Kernel learning for visual perception

Wang, Chen

2019

Wang, Chen. (2019). Kernel learning for visual perception. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/105527>

<https://doi.org/10.32657/10220/47835>

Kernel Learning for Visual Perception

Chen Wang

School of Electrical & Electronic Engineering

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

2019

Statement of Originality

I hereby certify that the intellectual content of this thesis is the product of my original research work and has not been submitted for a higher degree to any other University or Institution.

Aug. 2018

.....

Date

Chen Wang

.....

Chen Wang

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.

Mar. 2019

.....

Date

.....

Prof. Lihua XIE

Authorship Attribution Statement

This thesis contains material from 5 papers accepted at conferences in which I am listed as an author.

Chapter 3 and Chapter 4 are published as [Chen Wang, Le Zhang, Lihua Xie, and Junsong Yuan. Kernel Cross-Correlator. The Thirty-Second AAAI Conference on Artificial Intelligence \(AAAI-18\), 4179-4186, Feb. 2018.](#)

The contributions of the co-authors are as follows:

- I proposed the idea, designed the experiments, and prepared the manuscript.
- Le Zhang and I conducted the experiments of visual object tracking.
- The manuscript was revised by Le Zhang, Lihua Xie, and Junsong Yuan.

Chapter 5 is published as [Chen Wang, Tete Ji, Thien-Minh Nguyen, and Lihua Xie. Correlation Flow: Robust Optical Flow using Kernel Cross-Correlators. International Conference on Robotics and Automation \(ICRA\), 836-841, May. 2018.](#)

The contributions of the co-authors are as follows:

- I proposed the idea, designed the experiments, and prepared the manuscript.
- Tete Ji, Thien-Minh Nguyen, and I conducted the experiments.
- The manuscript was revised by Tete Ji, Lihua Xie, and Thien-Minh Nguyen.

Chapter 6 is an extension of [Chen Wang, Junsong Yuan, and Lihua Xie. Non-iterative SLAM. International Conference on Advanced Robotics \(ICAR\), 83-90, Jul. 2017.](#)

The contributions of the co-authors are as follows:

- I proposed the idea, conducted the experiments, and prepared the manuscript.
- The manuscript was revised by Lihua Xie and Junsong Yuan.

Chapter 7 is published as [Chen Wang, Handuo Zhang, Thien-Minh Nguyen, and Lihua Xie. Ultra-Wideband Aided Fast Localization and Mapping System. IEEE/RSJ International Conference on Intelligent Robots and Systems \(IROS\), 1602-1609, Sep. 2017.](#)

The contributions of the co-authors are as follows:

- I proposed the idea, designed the experiments, and prepared the manuscript.
- Handuo Zhang, Thien-Minh Nguyen, and I conducted the experiments.
- The manuscript was revised by Handuo Zhang, Thien-Minh Nguyen, and Lihua Xie.

Chapter 8 is accepted as [Chen Wang, Jianfei Yang, Lihua Xie, and Junsong Yuan. Kervolutional Neural Networks. Conference on Computer Vision and Pattern Recognition \(CVPR\), Jun. 2019.](#)

The contributions of the co-authors are as follows:

- I proposed the idea, designed the experiments, and prepared the manuscript.
- Jianfei Yang and I conducted the experiments.
- The manuscript was revised by Jianfei Yang and Lihua Xie.

Mar. 2019

.....

Date

Chen Wang

.....

Chen Wang

Acknowledgements

I wish to express my sincerest gratitude to my advisor, Professor Lihua Xie, who has provided much encouragement and freedom for my research. His attitude towards research no doubt set a good example of what a true researcher should be.

I also wish to express my deep gratitude to Associate Professor Junsong Yuan for his unconditional help in supervising me. His kindly sharing and deep knowledge in computer vision inspired my research.

I need to thank my cool friends who make my study memorable and joyful. Special thanks go to Chun-Lin Chen for his suggestions on many projects, especially in the first two years of my study. Special thanks go to Junjun Wang for the insightful suggestions on my Ph.D. life. Special thanks go to Minh-Chung Hoang for his sharing and support. Special thanks go to Xu Fang and Jianfei Yang for their help in the experiments, especially towards the end of my studies. Special thanks go to Le Zhang and Handuo Zhang for their kindly sharing of great technologies. Special thanks go to Kexin Guo, Thien-Minh Nguyen, and Abdul Hanif Zaini for the insightful discussions. Special thanks go to Tete Ji and Han Wang for their trusting and help. It is also my great privilege to study with Wei Cui, Yiyuan Han, Minzuo Jiang, Cunxiao Miao, Yan Zhao, Jianjie Zhu, Wei Meng, Jinming Xu and other talented friends, who made my stay in Singapore really like being at home.

Last but not least, I must thank my beloved parents, cousin as well as other family members, without whose unconditional love, support and understanding, this work would not have been possible. Love you, my family.

“Success is how high you bounce when you hit the bottom.”

— George S. Patton

To my dear family

Abstract

The visual perceptual system in animals allows them to assimilate information from their surroundings. In artificial intelligence, the objective of visual perception is to enable the capability of a computer system to interpret the surrounding environment using data acquired from cameras and other aided sensors. Since the last century, researchers in visual perception have delivered many marvelous technologies and algorithms for various applications, such as object detection and image recognition, etc. Despite the technological progresses, human beings are still confronted by the unsatisfactory performance of artificial visual perceptual systems. One of the main reasons is that the traditional methods usually rely on large amount of training data, powerful processors, and require great efforts and time for process modeling. The research goal of this thesis is to develop visual perceptual systems that requires less computational resources but with higher performance. To this end, the novel kernel learning methods for several basic visual perceptual tasks, including object tracking, localization, mapping, and image recognition, are proposed and demonstrated both theoretically and practically.

In visual object tracking, the state-of-the-art algorithms that leverage on kernelized correlation filters are limited by circulant training data and non-weighted kernel functions. This makes them only applicable for translation prediction and prevents their usage in other applications. To overcome the problems, a kernel cross-correlator (KCC) is introduced. First, by introducing the kernel trick, the KCC extends linear cross-correlation to non-linear space, which is more robust to signal noises and distortions. Second, connections to the existing works show that the KCC provides a unified solution for correlation filters. Third, the KCC is not only applicable to any training data and kernel functions, but also able to predict affine transforms with customized properties. Last, by leveraging the fast Fourier transform (FFT), the KCC eliminates direct calculation of kernel vectors, thus achieving better performance at a reasonable computational cost. Comprehensive experiments on visual tracking and human activity recognition using wearable devices have demonstrated its robustness, flexibility, and efficiency.

Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. It is calculated from sequences of ordered images and allows the estimation of motion as instantaneous image velocities, which is crucial for autonomous robot navigation. This thesis proposes a KCC-based algorithm to determine optical flow using a monocular camera, which is named as correlation flow (CF). CF can provide reliable and accurate velocity estimation and is robust to motion blur. In addition, a joint kernel scale-rotation correlator is proposed to estimate the altitude velocity and yaw rate which are not available by traditional methods. Autonomous flight tests on a quadcopter show that correlation flow can provide robust trajectory estimation with very low processing power.

In the problem of simultaneous localization and mapping (SLAM), traditional odometry methods resort to iterative algorithms which are usually computationally expensive or require well-designed initialization. To overcome this problem, a KCC-based non-iterative solution to RGB-D-inertial odometry system is proposed. To reduce the odometry and inertial drifts, two frameworks for non-iterative SLAM (NI-SLAM) are presented. One is to combine a visual loop closure detection, another one is to seek the aids from ultra wide-band (UWB) technology. Dominated by the FFT, the non-iterative front-end is only of $\mathcal{O}(n \log n)$ complexity, where n is the number of pixels. Therefore, both frameworks can provide reliable performance and are of very low computational complexity. The map fusion is conducted by element-wise operation, so that both time and space complexity are further reduced. Extensive experiments show that, due to the lightweight of the proposed non-iterative front-end, both frameworks of NI-SLAM can run at a much faster speed and yet still with comparable accuracy with the state-of-the-arts.

Convolutional neural network (CNN) is one of the most powerful tools in visual perception. It has enabled many state-of-the-art performances in image recognition, object detection, etc. However, little effort has been devoted to establishing convolution in non-linear space. In this thesis, a new operation, kervolution (kernel convolution), is introduced to approximate the non-linear behavior of the human perceptual system. It generalizes traditional convolution and increases the model capacity without introducing more parameters. Similarly, kervolution can also be calculated through element-wise multiplication via Fourier transform. The extensive experiments show that the kervolutional neural networks (KNN) achieve better

performance and faster convergence than traditional CNN on the MNIST, CIFAR, and ImageNet datasets.

In summary, the thesis demonstrates the superiority of the proposed kernel tools for visual perceptual problems, including KCC, CF, NI-SLAM and KNN. With the kernel tools, we may expect their usage in more applications, such as internet of things, robotics, transfer learning, reinforcement learning, etc.

