our experiments. Same as [31], this section considers the first seven sequences as a training set and the rest three for testing.

Figure 3.17 and 3.18 demonstrate the distribution of training data in the 3D world coordinate for the displacement of camera position and rotation, respectively. It illustrates that labels are more diverge in the z direction of Euler parameters showing the Displacement of Camera Rotation. Applying the proposed clustering technique to cluster data points into four clusters also proves it and shows that the z direction of Euler parameters is the most effective parameter in defining subspaces.

Figure 3.19 shows the histogram of the number of data points in each cluster

Figure 3.16: The Procedure of the Proposed Training Approach applied to VO

Figure 3.17: Clustered Displacement of Camera Rotation Labels in 4 Classes

Figure 3.18: Clustered Displacement of Camera Position Labels in 4 Classes

after applying the proposed clustering technique. The results show that the four clusters have a low difference in the number of observations. Even applying other clustering techniques cannot reach a result with a lower difference in the number of observations in different clusters. Figure 3.17 and 3.18 also show the final result of clustering data points, and points with the same color are in the same class. It is evident that data points are clustered according to the z value of Euler angles, which contains an extensive range of values and is the most diverge parameter. z value demonstrates the car's moving direction (Left, Right, Middle Left, or Middle Right).

Figure 3.19: Number of Observations per Class After Applying the Proposed Clustering Technique

According to the method mentioned in the previous section, CNN trains as a classifier after clustering data points and computing Optical Flow. The classifier network learns to output the subspace that the final regression result belongs to.

After training the network, the network fine-tunes as a regressor. During fine-tuning, the network uses prior-knowledge regarding classifying data in the most diverge direction. So the network learns to regress data in a local boundary, and the problem becomes more simple for CNN to solve. It is also important to note that the Euclidian loss function is used during training the network for the regression purpose.

Figure 3.20: Average Rotation Error Across Test Sequence Speeds

Figure 3.21: Average Rotation Error Across Test Sequence Path Length

Figure 3.20 to 3.23 show the average of rotation and translation error across test sequences of KITTI dataset regarding path and speed. These figures show that using the proposed method to train P-CNN outputs a network with less error on the test data compared to training P-CNN as a regressor from the beginning. It also shows that the average error is reduced by ten percent on average for both translation and rotation. Hence, clustering label of training data and training a classifier and afterward fine-tuning the learned model as the regressor in the problem of Visual Odometry leads to a better result on the test dataset. This improvement

Figure 3.22: Average Translation Error Across Test Sequence Speeds

Figure 3.23: Average Translation Error Across Test Sequence Path Length

is because of fine-tuning a network that knows the local boundary for each data. Hence, the regressor network solves the regression problem in a local domain for different types of input. Figures 3.24 to 3.26 also compare trajectories computed from the trained P-CNN network using the proposed method and training it as regressor from the beginning on the test data of the KITTI dataset.

Regarding training time, an additional training phase to train the network as a classifier requires in the proposed method. The classification problem does not need to solve completely before training the network as a regressor, and learning to extract basic knowledge by the network is enough. Hence, the proposed method does not significantly influence training time and using it to solve VO's problem increased training time for about 5 percent approximately.

Figure 3.24: Comparing trajectories computed from trained P-CNN network using proposed method and training it as regressor from beginning for sequence 08 of KITTI dataset

Figure 3.25: Comparing trajectories computed from trained P-CNN network using proposed method and training it as regressor from beginning for sequence 09 of KITTI dataset

Figure 3.26: Comparing trajectories computed from trained P-CNN network using proposed method and training it as regressor from beginning for sequence 10 of KITTI dataset

Further investigation is applied to analyze the effect of the number of clusters. Figure 3.27 illustrates the effect of the number of clusters on the accuracy of the final model. Figure 3.27 shows that the model reaches its maximum accuracy using the optimum number of clusters, where the distribution of observation over different clusters is flat, and each cluster demonstrates a meaning. The optimum number of clusters is equal to four in the VO problem, and each cluster represents the moving direction of the camera movement. Clustering data points to more clusters than the optimum value results in the drop of accuracy. Moving towards making a cluster for each data point results in an accuracy less than the time no clustering technique is used in the training phase. Hence, the optimum number of clusters is when different clusters have a similar number of observations, and each cluster represents a meaning.

Figure 3.27: Effect of Number of Clusters on The Final Model Accuracy

3.5 Concluding Remarks

In this chapter, a new approach to train CNNs for regression problems is introduced. In this method, the regression problem first formulates as a classification problem according to the label of training data, and after training the classifier network, the regression problem dissolves by fine-tuning the classification network as a regressor. In this manner, the regression problem solves in the local parts learned by the classification network. Results show that re-formulating the regression problem as a classification problem helps regression problems to learn a classifier in the direction of the most diverge parameter in the first step. Fine-tuning such a classification network as a regressor helps the regressor map input to the output according to each input data's local boundary. Hence, despite solving the regression problem globally, this approach uses the classifier's knowledge to solve the problem locally for each observation. Results show that this approach helps the learning procedure of the regression problems and improves their accuracy.

Furthermore, an unsupervised approach for breaking the environment into a reasonable number of sub-spaces is proposed in this chapter. Each sub-space has a unique and distinguished view of the environment. The proposed approach has

been examined in a scenario captured at NTU and a publicly available dataset. The results show that the proposed approach has achieved its goal and extracted clusters of the scene in a reasonable manner. However, some data points are clustered wrongly due to the presence of repetitive structures or similar viewpoints. It is possible to fix such an issue by benefiting from odometry modules and location information in the future if high accuracy is required (similar to the clustering technique introduced to satisfy the requirement of flat observations over clusters). The proposed module can be used to break down large-scale environments and assist place recognition and camera relocalization problems to be solved robustly. This approach is used to determine the number of clusters in the proposed training approach (for camera relocalization).

Chapter 4

Dealing with Moving Objects

4.1 Overview

Deep Learning methods can deploy a fast, robust, and lightweight model to solve the problem of 6-DoF camera relocalization in large-scale outdoor environments. However, two significant characteristics of captured images in a large-scale outdoor environment are moving objects, which should not be included in the representation of an environment, and motion blur, which widely exists in the images captured with moving cameras. None of the existing approaches study and investigate these two problems in their method. For the first time, this chapter proposes a deep network architecture that is trained based on the knowledge achieved by combining deblurring and semantic segmentation modules and examines the effect of this combination on a challenging dataset. Results show approximately 20 and 50 percent improvement in camera position and orientation relocalization error, respectively. Furthermore, a similar approach with different criteria is applied to the problem of visual odometry.

4.2 Preliminaries

Since this chapter utilizes image deblurring and semantic segmentation modules, a brief review of the aforementioned modules are provided in this section.

4.2.1 Image Deblurring

Shaking the camera, the motion of moving objects, and camera focus can result in a blurred captured image that is not sharp enough to extract robust features. To obtain a sharp image from the blurred one, researchers studied and proposed several methods.

Traditional methods [13, 59] are applying different constraints to heuristically model the image blurring problem and use prior knowledge to solve it on new queries. The drawback of such methods is their large number of tuning parameters

and their hypotheses which prevents them from having an outstanding performance on the real experiments.

On the other side, Learning-based approaches, especially end-to-end CNN ones, have achieved state of the art results recently. Nah et al. [116] proposed a multi-scale CNN which solves the problem of de-blurring for low-resolution images and progressively increases its resolution. While it outperforms traditional methods, it has a high number of weights which makes it slow. Tao et al. [163] used recurrent neural networks to have one network for all the scales and reduce the number of trainable parameters which prevents the network from overfitting and makes it faster. In this research, Tao et al. method is used due to its high performance on unseen environments, speed and lightweight model for robotic applications.

4.2.2 Semantic Segmentation

Semantic Segmentation and Object Detection are significant problems which are in focus of researchers in the last few years. Powerful Deep CNN systems are proposed in the last few years such as Fast/Faster-RCNN [56, 130] and Fully Convolutional Neural Network (FCN) [98] methods. The approach they are following is extracting a Region of Interest (RoI) in the first step from the image, and after finding the class of the object, they move towards finding correspondence pixels to each semantic in the problem of instance segmentation.

RoI methods [65] extended to RoIPool which applies to the feature representations and improves the speed. Other works [130] also proposed Region Proposal Network (RPN) to figure out regions with high probability of having a semantic or object.

Mask R-CNN proposed by He et al. [64] is a fast semantic segmentation approach which solves the problem of instance segmentation at the pixel level. The Mask R-CNN has the same steps as Faster R-CNN to extract RoI and class of the data. On top of that steps, it learns to extract a binary mask for each region which requires valuable features in each RoI. Hence, they add layers on top of Faster R-CNN and minimize a loss function based on the class of the object, the bounding box and the binary map to find the optimum result. Since their method can perform fast and achieve state of the art result this chapter uses it as a module

to extract semantics in the experiments.

4.3 DeepDSAIR: Deep 6-DoF Camera Relocalization using Deblurred Semantic-Aware Image Representation for Large-Scale Outdoor Environments

This section proposes a Deblurred Semantic-Aware Image Representation (DSAIR) module which is robust to motion blur and enhances modeling static scene. This module can attach to several applications where the static environment’s information is necessary, such as place recognition, visual odometry, and camera relocalization. The first step of the proposed module applies the deblurring technique proposed by Tao et al. [163] to make a deblurred training data from the input. It is important to note that the deblurring model has not been trained on the same camera used in the experiments, and this report uses the trained model achieved by Tao et al. [163], on the public dataset with a different type of camera. In this manner, the generalization ability of Tao et al. method examines in our results where the camera model of the input data is different from the knowledge of the trained model. After getting the network’s output result, it moves towards the semantic segmentation module and Semantic-Aware Image Representation layer. Then, in the proposed deep network architecture for the problem of 6-DoF camera relocalization, a various number of augmentation layers applies to the input image and the network trains to estimate camera position and orientation parameters separately. The network trains on raw, deblurred and semantic-aware images concurrently to become robust to noise and motion blur, and improve its generalization ability. The results show that using a deblurred module in the test time is enough to achieve the best result from the trained network. The workflow of the proposed method, shown in Figure 4.1 and 4.2, discuss in detail in the next subsections.

4.3.1 Deblurred Semantic-Aware Image Representation

The Deblurring Semantic-Aware Image Representation (DSAIR) layer creates a deblurred image in which the semantically moving objects are blurred to move the network’s focus towards the scene’s static parts and make the model robust to

Figure 4.1: Proposed Deep Semantic-Aware Image Representation (DS AIR) Module that applies before Deep CNN

Figure 4.2: Deep CNN Module

dynamic objects in training time.

The raw image firstly feeds into Tao et al. [163] deblurring Scale Recurrent Network to obtain a clear and sharp image. Then, using Mask-RCNN [64], the bounding box of the semantic objects in the scene extracts. From the various number of semantics, the ones related to moving targets, such as pedestrians, cars, bicycles, etc., can cause a problem in describing the static environment. Filtering out the moving objects from all the semantic labels helps to understand the static view that preserves the scene's main structure.

Semantic information is insufficient for the network to learn to extract meaningful knowledge related to the static scene. The primary requirement is an informative image representation that highlights the static scene locations where the network should concentrate during training and testing procedures. One solution is deleting the parts describing a moving target from the image. This approach is inefficient because of completely changing the appearance information, and if a new query with information in the deleted parts appears network fails and cannot generalize.

This section proposes a new representation for pixels inside a bounding box describing a dynamic object. The value of each pixel P with the position of (x_c, y_c) in the image coordinate inside the bounding box (bb) showing a moving object defines as

$$P_{bb}(x_c, y_c) = \sum_{x=x_c-L}^{x_c+L} \sum_{y=y_c-L}^{y_c+L} \frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_c)^2+(y-y_c)^2}{2\sigma^2}} \quad (4.1)$$

where the value of each pixel inside the bounding box assigns as the weighted summation of its neighbors in a window with the size of $(L+1) \times (L+1)$. Equation 4.1 blurs parts of an image containing a dynamic object. Choosing a higher L value increases the boundary of the pixels that affect the center pixel. However, since neighboring pixels are showing similar measurements a smaller value will keep it sharper. σ can be a fixed value or weighted as

$$\sigma = K \cdot \frac{\sqrt{(bb_x - x_c)^2 + (bb_y - y_c)^2}}{\frac{|bb|}{2}} \quad (4.2)$$

which shows the degree of blurriness for each pixel. bb_x and bb_y are the x and y

position of the nearest point on the bounding box to the point (x_c, y_c) . $\frac{|bb|}{2}$ shows half of the bounding box's size, and K is a tuning parameter. Since the boundary points are hard to detect by the semantic segmentation modules, using equation 4.2 preserves the pixels near the boundary from getting completely blurred. It gives the network a chance to extract and use some features from the boundaries to model the scene. Figure 4.3 demonstrates a scene with moving objects where the dynamic objects get blurred using the proposed DSAIR module. As it shows, some parts of the car on the right are still sharp due to the semantic segmentation module's error.

(a) Raw Image

(b) After applying DSAIR

Figure 4.3: Sample image of the scene before and after applying DSAIR module

Using the proposed image representation prevents the Deep CNN to extract any feature from the pixels related to dynamic objects. Moreover, the new representation makes the network aware of some knowledge in parts with moving objects. Hence, the final trained model represents the scene based on the static points and has a high generalization ability. Figure 4.1 shows the procedure of the proposed DSAIR layer. Switch *sw1* forces the module to use the deblurred image or not, and if it is true, then switch *sw2* forces the module to use semantic-aware image representation or not.

4.3.2 Network Architecture

Existing approaches are modeling and training the Deep CNN architecture end-to-end to extract both parameters of camera position and orientation simultaneously. They minimize a joint loss function based on the weighted Mean Square Error (MSE) of the estimated camera position and orientation parameters, with the ground truth. Previous works' wrong assumption is considering the same orientation parameters in random crops and using random crops data augmentation procedure. Hence, they divide the image into random 224×224 parts to improve the generalization ability of the network. 224×224 sub-images can be taken from the same position, but with a different orientation. Orientation depends on the center of each image, and data augmentation with random crops of the original image is incorrect and results in failure of camera orientation estimation. Only the 224×224 sub-image extracted from the center of the original image has the correct orientation parameters. Figure 4.4 is demonstrating the problem. The cropped sub-images of the original image are taken from the same position but with a different orientation.

The Deep CNN network architecture this section used to test the DSAIR module is the same as the latest state of the art method [168] with some modifications in the input layer. This network architecture consists of three main modules: GoogLeNet network architecture [162], a Long Short-Term Memory (LSTM), and a Fully Connected layer. The GoogLeNet module helps to extract meaningful feature representations from the input image. To have better knowledge about the structure of the scene and reduce the feature vector's dimension, the pro-

Figure 4.4: Effect of random cropping augmentation layer used by previous methods on the camera orientation parameters

duced feature vector of the GoogLeNet moves toward the LSTM layer to realize the correlation of the elements of the feature vector and reduce its dimension. In this manner, 2048 components of the feature vector representation extracted from GoogLeNet, are reshaped into 32×64 matrix and the LSTM applies on this matrix from up to down, left to right and their reverse direction. Hence, the correlation between feature vectors learns, and the dimension of the 2048D vector reduces. Later, the output of each of the four LSTM unit concatenates, and move toward a fully connected layer to regress into the final output. The detail of this structure is available in [168].

Despite [168], which beliefs in a single network to retrieve camera position and orientation parameters, this chapter considers two separate networks with suitable augmentation layers to relocalize position and orientation parameters individually. This approach helps to resolve the mentioned problem regarding the orientation parameters, which gets faulty with random crops. However, this solution will require more memory (twice previous methods) to record the network's weights.

In order to improve the network's generalization ability, for the module that recovers camera position, random 224×224 crops extract from each image during

the training procedure. After taking the output of the random crops augmentation layer, random rotation between -5 to 5 degree also applies to the image. These two data augmentation module will help the network to have more data, which improves its generalization ability, and also makes it robust to rotation of the camera. On the other hand, for the orientation network, the whole image considers without any crop or augmentation layer. Applying a geometric augmentation on the image will change the orientation parameters and prevent the network from learning meaningful feature representations. Figure 4.2 shows the workflow of the two Deep CNN modules. The input comes from the DSAIR module, resizes into a 455×256 image, and after applying different augmentations for different tasks, the final output estimates using the GoogLeNet, LSTM, and Fully Connected layers.

4.3.3 Training Procedure

To train the proposed network architecture, the Deblurring network [163], and Mask-RCNN [64] module are fixed, and their weights are loaded from the pre-trained models. Switches regarding using the Deblurring module and the output of Semantic-Aware module are randomly set to on or off in the training phase (*sw1* and *sw2*). In this manner, the network is trained on a dataset three times larger than the existing dataset (using the deblurred image, semantic-aware image, or their combination for training). Training the network on these images makes the network intelligent to automatically extract features from buildings and static objects, and not focusing on dynamic objects such as pedestrians and cars. Consequently, the network becomes robust to dynamic objects and motion blur.

Since Mask-RCNN [64] network is not one hundred percent accurate, some parts in the images detect incorrectly as dynamic objects, and DSAIR makes them blur. This error results in losing the essential pieces of information. Using the raw image and the DSAIR module's output at the same time during training makes our network robust to such error and grants it more generalization ability. This problem discusses and analyses in detail in the next section.

To train Deep CNN, the two-step training approach proposed in the previous chapter uses. In this manner, the problem first converts into a classification problem by clustering the training data label to N number of clusters according to

the type of the scene. Each cluster presents a local region of an image. Deep CNN trains for a few iterations as a classifier to classify input images to their local class or region. Then, the network finetunes as a regressor to solve the regression problem directly from the input. In this manner, the network firstly learns to map input images to local boundaries; for instance, it can figure out an image is captured from the front view of the building or the backside. Afterward, during the finetuning step, it solves the regression problem in that local boundary. Solving the regression problem in a local boundary is much easier for the network and will increase its accuracy. Figure 4.5 illustrates the procedure of the two-step training approach, which is presented with details in the previous chapter. As it shows, firstly, the training data gather and the labels of the training data clusters to N number of clusters. Afterward, the Deep CNN trains as a classifier to distinguish different parts of the scene. After learning such information, in step 4, the network fine-tunes to solve the regression problem based on the original ground truth (position and orientation). The MSE between the estimated measurements and the ground truth is used in each network’s training step as the loss function.

Figure 4.5: Multi-Step Training Procedure

4.3.4 Experiments and Results

This section evaluates the proposed method to investigate its performance and monitor its generalization ability to resolve the problem of 6-DoF camera relocalization for large-scale outdoor environments. The proposed method implemented using the Pytorch [123], and the experiments performed on an NVIDIA 1080TI GPU. The proposed approach evaluates on the Cambridge outdoor dataset [75]. This dataset is the only available dataset to examine camera relocalization modules in outdoor environments and consists of four different scenes with ground truth labels for each frame showing camera position and orientation parameters. Each of these scenes contains training and testing sequences to train and test the

implemented relocation module. Information regarding the dataset is available in Table 4.1. The scenes are large-scale, contain moving objects, and the camera is moving during capturing data, which results in motion blur on the training and testing images. Moreover, the images were taken at a different time of the days, which makes the dataset complex enough.

Table 4.1: Dataset Details

Scene	Number of Train Frames	Number of Test Frames	Spatial Extent (m)	Area or Volume (m^2)
King's College	1220	343	140 x 40 (m)	5600 (m^2)
Old Hospital	895	182	50 x 40 (m)	2000 (m^2)
Shop Facade	231	103	35 x 25 (m)	875 (m^2)
St Mary's Church	1487	530	80 x 60 (m)	4800 (m^2)

The De-blurring and Semantic Segmentation modules are the pre-trained models trained on a dataset with a camera different from the dataset used in this research. To train the Deep CNN module, the weights of the GoogLeNet initializes using the weights from the pre-trained model on the Places [184] dataset to solve the problem of object recognition. In this manner, the network has the power to extract valuable features and tunes itself to use the critical and invariant features to resolve the problem of 6-DoF camera relocation.

The batch size of 75 with the auxiliary loss weight of 0.3 and a dropout probability of 0.5 used to train the network. The size of the blurring window or L parameter in equation 4.1 is set to 2 which leads a window of size 5, and Adam optimization uses to find optimum weights with hyper-parameters $\epsilon = 1$, $\beta_1 = 0.1$ and $\beta_2 = 0.999$. According to the previous chapter's proposed training procedure, the labels of dataset clusters and the networks train as a classifier. Afterward, the network finetune as a regressor and learn to extract and map meaningful features to the desired output. During training the model $sw1$ and $sw2$ switches randomly set to *true* or *false*.

The result of the proposed method compares with other existing approaches in Table 4.2. The proposed method applies on top of [168] network architecture. To have a fair comparison, reported result in Table 4.2 for [168] is the result of our implementation in Pytorch. To check the robustness of the proposed method,

the effect of Deblurring and Semantic-Aware modules checks in the test time by feeding different inputs to the trained model (the result is reported in Tabel 4.2). It is important to notice that all the reported results for the proposed method obtain on a single model trained on the random values for $sw1$ and $sw2$, in Figure 4.1, in the training step. As results show, the proposed method's results compared to the existing approaches are improved for about 50 percent and 20 percent on average for camera orientation and position recovery, respectively.

Table 4.2: Comparing the median error of the proposed method with the existing approaches to solve the problem of 6-DoF camera relocalization

	King's College	Old Hospital	Shop Facade	St Mary's Church	Average
PoseNet [75]	1.92 m, 5.40°	2.31 m, 5.38°	1.46 m, 8.08°	2.65 m, 8.48°	2.08 m, 6.83°
Bayesian PoseNet [73]	1.74 m, 4.06°	2.57 m, 5.14°	1.25 m, 7.54°	2.11 m, 8.38°	1.92 m, 6.28°
PoseNet LSTMs [168] (Our Implementation)	1.21 m, 5.34°	1.96 m, 5.63°	1.41 m, 8.09°	1.85 m, 9.51°	1.60 m, 7.14°
Proposed Method (using the raw image as input in test time)	0.85 m, 1.12°	1.65 m, 2.49°	1.05 m, 4.35°	1.82 m, 6.26°	1.34 m, 3.55°
Proposed Method (using the deblurred image as input in test time)	0.86 m, 1.11°	1.64 m, 2.47°	1.01 m, 4.14°	1.80 m, 6.25°	1.32 m, 3.49°
Proposed Method (using the deblurred semantic-aware image as input in test time)	0.87 m, 1.12°	1.64 m, 2.42°	1.07 m, 4.39°	1.84 m, 6.28°	1.35 m, 3.55°

Results illustrate the great improvement of the camera orientation recovery. This improvement is due to removing the random cropping augmentation layer and training deep networks based on the DSAIR module. The results show that using only the Deblurred module in test time outputs the optimum result followed by the raw image and Semantic-Aware Image Representation. Figure 4.6(b) is showing the outcome of applying the Deblurring module on the raw input image (Figure 4.6(a)). While the Deblurring module did not train for this type of camera, the figure shows that the module can reduce the effect of motion blur significantly. After applying the deblurring module, the output image is sharp and clear, and meaningful deep feature representations can extract and match by the network. The results show that using the deblurred image at test time improves the accuracy for 0.02 meters and 0.06 degrees compare to raw images as input. It also proves that using both raw and de-blurred image at the same time during training makes the network robust to motion blur when the raw image feeds into the network in the test time.