Contents

Acknowledgements	ix
Abstract	xiii
List of Figures	xxi
List of Tables	xxv
Symbols and Acronyms	xxvii
1 Introduction	1
1.1 Scope and Overview	1
1.2 Major Contributions	3
1.3 Outline of the Thesis	4
2 Literature Review	9
2.1 Support Vector Machine	9
2.1.1 Hard SVM	10
2.1.2 Soft SVM	12
2.1.3 Kernel SVM	14
2.2 Convolutional Neural Network	16
2.2.1 Cross-Correlation & Convolution	17
2.2.2 Equivariance and Invariance	19
2.3 Kernel Ridge Regression	20
2.3.1 Circulant Data Structure	22
2.4 Cross-Correlator	25
2.5 Conclusion	26
3 Kernel Cross-Correlator	27
3.1 Introduction	27
3.2 Related Work	28
3.3 Formulation of Kernel Cross-Correlator	29
3.4 Special Cases of KCC	33
3.4.1 Kernel Translation Correlator	33
3.4.2 Kernel Rotation Correlator	34

3.4.3	Kernel Scale Correlator	35
3.5	Theoretical Analysis	35
3.5.1	Connection to MOSSE	35
3.5.2	Connection to KCF	36
3.5.3	Connection to MKCF, DSST	37
3.6	Experiments	37
3.6.1	Dataset	38
3.6.2	Implementation Details	38
3.6.3	Performance	40
3.7	Conclusion	41
4	Visual Object Tracking	43
4.1	Introduction	43
4.2	Related Work	43
4.2.1	Generative Trackers	44
4.2.2	Discriminative Trackers	45
4.3	Methodology	47
4.3.1	Update Rules	48
4.3.2	Prediction	49
4.4	Experiments	51
4.4.1	Dataset and Evaluation Metric	51
4.4.2	Comparison	53
4.5	Conclusion	55
5	Correlation Flow	57
5.1	Introduction	57
5.2	Related Work	59
5.3	2-D KCC	60
5.4	Correlation Flow	61
5.4.1	Translation Flow	61
5.4.2	Scale and Rotation Flow	64
5.5	Experiments	67
5.5.1	Implementation	67
5.5.2	Velocity Estimation	69
5.5.3	Autonomous Flight	70
5.5.4	Hovering Test	71
5.6	Conclusion	71
6	Non-iterative RGB-D-inertial Odometry	73
6.1	Introduction	73
6.2	Related Work	75
6.2.1	Visual Odometry and Map Fusion	76
6.2.2	Inertial Fusion	78
6.3	System Architecture	79

6.3.1	Iterative Framework	79
6.3.2	Non-iterative Framework	81
6.4	Data Decoupling	82
6.4.1	6 DoF to 3 DoF	82
6.4.2	3 DoF to 2 DoF	83
6.5	Key-frame Training	85
6.6	Translation Estimation	86
6.6.1	Image Translation Estimation	86
6.6.2	Depth Translation Estimation	88
6.7	Key-frame Selection	89
6.8	Map Refinement and Fusion	91
6.9	Visual Loop Closure	93
6.9.1	Relocalization	93
6.9.2	Pose Graph Optimization	93
6.10	Experiments	94
6.10.1	Implementation	94
6.10.2	Simulation on TUM dataset	96
6.10.3	Dense Mapping	97
6.10.4	Performance on Benchmark	98
6.10.5	Performance on Micro-device	102
6.11	Conclusion	103
7	UWB Aided NI-SLAM	105
7.1	Introduction	105
7.2	System Architecture	107
7.2.1	UWB Localization	107
7.2.2	RGB-D-Inertial Odometry	111
7.2.3	UWB Aided Odometry	112
7.3	Experiments	115
7.3.1	Implementation	115
7.3.2	Accuracy Evaluation	116
7.3.3	Efficiency Evaluation	117
7.3.4	Dense Mapping	118
7.3.5	System Generalization	118
7.4	Conclusion	120
8	Kervolutional Neural Networks	123
8.1	Introduction	123
8.2	Related Work	125
8.3	Kervolution	127
8.3.1	Sharing Weights	128
8.3.2	Translational Equivariance	129
8.3.3	Model Capacity and Features	130

8.3.4	Kervolutional Layers and Learnable Kernel	132
8.4	Ablation Study	133
8.4.1	Kernels	134
8.4.2	Hyperparameters	135
8.4.3	Layer Arrangement	135
8.4.4	Removing ReLU	136
8.5	Experiments	136
8.5.1	CIFAR	137
8.5.2	ImageNet	139
8.6	Discussion	140
8.7	Conclusion	140
9	Conclusion & Future Work	143
9.1	Conclusion	143
9.2	Future Work	145
9.2.1	Machine Learning	145
9.2.2	Application	146
	List of Author's Awards, Patents, and Publications	149
	Bibliography	151

List of Figures

1.1	The structure of the thesis.	2
2.1	An example of CNN (LeNet-5)	17
2.2	The difference of cross-correlation and convolution on two infinite continuous signals.	18
3.1	The color map examples of the 25 dimensional raw data from the wearable sensors. The horizontal axis is the time instant which is recorded with update rate of 30Hz. The data are normalized for better visualization.	38
3.2	The average confusion matrix of KCC and DTW over all the subjects. There are 5 trials for each of the 13 activities, resulting in a 65×65 confusion matrix. It is obvious that KCC is able to better distinguish human activities and produce less false positives.	39
3.3	The performance comparison on WARD dataset. The left and right vertical axes indicate the recognition accuracy (%) and average running time (μ s), respectively.	40
4.1	The tracking examples of KCC in terms of number of frames (#).	50
4.2	Success plot of state-of-the-art methods on the OTB dataset.	51
4.3	Success plot of state-of-the-art methods on different categories (I).	52
4.4	Success plot of state-of-the-art methods on different categories (II).	53
4.5	Success plot of state-of-the-art methods on scale variation.	53
5.1	The framework for camera velocity estimation based on the proposed correlation flow. A downward facing camera mounted on a quadcopter moves arbitrarily in a 3-D space. The horizontal and altitude velocities and the yaw angular rate are estimated by the kernel translation correlator (KTC) and the kernel scale-rotation correlator (KSRC), respectively.	58
5.2	The computation structure of correlation flow. The position of the maximum response indicates the transformation of images. Except for the FFT, all the operations are element-wise, resulting in efficient computations. Due to the introduction of kernels, correlation flow is robust to noises, motion blur, and image distortions.	61

5.3	An example of the KSRC. (a) previous image captured at 30Hz; (b) current image; (c) the response of KSRC. The location of the maximum response relative to the center indicates the transformation pattern. In this example, the image is rotated by 9 degrees and scaled with factor 1.2516.	63
5.4	One example of the velocity estimation from the first 60 seconds of a flight test. It is obvious that the estimation from correlation flow is smoother and more accurate than PX4Flow.	65
5.5	The estimated yaw rate and altitude velocity compared with ground truth from Vicon system.	68
5.6	The flight environment and the control scheme.	69
5.7	The position estimation by integrating the estimated velocity during fully autonomous flight. The drone tries to fly along a rectangle. . .	70
6.1	The proposed framework of non-iterative RGB-D-inertial odometry system. The 3-D movements are estimated in decoupled subspaces separately. This process is of complexity $\mathcal{O}(n \log(n))$ where n is the number of image points.	79
6.2	Differences between perspective and orthogonal projections. The perspective projection (a) projects points on the principle point of the camera, while orthogonal projection (b) projects points on the image plane. A quadrangular frustum pyramid can be a rectangle when the vertex coincides with the principle point. While for orthogonal image, the size ratio of the two rectangles will not change. This simple property is applied for data decoupling.	81
6.3	The point cloud obtained by a depth camera in Figure (a) is rectified by the attitude estimation from the back-end. Its perspective image (b) is then reprojected to orthogonal color image (c) and depth image (d). Since the missing points in (b) are also projected, there exists black holes in (c) and (d).	83
6.4	The output from KCC on an example sequence. (a) is the first key-frame; the top-left figure of (b), (c), and (d) are selected from its subsequent frames. Their KCC outputs are shown on the top-right of their corresponding figures whose PSR is 161, 71, and 41, respectively. The outputs are normalized and shown in 3-D space for better visualization. It is assumed that the KCC outputs can be approximated by 2-D Gaussian functions with the center on the peak, which is useful for fast computing of estimated covariance in (6.15).	87
6.5	The image (a) is an example of orthogonal depth image from Figure 6.3. The corresponding well-matched points are shown in (b) by pixel intensities. The higher the brightness is, the more confidence the matches have.	88

6.6	The first 200 frames of a sequence are extracted to show the changes of PSR with respect to frame number. Only when the PSR of the correlation output goes below T_K , a new key-frame will be created. The point clouds will be fused with the key-frame when PSR is higher than T_M	90
6.7	(a) is a new key-frame. (b) is the refined key-frame. It is shown that the ‘black holes’ in the original key-frame can be filled by subsequent matched frames based on (6.19). (c) and (d) are the same part of (a) and (b), respectively. Note that the keyboard is well complemented. The moving average in (6.19) will not make it blurred, as long as the image translation is predicted correctly.	91
6.8	Figure (a) and (b) are the dense reconstruction of our lab. The panorama of this room is given in Figure (c). This environment is quite challenging for feature-based methods, since there are many feature-less area and light reflections produced by pieces of glass. . .	96
6.9	The examples of color (first row) and depth (second row) images, in which lots of specular reflections and featureless regions can be seen. . .	97
6.10	The trajectory estimation on an ultra-low power CPU Atom x5-Z8350 with Scenario Design Power (SDP) of 2W.	101
6.11	A dense mapping of office cubicles. It is constructed in real-time on the ultra-low power CPU Atom x5-Z8350. The map is sent to a remote server in real-time only for visualization purpose (no post-processing).	101
7.1	The proposed framework of the UWB aided dense mapping system consists of three parallel threads. The UWB localization thread is to provide global drift-free loop closure. The thread of RGB-D-inertial odometry is served for local accurate trajectory estimation. The key-frames from the odometry thread together with the loop closure from UWB thread are sent to the mapping thread where an optimization is performed to generate live dense drift-free maps. . .	107
7.2	The proposed optimization scheme for UWB ranging localization. The pose graph is updated by inserting a new and removing the oldest pose in a sliding window, when a new ranging measurement is acquired. The trajectory in the window is optimized based on the range and smooth constraints. Note that only the latest optimized pose in the window is reported to detect the odometry drift, resulting in a real-time and dynamic system.	109
7.3	The UWB aided mapping framework using the objective (7.20). The range-smooth constraints are connected to the nearest tracking constraint.	113
7.4	The UWB aided mapping framework using the objective (7.23). The virtual range constraints r_i^{vr} are connected to tracking constraints directly.	113

7.5	Some examples of trajectory estimation w/wo UWB aids. The estimation with UWB aids is obviously much closer to the ground truth.	117
7.6	The live dense mapping for the IoT (Internet of Things) lab in NTU. The first two figures show the maps from two points of view that is generated in real-time, under which the panoramic view is presented for comparison.	119
7.7	The features that are extracted by ORB-SLAM. The specular reflections obviously raise issues for pure visual methods.	120
8.1	The comparison of learned filters on MNIST from the first layer (six channels and filter size of 5×5) of CNN and polynomial KNN. It is interesting that some of the learned filters (e.g. channel 4) from KNN are quite similar to CNN. This indicates that part of the kervolutional layer learns linear behavior, which is controlled by the linear part of the polynomial kernel.	130
8.2	The influences of kervolution on the convergence rate. The best validation accuracy (20 epochs) and training time to target accuracy (98%) are displayed in brackets within the legend of each figure, respectively. (a) and (b) demonstrate that the kernel functions have a significant impact on the convergence speed. (c) and (d) demonstrate that the hyperparameters of kervolutional layer play less important roles than kernels. (e) and (f) shows the effects of arrangement of kervolutional layers.	134

List of Tables

5.1	Comparison on horizontal velocity with PX4Flow. (m/s)	67
6.1	Simulation performance of ATE RMSE, mean, and median error on TUM dataset. ‘Dynamic’ means whether dynamic objects are contained in the corresponding sequence.	95
6.2	Root mean square error (RMSE) comparison on the benchmark. Only instant pose estimation is accepted.	98
6.3	Mean error and maximum error comparison on the benchmark. Only instant pose estimation is accepted.	98
6.4	Real-time efficiency comparison on Kinect dataset with PC platform. All the data are give by average update rate (Hz).	99
6.5	Average runtime (ms) on the ultra low power platform.†	102
7.1	The improved performance of the UWB aided NI-SLAM (NI-SLAM + UWB) system in terms of ATE RMSE and MAE (Unit: m) . . .	116
7.2	Average runtime of the UWB aided NI-SLAM (NI+UWB) system.	117
7.3	The improved performance for UWB aided ORB-SLAM in terms of efficiency and accuracy.	120
8.1	Test Accuracy (%) of the white-box FGSM attack with L^p -norm kervolution on MNIST. 10K images are randomly selected.	130
8.2	Validation error (%) of ResNets on CIFAR-10 and CIFAR-100 without data augmentation.	137
8.3	Validation error (%) on CIFAR on different architectures with data augmentation.	137
8.4	Validation error (%) of polynomial KNN using ResNet-32 on CIFAR-10+ using different hyper parameters.	138
8.5	Training time (h) of polynomial KNN using ResNet-32 on CIFAR-10+ using different hyper parameters.	138
8.6	The top-1 and top-5 validation errors on ImageNet, with single-crop and 10-crop testing, respectively.	140

Symbols and Acronyms

Symbols

$\mathbb{R}^n$	the n -dimensional Euclidean space
$\mathbb{C}^n$	the n -dimensional Complex space
$\ \cdot\ $	the norm of a vector or matrix in Euclidean space
$ \cdot $	the number of elements in a set
$\odot$	the Hadamard (element-wise) product
$\mathbf{x}^*$	the conjugate of a vector or matrix $\mathbf{x}$
$*$	the circular cross-correlation of two vectors or matrices
$\oplus$	the circular convolution of two vectors or matrices
$\otimes$	the kervolution of two vectors or matrices
$\langle \cdot \rangle$	the nearest integer of a real number
$\langle \cdot, \cdot \rangle$	the inner product of two vectors
mod	the modular arithmetic operation
∇f	the gradient vector
$\mathcal{F}(\cdot)$	the discrete Fourier transform (DFT) of a vector or matrix
$\mathbf{F}$	the DFT matrix
$\hat{\mathbf{x}}$	the DFT alias of a vector or matrix $\mathbf{x}$
$\mathcal{F}^{-1}(\cdot)$	the inverse DFT of a vector or matrix
$\mathbf{x}_{(i)}$	the left circular shift (translation) of a vector $\mathbf{x}$ by i elements
$\mathcal{O}(\cdot)$	the compleixty of an operation
$\mathbf{C}^H$	the conjugate transpose of a complex matrix $\mathbf{C}$
$\mathcal{T}(\cdot)$	the affine transform of a vector of matrix

Acronyms

DFT	discrete Fourier transform
FFT	fast Fourier transform
KCC	Kernel Cross-Correlator
KTC	Kernel Translation Correlator
KSC	Kernel Scale Correlator
KRC	Kernel Rotation Correlator
KSRC	Kernel Scale-Rotation Correlator
CF	Correlation Flow
OF	Optical Flow
TF	Translation Flow
SRF	Joint Scale-Rotation Flow
DoF	Degree of Freedom
SLAM	Simultaneous Localization and Mapping
VSLAM	Visual Simultaneous Localization and Mapping
NI-SLAM	Non-iterative RGB-D-inertial SLAM
ConvNet/CNN	Convolutional Neural Network
KervNet/KNN	Kervolutional Neural Network
i.i.d.	independent and identically distributed
<i>a.s.</i>	almost sure convergence of a random sequence

Chapter 1

Introduction

1.1 Scope and Overview

Visual perception is the ability to interpret the environment using light reflected by the surrounding objects. Researchers have long struggled to explain how does a visual system interpret what is actually seen. For an artificial intelligence system, the task of visual perception is to mimic the human behavior and to explain the environment by machines via measurements from cameras and other aided sensors. Due to the rich information and low cost of these sensors, visual perception has been one of the hottest research topics in computer science since the last century.

Although a large amount of effort has been devoted to visual perception, there are still many problems to be solved, such as unsatisfactory performance due to the limited computational resources and unstable performance in real-time systems. One of the main reasons is that the traditional methods usually rely on large amount of training data, powerful processors, and require great efforts and time for process modeling. To this end, this thesis proposes several novel kernel learning methods for the basic tasks of visual perception, including visual tracking and recognition, etc. The problem of visual localization and mapping are also addressed. The structure of this thesis is illustrated in Figure 1.1, in which the core concept is the ‘kernel learning’. Motivated by this strategy, both theoretical tools and their applications are developed, including kernel cross-correlators (KCC), KCC-based object tracking, correlation flow (CF), non-iterative RGB-D-inertial SLAM (NI-SLAM), and kervolutional neural networks (KNN).

FIGURE 1.1: The structure of the thesis.

Those methods are designed originally for visual perceptual systems, but are also applicable to other scenarios. To construct comprehensive robotic systems, some other techniques including inertial navigation and ultra-wide band communication technology are also utilized to complement weakness of visual systems.

1.2 Major Contributions

One of the major contributions of this thesis is the complexity reduction by the proposed kernel learning methods. More specifically, they can be stated as follows:

Kernel Cross-Correlator Cross-correlator plays a significant role in many visual perception tasks, such as object detection and tracking. Beyond the linear cross-correlator, this thesis proposes a kernel cross-correlator (KCC) that breaks traditional limitations. First, by introducing the kernel trick, the KCC extends the linear cross-correlation to non-linear space, which is more robust to signal noises and distortions. Second, connections to the existing work show that the KCC provides a unified solution for correlation filters. Third, the KCC is applicable to any kernel function and is not limited to circulant structure on training data, thus it is able to predict affine transformations with customized properties. Last, by leveraging the fast Fourier transform (FFT), the KCC eliminates direct calculation of kernel vectors, thus achieves better performance at a reasonable computational cost. Comprehensive experiments on visual tracking and human activity recognition using wearable devices demonstrate its robustness, flexibility, and efficiency. The source codes of both experiments are released at <https://github.com/wang-chen/KCC>.

Correlation Flow Robust velocity and position estimation is crucial for autonomous robot navigation. The optical flow based methods for autonomous navigation have been receiving increasing attentions in tandem with the development of micro unmanned aerial vehicles. This thesis proposes a kernel cross-correlator (KCC) based algorithm to determine optical flow using a monocular camera, which is named as correlation flow (CF). CF is able to provide reliable and accurate velocity estimation, which is robust to motion blur. In addition, CF can also estimate the altitude velocity and yaw rate, which are not available by traditional methods.

Autonomous flight tests on a quadcopter show that correlation flow can provide robust trajectory estimation with very low processing power. The source codes are released at https://github.com/wang-chen/correlation_flow.

Non-iterative SLAM A non-iterative solution to RGB-D-inertial odometry system is presented. Traditional odometry methods resort to iterative algorithms which are usually computationally expensive or require well-designed initialization. To overcome this problem, a combination of a non-iterative front-end (odometry) and an iterative back-end (loop closure) for the RGB-D-inertial SLAM system is proposed. The main contribution lies in the novel non-iterative front-end, which leverages on inertial fusion and kernel cross-correlator (KCC) to match point clouds in frequency domain. Dominated by the fast Fourier transform (FFT), our method is only of complexity of $\mathcal{O}(n \log n)$, where n is the number of points. Map fusion is conducted by element-wise operations, so that both time and space complexity are further reduced. Extensive experiments show that, due to the lightweight of the proposed front-end, the framework is able to run at a much faster speed yet still with comparable accuracy with the state-of-the-art. An alternative loop closure solution using the ultra-wide band technology is also proposed.