Using the Semantic-Aware Image as the input ($sw1 = true$ and $sw2 = true$) at

(a) The RAW input image

(b) The image after applying deblurring module

(c) The output image of DSAIR module

Figure 4.6: Illustrating the output of different stages of the DSAIR module. The motion blur of the raw input image (a) is handled efficiently by the Deblurring module and in (b) the edges and corners are sharp enough to extract meaningful feature representations. However, the semantic segmentation module wrongly detected the front view of the building as a dynamic object and DSAIR module makes it blur (c). The blurred part of the building has the most information to recover the position and orientation of the captured image.

the test time gives the worst result compared to others (raw and deblurred image) regarding position recovery. Figure 4.6 (a) and (c) show an image before and after applying the Semantic-Aware module. As it shows, the Semantic-Aware image is faulty, and the informative parts of the image are blurred. Blurring informative parts of the image results in the system's failure in test time, which is why the result of the network when semantic-aware image passes to network in test time is not good compared to others. However, this semantic segmentation error also exists in the training data, and as results show, the error between using the output of the deblurred module and semantic-aware image in test time is 0.03 meters and 0.06 degree, which is still acceptable and shows the trained network is robust enough to dynamic objects. As stated, this error is due to the wrong estimation of semantics in the image during the test time, and it is better to turn it off in the test time.

It is also important to clarify that, *KingsColledge* and *ShopFacade* dataset are full of dynamic objects, while the other two scenarios contain fewer dynamic objects. Results show that in these two scenarios, the proposed method has a significant impact. This influence is because of the existence of the high number of dynamic objects in the scene in these two scenarios, in which the network is robust to them because of using DSAIR module in the training phase. Hence, the network trained based on three different types of input performs well in a challenging environment with the existence of numerous dynamic objects.

Since the result of using deblurred images in test time is optimum, it is possible to ignore the semantic-aware module in the test time ($sw1 = true$ and $sw2 = false$ is enough in test time while it switches randomly in training time). Hence, the proposed method's speed is almost the same as the previous methods with an additional execution time of the de-blurring module, which performs in real-time. The only drawback of the proposed method is the storage required to save the networks' weights, which is twice the previous methods because of using two separate networks to recover position and orientation parameters. However, this amount is still lower than the storage required by the traditional methods to solve the problem in large-scale environments.

4.4 Removing the Effect of Moving Objects in Traditional Visual Odometry Systems

State of the art ORBSLAM2 [115] extracts ORB features from the images and matches extracted features in consecutive frames. Extracting and matching features from the whole image in consecutive frames of the scene, which consists of dynamic objects, results in wrong translation and rotation estimation. Only the static part of the scene can be trusted to estimate the camera movement. This section merges information extracted from Deep Learning approaches with the feature extraction part of the ORBSLAM2 to determine reliable features from the static parts of the scene. Figure 4.7 illustrates the work-flow of the proposed approach.

Figure 4.7: Workflow of the Proposed Deep Uncertainty Aware Feature Extraction Method

The Mask-RCNN [64] semantic segmentation module outputs the bounding box related to different objects. Among different detectable classes, classes pertaining dynamic objects such as a car, truck, and pedestrians can be dynamic in the scene. Hence, the scene's positions with a high probability of describing moving objects can be extracted using Mask-RCNN. Figure 4.8(a) is showing a result of applying

Figure 4.8: Dynamic Objects Detected After Merging Deep Uncertainties

Mask-RCNN on an input image. As it shows, different objects which semantically describe a moving object are detected, and green bounding boxes are showing them.

While Mask-RCNN can extract places that are semantically related to moving objects, there is no guarantee that they are related to dynamic objects in the scene. As shown in the left image of Figure 4.8(a), the cars parked on the sides of the street semantically show moving objects, but they are static. Extracting features from such static objects helps to extract powerful and robust features in order to track and estimate motion in consecutive frames accurately.

Optical Flow estimates the motion of the scene at the pixel level, and by using this information, it is possible to have some knowledge to distinguish dynamic and static objects. Figure 4.8(b) is demonstrating the optical flow of the input image. As it shows, the pixels related to moving objects have higher motion compared to the surroundings, and also parts of the image near to camera are facing higher motion than far pixels (considering the Left and Right side of the optical flow image proves it). The Optical Flow method proposed by Ilg et al. [71] and named FlowNet2 is utilized in this section. FlowNet2 shown that the deep model can operate well on motion estimation in the dynamic environments and performs well when the camera is moving.

In our approach, real dynamic objects in the scene are selected from the objects semantically related to dynamic objects (*Semantics*), extracted from Mask-RCNN, using a multi-step approach. In the first step, since the right and left side of the optical flow can suffer from higher motion, the optical flow image divides into three parts horizontally. Then, the mean (μ) and variance (σ^2) of the pixel's length of movement (equivalent to optical flow image pixel values) are computed from optical flow measurements for each part, and a Gaussian function fits to model the measurements of each part. If the variance of each part of the optical flow image is less than a specific threshold (α), there is no possibility of having a dynamic object in that part. Then, for the remaining parts, if there is an object semantically related to dynamic objects in that part ($s \in \text{Semantics} \ \& \ s \in \text{part}$), the Gaussian's maximum value for that part is defined by M in the first step. Then, if the output of the fitted Gaussian function, for the length of movement of each pixel inside the semantic bounding box (s), is less than $\frac{1}{2}$ of the M , that pixel has different behavior compared to other pixels in that part of the optical flow image and can demonstrate a moving object. It means the motion behavior of a pixel inside the bounding box (s , which is semantically related to dynamic object) is different from the motion behavior of that part significantly, which increases its probability of being related to moving objects.

For each of the outputs ($s \in \text{Semantics}$) of Mask-RCNN, if a pixel inside the bounding box has the probability of demonstrating a moving object, that bounding box keeps for more consideration; otherwise, that bounding box considers as a static object. Afterward, for the remaining objects, the average motion measures, on a bounding box with the half-height of the dynamic object. Since the pixels on the top of the objects are mostly related to the sky or buildings, using a bounding box with a half-height of the original box from the bottom of the detected object helps to remove the effect of different motion behavior. Then, pixels surrounding the averaged bounding box with a distance of 2 pixels are considered for further calculations. If the difference of the average motion between the inner bounding box, which is related to the object, and the surrounding pixels is more than a certain threshold (β), the object considers as a real moving object. Figure 4.8(c) is showing the final results, and the algorithm is shown on Algorithm 4.1.

Algorithm 4.1: Extract Real Moving Objects From the Scene

Input : Consecutive Frames (F_t, F_{t-1})
Output: Bounding Box of Real Dynamic Objects in The Scene

```

1  $Results \leftarrow []$ 
2  $Semantic \leftarrow$  Applying Mask-RCNN on  $F_t$  to Extract Bounding Box
   Related to Car, Pedestrian and Truck
3  $Flow \leftarrow$  Length of movement for each pixel from FlowNet using  $F_t$  and
    $F_{t-1}$ 
4  $[left, mid, right] \leftarrow$  divide  $Flow$  horizontally to three parts
5 foreach  $p \in [left, mid, right]$  do
6    $\mu \leftarrow mean(p)$ 
7    $\sigma^2 \leftarrow variance(p)$ 
8    $G \leftarrow Gaussian(\mu, \sigma^2, p)$ 
9    $M \leftarrow Maximum(G)$ 
10  if  $\sigma^2 < \alpha$  then
11    move to next  $p$ 
12  end if
13  foreach  $s \in Semantic$  and  $s \in p$  do
14     $Dynamic \leftarrow \mathbf{False}$ 
15    foreach  $pixel \in s$  do
16      if  $G(pixel) < \frac{1}{2} \times M$  then
17         $Dynamic \leftarrow \mathbf{True}$ 
18      end if
19    end foreach
20    if  $Dynamic == \mathbf{True}$  then
21       $BB \leftarrow$  bounding box of  $s$  with halved height
22       $BB_{AvgMotion} \leftarrow$  Average Motion of  $BB$ 
23       $SAvgMotion \leftarrow$  Average Motion of all pixels Surrounding  $BB$ 
        and  $s$  with distance of two pixels
24      if  $|BB_{AvgMotion} - SAvgMotion| > \beta$  then
25         $Results \leftarrow s$ 
26      end if
27    end if
28  end foreach
29 end foreach
30 return  $Results$ 

```

In this manner, the ORBSLAM2 can benefit from the high-level abstract information extracted from the various Deep Learning network architectures. This knowledge guides the feature extraction module to ignore the scene's segments, which are demonstrating a moving object and extract features from the static context. Hence, in the feature extraction step, features extracted from parts of the real dynamic object ignores.

4.4.1 Experiments and Results

In this section, the proposed method evaluates on the challenging KITTI [54, 53] visual odometry dataset. This dataset contains both car movements on the highways and in urban areas where lots of parked vehicles with rich features exist. The open-source implementation of ORBSLAM2 is used to implement and test the proposed method, and the whole system is tested on an Intel Core i7-3930k CPU @ 3.20GHz $\times$ 6 with a Nvidia 1080TI GPU.

After running the original ORBSLAM2 and the proposed method on all of the 11 sequences of the KITTI dataset, the average Translation and Rotation Error across different paths and speeds measured. These results are shown in Figure 4.9 to 4.12 . As it shows, the impact of the proposed method is high in highways where the car moves fast, and high-speed cars exist; hence filtering out dynamic objects in such scenarios had a significant effect on the overall performance of the system. On the other hand, plotting the error across the path shows that the proposed method's failure is less than the original ORBSLAM2. Overall results show approximately 5 percent improvement when the speed of the car is high.

Figure 4.9: Average Translation Error Across Different Speeds (Blue Line Shows The Result of The Proposed Method and The Red Line is The Original ORBSLAM2)

Figure 4.10: Average Rotation Error Across Different Speeds (Blue Line Shows The Result of The Proposed Method and The Red Line is The Original ORBSLAM2)

Figure 4.11: Average Translation Error Across Different Path Length (Blue Line Shows The Result of The Proposed Method and The Red Line is The Original ORBSLAM2)

Figure 4.12: Average Rotation Error Across Different Path Length (Blue Line Shows The Result of The Proposed Method and The Red Line is The Original ORBSLAM2)

4.5 Concluding Remarks

This chapter, in the first step, proposed a portable module (named DSAIR) that outputs various representations to train a deep CNN and make the network robust to motion blur and dynamic objects. The proposed module is utilized to solve the problem of 6-DoF camera relocalization. The proposed network architecture consists of pair networks that run in parallel, and one is to recover the camera position, and the other retrieves the camera orientation parameters. The first network benefits from random cropping and rotation augmentation layers to improve the network’s generalization ability and make it robust to the camera’s rotation. In contrast, the later network only uses the original image as input because of the influence of geometric augmentations on the orientation parameters. The results show that the proposed method significantly improves the performance compare to the existing approaches. Moreover, the results show that the Semantic-Aware module can be turned off at test time, which improves the speed. Using the whole DSAIR module in training time is enough to make the network robust to motion blur and dynamic objects. The proposed portable module can attach to the various computer vision applications.

Furthermore, a new approach to benefit from deep learning in traditional SLAM systems and improve their accuracy and robustness is proposed. The proposed method combines semantic and motion uncertainty information from deep learning models to detect real dynamic objects in the scene. Real dynamic objects are

defined as objects that have a movement and semantically demonstrate an active object, such as the car, pedestrian, etc..

The dynamic parts of the scene can be ignored in the feature extraction step by knowing the real dynamic objects. Extracting reliable features from the static environment is essential for estimating the camera motion robustly. Results show that this approach improves the ORBSLAM2 accuracy for approximately 5 percent when the car's speed is high. In the future, the information from the deep depth estimation modules can be considered as another source of uncertainty to extract features from near and far distances uniformly [23] in monocular SLAM.

Chapter 5

Learning to Extract Robust Handcrafted Features with a Single Observation via Evolutionary Neurogenesis

5.1 Overview

Today’s successful systems of intelligent machines that understand the visual information, plan, decide, and navigate in the environment are based on machine learning approaches, especially Convolutional Neural Networks (CNNs). CNNs model the brain hierarchical learning procedure and attempt to learn a set of filters, in convolutional layers. Convolving learned filters with the input results in the extraction of rich feature representations that can assist in solving a specific type of problem [82, 142]. CNNs, similar to the human brain, require a massive amount of training data, exposing different kinds of objects from different viewpoints, to obtain mature neurons with powerful feature extraction ability.

The main drawback of deep learning approaches is their requirement of a massive amount of annotated training data. Furthermore, deep learning approaches require a long training time to learn a model that extracts robust features. Annotated training data for solving a specific problem is usually limited, and data gathering is an expensive and time-consuming task. However, a different number of massive databases are introduced in recent years to train CNNs for various applications [37, 184].

While data gathering, data annotation, and training deep networks for different problems are expensive and time-consuming tasks, few-shot learning approaches investigate a fast technique to solve high-level problems based on a few observations [171]. Transfer learning is one of such approaches [178]. In transfer learning, the knowledge, learned by a deep network architecture to solve another problem with the existence of a massive amount of training data, is used to address a new challenge. In this way, the network knows how to extract robust features in the lower layers, and it will focus more on retrieving semantics that assists in solving the new challenge in higher layers. However, existing transfer learning approaches

still require a large amount of annotated training data. Another group of few-shot learning approaches are metric-learning methods [150]. These methods learn a support vector based on the feature representation of training images in order to create a cluster for each label; queries are labeled based on the label of their nearby cluster. Label propagation [97] and embedding propagation [131] are also other techniques researchers are using to address the few-shot learning problem.

To solve one-shot and few-shot learning problems efficiently, it is important to look at recent neuroscience achievements to see how the brain performs and expands its knowledge. There are different opinions by neuroscientists on the period of neurogenesis, which refers to born of new neurons in the brain, and recent results show that it rapidly reduces in the first ages of life [152]. After a new neuron borns, an Astrocyte helps it to connect to others [135]; generally, Astrocyte considers the pieces of knowledge of different neurons and finds the best connections that make the new neuron effective. After the new neuron connected to others, it improves its knowledge and learns a new concept from them. Usually, new neurons learn knowledge of many neurons simultaneously and are an elaborated version of neurons. Studies also show that the number of neurons in the human brain reduces over time [157]. This reduction still needs to be studied by neuroscientists [85] but can be due to the replacement of mature neurons by advanced newborn ones, which makes the brain faster and results in more creativity.

This chapter models brain neurogenesis and Astrocyte in the structure of CNNs to learn a new concept based on the prior knowledge of a pre-trained CNN. In this manner, a newborn convolutional neuron is created, and an evolutionary algorithm, which is a model of Astrocyte, moves the newborn neuron among the knowledge of a pre-trained CNN and trains it. This chapter applied this method to learn a corner detection module, as the most important kind of handcrafted and interest point features, but it can also be generalized to learn any handcrafted feature extraction system and solve higher-level problems.

Handcrafted feature or interest point extraction refers to extracting distinctive informative key points from images. These feature points could be edges and corners which are robust to noise and occlusion and can describe the image properly. Such features are the primary requirement used in various computer vision

applications such as place recognition [51], visual odometry [126] and object detection [57]. Due to the different applications of handcrafted features, many kinds of researches have been devoted to this area [63, 86, 122, 38], and researchers are working on extracting unique feature points that are reliable to scale, occlusion and illumination change. However, finding a final accurate and computationally efficient solution for this purpose is impractical.

The trade-off between computational cost and accuracy has always been challenging in the field of handcrafted feature extraction. While deep learning based methods can achieve satisfactory results, they cannot execute in real-time on low-power embedded systems and CPU; besides, they need a massive amount of annotated training data to obtain a robust result. On the other hand, traditional methods can perform in real-time, but they lack in accuracy and robustness.

This chapter applies evolutionary computing approaches to model the effect of Astrocyte and neurogenesis procedure in the human brain. In this manner, this chapter benefits from the simulated learning ability of the human brain, modeled by CNNs to introduce a new approach, named ECNN. The proposed approach helps in the fast extraction of more sophisticated handcrafted features with the presence of a single training data. ECNN uses a set of robust low-level filters of a pre-trained CNN to optimize a fitness function, which results in a robust feature extraction module. This approach not only helps to reduce the computational cost but also allows us to construct a filter that can detect the desired features rapidly. This method requires only one single image for the training part to achieve the desired feature extractor. The experimental part of this chapter focuses on detecting corners as the primary feature point among the handcrafted features family, which is forgotten by researchers due to their problem with scale and rotation transforms. However, the proposed method can be generalized to identify other types of handcrafted features and solve other problems as well. The contributions of this chapter can be summarized as below:

- Modeling the neurogenesis behavior for the first time by utilizing CNN and Particle Swarm Optimization (PSO) [144] techniques.
- Compacting the knowledge of convolutional neurons that can be used to reduce the number of neurons in a CNN architecture.

- Proposing a fitness function that assists in solving the corner detection problem via the proposed technique which can be used to learn other handcrafted features.
- Proposing a new robust and fast corner detection module, which trained with a single training data and can be executed much faster than other learning-based corner detection techniques on CPU.

A graphical comparison of the proposed handcrafted feature extraction module with existing ones can be seen in Fig. 5.1.

Handcrafted Feature Extraction Methods (such as Harris, FAST, etc.)

Interest Point Extraction Methods (such as SIFT, SURF, etc.)

Deep Learning based Interest Point Extraction (such as SuperPoint, LIFT, etc.)

Proposed Evolved CNN Handcrafted or Interest Point Extraction

Figure 5.1: Workflow of different feature extraction techniques (Mentioned methods can be referenced as Harris [63], FAST [133] , SIFT [100], SURF [15], SuperPoint [38], and LIFT [176])

As a comparison and short review of the effect of evolutionary computing in deep learning, recent studies have shown that evolutionary algorithms help to

extract meaningful feature representation and also can determine the structure of a network [68]. Stanley and Miikkulainen [156] have proposed an evolutionary approach to adaptively increase the complexity of neural networks and find the best composition. Later, connective compositional pattern producing networks have been introduced to improve the capacity of [156] to perform on deep networks and investigate their best architecture [155, 154]. Liu et al. [96] and Sun et al. [158] have also evolved DNNs to find the optimum structure of the network with the help of evolutionary algorithms. Despite previous works, this chapter is opening a new branch of studies in the applications of evolutionary algorithms in deep learning. All deep learning methods are based on training the whole deep network architecture on a massive amount of training data to extract meaningful feature representations. However, as recent studies show, different layers of CNNs have a different understanding and can obtain different knowledge from the image. This research aims to discover a new approach to use the knowledge of different layers of a pre-trained CNN (a CNN model trained on a massive dataset to solve a different task), and create a feature extraction tool quickly to solve another high-level problem. This chapter shows that by having a deep understanding of different layers' knowledge and benefiting from evolutionary computing, it is possible to create a platform to learn a robust feature extractor based on only one training data. Recent studies benefit from evolutionary computing to find the best structure of the network, but this chapter uses evolutionary computing to optimize a fitness function to obtain the required feature extractor.

5.2 Proposed Method

This section provides an evolutionary algorithm that uses a pre-trained CNN to model neurogenesis procedure in the human brain. We named this algorithm ECNN. The proposed algorithm is fast and only needs a single training data to train a robust corner detection module. Here, we only consider neurogenesis in the first convolutional layer of a pre-trained CNN to learn and extract corners from an observed image. This method can be generalized to higher layers to learn advanced concepts and make networks compact, fast, and efficient.

The proposed approach benefits from the knowledge learned by convolutional

layers of a pre-trained CNN on a massive amount of training data. The pre-trained network behaves like the young brain. The neurogenesis procedure can be abstracted as creating a new convolutional neuron that wants to connect to a specific layer's convolutional neurons in a pre-trained network. In this manner, the newborn neuron gains the knowledge of a pre-trained CNN by compacting the existing information in that layer, resulting in a faster computation with reduced memory.

For this purpose, the knowledge of the GoogLeNet network architecture [162] trained on Places205 dataset [184], for place recognition, is used as the primary resource of the proposed learning procedure in this chapter. The GoogLeNet network architecture is one of the well-known architectures widely utilized due to its great inception module. Solving the problem of place recognition requires robust extraction of interest points, which is the problem this research focuses on. Hence, training a deep network on Places205 resulted in robust feature representations among layers for the aforementioned task. This pre-trained model is mainly used because of its great performance in scene recognition, and it is possible to use another pre-trained model as well. Since we are focusing on handcrafted feature extraction, utilizing a model that is used to solve place recognition is meaningful as handcrafted features are the primary requirement for addressing such an issue. Since this chapter studies neurogenesis in the first layer of CNN, in the first step, 64 convolutional filters (referred to as neurons and kernels in literature as well) of the first convolutional layer of the pre-trained CNN (GoogLeNet network architecture) are extracted.

After extracting the filters of the first convolutional layer, which can be referred to as mature neurons, a newborn convolutional filter is created. The newborn convolutional neuron gains knowledge from pre-trained (mature) filters to solve a specific problem. For this purpose, an evolutionary algorithm is employed to model the behavior of Astrocyte in the human brain to connect the infant neuron (new neuron which is not trained) to the mature ones (pre-trained neurons) in order to learn a new concept. Hence, the connection strength between the newborn neuron and mature ones is critical and needs to be investigated. It is important to clarify that, from now onwards, mature neurons are referring to the extracted

neurons from the pre-trained network, and infant or newborn neurons are the newly considered neurons that are not trained.

The newborn convolutional neuron is formed based on the aggregation of the extracted filters (mature neurons) in order to gain knowledge and extract various kinds of interest points. The filter of the newborn convolutional neuron is the weighted average of the knowledge of the pre-trained neurons; hence, the necessary parameters to find are the weights of each filter. The weights here are representing the connection strength between the newborn neuron and mature (pre-trained) ones. A probabilistic fitness function and PSO [144] are used to model Astrocyte and find the best connection strength among the newborn convolutional neuron and mature ones; the best connection strength results in the global minimum of the fitness function. We have chosen PSO to find the global minimum of fitness function due to its simple concept, easy implementation, robustness to control parameters, and computational efficiency compared to other mathematical and heuristic optimization techniques [84]. Besides, PSO, similar to Astrocyte, moves particles to find the best particle values that optimize the fitness function. Furthermore, as can be seen in the experiments section, the results are repeatable by utilizing PSO on the proposed fitness function. In the next subsections, each step of the proposed method is discussed in detail.

5.2.1 CNNs Filter Visualization and Understanding

This section analyzes the knowledge learned by the first layer of CNNs. Zeiler and Fergus [180] have intensely discussed on different segments of CNNs. They have concluded that the first few layers of CNNs are learning to extract basic features, while the following layers are managing to understand more abstract elements based on the basic features, learned in the first few layers. Hence, this chapter is focused on the first convolutional layer to propose a corner detection module and analyze the effect of the neurogenesis. Knowledge of the first layer of the GoogLeNet network architecture trained on Places205 is visualized and shown on the Fig. 5.2(a).

Fig. 5.2(a) shows 64 filters with the size of $(7 \times 7) \times 3$ extracted from the GoogLeNet in RGB color space (3 in the size of the filter stands for the number

Figure 5.2: (a) shows the first layer's filters of the GoogLeNet, and (b) is the result of convolving those filters respectively on Lenna image.

of color channels). It illustrates that filters are learned in different rotations and strengths. These filters not only can help to extract interest points in different directions but also can contribute to consider the scale. Fig. 5.2(b) is the result of applying each of the 64 filters on the famous image of Lenna. As the result shows, learned filters can result in the extraction of features, especially edges, in various directions with different strengths. Hence, the knowledge embedded in the extracted filters of a CNN can play a significant role in the handcrafted feature detection.

The following section describes the workflow of the proposed ECNN approach for learning the feature extraction module based on the knowledge of a pre-trained CNN.

5.2.2 Newborn Convolutional Neuron Learning Procedure

Fig. 5.3 illustrates the training procedure of the proposed ECNN approach. Parameters used in this section are described in Table 5.1. In the visualized procedure, a newborn convolutional neuron is decided to connect to other mature (pre-trained) neurons by the modeled Astrocyte. An Astrocyte, performs like a local search and evolutionarily moves the newborn convolutional neuron, among various mature neurons of a specific layer in a pre-trained network to find the best connection strength. The modeled Astrocyte finds the best connection strength, by optimizing a fitness function, which results in learning a new concept and reducing the

number of mature neurons.

Figure 5.3: This figure illustrates the learning procedure of the proposed ECNN module which models the effect of the brain's neurogenesis and astrocyte (* stands as the convolution operation).

Table 5.1: Definition of parameters used in modeling newborn neurons

Parameters	Description
c	Color Channel (Red (R), Green (G), or Blue (B))
$\mathbf{g}_{newborn}$	Newborn convolutional neuron filters
$\mathbf{g}_{newborn}^c$	Newborn filter for the specific color channel c
$\mathbf{f}$	Neurons of the specific layer of the pre-trained network used to evolve the newborn neuron. Called as Mature Neurons
$\mathbf{f}^c$	Mature neuron's filters in a specific color channel c
$\mathbf{f}_i^c$	i^{th} mature neuron filter in a specific color channel c
$\mathbf{S}$	Strength of the connection of the newborn neuron with pre-trained ones (Computed by Astrocytes which is modeled by PSO)
$\mathbf{S}^c$	Strength of the connection of the newborn filter in a specific color channel c with others
$\mathbf{S}_i^c$	Strength of the connection of the newborn filter with i^{th} mature neuron filter in a specific color channel c (decimal number)
$\mathbf{S}^{N_P}$	N_P randomly initialized copies of $\mathbf{S}$ are considered as particles and evolved by PSO
N	Number of mature neurons (size of $\mathbf{f}^c$)
N_P	Number of particles (Population size of PSO)
G	Number of generations that an evolutionary algorithm should evolve the newborn neuron
$\mathbf{I}$	Training Image
$\mathbf{O}$	Ground Truth

In order to model Astorocyte by PSO [144], a 1-D array $\mathbf{S}$ is used. Each value in this array (S_i^c) is in the range of $[-1, 1]$ and demonstrates the strength of connection of a newborn convolutional neuron with the i^{th} mature (pre-trained) neuron filter ($\mathbf{f}_i^c$) extracted from the pre-trained network; c corresponds to the number of color spaces considered for the input which could be 3 values in RGB color space. It is important to mention that the size of $\mathbf{f}$ is equal to the number of pre-trained neurons (N), where there are c filters for each of them. Hence, the size of $\mathbf{S}$ is $N \times c$.

PSO requires a population of particles that evolve in the environment to find the best parameters. Hence, the population size of N_P is considered, and N_P copies of the connection strength vector $\mathbf{S}$ are used and named as particles and are shown by $\mathbf{S}^{N_P}$. These N_P particles ($\mathbf{S}^{N_P}$) are initialized randomly and updated iteratively to find the best connection strength among mature neurons and newborn convolutional one. For the rest of this chapter, the variables and the notations defined in Table 5.1 are used to explain the neurogenesis algorithm.

The knowledge of each convolutional neuron is a filter that convolves with the input, and outputs a matrix; the output matrix has high values for places which are similar to the knowledge of the neuron. Particles have to determine the best connection strengths (weights) regarding the effectiveness of knowledge of different mature neurons in the newborn neuron and assist the newborn neuron to obtain its required knowledge.

The final knowledge (filter) of a newborn convolutional neuron is defined as the weighted average of the knowledge of various mature neurons; weights here are referred to as the connection strength between the newborn neuron's filters and mature ones ($\mathbf{S}^c$). Hence, the knowledge of a newborn convolutional neuron ($\mathbf{g}_{newborn}^c$) can be mathematically modeled as

$$\mathbf{g}_{newborn}^c = \frac{\sum_{i=1}^N S_i^c \cdot \mathbf{f}_i^c}{\sum_{i=1}^N S_i^c} \quad (5.1)$$

where S_i^c is a decimal number that represents the effectiveness of i^{th} pre-trained mature neuron knowledge ($\mathbf{f}_i^c$) in the final knowledge of the newborn neuron ($\mathbf{g}_{newborn}^c$). Hence, in the end, $3 \times N$ decimal numbers related to the strength of the connection

between a newborn neuron and mature ones are required in order to generate the output kernels for different color channels in RGB color space ($\mathbf{g}_{newborn}^R, \mathbf{g}_{newborn}^G$ and $\mathbf{g}_{newborn}^B$). It is important to mention that $\mathbf{f}_i^c$ and $\mathbf{g}_{newborn}^c$ are 2-D filters (based on the GoogLeNet network architecture, and they could have other dimensions based on other network architectures), and $\mathbf{S}$ is a 1-D array for one particle among the N_P particles ($\mathbf{S}^{N_P}$).

Once the newborn convolutional neuron ($\mathbf{g}_{newborn}$) gained knowledge, its knowledge can be examined by convolving the evolved filters ($\mathbf{g}_{newborn}^c$) with the input $\mathbf{I}$. Normalizing the output in the range of $[0, 1]$ results in a score map that probabilistically shows the probability that each pixel of the input belongs to the particular class learned by the neuron. Defining a fitness function in a supervised or unsupervised manner plays a critical role in assisting the newborn neuron to achieve its required knowledge. The fitness function measures the probability estimation error between the gained knowledge and the knowledge required to be learned by the newborn neuron (in the supervised strategy, the output is compared with the ground truth $\mathbf{O}$). The probabilistic fitness function that is used to solve the problem of handcrafted feature extraction is defined in the following sections.

The newborn convolutional neuron evolves for the maximum number of G generations via PSO in order to find the best connection strength ($\mathbf{S}^c$). In this manner, in each iteration, the infant neuron (new neuron which is not trained) moves among the mature neurons (pre-trained neurons) of a specific layer by giving different connection strength to extracted mature neurons via particles. Then, the fitness is computed for each of the particles ($\mathbf{S}^{N_P}$), for instance, by comparing the neuron output with the ground truth in the supervised strategy. Afterward, based on the particles ($\mathbf{S}^{N_P}$) and their scores, the PSO algorithm updates particles to find the best connection strength, which benefits the newborn convolutional neuron most by optimizing the fitness function. Algorithm 5.1 illustrates the modeled evolutionary neurogenesis procedure in brief.

Algorithm 5.1: Pseudocode of the proposed approach for finding the best connection strength between a newborn neuron and the mature neurons

Input : Pre-trained (mature) Convolutional Filters ($\mathbf{f}$)
Input Training Data ($\mathbf{I}$)
Ground Truth ($\mathbf{O}$)
Maximum number of generations that the newborn filter should evolve (G)

Output: Learning the best connection strength ($\mathbf{S}$) between the newborn filter and mature ones

// Define Parameters

- 1 N = number of mature filters ($\mathbf{f}^c$)
- 2 $\mathbf{S}$ = 1-D array with the size of $N \times c$.
- 3 $\mathbf{S}^{N_P}$ = 2-D array with the size of $N_P \times (N \times c)$. N_P copies of $\mathbf{S}$ are considered as particles and evolved by PSO to find the best connection strength. Initialized with random values in the range of $[-1,1]$.

// Evolve newborn neuron for G iterations

- 4 **for** $iter \leftarrow 0$ **to** G **do**
 - // Compute Knowledge of the Particles at Current Iteration based on the Connection Strengths
 - 5 $\mathbf{g}_{newborn}^{N_P, c} = \frac{\sum_{i=1}^N S_i^{N_P, c} \cdot \mathbf{f}_i^c}{\sum_{i=1}^N S_i^{N_P, c}}$
 - // Examine Particle's knowledge on the Input
 - 6 $\mathbf{M}^{N_P} = \mathbf{g}_{newborn}^{N_P} * \mathbf{I}$
 - // Compute the fitness
 - 7 $\mathbf{score}^{N_P} = fitness(\mathbf{M}^{N_P}, \mathbf{O})$
 - // Keep best connection strength based on the scores
 - 8 $\mathbf{S} = Best(\mathbf{S}^{N_P}, \mathbf{score}^{N_P})$
 - // Use PSO to update the connection strength based on the score of N_P particles
 - 9 $\mathbf{S}^{N_P} = update_S_with_PSO(\mathbf{S}^{N_P}, \mathbf{score}^{N_P})$
- 10 **end for**
- 11 **return** $\mathbf{S}$

5.2.3 Problem Formulation

The convolutional filters (knowledge) of the newborn neuron ($\mathbf{g}_{newborn}^c$) are modeled as the weighted average of the extracted filters of the pre-trained network ($\mathbf{f}_i^c$). The best weights, connection strengths $\mathbf{S}$, need to be investigated with evolutionary computing. The evolved filters of the newborn neuron ($\mathbf{g}_{newborn}^c$) are defined as Eq. 5.1. The number of convolutional neurons in the first convolutional layer of the pre-trained GoogLeNet is equal to 64, and there are 3 color channels for each neuron that convolves with the correspondent input's channel ($N = 64, c = 3$).

Convolving intensity map of different color spaces of the input image, $\mathbf{I}^c$, with the corresponding filter, $\mathbf{g}_{newborn}^c$ ($c \in \{R, G, B\}$), results in an image with high-intensity value in the pixels with similar behavior to kernel $\mathbf{g}_{newborn}^c$. Each of the color maps of the convolved image is normalized in a range of $[0, 1]$ in the next step to demonstrate a score map $\mathbf{P}^c$. In this map, points with higher scores have more probability of demonstrating the neuron's learned knowledge, while lower values are not similar to the knowledge related to the neuron. In the problem of interest point detection, where neuron has to learn interest points, points with larger values have more probability of being an interest point. This process can be mathematically modeled as

$$\mathbf{P}^c = rescaled(\mathbf{g}_{newborn}^c * \mathbf{I}^c) \quad (5.2)$$

where

$$rescaled(\mathbf{x}) = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})} \quad (5.3)$$

and $*$ is the convolution operator. Hence, three probability maps are learned to be produced by newborn neurons using the evolutionary algorithms, and each probability map shows pixels' interest point response or their probability of being an interest point. Different operators can be applied to combine the probability maps and select the most active interest points. Two effective operators are summation and maximum which are defined as

$$\bar{\mathbf{P}} = \mathbf{P}^R + \mathbf{P}^G + \mathbf{P}^B \quad (5.4)$$

and

$$\bar{\mathbf{P}} = \max(\mathbf{P}^R, \mathbf{P}^G, \mathbf{P}^B) \quad (5.5)$$

respectively. The summation operator, which is considered in this chapter, focuses on the aggregation of different probability maps and results in the extraction of more robust features; on the other hand, the maximum operator tries to extract features from each of the probability maps individually. Based on the estimated probability map ($\bar{\mathbf{P}}$), the system can determine interest points. If the user requires K feature points, K pixels with the highest likelihood of $\bar{\mathbf{P}}$ are returned. It is also possible to return pixels higher than a specific probability and determine K automatically. The primary requirement of this formulation is to identify the best $\mathbf{g}_{newborn}$, which itself depends on finding the best weights ($\mathbf{S}$). This concept can be obtained by minimizing a fitness function for extracting interest points and is explained in the next sub-section.

5.2.4 Fitness Function for The Problem of Corner Detection

Proper formulation of the fitness function is an essential requirement to obtain the best probability map, and solve various problems. Having the training image $\mathbf{I}$, with the height of h and width of w , and its corresponding interest points' (corners') probability map $\mathbf{O}$ (ground truth), and the estimated probability map $\bar{\mathbf{P}}$ for the connection strength of $\mathbf{S}$, the fitness function to train a neuron to detect corners is defined as

$$fitness(\mathbf{I}, \mathbf{O}, \bar{\mathbf{P}}) = IP \times \alpha + NIP \times \beta + ND \times \gamma \quad (5.6)$$

where NIP is the sum of the probability estimation error for the pixels which are not interest point, IP is the sum of the probability estimation error for the pixels which are interest point, and ND is the number of interest points in the ground truth which are not detected (their estimated probability is not as high as other corners to be considered as true corners). Let Q be the positions in which the ground truth probability map ($\mathbf{O}$) is equal to one which demonstrates an interest

point or a corner. Hence, IP and NIP are defined as

$$IP = \sum_{(i,j) \in Q} 3\mathbf{O}(i,j) - \bar{\mathbf{P}}(i,j) \quad (5.7)$$

and

$$NIP = \sum_{(i,j) \notin Q} \bar{\mathbf{P}}(i,j) - 3\mathbf{O}(i,j) \quad (5.8)$$

and show the error of estimating an interest point and non-interest points, respectively. The global minimum point for IP and NIP is when $\bar{\mathbf{P}}$ for interest points gets one, and for non-interest ones gets zero. The number of interest points in the input image $\mathbf{I}$ is usually less than the number of pixels which are not interest points. Hence, the weights α and β in Eq. 5.6 are required to balance the error and give the same influence to interest and non-interest points in the fitness function. Hence, α is defined as

$$\alpha = \frac{h \times w - \text{size}(Q)}{\text{size}(Q)} \quad (5.9)$$

where $\text{size}(Q)$ shows the number of interest points in image $\mathbf{I}$ (number of points with a high probability, value of 1, in the ground truth image $\mathbf{O}$). By having $\beta = 1$ and using presented α , the proposed weighting scheme considers the same impact for interest and non-interest pixels in the fitness function. The hyper-parameter β forces our structure not to give any probability to pixels which are not related to interest points. Choosing a high β forces system to consider pixels without the probability of interest point more, while smaller β moves the focus of the optimization to both sides and reduces the error of estimation for both interest and non-interest points. Based on the experiments conducted in this chapter, the proposed method works best with the β in the range of [5,15]. Hyper-parameter γ helps to avoid local minimum which happens by finding the highest probability for some interest points, and near-zero probability for other interest points; choosing $\gamma = \frac{\alpha}{2}$ results in the convergence of the model to the global minimum which gives almost the same probability to all interest points and make all of them detectable. After defining the fitness function and formulating the problem, evolutionary optimization should be applied to model the Astrocyte behavior and find the best connection strength ($\mathcal{S}$) in Eq. 5.1.

5.2.5 Evolutionary Optimization

The significant part of the proposed approach is determining the weights S_i^c in the range of $[-1, 1]$. Optimizing the fitness function proposed in the previous section helps to find the weights which result in a probability map that shows a high probability for interest points and low probability for non-interest ones. Among various types of evolutionary algorithms, the PSO algorithm is applied in this chapter. Among various PSO implementations, the utilized PSO method is based on the algorithm proposed by Kennedy and Eberhart [76], with the modifications suggested by Mezura-Montes and Coello Coello [106] and Pedersen [124]; the utilized implementation exists under the original optimization toolbox of MATLAB (R2017b version). PSO is a optimization algorithm which uses batches of particles (called swarms) to move in the space in order to find the global minimum fitness value. The particles are able to communicate with each other, and each particle moves to a direction based on its previous movement, direction to the best variables it found individually, and the best global variables found by neighboring particles. In order to find the global minimum fitness value with PSO algorithm, as stated, each newborn neuron is modeled as a particle and represents a $\mathbf{S}$ array with the size of $c \times N$. First, N_P number of particles ($\mathbf{S}^{N_P}$) are considered, as the swarm population, and evolved by PSO to find the best value that can solve the desired problem. In each iteration, the fitness function for each of the N_P particles is computed, and then based on the particles' score, particles ($\mathbf{S}^{N_P}$) are updated to find the best connection strength ($\mathbf{S}$). This process is continued until exceeding the maximum number of iterations (G) or reaching a point that the fitness gets into a steady state.

To update $\mathbf{S}^{N_P}$ by PSO, step 16 in Algorithm 1, following values are computed based on the score of each particle in the particles population ($\mathbf{score}^{N_P}$). The best particle among particles' neighbor is named $\mathbf{S}_{neighbor_best}^{N_P}$ and has the lowest fitness value compared to other particles among the K -nearest neighboring particles (K is the size of the particles considered to be neighbored). Besides, $\mathbf{S}_{self_best}^{N_P}$ shows the weights ($\mathbf{S}$) of the best fitness value each particle has seen over generations. Based on these measurements, the velocity $\mathbf{v}^{N_P}$ for moving each particle and updating it is computed as

$$\mathbf{v}^{N_P} = W_1 \times \mathbf{v}^{N_P} + W_2 \times (\mathbf{S}_{self.best}^{N_P} - \mathbf{S}^{N_P}) + W_3 \times (\mathbf{S}_{neighbor.best}^{N_P} - \mathbf{S}^{N_P}) \quad (5.10)$$

where W_1 , W_2 and W_3 are weights and show effectiveness of different parameters. Based on these weights, the particle decides either to move in the direction of its prior speed, neighbor best particle, or the best place it was on its own. After computing the velocity of particles, $\mathbf{S}^{N_P}$ particles are updated by the following equation.

$$\mathbf{S}^{N_P} = \mathbf{S}^{N_P} + \mathbf{v}^{N_P} \quad (5.11)$$

Updating $\mathbf{v}^{N_P}$ and moving particles iteratively result in finding the best connection strengths $\mathbf{S}$ among the newborn neuron and the pre-trained ones that minimize the fitness function.

5.2.6 Newborn Neuron Execution in Test Time

Each of the CNNs' neurons learns to extract features for a specific rotation, and having a deep network with many neurons in each layer makes the network robust to the rotation [105]. For instance, for an input image with a rotation of θ , neurons that are sensitive to the rotation of θ are activated to produce the final result. Hence, data augmentation and observing various rotations and scales of the input image in the training time is significant for CNNs. A wide range of researches has been done to make CNN neurons rotation invariant [27].

Using the pieces of knowledge of a pre-trained CNN and evolving them to learn a new concept or produce a complex neuron is also not invariant to rotation and scale. Brain neurons are also not invariant to rotation and scale, and once human sees something that is not understandable on the paper, he rotates the paper and moves it forward or backward to make it understandable for his brain. In this chapter, we model this behavior and make the trained newborn neurons invariant to rotation and scale transforms by benefiting from the promising evolutionary computing, and the probabilistic formulation of the newborn neuron output and the fitness function. In order to make the evolved newborn neuron invariant to rotation and scale, rotation and scale layers are applied to the input, only during

the test time (after the newborn neuron is trained). The rotation layer rotates the input for 36 times by 10 degrees to produce different input with varying angles of rotation from 0 to 360 degrees. Afterward, for each of the rotations, the scale layer is applied to create images with different scales, which varies from half to double the size of the input.

In this manner, the output of the newborn neuron is the set of probability maps for each of the inputs in the scale space of the rotated inputs. The final probability for each pixel of the main input is defined as the average of the probabilities for that point in different rotations and scales. Having the averaged probability map for the input I , the points with the highest probability are the most similar points to the learned concept of the neuron. Hence, to extract a specific number of features, such as corners, the required number of features should be selected from the points with the highest probability. Fig. 5.4 demonstrates the work-flow of the proposed probabilistic rotation and scale invariant feature extraction module, which is named ECNN-RS (R and S are related to using rotation and scale layers).

Figure 5.4: Procedure of executing ECNN newborn neuron with Rotation and Scale layers that applies on the input of the evolved newborn neuron in test time

5.3 Experiments and Results

5.3.1 Implementation and Experimental Setup

The experiments applied to evaluate the performance of the proposed method are all implemented in MATLAB and C++ and tested on a system with Core i7-4770 CPU and GTX780Ti GPU. PSO module of MATLAB (R2017b version) optimization toolbox is utilized with its default parameters. The training phase is implemented in MATLAB, and the execution of the trained model on queries is implemented in C++. The corner detector is trained with a single image and its corresponding ground-truth, which are illustrated in Fig. 5.5. The swarm size (N_P) of 640 is used, and each of the particles is initialized randomly.

Figure 5.5: The training image (a) and its corresponding ground truth corners (b) which are used to train the corner detection module are illustrated in this figure. The ground truth image shows the probability map of (a) where corner pixels have the value of one and are the interest points required to be detected by the neuron

In order to make the comparison, a dataset that consists of 67 reference images from outdoor and indoor scenarios is created. Since there are different conditions such as illumination, rotation, noise, and scale change for each of the reference images, the number of total images that the method is tested on gets up to 3082 images. The experiments aim to evaluate the robustness of the feature detection modules in the presence of the mentioned conditions. A robust feature detector should be able to detect the same feature with the maximum distance of two pixels after applying different transforms. Hence, the metric which is used in the experiments is the ratio of features that can be detected repeatedly the same as the reference images after applying a transform. The reported results are averaged on the whole dataset.

In order to compare the proposed method with other corner and interest point detectors, OpenCV implementation of SIFT [100] and ORB [134], authors' released source code for deep learning-based methods (D2-Net [41], LIFT [176], LFN-Net [122], and SuperPoint [38]), and MATLAB implementation of other methods (Harris [63], SURF [15], BRISK [86], KAZE [6], Shi-Tomasi [143], and FAST [133]) are used. All methods are forced to extract 100 features from images; 100 feature points is a fair number due to the existence of at least 100 interest points with specific conditions in most scenarios, and it is the number of features usually utilized in various computer vision applications, especially in the area of robotics (e.g., solving the problem of visual odometry [126]). In order to gain a deep understanding of the proposed method and each of its components, three different models are considered for testing. ECNN is representing the results of the proposed method without using the rotation and scale layers in the test time, ECNN-R is the proposed method with the rotation layer, and ECNN-RS is the proposed method which benefits from rotation and scale layers simultaneously in the test time. The following subsections describe the different experiments conducted on the proposed approach.

5.3.1.1 Variation of Illumination

In order to analyze the effect of illumination change on different corner detection modules, an additive intensity value that varies from -175 to 175 is added to all

pixels of the reference image. Afterward, corners in the input reference image and transformed image are detected, and the ratio of pixels that match between two images is measured.