Kervolutional Neural Network Convolutional neural network (CNN) has enabled the state-of-the-art performance in computer vision and machine learning. This part addresses the issue that little effort has been devoted to establishing convolution in non-linear space. A new operation, kervolution (**k**ernel **conv**olution), is introduced to approximate the complex behavior of the human perception system leveraging on the kernel trick. It generalizes the traditional convolution and increases the model capacity without introducing more parameters. The networks consisting of kervolutional layers are named as kervolutional neural networks (KNN). It is demonstrated that KNN achieve better accuracy and faster convergence. The source codes are released at <https://github.com/wang-chen/KNN>.

1.3 Outline of the Thesis

Chapter 1 introduces the scope of this thesis and provides an overview of the visual perceptual system as well as issues and practical concerns involved, which

essentially motivates this research work. The main contributions and the outline of the thesis are also briefly stated.

Chapter 2 briefly reviews the mathematical background of prevalent learning methods in visual perception, including the support vector machine, convolutional neural network, and kernel ridge regression. Meanwhile, the prerequisite knowledge of this thesis, i.e. kernelized correlation filter and cross-correlator, are also introduced. Some drawbacks of these learning methods are pointed out, which are also the inspiration of the work in this thesis.

Chapter 3 proposes the kernel cross-correlator (KCC), which generalizes the linear cross-correlation to kernel space. This extends the equivariance of translation to affine transforms and makes the prediction robust to signal noises and occlusion. By specifying the training samples, KCC becomes the kernel translation correlator (KTC), kernel scale correlator (KSC), and kernel rotation correlator (KRC), respectively. The theoretical connections to existing works, including MOSSE and KCF are also presented. In summary, MOSSE is a linear case of KCC, while KCF is a special case of KTC for non-weighted kernel functions.

Chapter 4 proposes a KTC and KSC-based visual tracker. It first briefly reviews the state-of-the-art visual trackers and points out that the existing scale-invariant trackers are achieved via multiple translation-invariant trackers on different scales. This chapter proposes a joint update rule for KTC and KSC to predict the translational motion and scale variance simultaneously. The extensive experiments on the OTB dataset show that the KSC and KTC improve the performance dramatically.

Chapter 5 addresses the issue that the computational complexity will be doubled when both scale and rotation variance are presented. This chapter proposes a joint kernel scale-rotation correlator (KSRC) that retains the original complexity of $\mathcal{O}(n \log n)$ and the robustness to signal noises. The KTC and KSRC are applied to velocity estimation (named as *correlation flow*) for micro quad-copters, which are real-time systems and are very sensitive to computational power. Compared with PX4Flow, correlation flow reduces the estimation error by more than 50%. Autonomous flight and hovering tests demonstrate that correlation flow is able to provide robust trajectory estimation at very low computational cost.

Chapter 6 focuses on the application of KCC in simultaneous localization and mapping (SLAM) system. Leveraging on the highly non-linear optimization, the existing visual SLAM algorithms need iterative methods, such as gradient descent and Gaussian-Newton to find a numerical solution. However, they need well-designed initialization and cannot guarantee the global optimum. This chapter proposes to combine a non-iterative front-end (odometry) with an iterative back-end (loop closure) for the RGB-D-inertial SLAM system (named as NI-SLAM). The novel non-iterative front-end leverages on inertial fusion and KCC to match point clouds in frequency domain. Extensive experiments show that, due to the lightweight of the proposed front-end, the framework is able to run at a much faster speed yet still with comparable accuracy with the state-of-the-art.

Chapter 7 presents an alternative solution to the back-end of NI-SLAM system using the ultra wide-band (UWB) communication technology. With UWB aides, the NI-SLAM is able to correct the inevitable visual odometry drift without the need of feature detection and matching which are time consuming. A general framework based on graph optimization is developed which consists of three parallel threads, two of which carry out the visual-inertial odometry (VIO) and UWB localization respectively. The other mapping thread integrates visual tracking constraints into a pose graph with the proposed smooth and virtual range constraints, such that an optimization is performed to provide robust trajectory estimation. Experiments show that the proposed system is able to create dense drift-free maps in real-time even running on an ultra-low power processor in featureless environments.

Chapter 8 extends the basic convolution to *kervolution* (kernel convolution), resulting in the extension of convolutional neural networks (CNN) to kervolutional neural networks (KNN). CNN has enabled the state-of-the-art performance in many computer vision applications. However, little effort has been devoted to establishing convolution in non-linear space. The existing works mainly leverage on the activation layers, which can only provide point-wise non-linearity. To solve this problem, a new operation, kervolution (**kernel convolution**), is introduced to approximate complex behaviors of human perception systems leveraging on the kernel trick. It generalizes convolution, enhances the model capacity, and captures higher order interactions of features, via patch-wise kernel functions, but without introducing additional parameters. The extensive experiments show that the kervolutional

neural networks (KNN) achieve better accuracy and faster convergence in visual recognition than the baseline CNN.

Chapter 9 summarizes the thesis and envisions some future work.

Chapter 2

Literature Review

This chapter reviews the mathematical background of the most important learning algorithms in visual perception, i.e. support vector machine (SVM) (Section 2.1), convolutional neural network (CNN) (Section 2.2), and kernel ridge regression, while the application-oriented related works, including correlation filter, visual tracking, optical flow, visual simultaneous localization and mapping, and image recognition will be reviewed from Chapter 3 to Chapter 8, respectively. The drawbacks of existing learning methods will be pointed out in Section 2.3, which is also the inspiration of the work in this thesis.

The learning approaches can be categorized into two classes according to the objective, i.e. structural risk minimization (SRM) and empirical risk minimization (ERM). The learning approaches that are designed to minimize error on the training dataset is called the ERM of which the most common example is the convolutional neural networks (CNN). On the other hand, the SRM approaches root in the statistical learning theory, and the most common example is the support vector machine (SVM) which is to minimize an upper bound of the generalization error. The mathematical background of SVM and CNN will be reviewed, respectively.

2.1 Support Vector Machine

The support vector machine (SVM) is a kind of pattern classifier that has shown good generalization on many visual perceptual tasks [1], including face detection,

verification, and recognition [2], object detection and recognition [3], speech/speaker verification, and recognition [4], and information/image retrieval [5], etc.

The current standard SVM is first proposed to create nonlinear classifiers via the kernel trick in the context of optimal margin by Boser et al. [6] and later was recognized as SVM by Cortes and Vapnik [7]. Unlike neural networks, which minimize the empirical training error, SVM aims at minimizing an upper bound of the generalization error through maximizing the margin between the separating hyperplane and training data. SVM is known to generalize well even in high dimensional spaces under small training sample conditions. It is one of the most prevalent methods in machine learning, especially before the deep neural networks are found to be performed extremely well on ImageNet [8]. Nevertheless, SVM still plays an important role in the machine learning family, due to its good interpretability.

2.1.1 Hard SVM

The basic idea of the SVM is to find a hyperplane as the decision plane, which separates the positive (+1) and negative (−1) classes with the largest margin. Let $S = \{(\mathbf{x}_i, y_i)\}$ ($i = 0, \dots, N-1$) be a training set of examples, where each $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \{+1, -1\}$. Therefore, if the training is linearly separable, there exists a hyperplane $(\mathbf{w}, b)$, such that $y_i = \text{sign}(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ for all i , i.e.,

$$\forall i \in [N], y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 0. \quad (2.1)$$

All hyperplanes $(\mathbf{w}, b)$ that satisfy this condition are ERM hypotheses. The hard SVM is to find the best hyperplane, that separates the training set with the largest possible margin, which is defined to be the minimal distance between the training points and the hyperplane.

Recall that the distance between a point $\mathbf{x}$ and a hyperplane $(\mathbf{w}, b)$ is $|\langle \mathbf{w}, \mathbf{x} \rangle + b|/\|\mathbf{w}\|$. To ensure the uniqueness of the solution, we set $\|\mathbf{w}\| = 1$, hence the distance of a point $\mathbf{x}$ to the hyperplane $(\mathbf{w}, b)$, where $\|\mathbf{w}\| = 1$, is $|\langle \mathbf{w}, \mathbf{x} \rangle + b|$.

Therefore, the closest point in the training set to the separating hyperplane is $\min_{i \in [N]} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b|$, and the hard SVM is

$$\underset{(\mathbf{w}, b): \|\mathbf{w}\|=1}{\operatorname{argmax}} \min_{i \in [N]} |\langle \mathbf{w}, \mathbf{x}_i \rangle + b| \quad \text{s. t. } \forall i, y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 0, \quad (2.2)$$

which is equivalent to

$$\operatorname{argmax}_{(\mathbf{w}, b): \|\mathbf{w}\|=1} \min_{i \in [N]} y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b). \quad (2.3)$$

It is difficult to solve (2.3) directly, hence the hard SVM is usually solved as another equivalent quadratic optimization problem, i.e.,

$$\begin{aligned} (\mathbf{w}_0, b_0) &= \operatorname{argmin}_{(\mathbf{w}, b)} \|\mathbf{w}\|^2 \text{ s.t. } \forall i, y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \\ \mathbf{w}^* &= \frac{\mathbf{w}_0}{\|\mathbf{w}_0\|}, \quad \mathbf{b}^* = \frac{b_0}{\|\mathbf{w}_0\|}, \end{aligned} \quad (2.4)$$

where $(\mathbf{w}^*, \mathbf{b}^*)$ is the final solution.