5.3.1.2 Change of Scale

In order to analyze the effect of scale, the features of a reference image in the dataset are identified. Afterward, the image is scaled from 0.2 to 2.4 of its original size, and the features detected for different scales which are similar to the feature points of the reference image are identified, and then the ratio of matched features is measured.

5.3.1.3 Effect of Noise

Noise is another source of change that damages the input image, and feature detectors should be robust to it. In order to evaluate the performance of the proposed method in the existence of noise, a zero-mean Gaussian noise with the variance that varies from 0 to 0.2 is added to the reference image and the ratio of similar features that can be detected before and after adding the noise is measured.

5.3.1.4 Change of Rotation

Implementing a filter that is invariant to rotation is a big challenge, and lots of studies have been done to extract features that are robust to the rotation. Without using the proposed probabilistic rotation invariant feature extraction technique, the trained neuron is not robust to the rotation transform. Further studies have shown that the learned filter of ECNN is robust to the rotations between -5 to 5 degrees. In order to examine the robustness of various methods to the rotation transform, the reference image is rotated from 0 to 360 degrees and the extracted features for each degree are compared with the reference image. Then, the ratio of similar features detected from the rotated image and the reference image is measured.

5.3.1.5 Change of View Point

In order to evaluate the robustness of the proposed method in the existence of a change in the viewpoint, which contains different transforms such as rotation, scale,

and change in the illumination, the "Graffiti" and "Bricks" datasets are used [107]. The Graffiti dataset is from a structured scene that contains many corner-like structures, on the other hand, the Bricks dataset contains similar patterns in all parts of the image. Having different viewpoints from these two different scenes makes the dataset challenging to handle. For each dataset, there is one reference image and a few images from different viewpoints of the same scene. The feature points are extracted from all the images, and then with the help of the transform matrix, the ratio of features which are detected repeatedly in both of the reference image and different viewpoint is measured.

5.3.2 Discussion

Comparing the performance of the ECNN approach with the existing learning based feature detection techniques in Table 5.2 shows that the proposed method has significantly less number of parameters. This significant reduction in the number of parameters results in faster execution and training of the proposed method, and a considerable drop in memory usage. As the Table shows, the proposed method trains and executes faster than other learning-based methods, with only a single training data, which is a significant distinction in comparison to other learning-based methods. It is essential to notice that the other approaches cannot learn to extract features based on a single observation, and the best model for each work trained on a specific dataset is used. As the Table shows, benefiting from PSO and evolutionarily training a newborn neuron based on the knowledge of pre-trained network results in much faster learning ability (20 minutes on CPU vs more than 2 hours on GPU).

In order to check the capability of reproducing the results, four sets of experiments are considered, and in each set, the fitness function is optimized in 150 runs based on the proposed approach (600 runs are conducted in total). The mean, min, max, median, and variance of each experiment set's best fitness value over different runs is reported in Table 5.3. t -test is applied to check the null hypothesis that two experiment sets, represented by vectors $\mathbf{X}$ and $\mathbf{Y}$, come from normal distributions with equal means and equal but unknown variances [111]; the alternative hypothesis states that data in two vectors are coming from populations with

Table 5.2: This table compares the network parameters, learning and execution performance of the proposed ECNN module with other learning-based feature detection techniques. The reported execution time is for processing an image with a size of 777×1280 pixels.

	Number of Learnable Parameters	Number of Images Used for Training	Training Time	Execution Time on CPU in seconds
LIFT [176]	290k	≈ 2000	$\approx > 2$ hours on GPU	- (8.558447 on GPU)
LF-Net [122]	39k	2.5M	$\approx > 2$ hours on GPU	22.8457
SuperPoint [38]	940k	$32 \times 200k$ synthetic + 80k real	$\approx > 2$ hours on GPU	8.656
D2-Net [41]	24k	$\approx 2M$	$\approx > 2$ hours on GPU	8.302
ECNN	192	1	$\approx < 20$ minutes on CPU	0.989 (ECNN-R: 1.645) (ECNN-RS: 3.324)

Table 5.3: Mean, Minimum, Maximum, Variance and Median of best fitness value in each of the experiment sets. 150 runs are conducted in each of the experiment sets.

	Experiment Set #1	Experiment Set #2	Experiment Set #3	Experiment Set #4	Overall
Mean of Best Fitness	263651.305511880	263649.056849713	263535.239650400	264006.565448413	263710.5418651017
Minimum of Best Fitness	255601.412157000	255244.944314000	256556.568347000	256521.786387000	255244.9443140000
Maximum of Best Fitness	273285.549412000	273369.302353000	271055.508515000	270966.753109000	273369.3023530000
Variance of Best Fitness	9813158.19495463	10540143.8292961	8595277.54214859	8895892.76866592	9445198.505971884
Median of Best Fitness	263579.800196000	264105.130812500	263391.061120500	263835.797927000	263737.2302520000

unequal means. It is important to mention that the best fitness value achieved in each of the runs of experiment sets exists in the population vector of the experiment (vectors $\mathbf{X}$ and $\mathbf{Y}$ contain 150 values, which is the best fitness in each run of their corresponding experiment). The p -value between experiments is reported in Table 5.4. p -value is the probability of observing a test statistic as or more extreme than the observed value under the null hypothesis. A small p -value (usually ≤ 0.05) shows strong evidence against the null hypothesis. Considering the significance level of 5% shows that the null hypothesis cannot be rejected in Table 5.4 (The null hypothesis rejects if the p -value is less than the significance level). Hence, the experiments can be repeated.

Fig. 5.6(a) illustrates the convergence of the proposed method to the minimum fitness value for four different experiment sets based on the mean of the fitness value in each iteration. The difference in the obtained minimum fitness value in different

Table 5.4: p -value among experiments

	Experiment Set #1	Experiment Set #2	Experiment Set #3	Experiment Set #4
Experiment Set #1	1	0.9951	0.7406	0.3152
Experiment Set #2	0.9951	1	0.7502	0.3214
Experiment Set #3	0.7406	0.7502	1	0.1685
Experiment Set #4	0.3152	0.3214	0.1685	1

runs is because of facing a local minimum by the PSO algorithm. While using other variants of the optimization algorithms or different optimization parameters can help to find the global minimum fitness value better; our study shows that due to the robustness of the pre-trained CNN neurons, each of the learned feature detectors is robust enough to be used. It is also essential to notice that defining a proper fitness function that gives correct results on the position of its minimum value is critical. Besides, it is possible to combine knowledge of many trained newborn neurons to achieve better results and avoid local minimums.

Fig. 5.7(a) demonstrates a sample test input from the test dataset, and Fig. 5.7(b)-5.7(d) are demonstrating the extracted corner points using different variants of the proposed method. The features detected by the proposed methods (ECNN, ECNN-R, ECNN-RS) are complex, and as can be seen in figures some of the easy to detect corners might not be identified. These feature points are detected due to their similarity to the learned model, and different factors such as the scale of the features influences on the detector. The important aspect is robustness in detected corners in the presence of various conditions, so the robustness of the ECNN, ECNN-R, and ECNN-RS is compared with other techniques. In order to analyze the robustness of the proposed corner detector, its performance is evaluated by measuring the average ratio of features that have been detected repeatedly with the presence of various transforms. The summary of the results is illustrated in Tables 5.5 and 5.6. The detailed results can be seen in the Fig. 5.8-5.11. As the results show, the proposed method is robust to various transforms except for the scale, in which the deep learning based methods are robust to such transform with their deep architecture. However, the performance of the proposed method on the other transforms is significantly better than the deep learning methods. Overall, by comparing the average performance, the proposed technique results in a better corner detector compared to other techniques. ECNN-R and ECNN-RS perform equally in most cases except facing a scale transform in the scene, and comparing ECNN with ECNN-R shows the significant impact of the rotation layer. Hence, it is possible to select among ECNN, ECNN-R, and ECNN-RS for different applications and benefit from different speeds and accuracies. The sample results of the visualized corners in different conditions, on the created dataset, can be seen in

(a) Average of best fitness value in each experiment set during the optimization process

(b) Iteration #1 (c) Iteration #102 (d) Iteration #253 (e) Iteration #756

Figure 5.6: (a) shows the optimization of the fitness function over different iterations for each set of experiments; the plotted value for each iteration is the average of the best fitness value in all 150 runs conducted in each experiment set. (b),(c),(d) and (e) are showing the learned corners for different steps by red dots. In these figures, more corners than actual corners are asked to extract by trained corner detector for visualization. Having two red points on top and down left of (e) means the actual corners are detected, and the detector is moved to other low probability parts and detected points as corners are the most significant ones and are similar to a corner; hence the corner detector is well trained.

Figure 5.7: (a) is demonstrating a sample test data, and (b),(c), and (d) are showing the 200 corner points detected in this image, by a newborn neuron which is trained to detect corners.

the Supplementary Materials.

Further experiments are conducted by performing the pairwise comparison of the methods. In this manner, the Wilcoxon test [55] is applied by considering the significance level of 5%. Wilcoxon test is a statistical test that tests the null hypothesis, which states that for vectors $\mathbf{A}$ and $\mathbf{B}$, the $\mathbf{A} - \mathbf{B}$ is from a distribution with zero median; in other words, $\mathbf{A}$ and $\mathbf{B}$ are from the same distribution with the same median. We have considered the alternative hypothesis as $\mathbf{A} > \mathbf{B}$ (based on their distribution median); $\mathbf{A}$ and $\mathbf{B}$ are considered as two methods represented by vectors that contain the average ratio of repeated features in each test condition. The computed p -value between paired methods can be seen in the Table 5.7 ; $\mathbf{A}$ is a method in the row, and $\mathbf{B}$ is a method indicated in columns. If the Table's value is less than the significance level, the null hypothesis rejects, and the alternative hypothesis proves ($\mathbf{A} > \mathbf{B}$). It can be seen that the ECNN-RS and ECNN-R can reject the null hypothesis when compared to all other methods, which shows the significant difference and improvement of the proposed approach compared to others.

Table 5.5: The average ratio of matched features for viewpoint datasets. Top three results in each row are colored.

	FAST [133]	Shi-Tomasi [143]	Harris [63]	BRISK [86]	SURF [15]	KAZE [6]	SIFT [100]	ORB [134]	SuperPoint [38]	LIFT [176]	LF-Net [122]	D2-Net [41]	ECNN	ECNN-R	ECNN-RS
Bricks	0.2210	0.2060	0.2220	0.1740	0.2450	0.2400	0.1640	0.2290	0.0187	0.2400	0.2506	0.1520	0.2520	0.2800	0.2630
Graffiti	0.2460	0.2170	0.300	0.2220	0.1330	0.1420	0.2039	0.29	0.0235	0.1140	0.1122	0.0040	0.1450	0.2870	0.3040
Average	0.2335	0.2115	0.261	0.198	0.189	0.194	0.18395	0.2595	0.0211	0.177	0.1859	0.078	0.1985	0.2835	0.2835

Table 5.6: The average ratio of repeated features after applying different transforms. Top three results in each row are colored.

	FAST [133]	Shi-Tomasi [143]	Harris [63]	BRISK [86]	SURF [15]	KAZE [6]	SIFT [100]	ORB [134]	SuperPoint [38]	LIFT [176]	LF-Net [122]	D2-Net [41]	ECNN	ECNN-R	ECNN-RS
Illumination	0.5859	0.6326	0.6330	0.5589	0.6169	0.6157	0.5638	0.5699	0.5966	0.6136	0.5971	0.3185	0.6866	0.6653	0.6683
Scale	0.4479	0.5841	0.5917	0.5279	0.4658	0.4349	0.6316	0.5549	0.9055	0.5102	0.7478	0.1584	0.5935	0.5694	0.6342
Noise	0.2524	0.3111	0.3343	0.2349	0.5968	0.6481	0.4049	0.5211	0.5584	0.5155	0.4133	0.1332	0.4157	0.6381	0.6728
Rotation	0.7587	0.7958	0.8259	0.6641	0.7157	0.6986	0.4385	0.3265	0.0934	0.3963	0.1380	0.0951	0.3096	0.8459	0.8399
View Point	0.2335	0.2115	0.261	0.198	0.189	0.194	0.18395	0.2595	0.0211	0.177	0.1859	0.078	0.1985	0.2835	0.2835
Average	0.45568	0.50702	0.52918	0.43676	0.50364	0.51826	0.44455	0.44644	0.435	0.44252	0.41642	0.3916	0.44078	0.59764	0.61974

(a) Traditional Corner Detectors

(b) Traditional Interest Point Detectors

(c) Deep Learning based Interest Point Detectors

Figure 5.8: Comparison of the robustness of the proposed method with others in terms of illumination change. Each point of the figures shows the average ratio of features repeatedly detected after changing the illumination of each image of the dataset by adding a specific value shown on the horizontal axis.

Table 5.7: p -value of Wilcoxon test among different methods (cells get brighter as the value gets to zero and the null hypothesis rejects and alternative hypothesis which states row $\mathbf{A} >$ column $\mathbf{B}$ (based on their distribution median) satisfies; cells regarding values less than the significance level of 5% are purely white.)

		B														
	FAST [133]	Shi-Tomasi [143]	Harris [63]	BRISK [86]	SURF [15]	KAZE [6]	SIFT [100]	ORB [134]	SuperPoint [38]	LIFT [176]	LF-Net [122]	D2-Net [41]	ECNN	ECNN-R	ECNN-RS	
FAST	1	1	1	0.00224629	0.880984	0.95007	0.928627	0.646419	0.759196	0.581743	0.618519	1.32034e-09	0.85344	1	1	
	2.50735e-09	1	0.999968	3.29088e-08	0.0773231	0.169109	0.0938376	0.020821	0.234262	0.0164903	0.0605351	1.23757e-09	0.506137	0.999973	1	
Harris	2.67182e-09	3.37878e-05	1	4.96079e-08	0.0592761	0.136701	0.0197833	0.0173795	0.194219	0.00875758	0.0339677	1.23757e-09	0.444937	0.999867	0.999999	
BRISK	0.997827	1	1	1	0.999653	0.999639	0.988393	0.822763	0.793927	0.585871	0.684916	3.0284e-09	0.8581	1	1	
	0.121134	0.924199	0.941962	0.000361082	1	0.979443	0.0316426	0.00598071	0.0267256	0.00139716	0.00634923	1.23757e-09	0.0531103	1	1	
	0.0510292	0.833546	0.865603	0.00037524	0.0210868	1	0.00850958	0.000851593	0.00598071	9.56499e-05	0.0018373	1.23757e-09	0.0146446	0.999997	1	
SIFT	0.0728267	0.907923	0.980719	0.0119322	0.969103	0.991732	1	0.602284	0.992198	0.768972	0.914719	1.82198e-09	0.891756	0.999996	0.999999	
ORB	0.357523	0.979703	0.98307	0.180001	0.994196	0.999179	0.401803	1	0.85861	0.759196	0.0920775	1.23757e-09	0.657311	1	1	
	0.244112	0.768972	0.808678	0.209102	0.973922	0.994196	0.00803201	0.143774	1	0.393639	0.00160358	2.17149e-08	0.493863	0.999208	0.999208	
LIFT	0.422395	0.98394	0.99149	0.418257	0.99865	0.999908	0.234262	0.244112	0.610426	1	0.020821	1.23757e-09	0.331452	1	1	
LF-Net	0.385521	0.940724	0.966822	0.318853	0.993837	0.998224	0.0869439	0.999658	0.99845	0.979703	1	5.33572e-09	0.98013	0.999966	0.999993	
D2-Net	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
ECNN	0.148927	0.497954	0.559113	0.144216	0.94799	0.985732	0.110162	0.346466	0.510228	0.67226	0.0203686	8.96643e-10	1	0.999415	0.999979	
ECNN-R	3.66486e-09	2.83019e-05	0.000138992	1.81413e-08	1.32034e-09	2.81525e-06	4.29694e-06	1.60832e-08	0.000821356	1.55621e-07	3.5393e-05	1.23757e-09	0.000606169	1	0.999148	
ECNN-RS	8.22929e-09	2.56655e-07	1.2768e-06	3.90269e-09	6.42867e-09	2.92389e-08	8.85774e-07	3.44113e-09	0.000821356	3.70222e-08	6.99537e-06	1.32034e-09	2.18249e-05	0.00082784	1

(a) Traditional Corner Detectors

(b) Traditional Interest Point Detectors

(c) Deep Learning based Interest Point Detectors

Figure 5.9: Comparison of the robustness of the proposed method with others in the presence of Gaussian Noise with different variances (σ^2). Each point of the figures shows the average ratio of features repeatedly detected after adding Gaussian noise with a variance of σ^2 to each image of the dataset.

(a) Traditional Corner Detectors

(b) Traditional Interest Point Detectors

(c) Deep Learning based Interest Point Detectors

Figure 5.10: Comparison of the robustness of the proposed method with others in terms of change in the rotation of the image for θ degrees. Each point of the figures shows the average ratio of features repeatedly detected after rotating images in the dataset for θ degrees.

(a) Traditional Corner Detectors

(b) Traditional Interest Point Detectors

(c) Deep Learning based Interest Point Detectors

Figure 5.11: Comparison of the robustness of the proposed method with others in terms of change in the scale. Each point of the figures shows the average ratio of features repeatedly detected after scaling dataset images for a specific rate.

Understanding the neurons' learning capacity still requires to be studied, and there is no lower or upper bound on it, and it strongly depends on the correlation of different pieces of knowledge that have to be known by the neuron. Instantly, visualizing the estimated probability maps of the trained ECNN neuron on the training image over different color channels ($\mathbf{P}^c$), which can be seen in the Fig. 5.12, shows that the neuron trained to detect corners learned three different concepts related to vertical and horizontal edges, and corners. There exists a high value for measurements at corner points in different color channels of Fig. 5.12(b) which results in high probability ($\bar{\mathbf{P}}$) for that points for being corner compared to other points; top $\bar{\mathbf{P}}$ values in this score map have been shown in the Fig. 5.6(e). Hence, three different concepts are learned by the newborn neuron, and it not only can extract corners as its primary output by giving a high value of $\bar{\mathbf{P}}$ to corners but also can help to extract and distinguish vertical and horizontal edges as well, by giving smaller value of $\bar{\mathbf{P}}$ to edges compared to corners.

Another fundamental problem with the CNNs is their over-fitting in training time. The proposed ECNN module is trained with a single image, and the latest weights, evolved connection strengths, are used for testing. As the result shows, it is not over-fitted and can perform properly on various images. We believe this aspect is due to using knowledge of a pre-trained network, which is not overfitted, to evolve the newborn neuron. However, it requires to observe the behavior of the proposed method while solving other problems to be proved. Moreover, the knowledge of the pre-trained network used to train the newborn neuron plays a critical role. Having a well-trained network, with the high generalization ability helps the new convolutional neuron, which is not trained and is added to evolve, converge to the global minimum fitness value faster, and also output a generalizable filter. However, if the network filters are weak, the newborn neuron still can optimize its fitness function, by the promising evolutionary computing, and find a robust filter at a reasonable time, with an acceptable generalization ability. Overall, the results demonstrate the significant effect of biological-inspired neurogenesis in learning a new concept and compressing the knowledge of many neurons in one.

Figure 5.12: (a) shows the training data, and (b) is the normalized and scaled output probability map of convolving $\mathbf{g}_{newborn}^R$, $\mathbf{g}_{newborn}^G$ and $\mathbf{g}_{newborn}^B$ with the training image in the RGB color space. As it demonstrates, the trained neuron not only can work as a corner detector (as its primary output with high score) but also can be used to detect vertical and horizontal edges. In each black square, $\mathbf{g}_{newborn}^R$ extracts down right and top left, $\mathbf{g}_{newborn}^B$ extracts top right and down left, and $\mathbf{g}_{newborn}^G$ extracts all corners but have more influence on top and down left corners. Simultaneously, $\mathbf{g}_{newborn}^R$ have a low effect on extracting right and down side edges, $\mathbf{g}_{newborn}^B$ can help to detect the top and right edges, and $\mathbf{g}_{newborn}^G$ can help to extract and understand vertical and horizontal edges by giving different probabilities to different types of edges.

5.3.2.1 Future Directions

The proposed ECNN method moves the field forward by its fast and robust learning ability, which allows deploying new techniques in less time; with the evidence that a corner detector trained and executed much faster than other deep learning-based approaches on CPU (with reference to Table 5.2). Besides, the proposed approach can be generalized to reduce the number of neurons in trained CNNs, to improve their performance and speed significantly. For this purposes, a few parameters need to be set, which are the number of newborn neurons in each layer and the fitness function. Fitness function could be equivalent to the loss function, which

is minimized during training the whole network. Two different approaches can be taken by researchers to benefit from the proposed evolutionary neurogenesis technique, which can be mentioned as below:

- In the first approach, a different number of newborn neurons are considered for each layer. These newborn neurons are going to gain knowledge from the pre-trained neurons of their respective layer. The input will go through the first layer of newborn convolutional neurons, and the output will go to the second layer of newborn convolutional neurons, similar to the procedure of traditional CNNs. After the final layer, the output takes and compares with the ground truth to compute the loss function, which is equivalent to the fitness function. Newborn neurons in different layers will evolve altogether to minimize the fitness function and reach the global minimum.
- In the second approach, a set of newborn convolutional neurons will be considered on top of the traditional CNNs, after the final feature representation layer. The new neurons will gain knowledge from the last layer of the CNN to produce the required output. This will help to benefit from ECNN on top of the traditional CNNs to make a robust feature representation for the inputs.

The best number of newborn neurons that should be considered in each layer is unknown and, as discussed, depends on the correlation of different knowledge that a neuron must learn. To solve a specific problem or compress a CNN, an iterative procedure can be utilized to examine the network with the presence of a different number of new neurons in each layer. These steps should be taken in the future, and their advantages, disadvantages, and limitations need to be analyzed.

5.4 Concluding Remarks

This chapter has proposed a new approach to learn and extract handcrafted features based on only one single training image. The proposed method makes use of the knowledge of a pre-trained CNN and benefits from evolutionary algorithms to evolve (train) a newborn convolutional neuron and learn a new concept. The results show that the learned corner detector is robust to different transforms and

outperforms the existing methods for approximately 20 percent. The proposed method can be employed to extract other kinds of handcrafted features. Besides, it can also be used to create the user-defined feature extraction module, with only one single training image and its corresponding ground truth probability map. The proposed handcrafted feature extractor is specifically useful for robotic applications due to lower computational complexity and memory usage. Besides, as future works, the proposed ECNN module can be generalized to benefit from the knowledge of higher layers to solve different problems, and simultaneously improve the performance and speed of CNNs.