Proof. Let $(\mathbf{w}^*, \mathbf{b}^*)$ be a solution of (2.3) and assume the margin achieved by $(\mathbf{w}^*, \mathbf{b}^*)$ is γ^* . Hence for all i , we have $y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > \gamma^*$, i.e.,

$$y_i (\langle \mathbf{w}/\gamma^*, \mathbf{x}_i \rangle + b/\gamma^*) \geq 1. \quad (2.5)$$

Therefore, $(\mathbf{w}^*/\gamma^*, \mathbf{b}^*/\gamma^*)$ satisfies the quadratic optimization in (2.4). On the other hand, $\|\mathbf{w}_0\| \leq \|\mathbf{w}^*/\gamma^*\| = 1/\gamma^*$, hence

$$y_i \left(\left\langle \frac{\mathbf{w}_0}{\|\mathbf{w}_0\|}, \mathbf{x}_i \right\rangle + \frac{b_0}{\|\mathbf{w}_0\|} \right) \geq \frac{y_i}{\|\mathbf{w}_0\|} (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \frac{1}{\|\mathbf{w}_0\|} \geq \gamma^*, \quad (2.6)$$

which is equivalent to (2.5). Since $\left| \frac{\mathbf{w}_0}{\|\mathbf{w}_0\|} \right| = 1$, we know $\left(\frac{\mathbf{w}_0}{\|\mathbf{w}_0\|}, \frac{b_0}{\|\mathbf{w}_0\|} \right)$ is an optimal solution to (2.3). More details about its derivation can be found in the textbook of Shalev-Shwartz and Ben-David [9]. $\square$

It is more convenient to add one more feature to each instance of $\mathbf{x}_i$. This increases the dimension to $n + 1$ but the bias term b is not needed anymore. Intuitively, the homogenous case of (2.4) can be obtained by setting the bias term b to zero, i.e.

$$\min_{\mathbf{w}} \|\mathbf{w}\|^2 \text{ s.t. } \forall i, y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1. \quad (2.7)$$

2.1.2 Soft SVM

The assumption of hard SVM that the training set is linearly separable is quite strong. The soft SVM is a relaxation of hard SVM that can be applied even if the training set is not linearly separable.

A natural relaxation is to allow the constraint in (2.4) to be violated. To this end, we introduce the nonnegative variables ξ_i and replace each constraint $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ by the constraint $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$, where ξ_i measures how much the constraint is being violated. Soft SVM jointly minimizes the norm of $\mathbf{w}$, that corresponds to the margin, and the average of ξ_i , that corresponds to the violations. To trade-off the two terms, a parameter λ is introduced, which leads to the soft SVM optimization problem (2.8).

$$\begin{aligned} \min_{(\mathbf{w}, b, \xi)} & \left(\lambda \|\mathbf{w}\|^2 + \frac{1}{N} \sum \xi_i \right) \\ \text{s. t. } & \forall i, y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \text{ and } \xi_i > 0. \end{aligned} \quad (2.8)$$

Since ξ_i is nonnegative, it would be 0 if $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$ and would be $1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$ otherwise. Therefore, the optimization in (2.8) will always result in $\xi_i = \max\{0, 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)\}$, which is the hinge loss $\xi_i = l^{\text{hinge}}((\mathbf{w}, b), (\mathbf{x}_i, y_i))$. We define the average hinge loss on the training set S as $L_S^{\text{hinge}}((\mathbf{w}, b))$, thus the optimization can be rewritten as

$$\min_{(\mathbf{w}, b)} \left(\lambda \|\mathbf{w}\|^2 + L_S^{\text{hinge}}((\mathbf{w}, b)) \right) \quad (2.9)$$

It is also more convenient to convert (2.9) to the homogenous case, where the bias b is set to zero, which yields the following optimization problem (2.10).

$$\min_{(\mathbf{w}, b)} \left(\lambda \|\mathbf{w}\|^2 + L_S^{\text{hinge}}(\mathbf{w}) \right), \quad (2.10)$$

where

$$L_S^{\text{hinge}}(\mathbf{w}) = \frac{1}{N} \sum_i \max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\}. \quad (2.11)$$

Algorithm 1: SGD for Solving the Soft SVM (2.12)

```

parameter:  $T$ ;
initialization:  $\theta^{(1)} = \mathbf{0}$ ;
for  $t = 1, \dots, T$  do
     $\mathbf{w}^{(t)} = \frac{1}{\lambda_t} \theta^{(t)}$ ;
    Choose  $i$  randomly from  $[N]$ ;
    if  $y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle \geq 1$  then
         $\theta^{(t+1)} = \theta^{(t)}$ ;
    else
         $\theta^{(t+1)} = \theta^{(t)} + y_i \mathbf{x}_i$ ;
Result:  $\bar{\mathbf{w}} = \frac{1}{T} \sum_{t=1}^T \mathbf{w}^{(t)}$ 

```

2.1.2.1 Implementing SVM

Solving the optimization problem (2.7) and (2.10) requires iterative methods. Since there may exist a large number of training samples, the stochastic gradient descent (SGD) [10] is widely used, which selects the training samples randomly during each iteration, resulting in faster optimization. We update the parameter λ to $\frac{\lambda}{2}$ to get a more compact gradient of $\mathbf{w}$, resulting in the optimization problem of (2.12) that is naturally equivalent to (2.10).

$$\min_{(\mathbf{w}, b)} \left(\frac{\lambda}{2} \|\mathbf{w}\|^2 + \frac{1}{N} \sum_i \max\{0, 1 - y_i \langle \mathbf{w}, \mathbf{x}_i \rangle\} \right). \quad (2.12)$$

Taking the first derivation of (2.12) with respect to $\mathbf{w}$ and setting it to zero, we have the following update rule.

$$\mathbf{w}^{(t+1)} = -\frac{1}{\lambda t} \sum_{j=1}^t \mathbf{v}_j, \quad (2.13)$$

where t is the number of randomly selected training samples during each iteration and $\mathbf{v}_j = \mathbf{0}$ if $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1$ and $\mathbf{v}_j = -y_j \mathbf{x}_j$ otherwise. Denote $\theta^{(t)} = -\sum_{j=1}^t \mathbf{v}_j$, the SGD for solving the soft SVM can be described in Algorithm 1.

2.1.3 Kernel SVM

The soft SVM alleviates the linear non-separable learning problem, while there exist many cases that the training set is not separable in the original training space. For example, consider the integer domain points $x \in \mathbb{Z}$, where the labels are $+1$ for all x such that $|x| > 2$ and -1 otherwise. To solve this problem, we define a mapping function $\phi : \mathbb{R} \mapsto \mathbb{R}^2$ as $\phi(x) = (x, x^2)$, so that the data can be easily separated using the function $y = \text{sign}(\langle \mathbf{w}, \phi(x) \rangle - b)$, where $\mathbf{w} = (0, 1)$ and $b = 5$.

The range of the mapping function ϕ is called *feature space*. The success of this paradigm depends on a good feature space for a given learning task. In general, any feature mapping function may be chosen, resulting in a higher dimensional feature space, even an infinite space, e.g. the Hilbert space. Intuitively, if the range of ϕ is too large, we need more samples in order to learn a hyperplane, and the computational complexity is also too large to calculate the features for each sample. To solve this problem, the kernel trick is introduced in soft SVM.

The kernel function is to describe the inner product of two samples in the feature space, i.e.,

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle. \quad (2.14)$$

Before applying the kernel function, we need the following lemma.

Lemma 2.1.1 (Representer Theorem, Schölkopf et al. [11]). Assume that ϕ is a mapping function $\phi : \mathbb{R}^n \mapsto \mathbb{R}^d$. Then, there exists a vector $\mathbf{a} \in \mathbb{R}^N$ such that

$$\mathbf{w} = \sum_{i=0}^{N-1} a_i \phi(\mathbf{x}_i) \quad (2.15)$$

is an optimal solution to the objective (2.10).

On the basis of the representer theorem, we can optimize (2.10) with respect to the coefficients $\mathbf{a}$ instead of $\mathbf{w}$. First of all, we have

$$\langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle = \left\langle \sum_{j=0}^{N-1} a_j \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \right\rangle = \sum_{j=0}^{N-1} a_j \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \rangle \quad (2.16)$$

and

$$\|\mathbf{w}\|^2 = \left\langle \sum_{j=0}^{N-1} a_j \phi(\mathbf{x}_j), \sum_{j=0}^{N-1} a_j \phi(\mathbf{x}_j) \right\rangle = \sum_{i,j=0}^{N-1} a_i a_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle. \quad (2.17)$$

Substitute (2.16) and (2.17) into (2.10), the soft SVM becomes

$$\min_{\mathbf{a} \in \mathbb{R}^N} \left(\lambda \mathbf{a}^T \mathbf{K} \mathbf{a} + \frac{1}{N} \sum_{i=0}^{N-1} \max\{0, 1 - y_i (\mathbf{K} \mathbf{a})_i\} \right), \quad (2.18)$$

where $\mathbf{K}$ s.t. $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ is the *kernel matrix* and $(\mathbf{K} \mathbf{a})_i$ is the i^{th} element of the vector $\mathbf{K} \mathbf{a}$. In this sense, we may calculate the kernel function (2.14) instead of explicitly calculating the feature mapping ϕ .

The prediction can also be calculated via the coefficients $\mathbf{a}$ and the kernel function.

$$y(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle = \sum_{j=0}^{N-1} a_j \langle \phi(\mathbf{x}_j), \phi(\mathbf{x}) \rangle = \sum_{i=0}^{N-1} a_i \kappa(\mathbf{x}_i, \mathbf{x}). \quad (2.19)$$

2.1.3.1 Implementing Kernel SVM

To tackle the soft SVM with kernels, we maintain the coefficients $\mathbf{a}$ instead of $\mathbf{w}$ which may be of infinite dimensions. Similar to the derivation in Section 2.1.2.1, we may obtain the update rule of $\mathbf{a}$ by setting its first derivative of (2.18) to zero, i.e.,

$$a_j^{(t+1)} = -\frac{1}{\lambda t} \sum_{j=1}^t v_j, \quad (2.20)$$

where $v_j = 0$ if $y_j \sum_{i=1}^N \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 1$ and $v_j = -y_j$ otherwise. Therefore, a simple SGD for solving the soft SVM can be described as in Algorithm 2 [12].