Chapter 6

Deep Learning based Inertial Odometry

6.1 Overview

Inertial Measurement Units (IMU) suffer from bias and measurement noise, which makes it much more complicated to tackle the problem of Inertial Odometry (IO). Due to the error propagation over time, an inaccurate estimation or a small error will cause the odometry and localization system unreliable and unusable in a split of seconds while estimating robot position. In the first steps, this chapter presents a novel triple channel deep Inertial Odometry network architecture based on the physical and mathematical models of IMUs. The proposed method simulates the noise model in the training phase and becomes robust to noise during testing. Besides, the proposed network architecture also considers the time interval between two consecutive IMU readings (sampling time) so that it is robust to change of IMU frequency and miss of IMU information. The proposed network architecture outputs the length of translation and orientation changes and outperforms all the existing solutions on the IMU readings of challenging Micro Aerial Vehicle dataset and improves the accuracy by approximately 25 percent.

While the mentioned proposed approach can estimate the length of position and orientation changes, further research is applied to illustrate the power of the first deep learning framework for estimating the full 3D orientation of the flying robots (as yaw, pitch, and roll in quaternion coordinates) accurately with the presence of a single IMU. A particular IMU should be utilized during the training and testing of the proposed systems. Besides, a method based on the Genetic Algorithm is introduced to measure the IMU bias in each execution. The results show that the proposed method improved the flying robots' ability to estimate their orientation displacement by approximately 80% with the presence of a single particular IMU. The proposed approach also outperforms existing solutions that utilize a monocular camera and IMU simultaneously by approximately 30%.

6.2 Preliminaries

IMU measures 3-axis linear acceleration, $\hat{a}_t$, and 3-axis angular velocity, $\hat{w}_t$, in the IMU body frame at each time step (t) which can be adapted to compute orientation, velocity, and position of the robot. These measurements are influenced by bias and measurement noise. The relation between IMU measurements and the exact and actual linear acceleration (a_t) and angular velocity (w_t) can be modeled as

$$\hat{a}_t = a_t + b_{a_t} + R_w^t g^w + n_{a_t} \quad (6.1)$$

$$\hat{w}_t = w_t + b_{w_t} + n_{w_t} \quad (6.2)$$

where b_{a_t} and b_{w_t} are accelerometer and gyroscope bias, and n_{a_t} and n_{w_t} are additive zero-mean Gaussian noises with variances σ_a^2 and σ_w^2 . The bias of IMU is different in each run and changes very slowly over time. The effect of gravity on accelerometer models by g^w , which is the gravity vector in the world frame. g^w rotates to the IMU body frame with R_w^t in order to influence a_t . By considering the kinematic of IMU and assuming that a and w are constant in-between times $[t, t + \Delta T]$, where ΔT is the smallest time interval between two consecutive IMU readings, the orientation, velocity, and position of the IMU can be measured by

$$R_{t+\Delta T} = R_t \text{Exp}(w_t \Delta T) \quad (6.3)$$

$$V_{t+\Delta T} = V_t + a_t \Delta T \quad (6.4)$$

$$P_{t+\Delta T} = P_t + v_t \Delta T + \frac{1}{2} a_t \Delta T^2 \quad (6.5)$$

respectively [48]. However, the exact value of a and w are not available, and noise and bias would change them dramatically. Hence, in order to get the best orientation and position estimation from the IMU measurements, the bias and measurement noise needs to be modeled perfectly, and the noisy measurements ($\hat{a}_t, \hat{w}_t$) have to transfer to real measurements (a_t, w_t), which is a difficult task.

6.3 Estimating the Length of Rotation and Position Changes with a Single Particular IMU

To the best of our knowledge this section is the first work to examine the network architectures researchers introduced to solve the problem of Inertial Odometry, and in addition, it proposes a new architecture that outperforms all existing networks. Evaluating other Inertial Odometry networks for the first time in this section is critical to understand the behavior of different networks and analyze their architecture. The proposed network has a triple channel architecture to discover a suitable feature representation from the accelerometer, gyroscope and also the sampling time respectively. The noise model also included in the input layer, which makes the network robust to noise and outliers. To highlight, contributions of this section are,

- Introducing a novel triple-channel Long Short-Term Memory (LSTM) network architecture according to different types of measurements of IMU.
- Using IMU sampling time (time interval between two consecutive IMU readings) as another input of the inertial network for the first time, which is a critical input for estimating sensor movement from IMU measurements.
- Considering the noise model of the IMU sensor in the formulation of the deep inertial odometry network for the first time, and modeling its behavior with a specific Gaussian noise layer for each channel of the proposed network architecture.
- Splitting IMU measurements to a specific window and learning a sequential network for the split based on different approaches are discussed in this section.
- Discussing and evaluating different inertial odometry network architectures used by visual-inertial odometry methods for autonomous vehicles and smart-phones and comparing their performance, pros and cons.

6.3.1 Proposed Method

IMU suffers from measurement noise and is not that reliable to handle odometry alone. In order to implement a robust inertial odometry system, it is essential to analyze the inertial networks and find a structure that can outperform existing inertial odometry techniques. This section proposes a new deep inertial odometry network architecture based on IMU SINS model (refer to Chapter 2 Section 2.5)

to output the magnitude of direction change and scale of movement for a set of IMU readings.

Since values of the accelerometer are in $\frac{\text{meter}}{\text{second}^2}$ unit and gyroscope measurement are in $\frac{\text{radian}}{\text{second}}$ unit, type and distribution of these two observations are different, and concatenation of these two different vectors in the input space is a wrong approach. Normalizing the vectors in the input space and concatenating them is another approach to discuss but since the noise model for each of these measurements is different, concatenation of measures in the input space after normalization prevents the network to generalize well and learn the noise model.

There exist some sampling delay or missed IMU information in IMU readings which affects the position estimation. Hence, it is essential to consider the time required to read the next IMU information or sampling time (ΔT) as an additional input. The mathematical model of IMU also demonstrates that time of reading IMU measurements from sensor plays a critical role in estimating sensor movement. Hence, the sampling time, in nanoseconds, needs to be considered as another individual input in another channel; this network is the first network that considers the sampling time as an input.

This work, as the first work in the area of inertial odometry, uses slow fusion to follow IMU SINS model and proposes a triple-channel network architecture. The slow fusion model fuses global information (feature representations) extracted from different input channels in higher layers [72]. Two of the channels learn a feature representation by a specific dimension for two different IMU measurement units (accelerometer and gyroscope), and the third channel learns a feature representation for the sampling time. Using slow fusion helps to extract meaningful feature representations from each of the inputs and the system can analyze the information from each source in the first step; After retrieving correlated information in different channels, fusing information assists to achieve better results. SINS workflow also examines inputs in the first steps before fusing the measurements. Figure 6.1 illustrates the architecture of the proposed Inertial Odometry network.

A notable mathematical point for IMU is the possibility of modeling each of accelerometer and gyroscope measurement noises with an additive zero-mean Gaussian noise with different variances (σ_a^2 for accelerometer and σ_g^2 for gyroscope).

Figure 6.1: Proposed AbolDeepIO Network Architecture ($N1$, $N2$, and $N3$ are the number of LSTM layers used for accelerometer, gyroscope and sampling time inputs respectively. $N4$ is the number of LSTM layers used after concatenation of features.)

Figure 6.2: Windowing Approach used by AbolDeepIO

Probability density function (PDF) analysis applies to the IMU error over time [120] to extract σ_a^2 and σ_g^2 variables for a specific IMU. IMU providers usually provide these parameters, and their values are provided in each IMU data-sheet. Using a random noise layer, with different variances for each type of channels' input, helps network to inspect noisy observations in the training phase and get robust to noise. This layer randomly adds Gaussian noise to the input data in the training phase.

It is also important to note that the accelerometer and gyroscope inputs are $(N, 3)$ vectors and the sampling time is a $(N, 1)$ vector, where N is the size of windowing. Figure 6.2 demonstrates the procedure of windowing approach. In this approach, IMU readings are split to windows of size N . For instance, if the

IMU frequency is 100Hz and windowing size is 10, the 100 IMU information read in one second will divide into ten windows where each contains ten consecutive IMU readings. Since IMU displacement at time t is dependent on its previous and future acceleration and gyroscope information, a window applies to the IMU measurements to handle this relationship. Due to error propagation, in each window, network learns to output displacement of IMU position and orientation for the time N . Windowing approach exists in traditional works like OKVIS [87], but in contrast to the previous works, this work considers one IMU measurement from the future (measurement number $N + 1$ which have some effects on its previous measurements) inside the window. It means when windowing size is N , the information obtained at time $N + 1$ also uses, and the additional input is the information of the next consecutive IMU reading, and the IMU displacement at time N is the desired goal. Since IMU frequency is high, the system can still work in real-time and one reading from future is not harmful to the system. In Figure 6.2, the global model of the windowing approach is demonstrated, and the output for node number N , among $N + 1$ nodes, is selected.

After applying additive zero-mean Gaussian noise on both accelerometer and gyroscope measurements, independent LSTM layers (N_1 , N_2 , and N_3) are designed for each input channel. The LSTM layers are learning to extract sequential feature representation from a set of observed data. Besides, due to the high variance of noise in the IMU measurements, it is essential to keep the network from over-fitting and so dropout [153] is used in each layer. Value of dropout, to randomly drop one cell in each LSTM layer is discussed in the experiments and results section.

After moving from input space towards the LSTM layers, the network learns to map IMU measurements and the sampling time from input space to the feature space. Hence, in the second step of the network architecture, feature information extracted from accelerometer, gyroscope and sampling time can combine by concatenating features next to each other. Concatenation in this step is the concatenation in the feature space, and the network itself learns to move different input values into the same domain feature space for concatenation.

After concatenating features together, N_4 number of LSTM layers come along

next to each other to map this high dimension feature vector to scale of displacement of IMU translation and rotation. The outputs of our network are ΔL and $\Delta\theta$ which show the magnitude of translation and rotation changes of the sensor in 3D axis for consecutive IMU readings respectively. Hence, for consecutive IMU readings, in a window with the size of N , the outputs of the network (ΔL and $\Delta\theta$) defines as

$$\Delta L_t = \sum_{i=0}^t \sqrt{dp_{x_i}^2 + dp_{y_i}^2 + dp_{z_i}^2} \quad (6.6)$$

$$\Delta\theta_t = \sum_{i=0}^t \sqrt{dq_{w_i}^2 + dq_{x_i}^2 + dq_{y_i}^2 + dq_{z_i}^2} \quad (6.7)$$

where $t = N$ (t represents the time step in the window and is in the range of $[0, N]$). In these equations, dp_{x_i} , dp_{y_i} and dp_{z_i} are the sensor displacements in 3D Cartesian coordinates and dq_{w_i} , dq_{x_i} , dq_{y_i} and dq_{z_i} are the sensor displacements in quaternion coordinates between two consecutive IMU measurements at time step i . Modeling learning procedure in this manner is because IMU measurements are relative to the base node and not the world coordinate, and so it is not possible to design the network to output and learn the exact x , y and z measurements directly. Hence, the inputs and outputs of our model are

$$(\hat{a}, \hat{w}, \Delta T)_{N*7} \xrightarrow{F_p} (\Delta L, \Delta\theta) \quad (6.8)$$

where $\hat{a}$, $\hat{w}$ and ΔT are inputs of the network F with parameters and weights p , and the outputs are ΔL and $\Delta\theta$ (N shows the size of windowing, and 7 is the size of the input: 3 for accelerometer, 3 for gyroscope and one for sampling time). $\hat{a}$ and $\hat{w}$ are representing the accelerometer and gyroscope measurements.

To learn output from a sequence of data points, defining a proper loss function, and choosing a beneficial learning approach plays a critical role. In order to solve this problem, finding optimum parameters (p) of network F is required which can be solved by minimizing the *loss* function and solving

$$p^* = \underset{p}{\operatorname{argmin}} \operatorname{loss}(F_p(\hat{a}, \hat{w}, \Delta T), (\Delta L, \Delta\theta)) \quad (6.9)$$

where $F_p(\hat{a}, \hat{w}, \Delta T)$ represents outputs of the network F_p with inputs $\hat{a}$, $\hat{w}$ and ΔT .

The *loss* function to be minimized is the error between estimated measurements from F_p , and the ground truth ΔL and $\Delta\theta$. The magnitude of ΔL and $\Delta\theta$ are not in the same unit and require to have equal effect in the *loss* function. Hence, to minimize the deep network, *loss* function is defined as

$$loss = \lambda_1 \sum_{i=k}^N \sqrt{(\Delta\bar{L}_i - \Delta L_i)^2} + \lambda_2 \sum_{i=k}^N \sqrt{(\Delta\bar{\theta}_i - \Delta\theta_i)^2} \quad (6.10)$$

which balances error between both magnitude of sensor displacement and orientation change using tuning parameters λ_1 and λ_2 . In this equation $\Delta\bar{L}_i$, ΔL_i , $\Delta\bar{\theta}_i$, and $\Delta\theta_i$ stand for estimated sensor displacement, exact sensor displacement, measured sensor quaternion change and exact sensor quaternion change at time step i respectively. The index i iterates over the window and measures the error at each time step. Figure 6.3 illustrates the two different approaches to train sequential LSTMs on the IMU measurements window. The first approach shown on the left is many-to-one and the second approach on the right is many-to-many. In the first approach, only the last output is essential, and all other outputs of the network and their error are not necessary to us ($k = N$). While this approach helps the network to focus on its desired output, it does not benefit from local information for each time step. The second approach, many-to-many, benefit from both local and desired global goal information ($k = 0$). Performance of each of these approaches is analyzed in experiments and results and proves that many-to-many approach performs best.

6.3.2 Experiments and Results

This section examines the proposed Deep Inertial Odometry framework on a public MAV dataset [18]. The ASL EuRoC MAV Dataset [18] consists of data from two different places. The MAV holds two pairs of cameras and one MEMS IMU ADIS16448 which performs at a frequency of 200 Hz. To produce an accurate Ground-Truth they have used both Vicon Motion Capture and Leica MS50 laser tracker. The implementation of the proposed method is under Keras [61] with TensorFlow [1] backend, and the network is trained on a Nvidia 1080 Ti GPU.

To evaluate the performance of the proposed method and compare it with the existing approaches, IONet [24] and Inertial Odometry part of VINet [30] trained

Figure 6.3: Different Approaches to Train Proposed Network Architecture According to The Window Observations

Table 6.1: Train and Test Splits of ASL EuRoC MAV Dataset [18]

#	Training Data (Number of IMU Measurements)	Testing Data (Number of IMU Measurements)
1	Machine Hall01 - Easy (36383)	Machine Hall02 - Easy (29993)
2	Machine Hall03 - Medium (26302)	Machine Hall04 - Difficult (19753)
3	Machine Hall05 - Difficult (22212)	Vicon Room 1 03 - Difficult (21500)
4	Vicon Room 1 02 - Medium (17100)	Vicon Room 2 02 - Medium (23490)
5	Vicon Room 2 01 - Easy (22800)	Vicon Room 1 01 - Easy (29120)
6	Vicon Room 2 03 - Difficult (23370)	-

on the same dataset along training the proposed method. The popular traditional method SINS which used by traditional visual-inertial methods also tested. EuRoC MAV Dataset consists of 11 recordings. To train a network on this dataset, it needs to split into training and testing components which shown in Table 6.1. Each training and testing dataset consists of raw IMU measurements of MAV while it flies. The number of IMU measurements used from each dataset is stated in front of each dataset in Table 6.1.

In order to prove the best architecture of the proposed approach, analyze the effect of each module, and compare its performance with other methods, seven different architecture of the proposed method are considered and shown in Table 6.2; $N1$, $N2$, $N3$, and $N4$ in this table are the number of layers in each part of the network according to Figure 6.1. AbolDeepIO1 is the reference network architecture and others are different with this network in one factor. It is also

Table 6.2: Properties of Different Network Architectures of the Proposed Method To Train and Evaluate

Network Name	Number of LSTM Layers and hidden cells in Step 1 (N1, N2, and N3)	Number of LSTM Layers and hidden cells in Step 2 (N4)	Include Sampling Time as Input	Dropout	Include Gaussian Noise Layer	Many-to-One or Many-to-Many Training Approach
AbolDeepIO1	100-150-100(N1=3) 100-150-100 (N2=3) 100 (N3=1)	150-100-50 (N4=3)	yes	0.25	yes	Many-to-Many
AbolDeepIO2	100-150-100 (N1=3) 100-150-100 (N2=3) 100 (N3=1)	150-100-50 (N4=3)	yes	0.35	yes	Many-to-Many
AbolDeepIO3	100-150-100 (N1=3) 100-150-100 (N2=3) 100 (N3=1)	150-100-50 (N4=3)	yes	0.15	yes	Many-to-Many
AbolDeepIO4	100-150-100(N1=3) 100-150-100 (N2=3) 100 (N3=1)	150-100-50 (N4=3)	yes	0.25	yes	Many-to-One
AbolDeepIO5	50-50-50 (N1=3) 50-50-50 (N2=3) 10 (N3=1)	100-10 (N4=2)	yes	0.25	yes	Many-to-Many
AbolDeepIO6	100-150-100(N1=3) 100-150-100 (N2=3) 0 (N3=0)	150-100-50 (N4=3)	no	0.25	yes	Many-to-Many
AbolDeepIO7	100-150-100(N1=3) 100-150-100 (N2=3) 100 (N3=1)	150-100-50 (N4=3)	yes	0.25	no	Many-to-Many

important to note that, networks are trained on the training split for two window sizes of 10 and 20 to see the effect of different window size. Besides, according to the ADIS16448 IMU datasheet, the σ_a^2 and σ_g^2 are 0.2 and 0.02 respectively for this type of IMU.

After training all the methods on the training split, the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) of predicting each of ΔL and $\Delta\theta$ values measures. Both MAE and RMSE show the average prediction error, but RMSE gives higher weight to the significant errors during averaging. Table 6.3 to Table 6.6, show the MAE/RMSE of each of the proposed and existing methods, on the MAV test datasets.

Results show that in most of the cases, MAE and RMSE of the proposed method are less than existing methods. It is important to notice that, a small error here, demonstrates a significant change of odometry result. Looking at the average MAE and RMSE on all test data (overall row) shows that there is about 20 percent improvement in the value of MAE. Regarding RMSE, the effect of considerable noise is not too much, and still, the RMSE of the proposed method is less than existing methods, and the proposed method does not have too much overestimation or underestimation.

Comparing the results of AbolDeepIO5 and AbolDeepIO1 architecture shows that reducing the number of hidden cells in each layer will affect the network and network cannot perform well. Hence, reducing the number of neurons of

AbolDeepIO will cause its accuracy reduction. However, the result of the network with less number of hidden cells is still better than existing methods. It is important to notify that there is a significant trade-off between a complex architecture and speed of the approach. It takes 7 and 9 milliseconds to process an IMU frame using the proposed method with the shallow and deep models respectively. This time for VINET and IONet are 100 and 6 milliseconds respectively. For SINS which uses only integral to solve the problem of odometry, it takes 1 millisecond. However, the error propagated over time for SINS, and other deep approaches are too large which makes the computational time of the proposed method beneficial for accurate odometry compare to existing solutions.

Impact of the sampling time input parameter on estimating odometry is another major thing that needs to be analyzed. AbolDeepIO6 have zero number of layers in $N3$ which results in not including sampling time as input. Comparing AbolDeepIO6 with AbolDeepIO1 shows that the network performs for about 5 percent better when the sampling time is the input of the network. To extract position change in the mathematical model of IMU integral uses which is dependent on time. Hence, using time of reading IMU measurements as input helps to improve the result by 5 percent. It also helps network to become robust to change of IMU frequency, which exists in the dataset.

While LSTMs are good for learning the relationship between consecutive data points, they cannot memorize things well and using dropout for LSTMs requires lots of consideration. To see the effect of dropout parameter, AbolDeepIO1 needs to compare with AbolDeepIO2 and AbolDeepIO3 which have higher and lower dropout value respectively. Comparing results show that using a higher dropout may help network to execute better but the lower value will make it weak. Hence, dropout value higher than 0.25 is sufficient for training this kind of network. It also needs to notice that, a large dropout value will also affect the results of the proposed approach and make other existing methods to perform better. Based on the experiments dropout between 0.25 to 0.35 is acceptable.

Another thought to compare is the training approach which could be many-to-many or many-to-one. To see the effect of using each of these approaches AbolDeepIO1 compares with AbolDeepIO4. Results show that the many-to-many

Table 6.3: MAE/RMSE of $\Delta\theta$ for Windowing Size of 20

Test Dataset#	Method									
	SINS	IONet	VINet	AbolDeepIO1	AbolDeepIO2	AbolDeepIO3	AbolDeepIO4	AbolDeepIO5	AbolDeepIO6	AbolDeepIO7
1	0.0084/0.0112	0.0049/0.0063	0.0039/0.0059	0.0029 /0.0046	0.0032/ 0.0041	0.0041/0.0072	0.0036/0.0050	0.0037/0.0056	0.0034/0.0051	0.0037/0.0053
2	0.0136/0.0899	0.0099/0.09	0.009/ 0.089	0.0076 /0.090	0.0077/0.0899	0.0090/0.0902	0.0080/0.0903	0.0082/0.0900	0.0082/0.0903	0.0081/0.0902
3	0.0360/0.2038	0.0325/0.2053	0.0344/0.2050	0.0305/0.2045	0.0314/0.2044	0.0343/ 0.2036	0.0300 /0.2046	0.0323/0.2038	0.0313/0.2038	0.0307/0.2046
4	0.0282/0.1342	0.0188/0.1345	0.0194/0.1348	0.0194/0.1345	0.0171/0.1335	0.0235/0.1336	0.0170 / 0.1335	0.0211/0.1338	0.0194/0.1338	0.0181/0.1339
5	0.0358/ 0.2478	0.0293/0.2485	0.0287/0.2484	0.0271/0.2487	0.0268 /0.2487	0.0295/0.2485	0.0281/0.2486	0.0281/0.2484	0.0276/0.2488	0.0281/0.2488
Overall	0.0242/ 0.1621	0.0188/0.1627	0.0187/0.1626	0.0172/0.1626	0.017 /0.1624	0.0197/0.1622	0.0171/0.1625	0.0183/0.1622	0.0177/0.1624	0.0175/0.1626

Table 6.4: MAE/RMSE of ΔL for Windowing Size of 20

Test Dataset#	Method									
	SINS	IONet	VINet	AbolDeepIO1	AbolDeepIO2	AbolDeepIO3	AbolDeepIO4	AbolDeepIO5	AbolDeepIO6	AbolDeepIO7
1	0.0359/0.0432	0.0237/0.0283	0.0235/0.0286	0.0186/0.0250	0.0190/0.0251	0.0191/0.0265	0.0185 / 0.0243	0.0198/0.0265	0.0235/0.0292	0.0187/0.0256
2	0.0815/0.1022	0.0495/0.0650	0.0589/0.0754	0.0293/0.0407	0.0347/0.0467	0.0290/0.0400	0.0300/0.0415	0.0305/0.0441	0.0589/0.0756	0.0271 / 0.0370
3	0.0618/0.0726	0.0360/0.0480	0.0349/0.0451	0.0286/0.0381	0.0265 / 0.0349	0.0287/0.0379	0.0290/0.0381	0.0289/0.0378	0.0330/0.0423	0.0281/0.0371
4	0.0596/0.0596	0.0311/ 0.0386	0.0333/0.0419	0.0320/0.0412	0.0316/0.0408	0.0332/0.0432	0.0319/0.0409	0.0300 /0.0391	0.0324/0.0407	0.0326/0.0423
5	0.0273/0.0328	0.0299/0.0359	0.0225/0.0273	0.0226/0.0287	0.0219 / 0.0275	0.0227/0.0308	0.0245/0.0315	0.0249/0.0313	0.0226/ 0.0267	0.0234/0.0317
Overall	0.0501/0.063	0.0329/0.0431	0.0328/0.0444	0.0255/ 0.0343	0.0259/0.0347	0.0258/0.0353	0.0261/0.0349	0.0262/0.0353	0.0323/0.0438	0.0254 /0.0345

approach can perform better than many-to-one because of benefiting from ground truth label of middle states.