Many other algorithms have been suggested for solving the SVM. Similarly, those methods involve expensive matrix operations such as the Cholesky decomposition of a large submatrix of the kernel matrix, that requires large memory. For practitioners who would like to develop their own SVM classifier, coding these algorithms is very difficult. Although SVM generalizes not well in some large dataset such as ImageNet [13], where the convolutional neural network is preferred nowadays, it

Algorithm 2: SGD for Solving the Soft SVM with kernels (2.18)

parameter: T ;

initialization: $\beta^{(1)} = \mathbf{0}$;

for $t = 1, \dots, T$ **do**

$\mathbf{a}^{(t)} = \frac{1}{\lambda_t} \beta^{(t)}$;

 Choose i randomly from $[N]$;

if $y_i \sum_{j=1}^N \kappa(\mathbf{x}_i, \mathbf{x}_j) < 1$ **then**

$\beta_i^{(t+1)} = \beta_i^{(t)} + y_i$;

for all $j \neq i$ **do**

$\beta_j^{(t+1)} = \beta_j^{(t)}$

Result: $\bar{\mathbf{a}} = \frac{1}{T} \sum_{t=1}^T \mathbf{a}^{(t)}$

still plays important roles in the development of machine learning. Its mathematical derivations are quite useful for the work in this thesis, especially the kernel cross-correlator in Chapter 3 and the convolutional neural network in Chapter 8.

2.2 Convolutional Neural Network

The convolutional neural network (CNN) that is first proposed by LeCun et al. [14] is a specialized kind of artificial neural networks for processing data. It has been tremendously successful in numerous practical applications, such as image classification [15], object detection [16], the game of go [17], etc.

Usually, a feed forward network contains several kinds of layers, e.g., the convolution, pooling, activation, and fully connected layers, etc. A very simple example, LeNet-5 is illustrated in Figure 2.1, which is designed for hand-written digit recognition (Figure Source: Lecun et al. [18]). Each layer can be described as a function, whose input is the output of its previous layer. For example, we may have three layers, corresponding to three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ that are connected in a chain. This forms the output:

$$f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x}))). \quad (2.21)$$

The overall chain length is the *depth* of this model.

FIGURE 2.1: An example of CNN (LeNet-5)

The name of CNN indicates that it mainly employs the convolutional layers as its main operation. This section mainly describes the convolution and the equivalence/invariance of an operator. Note that the convolution used in a convolutional neural network sometimes does not correspond precisely to the definition of convolution as in pure mathematics. We will describe the circular convolution function, for which the convolution theorem can be applied. A similar operation, cross-correlation, that is sometime implemented instead of convolution, will also be described. Cross-correlation is also the basics of the work in this thesis.

2.2.1 Cross-Correlation & Convolution

In signal processing, *cross-correlation* is a measure of similarity of two signals as a function of the displacement of one relative to the other [19]. For discrete real valued signals, *convolution* is different from cross-correlation in the reversed signals. Denote two real finite discrete signals as $\mathbf{x}, \mathbf{h} \in \mathbb{R}^n$, the *circular cross-correlation* $(*)$ is defined in (2.22).

$$(\mathbf{x} * \mathbf{h})[m] = \sum_{i=0}^{n-1} \mathbf{x}[i] \mathbf{h}[(m+i)_{\text{mod } n}] = \mathbf{x}^T \mathbf{h}_{(m)}, \quad (2.22)$$

where the subscript $\cdot_{(m)}$ denotes the left circular rotation of a vector by m elements, the bracket $[m]$ is to access the m^{th} element of a vector, and ‘mod’ is the modular arithmetic operation.

The *circular convolution* $(\oplus)$ is defined in (2.23).

$$(\mathbf{x} \oplus \mathbf{h})[m] = \sum_{i=0}^{n-1} \mathbf{x}[i] \mathbf{h}[(m-i)_{\text{mod } n}] = \mathbf{x}^T \bar{\mathbf{h}}_{(m)}, \quad (2.23)$$

FIGURE 2.2: The difference of cross-correlation and convolution on two infinite continuous signals.

where the $\bar{\mathbf{h}}$ denotes the time reversal $\mathbf{h}$. With a little ambiguity, the symbol $\bar{\mathbf{h}}_{(m)}$ in convolution is often replaced by $\mathbf{h}_{(m)}$ for simplicity, especially in CNN [20].

The difference of cross-correlation and convolution on two infinite continuous signals is illustrated in Figure 2.2. For clarification, the second signal $\mathbf{h}$ is often named as the *cross-correlator* and *filter* for cross-correlation and convolution, respectively. In visual object tracking, the cross-correlator is also called *correlation filter*. Without loss of generality, the name ‘cross-correlator’ is preferred in this thesis rather than ‘correlation filter’, since the former is designed not only for filtering purpose. Intuitively, the main point is that the sliding inner product for cross-correlation starts from the right and the correlator is not reversed, while for convolution, it starts from the left and the filter is reversed.

The definition of circular cross-correlation (2.22) and circular convolution (2.23) are useful for understanding their behaviors, while they are often calculated using discrete Fourier transform (DFT) based on the circular convolution theorem and circular cross-correlation theorem, which is also known as *convolution theorem* and *cross-correlation theorem* for short. Denote $\hat{\mathbf{x}}$ as the DFT of $\mathbf{x}$, i.e. $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{x})$.

Theorem 2.2.1 (Cross-Correlation Theorem, Bracewell [21]). The DFT of the circular cross-correlation (2.22) on two finite discrete signals is equivalent to element-wise conjugate multiplication of the DFT of individual signals, i.e.

$$\mathcal{F}(\mathbf{x} * \mathbf{h}) = \hat{\mathbf{x}} \odot \hat{\mathbf{h}}^*, \quad (2.24)$$

where the superscript $*$ denotes the complex conjugate and $\odot$ is the element-wise multiplication.

Theorem 2.2.2 (Convolution Theorem, Bluestein [22]). The DFT of the circular convolution (2.23) on two finite discrete signals is equivalent to element-wise multiplication of the DFT of individual signals, i.e.

$$\mathcal{F}(\mathbf{x} \oplus \mathbf{h}) = \hat{\mathbf{x}} \odot \hat{\mathbf{h}}. \quad (2.25)$$

Cross-correlation theorem (2.24) and convolution theorem (2.25) indicate that a circular convolution and cross-correlation of two finite sequences can be obtained via the inverse DFT of the element-wise product of the individual transforms. This is crucial for many applications, in which the DFT is often calculated by efficient fast Fourier transform (FFT), such as the Rader's algorithm [23], which is only of complexity $\mathcal{O}(n \log n)$, where n is the length of the signal.

2.2.2 Equivariance and Invariance

One of the most important properties for operators is the equivariance and invariance to a transform. The CNN is invariant to minor input translation, which is one of the main reasons for the success of CNN. The definition of equivariance and invariance stated in this chapter will be useful for Chapter 3 and Chapter 8.

Definition 2.2.1 (Equivariance, Gaunce Jr et al. [24]). Equivariance to a transformation means that the output of an operator $f(\mathbf{x})$ on a transformed input $\mathcal{T}_l(\mathbf{x})$ is equal to the transform $\mathcal{T}_r$ of the operator output ($\mathcal{T}_l = \mathcal{T}_r$), i.e.

$$f(\mathcal{T}_l(\mathbf{x})) = \mathcal{T}_r(f(\mathbf{x})). \quad (2.26)$$

Sometimes, the definition is slightly different, i.e. an operator is equivariant to a transform when the effect of the transform is detectable in the output [25]. This means that the left and right transforms in (2.26) can be different ($\mathcal{T}_l \neq \mathcal{T}_r$).

Definition 2.2.2 (Invariance, Gaunce Jr et al. [24]). Invariance to a transformation means that the output of an operator $f(\mathbf{x})$ on a transformed input $\mathcal{T}(\mathbf{x})$ doesn't change the operator output, i.e.

$$f(\mathcal{T}(\mathbf{x})) = f(\mathbf{x}) \quad (2.27)$$

Theorem 2.2.3. Cross-correlation (2.22) and convolution (2.23) are equivariant to translation, i.e.

$$(\mathbf{x} * \mathbf{h})_{(i)} = \mathbf{x}_{(i)} * \mathbf{h}, \quad (2.28)$$

$$(\mathbf{x} \oplus \mathbf{h})_{(i)} = \mathbf{x}_{(i)} \oplus \mathbf{h}, \quad (2.29)$$

where $\mathbf{x}_{(i)}$ denotes the circular translation of $\mathbf{x}$ by i elements.

Proof. From the definition of (2.22), we know that the m^{th} element of $(\mathbf{x} * \mathbf{h})_{(i)}$ is $\sum_{j=0}^{n-1} \mathbf{x}[j] \mathbf{h}[(m + j + i) \bmod n] = \sum_{j=0}^{n-1} \mathbf{x}[(i + j) \bmod n] \mathbf{h}[(m + j) \bmod n] = \mathbf{x}_{(i)}^T \mathbf{h}_{(m)}$, which is the m^{th} element of $\mathbf{x}_{(i)} * \mathbf{h}$. This completes the proof. $\square$

The proof for convolution can be done similarly. Therefore, the convolutional layers of an artificial neural network are equivariant to input translation. The translational invariance of convolutional neural networks is achieved by concatenating pooling layers to convolutional layers [20].

2.3 Kernel Ridge Regression

It has been shown that the *kernel trick* is introduced to SVM for non-linear classification. This chapter will show that the kernel trick is also helpful for non-linear ridge regression.