Effect of the Gaussian noise layer also needs to be analyzed. To analyze the impact of this layer, AbolDeepIO1 compares with the AbolDeepIO7. Both models have the same structure, but AbolDeepIO7 do not include the Gaussian noise layer. As results show, the AbolDeepIO7 can outperform existing inertial odometry techniques significantly. Hence, the main impact of the improvements in the odometry come from the proposed triple-channel deep network architecture. The results show that the Gaussian noise layer can help to make network robust to noise (in terms of RMSE) and improve the accuracy of the network for approximately 2 percent.

Table 6.5: MAE/RMSE of $\Delta\theta$ for Windowing Size of 10

Test Dataset#	Method									
	SINS	IONet	VINet	AbolDeepIO1	AbolDeepIO2	AbolDeepIO3	AbolDeepIO4	AbolDeepIO5	AbolDeepIO6	AbolDeepIO7
1	0.0039/0.0052	0.0019/0.0025	0.0058/0.0063	0.0015/ 0.0020	0.0015/0.0022	0.0015 /0.0026	0.0018/0.0024	0.0022/0.0031	0.0016/0.0024	0.0019/0.0026
2	0.0065/0.0635	0.0046/0.0637	0.0073/0.0639	0.0036 /0.0636	0.0038/0.0636	0.0038/0.0636	0.0041/ 0.0635	0.0046/0.0636	0.0040/0.0637	0.0041/0.0637
3	0.0175/0.1445	0.0160/0.1450	0.0200/0.1458	0.0153/0.1447	0.0148/0.1447	0.0165/0.1446	0.0142 /0.1448	0.0153/0.1445	0.0151/ 0.1445	0.0156/0.1447
4	0.0133/0.0945	0.0095/0.0948	0.0127/0.0951	0.0086/ 0.0943	0.0084/0.0944	0.0103/0.0948	0.0081 /0.0944	0.0096/0.09436	0.0089/0.0944	0.0088/0.0944
5	0.0174/0.1723	0.0142/0.1725	0.0167/0.1725	0.0134 /0.1725	0.0136/0.1726	0.0138/0.1726	0.0141/0.1725	0.0147/ 0.1724	0.0139/0.1726	0.0141/0.1726
Overall	0.0116/0.1136	0.0090/0.1138	0.0123/0.1141	0.0084/0.1136	0.0083 /0.1137	0.0090/0.1137	0.0084/0.1137	0.0091/ 0.1136	0.0086/0.1136	0.0088/0.1137

Table 6.6: MAE/RMSE of ΔL for Windowing Size of 10

Test Dataset#	Method									
	SINS	IONet	VINet	AbolDeepIO1	AbolDeepIO2	AbolDeepIO3	AbolDeepIO4	AbolDeepIO5	AbolDeepIO6	AbolDeepIO7
1	0.0212/0.0251	0.0115/0.0140	0.0116/0.0143	0.0097/0.0128	0.0095/0.0125	0.0099/0.0132	0.0096/0.0126	0.0106/0.0139	0.0098/0.0129	0.0096/0.0131
2	0.0437/0.0544	0.0274/0.0350	0.0293/0.0376	0.0191/0.0259	0.0199/0.0265	0.0199/0.0271	0.0193/0.0264	0.0197/0.0266	0.0184/0.0248	0.0165/0.0227
3	0.0346/0.0399	0.0167/0.0218	0.0172/0.0222	0.0142/0.0182	0.0137/0.0176	0.0148/0.0189	0.0148/0.0191	0.0147/0.0187	0.0153/0.0193	0.0143/0.0187
4	0.0337/0.0387	0.0159/0.0199	0.0166/0.0209	0.0160/0.0203	0.0158/0.0202	0.0150/0.0192	0.0162/0.0207	0.0158/0.0202	0.0157/0.0200	0.0166/0.0213
5	0.0172/0.0198	0.0137/0.0169	0.0112/0.0135	0.0119/0.0146	0.0122/0.0150	0.0120/0.0146	0.0122/0.0153	0.0127/0.0155	0.0125/0.0153	0.0119/0.0163
Overall	0.0286/0.0358	0.0164/0.0217	0.0163/0.0221	0.0137/0.0182	0.0138/0.0183	0.0139/0.0185	0.014/0.0187	0.0143/0.0189	0.0139/0.0183	0.0134/0.0183

(a) Average ΔL Error Propagation Over Time (b) Average $\Delta \theta$ Error Propagation Over TimeFigure 6.4: Average of ΔL and $\Delta \theta$ error propagation over time for all datasets

Another feature to compare is the effect of windowing size on the results. To compare it and also see the effect of error propagation over time, the average of IMU estimated ΔL and $\Delta \theta$ error propagated over time for all dataset (for the specific time that all datasets have measurements) is plotted in Figure 6.4. Esti-

mated error for each time step is the average error of estimation at each time step for all datasets, plus the previous time step estimation error. In this manner, the error propagates over time to estimate the position and orientation of the IMU and effect of error propagation is obvious. As Figure 6.4 shows, the average error propagation for both scales of camera translation and rotation is minimum for the proposed network architecture. Besides, as illustrated, the proposed method with the window size of 20 performs better than window size of 10. Also, the error propagation for the proposed method, for IMU position displacement estimation, is almost linear over time which shows that with the existence of sudden motions of MAV and change of IMU frequency, the estimation error of the proposed architecture is almost linear and the model generalizes itself well in different situations. Since the structure of other deep networks is similar, they are propagating similar error over time, for IMU position displacement estimation, and there is not much difference in their error propagation.

6.4 Robust 3-D Orientation Estimation with a Single Particular IMU

In this section, for the first time to our knowledge, an end-to-end deep network architecture is proposed to estimate the full 3D orientation of flying robots with a single particular IMU in the quaternion coordinate. The contributions of this section are:

- proposing the *OriNet* deep network architecture to estimate the 3D orientation of a robot (world frame) over time, with the presence of a single particular IMU.
- defining terms of *state* and *local time window* space, which helps the network to transfer IMU measurements between the world frame and IMU body frame.
- sliding a local time window over nearby IMU measurements, and outputting various estimation for each time step by considering a different combination of adjacent IMU measurements. Fusing such estimations helps to model uncertainty and achieve a robust solution.
- proposing a training approach which moves the network from exploration of the robust weights to exploitation of the weights near the explored robust weights during training.

Figure 6.5: Challenges and proposed solutions for estimating 3D orientation with a single particular IMU

- proposing a novel method based on the Genetic Algorithm (GA) to calibrate a trained model with the bias of IMU in each execution.

The proposed network architecture is evaluated on the EuRoC Micro Aerial Vehicle (MAV) dataset [18], and the results show that it not only outperforms existing IO methods but also achieves a result better than VIO methods.

6.4.1 Proposed Architecture

This section presents a novel network architecture to estimate the full 3D orientation (yaw, pitch, and roll) in the world frame, with the presence of a single particular IMU. The frame that the orientation of the robot or particular object with connected IMU measures in that is called the world frame; for instance, if an IMU connects to a MAV and the orientation of MAV is required, MAV orientation creates the world frame. It is also important to mention that the estimated orientation is in the quaternion coordinates (as Q_4 where 4 stands for the size of the output, which is correspondent to the estimation of q_w , q_x , q_y and q_z sub-coordinates). Figure 6.5 demonstrates problems and solutions investigated in this section.

IMU measurements are considered in two spaces in this section, which are *state*

space, and *local time window space*. State space is the global state of the IMU over time, and once quaternion for the time of a particular IMU measurement is computed, a new (quaternion) state is created. In order to create a state space, the LSTM unit is applied on a local time window space with a size of N , which contains N continues IMU measurements.

In the beginning, the network is required to know the exact quaternion of the IMU in the initial state. The local time window starts from the IMU measurement number two, and the exact quaternion in the initial state creates the first state ($Q_{s=1}$). After processing the first local time window, it is shifted by one IMU measurement (starts from measurement number 3) and quaternion regarding the time step of the second IMU measurement is computed based on the outputs of the local time window before shifting, and the second quaternion state ($Q_{s=2}$) is created. From now onwards, index t refers to a time step in a local time window space ($1 \leq t \leq N$), and s demonstrates a state in the state space ($s \geq 1$).

OriNet consists of two parts, and the first part includes four LSTM channels where each get $\hat{w}$ of the current and previous time step, sampling time (time interval between two consecutive IMU readings), T , at each time step in a local time window, and prior state quaternion (Q_{s-1}), as input ($\hat{w}$ is representing the gyroscope measurements). Feature representations learned in these channels are fused, by using the concatenation of features in the slow fusion gate. Then, the fused feature representation moves to another LSTM and Fully Connected (FC) gates to output the displacement of quaternion sub-coordinates at any time step, in the local window, relative to its previous time step ($\Delta Q_t = Q_t - Q_{t-1}$). Figure 6.6 shows the proposed framework, which is discussed in this section.

It is essential to mention that the accelerometer measurements are not considered as input in our system; Because, based on our experiments, it reduces the performance of the network. While an accelerometer can help to reduce noise and extract robust orientation when IMU is stationary, it can reduce the overall performance. Designing a zero-velocity detection system and benefiting from accelerometer information when IMU is in a stationary state is a good approach which we will focus on in the future.

Similar to our previous work [44], mentioned in the previous section, an additive

Figure 6.6: The procedure of the proposed *OriNet* module (b_1, b_2 and b_3 are the gyroscope bias, b_w)

zero-mean gaussian noise layer with the variance of σ_g^2 is applied in the input layer. This layer adds noise to the input in the training time, and so makes the network robust to measurement noise. The value of σ_g^2 is measured by the Probability Density Function (PDF) analysis and is usually available in the IMU datasheets.

Since the gyroscope measures angular velocity relative to the world in the IMU body frame, knowing the latest state quaternion, in the world frame, is essential in order to estimate the next state quaternion (As IMU rotates the IMU body frame changes, and the angular velocity effects differently on the world frame quaternion). In recent deep learning-based VIO frameworks, the impact of previous state quaternion on the gyroscope and accelerometer measurements is ignored, and only the gyroscope and accelerometer measurements at each time step are passed to the LSTM units. One of the main issues that made scientists ignore the previous state quaternion is the structure of LSTM, which does not have a model to accept the latest state info as input; It is due to the lack of state definition which results in considering a local window of data as the only inputs of an LSTM unit. The previous state's output is essential for estimation of the current state, and so, in this work, we benefit from the definition of state and local time window to solve this issue, and previous state quaternion (Q_{s-1}) is considered as the input of the network. In this manner, it is assumed that the initial IMU quaternion ($Q_{s=1}$) is available at the beginning of each test execution. Hence, one limitation of our work is its requirement of the IMU quaternion in the world frame at the first time step in test time, which is not challenging, and is equivalent of starting algorithm from

a static quaternion at all the time; for instance you should always put your drone (world frame) which contains IMU on the flat ground looking North, and run the algorithm.

OriNet is applied on a local time window with N measurements. Each time window outputs the displacement of quaternion sub-coordinates (Δq_w , Δq_x , Δq_y , and Δq_z) for each of the time steps, among N , relative to its previous time step. As the time window shifts by one, the computed ΔQ , regarding the first IMU measurement ($t = 1$) in the previous window (state $s - 1$), which is not in the new window (state s), sums by its prior state quaternion (Q_{s-2}), to measure and create the new quaternion state (Q_{s-1}), which is going to be the input for the current state (s) time window. The new quaternion state always can be achieved by following similar steps and summing the displacement of the quaternion, for the first measurement dropped from local time window when window shifts, with its prior state quaternion ($Q_{s-1} = \Delta Q_{s-1} + Q_{s-2}$). In the training time, the ground truth quaternions are used for training, and in the test time, only the first quaternion is required to be known. It is also important to note that, since in the quaternion coordinate $Q = -Q$, in this work, the network is trained by considering both Q and $-Q$, and outputting the smooth Q over time.

Since the local time window shifts one by one, there will be N estimations for each time step. For instance, if $N = 10$, there exists one ΔQ estimation for measurement number 11 when the window begins from measurement number two, and we have another one when we shift the window by one to compute the first estimation for the measurement number 12. It is possible to benefit from all of these estimations and probabilistically measure the final output at each time step. The actual output of the network should be the averaged value of all estimations plus or minus an error term ζ , which can be modeled as

$$\Delta Q_s = \frac{1}{N} \sum_{i=1}^N \Delta Q_{s_i} \pm \zeta \quad (6.11)$$

where ΔQ_s is the quaternion displacement at state s relative to its previous state, and ΔQ_{s_i} is the existing estimations for that state. Figure 6.7 is showing the average of all estimations, and ζ for sub-quaternion of q_x in a short period. As it shows, finding the value of ζ is an important requirement for an accurate and

Figure 6.7: Importance of ζ which should be handled by ΔQ_{Net}

robust estimation. In this manner, to control the effect of ζ , a two-layered Multi-Layer Perceptron (MLP) is designed, which gets ΔQ_{s_i} as input and attempts to regress them to the desired ΔQ_s . We define the first term of equation 6.11, which performs averaging on the N estimations, as the averaging gate with the outcome of ΔQ_{Avg} , and the network which helps to handle ζ is ΔQ_{Net} (with the output of ΔQ_{Net}).

The final output of the network, called ΔQ_w , is computed by weighting gate which is defined as

$$\Delta Q_w = \frac{w_1 \Delta Q_{Avg} + w_2 \Delta Q_{Net}}{w_1 + w_2} \quad (6.12)$$

where w_1 and w_2 are weights which obtain by tuning the ΔQ_w on the training data. The loss function regarding the first part of the network, $Loss_1$ shown in Fig. 6.6, is defined as the Mean Square Error (MSE) of the estimated and Ground Truth values regarding the displacement of quaternion sub-coordinates for all measurements that exist in a local time window. Besides, the loss function of ΔQ_{Net} , $Loss_2$ in Fig. 6.6, is the MSE of the Ground Truth and the estimated values regarding the displacement of quaternion sub-coordinates for a particular time.

6.4.2 From Large to Small Dropout Training Approach

Numerous studies have been tried to find the best way of training deep neural networks to avoid overfitting, find robust weights, and solve regression problems[2]. In [44], we have shown that using dropout can help inertial networks to become robust to noise and perform well. In this section, a new methodology is applied, and the network starts training with a dropout of 0.25 in LSTM layers at the

beginning. As time passes, and the loss gets into a stationary state for a short period (i.e. 400 epochs), dropout is reduced by 20%, and the training procedure continues until reaching a new stationary state again. This approach resumes until dropout reaches zero value.

The proposed approach helps the network to learn the robust weights when the dropout is high, which makes network robust to noise. The first stages of training can be referred to as exploration steps, in which the network tries to find the best robust weights without overfitting. As time passes and dropout reduces, the network moves to the exploitation steps and tries to find the best weights nearby the previously computed robust weights, which keeps the network robust. The results show that this approach helps the network to be robust to noise and also achieve better performance.

6.4.3 OriNet-IMU Bias Calibration

IMU suffers from bias and measurement noise. The network gets robust to the noise by benefiting from the input noise layer and the proposed training approach. In order to handle bias, we ignored the small changes of bias over time and assumed it as a fixed value, which decides at the beginning of each execution. We utilized the Genetic Algorithm (GA)[35] to optimize a fitness function at the beginning of each test, which computes the bias of IMU. In this manner, it is assumed that IMU is going to be stationary in the first 5 seconds (can have small rotations withing a fixed trajectory for better calibration). The IMU measurements in the stationary period are used to calibrate the IMU with the deep network and compute the bias.

Considering *OriNet* as F , which gets angular velocity $\hat{w}$ and sampling time (T) for all time steps in a local time window, and previous state quaternion (Q_{s-1}) as input, and outputs the Q_s , the fitness function to be optimized is the MSE of the actual quaternion (Q'_s) and estimated quaternion in duration of 5 seconds, at the beginning of the execution. The problem can be formulated as

$$\min_{b_w} \{ (F(\hat{w}_t + b_w, \hat{w}_{t-1} + b_w, T, Q_{s-1}) - Q'_s)^2 \} \quad (6.13)$$

where b_w is the bias parameter that needs to be investigated by the GA. Hence, a population of 20 chromosomes, with the dimensionality of 1×3 are randomly

initialized in the range of $[-0.5, 0.5]$; Each chromosome has three values corresponding to the 3 values measured from the gyroscope. During the bias estimation process, the ΔQ_{Net} gate is only considered for computing the quaternion as the final output, and other outputs (ΔQ_w and ΔQ_{Avg}) are ignored. In this manner, the ΔQ_{Net} bias is also considered, and the bias of the gyroscope is computed based on all networks.

6.4.4 Experiments and Results

6.4.4.1 Testing Environment

This section evaluates the proposed method on the EuRoC MAV dataset[18]. This dataset captured images with stereo cameras at 20 frames per second (FPS) and recorded IMU measurements at 200 Hz, from the time MAV is in a stationary state until it takes off, flies and lands on its initial position. They have used a particular IMU in all scenarios. This dataset is a challenging dataset for the focused task because the MAV has lots of sudden motions during its flight, which makes it difficult to estimate MAV orientation at each state. Similar to the previous section, the dataset is split into training and testing sets which are shown in Table 6.1, and the number of neurons in each layer of the proposed *OriNet* architecture is shown in Table 6.7. *OriNet* can evaluate a local time window in 3 milliseconds, by benefitting from cuDNN implementation of LSTM, which is a significant improvement compared to 6-9 milliseconds execution time for other deep learning-based approaches. The evaluation goes into detail to understand the effect of each of the proposed modules, which is extremely important. The size of the time window, N , is considered as 11 in all experiments. It is also important to mention that in all experiments, similar to [104, 166] and the single state alignment approach in [182], the frame of the estimated quaternions is aligned with the Ground Truth based on the estimated and Ground Truth values at the first state. After aligning the first state quaternion, based on the quaternion displacement between next time steps and the aligned quaternion (transform in the first state), the new aligned states are created.

In order to analyze the impact of the proposed modules, five different variants of the proposed framework are considered, trained, and tested. The summary of

Table 6.7: Number of units in each layer of the *OriNet*

Layer			
LSTM before SLOW FUSION	LSTM after SLOW FUSION	Fully Connected (FC)	ΔQ_{Net}
100 (Bidirectional Layer)	250-100 (Two Bidirectional Layers)	20-4 (Two Time Distributed Dense Layers)	150-4 (Two Dense Layers)

the considered networks is shown in Table 6.8. The performance of each variant is compared with other variants and existing deep learning-based IO frameworks, as well as the traditional SINS framework. Existing deep IO solutions are not estimating the 3D orientation with the presence of single IMU, and are most capable of measuring the length of quaternion displacement in a local time window; Hence, in this work, similar to [44], the length of quaternion displacement estimation in a local time window is computed from the output of our method and compared with others in terms of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Besides, the performance of the proposed method for estimating the full quaternion is compared with the traditional inertial odometry techniques which are capable of tracking the orientation along the path. Furthermore, the performance is also compared with the VINS-MONO, VIO framework, which is diagnosed as the top Monocular VIO framework in [36].

Table 6.8: Considered variants of the proposed method

#	Bias Calibration	Average Gate	ΔQ_{Net} Gate	Weighting Gate	Proposed Training Approach
OriNet1	✓	✗	✓	✗	✓
OriNet2	✗	✗	✓	✗	✓
OriNet3	✗	✓	✗	✗	✓
OriNet4	✗	✗	✓	✗	✗
OriNet5	✓	✓	✓	✓	✓

6.4.4.2 Comparing IO Methods in terms of Quaternion Displacement Estimation and Analyzing Modules

This section analyzes the proposed modules by comparing the performance of the considered variants in terms of length of quaternion displacement estimation in a local time window. Length of quaternion displacement for a local time window is defined as

$$|\Delta Q| = \sum_{i=1}^N \sqrt{\Delta q_{w_i}^2 + \Delta q_{x_i}^2 + \Delta q_{y_i}^2 + \Delta q_{z_i}^2} \quad (6.14)$$

where Δq_{w_i} , Δq_{x_i} , Δq_{y_i} , and Δq_{z_i} are sub-quaternions displacement in between two consecutive IMU readings. The MAE and RMSE of different networks on the test split are shown in Table 6.9; as the table shows, the proposed method can outperform the existing IO techniques, as well as inertial part of VINET, significantly. The analysis of the results and proposed modules come in the following.

Table 6.9: Comparing variants of the proposed method with others in terms of MAE/RMSE of $|\Delta Q|$ estimation

Test Dataset#	Methods									
	SINS	IONet	VINet	AbolDeepIO	OriNet1	OriNet2	OriNet3	OriNet4	OriNet5	
1	0.0039/0.0052	0.0019/0.0025	0.0058/0.0063	0.0015/0.0020	5.98e-05/8.39e-05	6.15e-05/8.70e-05	5.42e-05/7.92e-05	0.0013/0.0026	3.94e-05/0.000061	
2	0.0065/0.0635	0.0046/0.0637	0.0073/0.0639	0.0036/0.0636	3.85e-05/7.87e-05	5.25e-05/9.71e-05	4.28e-05/9.27e-05	0.0014/0.0024	5.1e-05/8.69e-05	
3	0.0175/0.1445	0.0160/0.1450	0.0200/0.1458	0.0153/0.1447	0.00016/0.00024	0.00014/0.00023	0.00014/0.00023	0.00016/0.00028	0.00015/0.00024	
4	0.0133/0.0945	0.0095/0.0948	0.0127/0.0951	0.0086/0.0943	0.00014/0.00024	0.00015/0.00025	0.00018/0.00029	0.00014/0.00028	0.00014/0.00023	
5	0.0174/0.1723	0.0142/0.1725	0.0167/0.1725	0.0134/0.1725	8.29e-05/0.00010	9.60e-05/0.00012	7.40e-05/0.000099	0.00012/0.00015	7.07e-05/9.33e-05	
Overall	0.0116/0.1136	0.0090/0.1138	0.0123/0.1141	0.0084/0.1136	9.38e-05/0.000159	9.79e-05/0.000165	9.49e-05/0.000172	0.00042/0.00130	8.61e-05/0.000154	

Impact of using the previous state quaternion: *OriNet4* is trained from scratch without dropout, and its main differences with *AbolDeepIO*[44] are: 1) it benefits from the previous state quaternion (Q_{s-1}), 2) it outputs the ΔQ_{Net} which is an aggregation of various estimations for a particular time step. Comparing the results shows that, benefiting from the quaternion and aggregating the estimations with ΔQ_{Net} improves the performance of the network by 95%. This significant improvement is mainly because IMU measurements are in the IMU body frame, and passing the quaternion state, in the world frame, as another input to the network is critical. Having Q_{s-1} at the beginning of each local time window helps the network to transfer IMU measurements between world and IMU body frame, and correctly propagate measurements in order to estimate the quaternion in the world frame. It is essential to transfer IMU measurements between the world and IMU frames because, for instance, the IMU measurements regarding the yaw angle of the world frame are obviously different when IMU is at 90° and 0° (it is also different in between). Besides, similar to an EKF, the network can learn to perform an analysis and do the update process based on the previous state measurements and current values, by giving more weights to the previous states in some situations and learning to apply some filters. These facts are forgotten, either by deep learning-based IO or VIO methods like VINet[30], and the proposed approach can improve the performance of deep learning-based VIO systems significantly.