Assume the training samples $\mathbf{x}_i$ ($\mathbf{x}_i \in \mathbb{R}^n$, $i = 0, 1, \dots, N - 1$) have different regression targets $\mathbf{y}_i$. To obtain a non-linear model, we introduce a non-linear feature mapping function $\phi(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^d$ ($d \gg n$) on the training samples, as in Section 2.1.3. The kernel trick is to bypass explicit calculation of feature mapping

$\phi(\mathbf{x})$, where the dimensionality d is always very high or even infinite. To predict the targets of samples $\mathbf{x}_i$, we construct a ridge regression model in (2.30).

$$y_i = \mathbf{w}^T \phi(\mathbf{x}_i), \quad (2.30)$$

where the parameter $\mathbf{w}$ is determined by minimizing a regularized sum of squared error (SSE) in (2.31).

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=0}^{N-1} (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}, \quad (2.31)$$

where $\lambda > 0$ is a regularizer to prevent over-fitting. To solve the optimization problem (2.31), we set the gradient of $J(\mathbf{w})$ to zero, so that the solution $\mathbf{w}$ takes the form of a linear combination of $\phi(\mathbf{x}_i)$ with coefficients that are functions of $\mathbf{w}$:

$$\mathbf{w} = -\frac{1}{\lambda} \sum_{i=0}^{N-1} (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i) \phi(\mathbf{x}_i) = \sum_{i=0}^{N-1} a_i \phi(\mathbf{x}_i) = \Phi^T \mathbf{a}, \quad (2.32)$$

where Φ is the design matrix with $\phi(\mathbf{x}_i)^T$ as the i_{th} row. The vector $\mathbf{a} = [a_1, \dots, a_N]^T$, where a_i is defined as

$$a_i = -\frac{1}{\lambda} (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i). \quad (2.33)$$

We then reformulate this problem in terms of the vector $\mathbf{a}$, which is a *dual representation*. Substitute $\mathbf{w} = \Phi^T \mathbf{a}$ into (2.31), we have

$$J(\mathbf{a}) = \frac{1}{2} \mathbf{a}^T \Phi \Phi^T \Phi \Phi^T \mathbf{a} - \mathbf{a}^T \Phi \Phi^T \mathbf{y} + \frac{1}{2} \mathbf{y}^T \mathbf{y} + \frac{\lambda}{2} \mathbf{a}^T \Phi \Phi^T \mathbf{a}, \quad (2.34)$$

where $\mathbf{y} = [y_1, \dots, y_N]^T$. The *Gramian* matrix $\mathbf{K} = \Phi \Phi^T$ is an $N \times N$ symmetric matrix with its $(i, j)^{th}$ element

$$\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = \phi(\mathbf{x}_j)^T \phi(\mathbf{x}_i) = \mathbf{K}_{ji} = \kappa(\mathbf{x}_i, \mathbf{x}_j), \quad (2.35)$$

where we have introduced the kernel function in (2.14). Then the optimization problem (2.31) can be rewritten in terms of the *kernel matrix* $\mathbf{K}$.

$$J(\mathbf{a}) = \frac{1}{2} \mathbf{a}^T \mathbf{K} \mathbf{K} \mathbf{a} - \mathbf{a}^T \mathbf{K} \mathbf{y} + \frac{1}{2} \mathbf{y}^T \mathbf{y} + \frac{\lambda}{2} \mathbf{a}^T \mathbf{K} \mathbf{a}, \quad (2.36)$$

We again set the gradient of $J(\mathbf{a})$ with respect to $\mathbf{a}$ to zero, the solution $\mathbf{a}$ can be obtained as

$$\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}. \quad (2.37)$$

For a test sample $\mathbf{z}$, the prediction is calculated as

$$y(\mathbf{z}) = \mathbf{w}^T \phi(\mathbf{z}) = \mathbf{a}^T \Phi \phi(\mathbf{z}), \quad (2.38)$$

where $\mathbf{w}$ is replaced by (2.32). We denote $\Phi \phi(\mathbf{z}) = [\phi(\mathbf{x}_1)^T \phi(\mathbf{z}), \dots, \phi(\mathbf{x}_N)^T \phi(\mathbf{z})] = [\kappa(\mathbf{x}_1, \mathbf{z}), \dots, \kappa(\mathbf{x}_N, \mathbf{z})]$ as the kernel vector $\boldsymbol{\kappa}_{\mathbf{x}}(\mathbf{z})$ with elements $\kappa(\mathbf{x}_i, \mathbf{z})$. Then substitute (2.37) into (2.38), we have

$$y(\mathbf{x}) = \boldsymbol{\kappa}_{\mathbf{x}}^T(\mathbf{z})(\mathbf{K} + \lambda)^{-1} \mathbf{y} \quad (2.39)$$

Therefore, we see that the original formulation is recovered and the non-linear feature mapping function is calculated implicitly. This dual version of ridge regression is first proposed in [26] and more details can be found in [27].

2.3.1 Circulant Data Structure

According to (2.37), we need to calculate the inverse of a large matrix $(\mathbf{K} + \lambda \mathbf{I})$, which is sometimes online infeasible. Fortunately, if the training samples are circulant examples of each other, i.e. the design matrix Φ is a circulant matrix, we may calculate the solution in (2.37) via discrete Fourier transform (DFT). We first define the circulant matrix.

Definition 2.3.1 (Circulant Matrix, Davis [28]). An $n \times n$ matrix $\mathbf{C}$ is a circulant matrix, if each row vector of $\mathbf{C}$ is shifted one element to the left relative to the preceding row vector, i.e.

$$\mathbf{C} = \begin{bmatrix} c_0 & c_1 & \dots & c_{n-2} & c_{n-1} \\ c_1 & c_2 & c_3 & & c_0 \\ \vdots & c_3 & c_4 & \ddots & \vdots \\ c_{n-2} & & \ddots & \ddots & c_{n-3} \\ c_{n-1} & c_0 & \dots & c_{n-3} & c_{n-2} \end{bmatrix} = [\mathbf{c}_{(0)}^T, \mathbf{c}_{(1)}^T, \dots, \mathbf{c}_{(n-1)}^T]^T, \quad (2.40)$$

where the subscript $\cdot_{(m)}$ denotes the left circular rotation of a vector by m elements. Note that the circulant matrix $\mathbf{C}$ can be generated by its first row $\mathbf{c}$, since $\mathbf{c}^T = \mathbf{c}_{(0)}^T = [c_0, c_1, \dots, c_{n-1}]^T$.

Recall that $\hat{\mathbf{x}}$ is the discrete Fourier transform of $\mathbf{x}$, i.e. $\hat{\mathbf{x}} = \mathcal{F}(\mathbf{x})$. From the DFT theorem, we have $\hat{\mathbf{x}} = \mathbf{F}\mathbf{x}$, where $\mathbf{F}$ is a constant DFT matrix that is only dependent on the length of $\mathbf{x}$. For a circulant matrix, we have the following theorem.

Theorem 2.3.1 (Gray [29]). An $n \times n$ circulant matrix $\mathbf{C}$ can be generated from the discrete Fourier transform of its first row $\mathbf{c}$, i.e.

$$\mathbf{C} = \mathbf{F} \text{diag}(\hat{\mathbf{c}}) \mathbf{F}^H, \quad (2.41)$$

where $\mathbf{F}$ is an $n \times n$ DFT matrix, and the $\mathbf{F}^H$ is the complex conjugate transpose.

Assume that the training samples are circulant to each other, i.e. $\mathbf{x}_i = \mathbf{x}_{(i)}$, so that all the samples are circular shifts of the first sample $\mathbf{x}$. From (2.35), we have

$$\mathbf{K} = \begin{bmatrix} \kappa(\mathbf{x}_{(0)}, \mathbf{x}_{(0)}) & \kappa(\mathbf{x}_{(0)}, \mathbf{x}_{(1)}) & \cdots & \kappa(\mathbf{x}_{(0)}, \mathbf{x}_{(N-1)}) \\ \kappa(\mathbf{x}_{(1)}, \mathbf{x}_{(0)}) & \kappa(\mathbf{x}_{(1)}, \mathbf{x}_{(1)}) & \cdots & \kappa(\mathbf{x}_{(1)}, \mathbf{x}_{(N-1)}) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_{(N-1)}, \mathbf{x}_{(0)}) & \kappa(\mathbf{x}_{(N-1)}, \mathbf{x}_{(1)}) & \cdots & \kappa(\mathbf{x}_{(N-1)}, \mathbf{x}_{(N-1)}) \end{bmatrix} \quad (2.42)$$

Therefore, we can see that if $\kappa(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}_{(i)}, \mathbf{x}'_{(i)})$, where $0 \leq i \leq N-1$, then $\mathbf{K}^H$ is an $N \times N$ circulant matrix. Therefore, we have the following theorem.