Impact of the proposed training approach: *OriNet4* is trained from scratch, without using the proposed training approach and dropout; on the other hand, *OriNet2* has the same configuration as *OriNet4*, but it is trained with the proposed training approach. It can be seen that the proposed training approach results in a 76% improvement in MAE. When the dropout is high, it is hard to reach the optimum loss and overfit; Hence, learning weights in the presence of high dropout at the first stages, results in a robust model which is not overfitted. However, having a high dropout value may avoid the network to reach its optimum result on validation and test set. Benefiting from the proposed method and reducing the dropout value once the model achieved a stationary state helps to avoid overfitting, and at the same time, it helps in reaching the optimum model, which is robust and gives the best output on the validation and test sets.

Figure 6.8 is illustrating the best validation loss, during training the network with and without using the proposed training approach, and verifies our discussion on the effect of the proposed training approach. As can be seen, the proposed training approach cannot reach the optimum validation loss in the early stages, and it moves from exploration of the robust weights in the early stages to the exploitation of the weights in the later ones. Besides, comparing the validation loss of the proposed method, by the models trained from scratch with different dropout values, shows that the proposed method can achieve a better validation loss and distance between their validation loss increases by getting into the exploitation steps.

Importance of the ΔQ_{Net} , Average, and Weighting Gates: In order to analyze the effect of the three gates, *OriNet2* and *OriNet3* need to be compared in the beginning to see the impact of each of the ΔQ_{Net} and averaging gates. Comparing the overall result shows that using the averaging gate, as the final output, results in better performance, in terms of MAE. However, the RMSE of the ΔQ_{Net} is better. RMSE gives higher weight to the significant errors during averaging, and so the ΔQ_{Net} has less significant errors compared to average. As illustrated in Fig. 6.9, the most active neuron in ΔQ_{Net} gives high weight to almost all quaternion displacement estimations, which results in the near average estimate

Figure 6.8: Brown line shows the best validation loss obtained by training the network from scratch and using the dropout value indicated on the horizontal axis. The blue line shows the best validation loss achieved by training the network with the proposed training approach; The proposed training method reduces the dropout during training, and so the utilized dropout at each training epoch is indicated on the horizontal axis.

Figure 6.9: Weights of most active neurons in ΔQ_{Net} input space

of all outputs, and other neurons are considering to perform a probabilistic analysis based on various estimations. Considering other neurons shows that the ΔQ_{Net} trusts on the estimations 9-11 more; such estimations are the ones estimated when there is a small distance between the start of the local time window and state estimation. For instance, estimation 11 is for the time that the window starts right at the time of measurement ($\Delta Q_{t=1}$ considers from the window), and for the estimation 10, the window starts from one measurement behind the state we are considering ($\Delta Q_{t=2}$ considers from the window).

Weighting gate helps to benefit from the output of the Average and ΔQ_{Net} gates simultaneously and reach less error with the small effect of significant errors. Comparing the results of *OriNet5* with *OriNet1* shows the considerable impact of the weighting gate, compared to using ΔQ_{Net} , which results in 8% improvement

Figure 6.10: Best and mean fitness value for the test dataset #4

in MAE. Besides, the network has the least RMSE. It is essential to notice that, when the weighting gate is on, the ΔQ_{Net} and averaging gates also need to be on, because the weighing gate uses the output of these two gates; but at the end, the output of the weighing gate only considers as the final output.

Importaance of OriNet-IMU bias calibration: *OriNet1* and *OriNet2* have a same configuration, but the *OriNet1* performs bias calibration at the beginning of each test set. Comparing the results of *OriNet1* and *OriNet2* demonstrates a 4% improvement, by performing bias calibration, which shows the importance of this module for estimating the bias of IMU in the test time. Figure 6.10 illustrates the fitness value achieved in different generations, and as it can be seen, the optimum value is almost achieved at 50th generation. The results show that the biases are in the range of $[-0.0019, 0.0027]$ in all test sets, and so it is possible to change the search domain to $[-0.005, 0.005]$ and find biases faster (within 20 generations).

6.4.4.3 Comparing Full Quaternion Estimation Capability of OriNet with IO and VIO Frameworks

This section compares the performance of the proposed method with the VINS-MONO[126] VIO method and two other IO techniques (Madgwick et al. [104] and Complementary Filter[166]), which are capable of extracting the full 3D quater-

nion along time. Estimations at times that image is captured and VINS-MONO outputs the orientation are only considered. The MAE of estimating orientation in each of the quaternion subspaces, along the moving path, is shown in Table 6.10. As it shows, the proposed method can outperform IO methods significantly. Besides, considering $|Q|$, which demonstrates the averaged length of error, shows that the proposed method can approximately outperform VINS-MONO by 30%. Furthermore, the quaternion distance (rotation required to get from one quaternion to another) between two quaternions p and q is also defined as [70]

$$\theta_D = 2 \arccos(|\langle p, q \rangle|) \quad (6.15)$$

where

$$\langle p, q \rangle = p_1 * q_1 + p_2 * q_2 + p_3 * q_3 + p_4 * q_4 \quad (6.16)$$

and $|\cdot|$ is the modulus function. The averaged quaternion distance between estimated and Ground Truth quaternion over each test dataset ($\bar{\theta}_D$) is computed and reported. As shown in Table 6.10, the proposed method has achieved the least quaternion distance compared to others, and it beats the VINS-MONO by approximately 28%. The Estimated and Ground Truth quaternion of the test dataset #4 is shown in Fig. 6.11. As it can be seen, *OriNet* can perform robustly during high motions on q_x and q_z estimations, while the VINS-MONO over-fits and under-fits the estimations.

Table 6.10: Comparing OriNet full quaternion estimation capability with IO and VIO frameworks. q_w , q_x , q_y , and q_z columns refer to the MAE in each coordinate, $|Q|$ is the mean length of the error, and $\bar{\theta}_D$ is the mean distance between the estimated and ground-truth quaternions in degrees.

Test Dataset #	VINS-MONO[126]						OriNet						Complementary Filter[166]						Madgwick et al.[104]					
	q_w	q_x	q_y	q_z	$ Q $	$\bar{\theta}_D$	q_w	q_x	q_y	q_z	$ Q $	$\bar{\theta}_D$	q_w	q_x	q_y	q_z	$ Q $	$\bar{\theta}_D$	q_w	q_x	q_y	q_z	$ Q $	$\bar{\theta}_D$
1	0.0092	0.0817	0.0135	0.0647	0.1068	12.2640	0.0121	0.0233	0.0073	0.0339	0.0447	5.1242	0.0152	0.0333	0.0177	0.0331	0.0564	6.5376	0.2122	0.3234	0.3155	0.2209	0.5459	70.4467
2	0.0175	0.0141	0.0270	0.0174	0.0438	5.0389	0.0083	0.0167	0.0604	0.0167	0.0678	7.7748	0.0514	0.0715	0.0427	0.0530	0.1186	13.7461	0.2410	0.115	0.3537	0.0757	0.4611	54.0868
3	0.0304	0.0360	0.0497	0.0234	0.0802	9.2405	0.0381	0.0567	0.0632	0.0424	0.1152	13.2222	0.0294	0.0345	0.0472	0.0230	0.0955	9.028	0.2801	0.7115	0.6526	0.2119	1.0791	98.2662
4	0.01435	0.1460	0.0180	0.1567	0.2191	25.2119	0.0163	0.0178	0.0439	0.0585	0.0836	9.5920	0.2222	0.2586	0.3407	0.1772	0.5670	66.532	0.1398	0.2006	0.2071	0.2167	0.4404	51.1945
5	0.0054	0.0083	0.0103	0.0141	0.0223	2.5728	0.0123	0.0128	0.0215	0.0258	0.0436	5.0076	0.1677	0.1445	0.2322	0.1216	0.3653	42.3221	0.2493	0.2275	0.3371	0.1541	0.5296	62.7062
Overall	0.0141	0.0574	0.0218	0.0549	0.0926	10.6616	0.0166	0.0243	0.0353	0.0350	0.0670	7.6961	0.0970	0.1069	0.1359	0.0817	0.2386	27.4691	0.2237	0.3089	0.3632	0.1791	0.5977	67.0723

Figure 6.11: Estimated and exact quaternion of test dataset #4

6.5 Concluding Remarks

This chapter, firstly, presents a novel triple channel deep neural network architecture for the problem of Inertial Odometry. The proposed network architecture learns to extract feature representations from accelerometer, gyroscope and time interval between two consecutive IMU readings (sampling time), and concatenates feature representations in the feature space to absorb desired output. Results confirm that the proposed method outperforms all the existing techniques. The proposed architecture includes a Gaussian noise gate after reading inputs from IMU and simulates the effect of noise on the IMU readings in the training phase which makes it robust to noise and improves its generalization ability. It also shows that including sampling time as an input improves MAE for about 5 percent and the network is robust to change of IMU frequency. Windowing is another technique used in this work, and it shows that using a larger window is better. However, in the future, this window can become adaptive and be selected according to the type of data.

Furthermore, this chapter proposed a deep learning-based framework for estimating the 3D orientation of flying robots in quaternion coordinates. The proposed method is compared with existing visual-inertial navigation systems, as well as inertial methods. The results show the notable power of the proposed method compared to existing approaches. The proposed framework benefits from the pre-

vious state information and aggregates various estimations achieved for each state, while shifting the local time window, to find the output robustly. Besides, a novel training approach is used, which forces the network to do the exploration of the best robust weights, which are not overfitted, in the first training stages, and move to the exploitation of the weights in later stages. A Genetic Algorithm based approach is also proposed to calibrate trained network with the IMU bias, which can be improved and utilized in other applications. The proposed method can significantly improve the performance of the existing deep learning-based visual-inertial navigation systems. Since deep learning shows its promising behavior in robust orientation estimation, with a single IMU, in the future, the best model for utilizing IMU for full position estimation needs to be investigated, to see deep learning capability in retrieving position from a single particular IMU. Moreover, the generalization capability of the proposed framework when a new IMU utilizes, and the IMU frequency and noise model changes, should be investigated, and an approach which makes a trained network capable of being used with other IMUs have to be studied. As another future work, features extracted from the proposed network architectures, need to be combined with visual information to solve the problem of odometry completely in a robust manner.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis has introduced a series of algorithms that assist deep learning approaches to perform better in the regression problems of Visual Odometry (VO), Inertial Odometry (IO), and 6-DoF Camera Relocalization. The proposed ideas are not limited to solving the aforementioned problems in the area of autonomous vehicles, and they can be extended to enhance in solving other problems robustly.

Chapter 3 has introduced an approach for robust training of Deep Neural Networks (DNNs) to solve regression problems. The proposed approach clusters output labels in the first step. Afterward, the DNN trains as a classifier to understand the sub-space that each observation belongs to. After training the network as a classifier, it finetunes as a regressor to solve the regression problem. In this manner, the network learns the local boundary that each new query is related to that in the first step; then, the network solves the regression problem for each observation in its own boundary. Hence, each observation's output space domain gets smaller, and the network can train better (more robust and accurate).

The main requirement of the proposed training approach is segmenting the input to local boundaries or clusters. The number of segments or local boundaries for the input data can be defined based on the problem complexity. For the problem of 6-DoF camera relocalization, an unsupervised approach that assists in segmenting the environment in an unsupervised manner is proposed in Chapter 3. The proposed approach finds the best number of segments in the environment by analyzing the distance between clusters by considering the captured images' feature representation. However, for some problems such as VO, it is a bit complicated to automatically find the best number of clusters due to the problem's nature. Hence, the best approach for such problems is to visualize the data point's label and manually determine the number of clusters. Analyzing the results shows that the best number of clusters is when each cluster has an efficient number of observa-

tions and has an abstract meaning. An approach to cluster labels efficiently, after determining the number of clusters, is also proposed in Chapter 3.

Chapter 4 studied the effect of moving objects and motion blur in camera re-localization and VO problems. First of all, an image representation that assists end-to-end DNNs to solve problems robustly with the presence of moving objects and motion blur is introduced; semantic segmentation is utilized to detect semantically moving objects and create the image representation. Then, the DNN is trained based on various image representations that increased the training data's size and made the DNN robust to the aforementioned issues. The experiment and results show the great impact of the proposed image representation on training the network. The study is extended to solve VO's problem and combine the power of DNNs with traditional approaches to make traditional VO techniques robust to moving objects. Semantic Segmentation and Optical Flow are utilized to detect moving objects in the scene in a novel manner and extract feature points from the scene's static parts. The results show the proposed methodology makes traditional networks perform more robust and accurate, especially in highways where high-speed moving objects exist.

Since the essential component for a robust and accurate odometry system is the detection of robust feature points in the scene which are consistent and trackable over time, an approach that learns to extract robust handcrafted features is introduced in the Chapter 5. The proposed approach learns to extract handcrafted features with the presence of only one single training data, and it makes a bridge between neuroscience, evolutionary computing, and neural networks. The proposed approach trains a neuron based on the previously trained neurons. So, the final trained neuron is the sophisticated version of multiple neurons and have rich knowledge. Experiments and results show that the proposed approach helps in training a robust neuron fast with the presence of less amount of training data. The proposed approach can be used to solve the problem of handcrafted feature extraction and can be further improved to solve high-level problems such as object detection.

Chapter 6 has moved the Inertial Odometry research area one step forward by showing that it is possible to solve the odometry problem with the presence of a

single particular Inertial Measurement Unit (IMU); approaches that help to extract the scale of movement, and IMU orientation in the world frame are proposed and examined. Besides, a method based on evolutionary algorithms is proposed for the first time to calibrate the network with the sensor bias; this will help other areas and make DNNs generalizable to different sensors. It has been demonstrated that by training a deep network, it is possible to extract IMU orientation in the world frame with the same accuracy as Visual-Inertial systems.

7.2 Future Work

The theoretical and practical analyzes conducted in this thesis demonstrate the power of Deep Learning in robotic applications, especially in VO and IO modules. The power of deep learning and proposed approaches inspire researchers to do more in-depth research. The possible directions that can be focused in the next few years can be listed as below.

- Benefiting from rich feature representations of deep neural networks and mathematical models of traditional techniques is an approach that researchers should focus more on. This approach helps researchers benefit from rich feature representations of deep learning and fast and robust execution of traditional approaches simultaneously to introduce real-time methods that require less powerful hardware. For instance, the proposed module in Chapter 4 about dealing with moving objects in VO can be further extended to utilize the deep uncertainty of recent deep learning-based monocular depth estimation systems [23] to select features uniformly from near and far ranges.
- Investigating the best approach for training DNNs as a regressor needs more focus. The proposed method has shown that learning sub-spaces and solving the regression problem inside sub-spaces can increase accuracy. This approach can be further extended by utilizing Probability Theory and Bayesian models. Since there are many regression problems in the area of robotics, it could be a great direction. Furthermore, the effect of the map size on the proposed approach also needs to be studied.
- Solving problems with fewer labeled data is another approach that requires

more focus. The proposed training approach in Chapter 5 is capable of learning handcrafted features with a single observation fast. We believe that this approach can be further extended to solve other high-level problems and is a direction that requires lots of focus, not only because it is fast and robust but also because it is inspired by neuroscience and human neuron's training procedure. Labeling data points is costly, and it is important to learn with fewer observations. On the other hand, finding the best compact model plays a critical role in deploying a method for real-time applications. In this manner, expanding the method proposed in Chapter 5 and making the connection between evolutionary computing, neuroscience, and deep learning stronger is critical.

- Inertial Odometry is an important problem, and as shown in this research, it is possible to estimate IMU orientation in the world frame robustly via DNNs. Further investigation is still required in this field to find an approach that can estimate the full position and orientation of IMU in the world frame accurately. An accurate IO system helps in indoor localization, robotics, and virtual reality, and is a significant topic to be studied. Utilizing Bayesian models and Generative Adversarial Neural Networks could be great in this field since they can help in modeling uncertainty and get robust to the sensor's noise.
- One of the major problems with deep learning based approaches in the area of robotics is that once you train a network by a specific type of sensor, you will need to retrain it for another type of sensor (needs to be retrained once the type of camera is changed or a different IMU is utilized). You will also need to retrain it if the experiment conditions are changed (for instance, training data are captured in the autumn, and you test in winter under snow). These conditions widely exist in robotics applications. Hence, it is essential to increase DNNs generalization ability and make them robust to the aforementioned issue.
- This study has mainly focused on finding the best approaches to solve the visual and inertial odometry problems individually. The next step, which

needs to be focused on in the future, is the fusion of introduced modules to get a robust system that can robustly solve the odometry problem.

- The proposed dynamic obstacle detection technique in Chapter 4 needs to be analyzed and compared with recent dynamic obstacle detection modules [28]. While the proposed approach performs well under visual odometry, the proposed dynamic obstacle detection module needs to be compared with other dynamic obstacle detection modules to see its pros and cons and be improved.

Except for the mentioned research directions, the methods proposed in this thesis can be applied to other applications such as virtual reality and indoor localization. We strongly encourage researchers to extend the introduced modules and utilize them on other applications. Modeling the proposed methods on other applications will improve their performance, help find the limitations, and improve the limitations of the proposed modules.

List of Publications

Journal Papers

- **Mahdi Abolfazli Esfahani**, Han Wang, Keyu Wu, Shenghai Yuan, “OriNet: Robust 3D Orientation Estimation with a Single Particular IMU”, *IEEE Robotics and Automation Letters (RA-L)*, 2020
- **Mahdi Abolfazli Esfahani**, Han Wang, Keyu Wu, Shenghai Yuan, “AbolD-deepIO: A novel deep inertial odometry network for autonomous vehicles”, *IEEE Transactions on Intelligent Transportation Systems*, 2019
- **Mahdi Abolfazli Esfahani**, Han Wang, Keyu Wu, Shenghai Yuan, ”DeepD-SAIR: Deep 6-DOF camera relocalization using deblurred semantic-aware image representation for large-scale outdoor environments”, *Image and Vision Computing*, 2019
- **Mahdi Abolfazli Esfahani**, Han Wang, Keyu Wu, Shenghai Yuan, “From local understanding to global regression in monocular visual odometry”, *International Journal of Pattern Recognition and Artificial Intelligence*, 2020
- Keyu Wu, Han Wang, **Mahdi Abolfazli Esfahani**, Shenghai Yuan, “Achieving Real-Time Path Planning in Unknown Environments through Deep Neural Networks”, *IEEE Transactions on Intelligent Transportation Systems*, 2020
- Keyu Wu, Han Wang, **Mahdi Abolfazli Esfahani**, Shenghai Yuan, “Tdpp-net: Achieving three-dimensional path planning via a deep neural network” architecture”, *Neurocomputing*, 2019
- Keyu Wu, Han Wang, **Mahdi Abolfazli Esfahani**, Shenghai Yuan, “BND*-DDQN: Learn to Steer Autonomously through Deep Reinforcement Learning”, *IEEE Transactions on Cognitive and Developmental Systems*, 2019
- Keyu Wu, **Mahdi Abolfazli Esfahani**, Shenghai Yuan, Han Wang, “Learn to steer through deep reinforcement learning”, *Sensors*, 2018

- **Mahdi Abolfazli Esfahani**, Han Wang, Benyamin Bashari, Keyu Wu, Shenghai Yuan, “Learning to Extract Robust Handcrafted Features with a Single Observation via Evolutionary Neurogenesis”, *Applied Soft Computing*, 2021
- Keyu Wu, Han Wang, **Mahdi Abolfazli Esfahani**, Shenghai Yuan, “Learn to Navigate Autonomously through Deep Reinforcement Learning”, *IEEE Transactions on Industrial Electronics*, 2021
- Hasith Karunasekera, **Mahdi Abolfazli Esfahani**, Handuo Zhang, Hui Zhou, and Han Wang, “Learning Features for Multiple Object Tracking as an Extension to Detection”, *Image and Vision Computing*, (Under Review)

Conference Papers

- **Mahdi Abolfazli Esfahani**, Han Wang, Keyu Wu, Shenghai Yuan, “OriNet: Robust 3D Orientation Estimation with a Single Particular IMU”, *International Conference on Robotics and Automation (ICRA)*, 2020
- **Mahdi Abolfazli Esfahani**, Han Wang, Keyu Wu, Shenghai Yuan, “Unsupervised Scene Categorization, Path Segmentation and Landmark Extraction while Traveling Path”, *International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2020
- **Mahdi Abolfazli Esfahani**, Keyu Wu, Shenghai Yuan, Han Wang, “Towards Utilizing Deep Uncertainty In Traditional SLAM”, *IEEE 15th International Conference on Control and Automation (ICCA)*, 2019
- **Mahdi Abolfazli Esfahani**, Keyu Wu, Shenghai Yuan, Han Wang, “A New Approach to Train Convolutional Neural Networks for Real-Time 6-DOF Camera Relocalization”, *IEEE 14th International Conference on Control and Automation (ICCA)*, 2018
- Guiju Ping, **Mahdi Abolfazli Esfahani**, Han Wang, “Unsupervised 3D Primitive Shape Detection Using Mathematical Models”, *International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2020

-
- Keyu Wu, **Mahdi Abolfazli Esfahani**, Shenghai Yuan, Han Wang, “Depth-based Obstacle Avoidance through Deep Reinforcement Learning”, *5th International Conference on Mechatronics and Robotics Engineering*, 2019

References

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
- [2] Mahdi Abolfazli Esfahani, Keyu Wu, Shenghai Yuan, and Han Wang. From local understanding to global regression in monocular visual odometry. *International Journal of Pattern Recognition and Artificial Intelligence*, 2019.
- [3] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.
- [4] Norhafizan Ahmad, Raja Ariffin Raja Ghazilla, Nazirah M Khairi, and Vijayabaskar Kasi. Reviews on various inertial measurement unit (imu) sensor applications. *International Journal of Signal Processing Systems*, 1(2):256–262, 2013.
- [5] Mary B Alatis and Gerhard P Hancke. Pose estimation of a mobile robot based on fusion of imu data and vision data using an extended kalman filter. *Sensors*, 17(10):2164, 2017.
- [6] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison. Kaze features. In *European Conference on Computer Vision*, pages 214–227. Springer, 2012.
- [7] Mahdi Alehdaghi, Mahdi Abolfazli Esfahani, and Ahad Harati. Parallel ransac: Speeding up plane extraction in rgb-d image sequences using gpu. In *Computer and Knowledge Engineering (ICCKE), 2015 5th International Conference on*, pages 295–300. IEEE, 2015.
- [8] Yasin Almalioglu, Muhamad Risqi U Saputra, Pedro PB de Gusmao, Andrew Markham, and Niki Trigoni. Ganvo: Unsupervised deep monocular visual

- odometry and depth estimation with generative adversarial networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5474–5480. IEEE, 2019.
- [9] Adrien Angeli, David Filliat, Stéphane Doncieux, and Jean-Arcady Meyer. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, 24(5):1027–1037, 2008.
- [10] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan, and Napsiah Bt Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5(1):1897, 2016.
- [11] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016.
- [12] Relja Arandjelović and Andrew Zisserman. Dislocation: Scalable descriptor distinctiveness for location recognition. In *Asian Conference on Computer Vision*, pages 188–204. Springer, 2014.
- [13] Yuval Bahat, Netalee Efrat, and Michal Irani. Non-uniform blind deblurring by reblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3286–3294, 2017.
- [14] Axel Barroso-Laguna, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Key. net: Keypoint detection by handcrafted and learned cnn filters. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5836–5844, 2019.
- [15] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.
- [16] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *Intelligent*

- Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 298–304. IEEE, 2015.
- [17] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*, pages 25–36. Springer, 2004.
- [18] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.
- [19] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.
- [20] Song Cao and Noah Snavely. Graph-based discriminative learning for location recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 700–707, 2013.
- [21] Song Cao and Noah Snavely. Minimal scene descriptions from structure from motion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 461–468, 2014.
- [22] Francois Caron, Emmanuel Duflos, Denis Pomorski, and Philippe Vanheegehe. Gps/imu data fusion using multisensor kalman filtering: introduction of contextual aspects. *Information fusion*, 7(2):221–230, 2006.
- [23] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8001–8008, 2019.
- [24] Changhao Chen, Xiaoxuan Lu, Andrew Markham, and Niki Trigoni. Ionet: Learning to cure the curse of drift in inertial odometry. *arXiv preprint arXiv:1802.02209*, 2018.

-
- [25] Zetao Chen, Adam Jacobson, Niko Sünderhauf, Ben Upcroft, Lingqiao Liu, Chunhua Shen, Ian Reid, and Michael Milford. Deep learning features at scale for visual place recognition. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3223–3230. IEEE, 2017.
 - [26] Zetao Chen, Obadiah Lam, Adam Jacobson, and Michael Milford. Convolutional neural network-based place recognition. *arXiv preprint arXiv:1411.1509*, 2014.
 - [27] Gong Cheng, Junwei Han, Peicheng Zhou, and Dong Xu. Learning rotation-invariant and fisher discriminative convolutional neural networks for object detection. *IEEE Transactions on Image Processing*, 28(1):265–278, 2019.
 - [28] Jiyu Cheng, Hong Zhang, and Max Q-H Meng. Improving visual localization accuracy in dynamic environments based on dynamic region removal. *IEEE Transactions on Automation Science and Engineering*, 17(3):1585–1596, 2020.
 - [29] Ronald Clark, Sen Wang, Andrew Markham, Niki Trigoni, and Hongkai Wen. Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017.
 - [30] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *AAAI*, pages 3995–4001, 2017.
 - [31] Gabriele Costante, Michele Mancini, Paolo Valigi, and Thomas A Ciarfuglia. Exploring representation learning with cnns for frame-to-frame ego-motion estimation. *IEEE robotics and automation letters*, 1(1):18–25, 2016.
 - [32] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.
 - [33] Robert Cupec, Emmanuel Karlo Nyarko, Damir Filko, Andrej Kitanov, and Ivan Petrović. Place recognition based on matching of planar surfaces and

- line segments. *The International Journal of Robotics Research*, 34(4-5):674–704, 2015.
- [34] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.
- [35] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.
- [36] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2502–2509. IEEE, 2018.
- [37] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009.
- [38] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018.
- [39] Michael Donoser and Dieter Schmalstieg. Discriminative feature-to-point matching in image-based localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 516–523, 2014.
- [40] Emmanuel Dufourq and Bruce A Bassett. Eden: Evolutionary deep networks for efficient machine learning. In *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*, pages 110–115. IEEE, 2017.
- [41] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint description and detection of local features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8092–8101, 2019.

-
- [42] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.
 - [43] MA Esfahani and H Pourreza. Kinect depth recovery based on local filters and plane primitives. In *Integral Methods in Science and Engineering, Volume 2*, pages 53–63. Springer, 2017.
 - [44] Mahdi Abolfazli Esfahani, Han Wang, Keyu Wu, and Shenghai Yuan. Abold-eepio: A novel deep inertial odometry network for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2019.
 - [45] Mahdi Abolfazli Esfahani, Keyu Wu, Shenghai Yuan, and Han Wang. A new approach to train convolutional neural networks for real-time 6-dof camera relocalization. In *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, pages 81–85. IEEE, 2018.
 - [46] Brian S Everitt, Sabine Landau, Morven Leese, and Daniel Stahl. Cluster analysis. –john wiley & sons. *Ltd., New York*, page 330, 2011.
 - [47] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
 - [48] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Georgia Institute of Technology, 2015.
 - [49] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.
 - [50] Eric Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer graphics and applications*, 25(6):38–46, 2005.
 - [51] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

-
- [52] Abel Gawel, Carlo Del Don, Roland Siegwart, Juan Nieto, and Cesar Cadena. X-view: Graph-based semantic multi-view localization. *IEEE Robotics and Automation Letters*, 3(3):1687–1694, 2018.
 - [53] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
 - [54] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.
 - [55] Jean Dickinson Gibbons and Subhabrata Chakraborti. *Nonparametric Statistical Inference: Revised and Expanded*. CRC press, 2014.
 - [56] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
 - [57] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
 - [58] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.
 - [59] Amit Goldstein and Raanan Fattal. Blur-kernel estimation from spectral irregularities. In *European Conference on Computer Vision*, pages 622–635. Springer, 2012.
 - [60] Petr Gronat, Guillaume Obozinski, Josef Sivic, and Tomas Pajdla. Learning and calibrating per-location classifiers for visual place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 907–914, 2013.

-
- [61] Antonio Gulli and Sujit Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.
 - [62] Abner Guzman-Rivera, Pushmeet Kohli, Ben Glocker, Jamie Shotton, Toby Sharp, Andrew Fitzgibbon, and Shahram Izadi. Multi-output learning for camera relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1114–1121, 2014.
 - [63] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.
 - [64] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.
 - [65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European conference on computer vision*, pages 346–361. Springer, 2014.
 - [66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
 - [67] Simon Hecker, Dengxin Dai, and Luc Van Gool. End-to-end learning of driving models with surround-view cameras and route planners. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 435–453, 2018.
 - [68] Sebastian Herzog, Christian Tetzlaff, and Florentin Wörgötter. Evolving artificial neural networks with feedback. *Neural Networks*, 123:153–162, 2020.
 - [69] Sebastian Hilsenbeck, Dmytro Bobkov, Georg Schroth, Robert Huitl, and Eckehard Steinbach. Graph-based data fusion of pedometer and wifi measurements for mobile indoor positioning. In *Proceedings of the 2014 ACM international joint conference on pervasive and ubiquitous computing*, pages 147–158. ACM, 2014.

-
- [70] Du Q Huynh. Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.
 - [71] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.
 - [72] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.
 - [73] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016.
 - [74] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5974–5983, 2017.
 - [75] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-dof camera relocalization. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2938–2946. IEEE, 2015.
 - [76] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN’95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.
 - [77] Hooman Khosravi, M Abolfazli-E, and M-R Akbarzadeh-T. A nature-inspired transition from differential evolution to particle swarm optimization. In *2015 5th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 87–92. IEEE, 2015.

-
- [78] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.
 - [79] Kishore Reddy Konda and Roland Memisevic. Learning visual odometry with a convolutional network. In *VISAPP (1)*, pages 486–490, 2015.
 - [80] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
 - [81] Zuzana Kukelova, Martin Bujnak, and Tomas Pajdla. Real-time solution to the absolute pose problem with unknown radial distortion and focal length. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2816–2823, 2013.
 - [82] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.
 - [83] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.
 - [84] Kwang Y Lee and Jong-Bae Park. Application of particle swarm optimization to economic dispatch problem: advantages and disadvantages. In *2006 IEEE PES Power Systems Conference and Exposition*, pages 188–192. IEEE, 2006.
 - [85] Benedetta Leuner, Elizabeth Gould, and Tracey J Shors. Is there a link between adult neurogenesis and learning? *Hippocampus*, 16(3):216–224, 2006.
 - [86] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. *BRISK: Binary robust invariant scalable keypoints*. IEEE, 2011.
 - [87] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual-inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

-
- [88] Peiliang Li, Tong Qin, Botao Hu, Fengyuan Zhu, and Shaojie Shen. Monocular visual-inertial state estimation for mobile augmented reality. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 11–21. IEEE, 2017.
 - [89] Ruihao Li, Qiang Liu, Jianjun Gui, Dongbing Gu, and Huosheng Hu. Indoor relocalization in challenging environments with dual-stream convolutional neural networks. *IEEE Transactions on Automation Science and Engineering*, 15(2):651–662, 2018.
 - [90] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 7286–7291. IEEE, 2018.
 - [91] Sijin Li and Antoni B Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *Asian Conference on Computer Vision*, pages 332–347. Springer, 2014.
 - [92] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In *European conference on computer vision*, pages 15–29. Springer, 2012.
 - [93] Yunpeng Li, Noah Snavely, and Daniel P Huttenlocher. Location recognition using prioritized feature matching. In *European conference on computer vision*, pages 791–804. Springer, 2010.
 - [94] Hyon Lim, Sudipta N Sinha, Michael F Cohen, and Matthew Uyttendaele. Real-time image-based 6-dof localization in large-scale environments. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1043–1050. IEEE, 2012.
 - [95] Yi Lin, Fei Gao, Tong Qin, Wenliang Gao, Tianbo Liu, William Wu, Zhenfei Yang, and Shaojie Shen. Autonomous aerial navigation using monocular visual-inertial fusion. *Journal of Field Robotics*, 35(1):23–51, 2018.

-
- [96] Jia Liu, Maoguo Gong, Qiguang Miao, Xiaogang Wang, and Hao Li. Structure learning for deep neural networks based on multiobjective optimization. *IEEE transactions on neural networks and learning systems*, 29(6):2450–2463, 2018.
 - [97] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.
 - [98] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
 - [99] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.
 - [100] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
 - [101] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. *arXiv preprint arXiv:1511.06727*, 2015.
 - [102] Dimitrios Lymberopoulos, Jie Liu, Xue Yang, Romit Roy Choudhury, Vlado Handziski, and Souvik Sen. A realistic evaluation and comparison of indoor location technologies: Experiences and lessons learned. In *Proceedings of the 14th international conference on information processing in sensor networks*, pages 178–189. ACM, 2015.
 - [103] Simon Lynen, Torsten Sattler, Michael Bosse, Joel A Hesch, Marc Pollefeys, and Roland Siegwart. Get out of my lab: Large-scale, real-time visual-inertial localization. In *Robotics: Science and Systems*, 2015.
 - [104] Sebastian OH Madgwick, Andrew JL Harrison, and Ravi Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm.

- In *2011 IEEE international conference on rehabilitation robotics*, pages 1–7. IEEE, 2011.
- [105] Diego Marcos, Michele Volpi, and Devis Tuia. Learning rotation invariant convolutional filters for texture classification. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2012–2017. IEEE, 2016.
- [106] Efrén Mezura-Montes and Carlos A Coello Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.
- [107] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10):1615–1630, 2005.
- [108] Michael J Milford and Gordon F Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *2012 IEEE International Conference on Robotics and Automation*, pages 1643–1649. IEEE, 2012.
- [109] Vikram Mohanty, Shubh Agrawal, Shaswat Datta, Arna Ghosh, Vishnu Dutt Sharma, and Debashish Chakravarty. Deepvo: A deep learning approach for monocular visual odometry. *arXiv preprint arXiv:1611.06069*, 2016.
- [110] Hans P Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1980.
- [111] Barry K Moser, Gary R Stevens, and Christian L Watts. The two-sample t test versus satterthwaite’s approximate f test. *Communications in Statistics-Theory and Methods*, 18(11):3963–3975, 1989.
- [112] Peter Muller and Andreas Savakis. Flowdometry: An optical flow and deep learning based approach to visual odometry. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 624–631. IEEE, 2017.

- [113] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.
- [114] Raúl Mur-Artal and Juan D Tardós. Fast relocalisation and loop closing in keyframe-based slam. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 846–853. IEEE, 2014.
- [115] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.
- [116] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, volume 1, page 3, 2017.
- [117] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.
- [118] John-Olof Nilsson, Amit K Gupta, and Peter Händel. Foot-mounted inertial navigation made easy. In *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on*, pages 24–29. IEEE, 2014.
- [119] John-Olof Nilsson, Dave Zachariah, Isaac Skog, and Peter Händel. Cooperative localization by dual foot-mounted inertial sensors and inter-agent ranging. *EURASIP Journal on Advances in Signal Processing*, 2013(1):164, 2013.
- [120] K Nirmal, AG Sreejith, Joice Mathew, Mayuresh Sarpotdar, Ambily Suresh, Ajin Prakash, Margarita Safonova, and Jayant Murthy. Noise modeling and analysis of an imu-based attitude sensor: improvement of performance by filtering and sensor fusion. In *Advances in Optical and Mechanical Technologies for Telescopes and Instrumentation II*, volume 9912, page 99126W. International Society for Optics and Photonics, 2016.

-
- [121] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *Proceedings of the IEEE international conference on computer vision*, pages 3456–3465, 2017.
- [122] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: learning local features from images. In *Advances in Neural Information Processing Systems*, pages 6237–6247, 2018.
- [123] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
- [124] Magnus Erik Hvass Pedersen. Good parameters for particle swarm optimization. *Hvass Lab., Copenhagen, Denmark, Tech. Rep. HL1001*, pages 1551–3203, 2010.
- [125] Mark G Petovello. *Real-time integration of a tactical-grade IMU and GPS for high-accuracy positioning and navigation*. Citeseer, 2003.
- [126] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.
- [127] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 12240–12249, 2019.
- [128] Anurag Ranjan, Varun Jampani, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Adversarial collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. *arXiv preprint arXiv:1805.09806*, 2018.
- [129] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural net-

- works. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.
- [130] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1137–1149, 2017.
- [131] Pau Rodríguez, Issam Laradji, Alexandre Drouin, and Alexandre Lacoste. Embedding propagation: Smoother manifold for few-shot classification. *arXiv preprint arXiv:2003.04151*, 2020.
- [132] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1508–1515. IEEE, 2005.
- [133] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.
- [134] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.
- [135] Mirko Santello, Nicolas Toni, and Andrea Volterra. Astrocyte function from information processing to cognition and cognitive impairment. *Nature neuroscience*, page 1, 2019.
- [136] Torsten Sattler, Michal Havlena, Filip Radenovic, Konrad Schindler, and Marc Pollefeys. Hyperpoints and fine vocabularies for large-scale location recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2102–2110, 2015.
- [137] Torsten Sattler, Michal Havlena, Konrad Schindler, and Marc Pollefeys. Large-scale location recognition and the geometric burstiness problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1582–1590, 2016.

- [138] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 667–674. IEEE, 2011.
- [139] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):1744–1756, 2017.
- [140] Paul G Savage. Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms. *Journal of guidance, control, and dynamics*, 21(1):19–28, 1998.
- [141] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.
- [142] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [143] Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, 1993.
- [144] Yuhui Shi et al. Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 81–86. IEEE, 2001.
- [145] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013.
- [146] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [147] Sudipta N Sinha, Jan-Michael Frahm, Marc Pollefeys, and Yakup Genc. Gpu-based video feature tracking and matching. In *EDGE, workshop on edge computing using new commodity architectures*, volume 278, page 4321, 2006.

-
- [148] Isaac Skog, John-Olof Nilsson, and Peter Händel. Evaluation of zero-velocity detectors for foot-mounted inertial navigation systems. In *2010 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–6. IEEE, 2010.
 - [149] Stephen M Smith and J Michael Brady. Susan—a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997.
 - [150] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.
 - [151] Arno Solin, Santiago Cortes, Esa Rahtu, and Juho Kannala. Inertial odometry on handheld smartphones. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 1–5. IEEE, 2018.
 - [152] Shawn F Sorrells, Mercedes F Paredes, Arantxa Cebrian-Silla, Kadellyn Sandoval, Dashi Qi, Kevin W Kelley, David James, Simone Mayer, Julia Chang, Kurtis I Auguste, et al. Human hippocampal neurogenesis drops sharply in children to undetectable levels in adults. *Nature*, 555(7696):377, 2018.
 - [153] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
 - [154] Kenneth O Stanley. Exploiting regularity without development. In *Proceedings of the AAAI Fall Symposium on Developmental Systems*, page 37. AAAI Press Menlo Park, CA, 2006.
 - [155] Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007.

-
- [156] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.
 - [157] Sébastien Sultan, Liyi Li, Jonathan Moss, Francesco Petrelli, Frédéric Cassé, Elias Gebara, Jan Lopatar, Frank W Pfrieger, Paola Bezzi, Josef Bischofberger, et al. Synaptic integration of adult-born hippocampal neurons is locally controlled by astrocytes. *Neuron*, 88(5):957–972, 2015.
 - [158] Yanan Sun, Gary G Yen, and Zhang Yi. Evolving unsupervised deep neural networks for learning meaningful representations. *IEEE Transactions on Evolutionary Computation*, 23(1):89–103, 2019.
 - [159] Niko Sünderhauf, Sareh Shirazi, Feras Dayoub, Ben Upcroft, and Michael Milford. On the performance of convnet features for place recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4297–4304. IEEE, 2015.
 - [160] Niko Sünderhauf, Sareh Shirazi, Adam Jacobson, Feras Dayoub, Edward Pepperell, Ben Upcroft, and Michael Milford. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. *Proceedings of Robotics: Science and Systems XII*, 2015.
 - [161] Linus Svärm, Olof Enqvist, Fredrik Kahl, and Magnus Oskarsson. City-scale localization for cameras with known vertical direction. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1455–1461, 2017.
 - [162] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
 - [163] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8174–8182, 2018.

-
- [164] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1808–1817, 2015.
 - [165] Akihiko Torii, Josef Sivic, Tomas Pajdla, and Masatoshi Okutomi. Visual place recognition with repetitive structures. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 883–890, 2013.
 - [166] Roberto Valenti, Ivan Dryanovski, and Jizhong Xiao. Keeping a good attitude: A quaternion-based orientation filter for imus and margs. *Sensors*, 15(8):19302–19330, 2015.
 - [167] Brandon Wagstaff and Jonathan Kelly. Lstm-based zero-velocity detection for robust inertial navigation. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8. IEEE, 2018.
 - [168] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using lstms for structured feature correlation. In *Int. Conf. Comput. Vis.(ICCV)*, pages 627–637, 2017.
 - [169] Rui Wang, Martin Schwörer, and Daniel Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *International Conference on Computer Vision (ICCV), Venice, Italy*, 2017.
 - [170] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2043–2050. IEEE, 2017.
 - [171] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

-
- [172] MJ Westoby, J Brasington, NF Glasser, MJ Hambrey, and JM Reynolds. ‘structure-from-motion’photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179:300–314, 2012.
- [173] Brian Williams, Mark Cummins, José Neira, Paul Newman, Ian Reid, and Juan Tardós. A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems*, 57(12):1188–1197, 2009.
- [174] Zhuoling Xiao, Hongkai Wen, Andrew Markham, and Niki Trigoni. Lightweight map matching for indoor localisation using conditional random fields. In *Proceedings of the 13th international symposium on Information processing in sensor networks*, pages 131–142. IEEE Press, 2014.
- [175] Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. Every pixel counts: Unsupervised geometry learning with holistic 3d motion understanding. *arXiv preprint arXiv:1806.10556*, 2018.
- [176] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016.
- [177] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2018.
- [178] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.
- [179] Amir Roshan Zamir and Mubarak Shah. Image geo-localization based on multiplenearest neighbor feature matching usinggeneralized graphs. *IEEE transactions on pattern analysis and machine intelligence*, 36(8):1546–1558, 2014.
- [180] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

-
- [181] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 340–349, 2018.
 - [182] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE, 2018.
 - [183] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
 - [184] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.

Appendix A

Visual Comparison of Various Corner Detection Modules

Figure A.1: This figure illustrates and compares the detected corners of the proposed method with others in the presence of the rotation. The first row is the reference image, and corners detected with the proposed ECNN-RS, Harris and KAZE methods are shown with green squares. The second and third rows are the image after rotation. For the second row, the image is rotated for -120° , and in the third row, it rotates for $+30^\circ$. Green squares are detected features which are the same as the features in the reference image (first row), and red squares are new features which do not have a correspondent in the reference image.

Figure A.2: This figure illustrates and compares the detected corners of the proposed method with others in the presence of the change in the illumination. The first row is the reference image, and corners detected with the proposed ECNN-RS, Harris and KAZE methods are shown with green squares. The second and third rows are the image after illumination change. For the second row positive measurement of +100 is added to all pixels, and in the third row negative value of -100 is added. Green squares are detected features which are the same as the features in the reference image (first row), and red squares are new features which do not have a correspondent in the reference image.

Figure A.3: This figure illustrates and compares the detected corners of the proposed method with others in the presence of the zero-mean Gaussian noise. The first row is the reference image, and corners detected with the proposed ECNN-RS, Harris and KAZE methods are shown with green squares. The second and third rows are the images with the presence of Gaussian noise with different variances (σ^2). For the second row, $\sigma^2 = 0.18$ is used, and in the third row $\sigma^2 = 0.12$. Green squares are detected features which are the same as the features in the reference image (first row), and red squares are new features which do not have a correspondent in the reference image.

Figure A.4: This figure illustrates and compares the detected corners of the proposed method with others in the presence of the change of scale. The first row is the reference image, and corners detected with the proposed ECNN-RS, Harris and KAZE methods are shown with green squares. The second and third rows are the images scaled by 0.6 and 1.6 of the reference image. Green squares are detected features which are the same as the features in the first row(reference image), and red squares are new features which do not have a correspondent in the reference image.

Figure A.5: This figure illustrates and compares the detected corners of the proposed method with others in the presence of the change in the viewpoint (on the dataset referenced in the paper). The first row is the reference image, and corners detected with the proposed ECNN-RS, Harris and KAZE methods are shown with green squares. The second and third rows are the images from different point of views. Green squares are detected features which are the same as the features in the reference image (first row), and red squares are new features which do not have a correspondent in the reference image.