Theorem 2.3.2 (Henriques et al. [30]). The dual solution (2.37) of the ridge regression can be converted to element-wise operation (2.43) via fast Fourier transform

$$\hat{\mathbf{a}} = \frac{\hat{\mathbf{y}}}{\hat{\kappa}_{\mathbf{x}}(\mathbf{x}) + \lambda}, \quad (2.43)$$

where $\kappa_{\mathbf{x}}(\mathbf{x}) = [\kappa(\mathbf{x}, \mathbf{x}_0), \kappa(\mathbf{x}, \mathbf{x}_1), \dots, \kappa(\mathbf{x}, \mathbf{x}_{N-1})]$ and $\mathbf{x} = \mathbf{x}_0$, if the conditions (2.44) and (2.45) are satisfied.

$$\mathbf{x}_i = \mathbf{x}_{(i)} \quad (2.44)$$

$$\kappa(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}_{(i)}, \mathbf{x}'_{(i)}) \quad (2.45)$$

Proof. Multiply the DFT matrix $\mathbf{F}$ on both side of (2.37), so that we have

$$\mathbf{F}\mathbf{a} = \hat{\mathbf{a}} = \mathbf{F}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y} \quad (2.46)$$

From the above analysis, we know that if (2.44) and (2.45) hold, then $\mathbf{K}^H$ is a circulant matrix. Since $\mathbf{F}\mathbf{F}^H = \mathbf{I}$, from Theorem 2.3.1, we have

$$\begin{aligned} \hat{\mathbf{a}} &= \mathbf{F}(\mathbf{F}^H \text{diag}(\hat{\boldsymbol{\kappa}}_{\mathbf{x}}(\mathbf{x}))\mathbf{F} + \lambda\mathbf{I})^{-1}\mathbf{y} \\ &= \mathbf{F}(\mathbf{F}^H(\text{diag}(\hat{\boldsymbol{\kappa}}_{\mathbf{x}}(\mathbf{x})) + \lambda\mathbf{I})\mathbf{F})^{-1}\mathbf{y} \\ &= \mathbf{F}\mathbf{F}^H(\text{diag}(\hat{\boldsymbol{\kappa}}_{\mathbf{x}}(\mathbf{x})) + \lambda\mathbf{I})^{-1}\mathbf{F}\mathbf{y} \\ &= (\text{diag}(\hat{\boldsymbol{\kappa}}_{\mathbf{x}}(\mathbf{x})) + \lambda\mathbf{I})^{-1}\hat{\mathbf{y}}. \end{aligned} \quad (2.47)$$

Since the inverse of a diagonal matrix only contains independent diagonal element-wise inversion, we rewrite (2.47) as

$$\hat{\mathbf{a}} = \frac{\hat{\mathbf{y}}}{\hat{\boldsymbol{\kappa}}_{\mathbf{x}}(\mathbf{x}) + \lambda}, \quad (2.48)$$

where $\div$ is the vector element-wise division. This completes the proof. $\square$

Theorem 2.3.2 is useful for signal translation prediction and is known as *kernelized correlation filter* (KCF). It has been widely used for visual object tracking due to its accuracy and robustness to image noise and occlusion. One of the main advantages of Theorem 2.3.2 is that it replaces the inversion of a large matrix by element-wise division via FFT, whose complexity is only of $\mathcal{O}(N \log N)$.

However, the limitation of Theorem 2.3.2 is also obvious. It requires that the training data samples be circulant to each other (Equation (2.44)) and the kernel function treats all dimensions of data equally (Equation (2.45)). These two strong conditions sometimes cannot be satisfied, which excludes its usages in many applications. In this thesis, a new theorem is developed in Chapter 3 based on cross-correlation, so that any training samples and kernel functions can be applied, meaning that any affine transforms can be predicted. Moreover, its computational complexity is still bounded by $\mathcal{O}(N \log N)$. To this end, we first introduce the *cross-correlator*, which is the basics of Chapter 3.

2.4 Cross-Correlator

Cross-correlation is often applied to computer vision for visual tracking. Given known correlation outputs $\mathbf{g}_i$ ($\mathbf{g}_i \in \mathbb{R}^n, i = 0, 1, \dots, N-1$), the objective is to find a correlator $\mathbf{h}$ that maps signals $\mathbf{x}_i$ to their targets $\mathbf{g}_i$ ($\mathbf{g}_i \in \mathbb{R}^n$), i.e.

$$\mathbf{g}_i = \mathbf{x}_i * \mathbf{h}. \quad (2.49)$$

To find the correlator $\mathbf{h}$, the sum of squared error is minimized in Fourier domain to leverage the simple element-wise operations.

$$\min \mathbf{J}(\hat{\mathbf{h}}) = \min \sum_{i=0}^{N-1} \|\hat{\mathbf{g}}_i - \hat{\mathbf{x}}_i \odot \hat{\mathbf{h}}^*\|^2 \quad (2.50)$$

The function in (2.50) is real valued, positive, and convex, hence it has only single optima. Therefore, we have the following theorem.

Theorem 2.4.1 (Bolme et al. [31]). The problem (2.50) has a closed-form solution (2.51) in complex domain.

$$\hat{\mathbf{h}} = \frac{\sum_{i=0}^{N-1} \hat{\mathbf{x}}_i \odot \hat{\mathbf{g}}_i^*}{\sum_{i=0}^{N-1} \hat{\mathbf{x}}_i \odot \hat{\mathbf{x}}_i^*}, \quad (2.51)$$

where all the operations are element-wise.

Proof. This proof is inspired by [32]. Take the first derivative of $\mathbf{J}(\hat{\mathbf{h}})$ with respect to $\hat{\mathbf{h}}^*$ and set it to zero, then we have

$$\nabla \mathbf{J}_{\hat{\mathbf{h}}^*} = \frac{\partial}{\partial \hat{\mathbf{h}}^*} \sum_{i=0}^{N-1} (\hat{\mathbf{x}}_i \odot \hat{\mathbf{h}}^* - \hat{\mathbf{g}}_i) \odot (\hat{\mathbf{x}}_i \odot \hat{\mathbf{h}}^* - \hat{\mathbf{g}}_i)^* = 0, \quad (2.52)$$

where all the elements of $\nabla \mathbf{J}_{\hat{\mathbf{h}}^*}$ can be computed element-wisely. Hence

$$0 = \frac{\partial}{\partial \hat{\mathbf{h}}^*} \sum_{i=0}^{N-1} (\hat{\mathbf{x}}_i \odot \hat{\mathbf{h}}^* \odot \hat{\mathbf{x}}_i^* \odot \hat{\mathbf{h}} - \hat{\mathbf{x}}_i \odot \hat{\mathbf{h}}^* \odot \hat{\mathbf{g}}_i^* - \hat{\mathbf{g}}_i^* \odot \hat{\mathbf{x}}_i^* \odot \hat{\mathbf{h}} + \hat{\mathbf{g}}_i \odot \hat{\mathbf{g}}_i^*). \quad (2.53)$$

The derivative is obtained by taking $\hat{\mathbf{h}}$ as a variable that is independent of $\hat{\mathbf{h}}^*$, thus

$$0 = \sum_{i=0}^{N-1} (\hat{\mathbf{x}}_i \odot \hat{\mathbf{x}}_i^* \odot \hat{\mathbf{h}} - \hat{\mathbf{x}}_i \odot \hat{\mathbf{g}}_i^*). \quad (2.54)$$

Then we can solve for $\hat{\mathbf{h}}$ as

$$\hat{\mathbf{h}} = \frac{\sum_{i=0}^{N-1} \hat{\mathbf{x}}_i \odot \hat{\mathbf{g}}_i^*}{\sum_{i=0}^{N-1} \hat{\mathbf{x}}_i \odot \hat{\mathbf{x}}_i^*}, \quad (2.55)$$

which completes the proof. $\square$

Solution (2.51) is a type of cross-correlator (correlation filter) and is known as *minimum output sum of squared error* (MOSSE), which is widely used in visual object tracking. Compared to KCF (2.43), MOSSE is linear, intuitive, and efficient, but sensitive to image noise and occlusion. To solve this problem, we will extend the MOSSE to kernel cross-correlator (KCC) in Chapter 3, so that it can predict any affine transforms with any kernel functions, which cannot be achieved by KCF.

2.5 Conclusion

This chapter introduces the mathematical background of prevalent learning methods in visual perception, especially the SVM, CNN, and kernel ridge regression. By applying the circulant data structure on the kernel ridge regression, the KCF is also derived. Meanwhile, the drawbacks of these kernel learning methods are also pointed out. To solve these problems, the prerequisite knowledge of this thesis is also introduced, i.e. cross-correlator, which is the basics of Chapter 3.

Chapter 3

Kernel Cross-Correlator

3.1 Introduction

As mentioned in Section 2.2.1, cross-correlator or correlation filter (CF) usually refers to the function or device to compute cross-correlation for signals. It has been widely used in visual perceptual tasks such as object detection, recognition, and tracking. The recent extensions in visual tracking, which are usually named as correlation filters, e.g., minimum output sum of squared error (MOSSE) [31], rekindle the interest of real-time applications due to their robust translation prediction. One of the main advantages of CF is the efficient calculation with a complexity of $\mathcal{O}(n \log n)$ in Fourier domain versus $\mathcal{O}(n^3)$ in spatial domain, where n is the length of a signal. The kernelized correlation filter (KCF) brings kernel ridge regression to circulant structure and obtains impressive performance on object tracking [30].

In spite of the theoretical improvements, the existing CFs are still limited in some aspects. First, only the translational changes of the input can be estimated by viewing the corresponding shift of the maximum output. The estimation for other transforms are usually achieved by combining multiple CFs with different input transforms, e.g. the scale filters [33]. Second, KCF speeds up the ridge regression but is only applicable for training data with circulant structure (each sample is circular translation of others) and non-weighted kernel functions, which excludes its usage in other applications where the aforementioned conditions turn out to be invalid [30]. To address these problems, a kernel cross-correlator (KCC) will be proposed. In summary, the main contributions are:

- By introducing the kernel trick, the KCC extends the cross-correlation to non-linear space, which is more robust to signal noises and distortions;
- Compared to KCF, KCC can be applied to any data structure using any kernel functions, hence, its properties are customizable and can be used to predict any affine transforms, especially the rotation, scale, and translation;
- KCC generalizes the existing CFs and provides unified solutions, which improves the performance of visual tracking while still keeps efficiency;
- KCC provides a similarity measurement for signals with transform. The extensive experiments on human activity recognition using wearable devices demonstrate its robustness and flexibility.

Considering that the CFs have shown their great capabilities in numerous applications including template matching, visual tracking, face verification, etc., we believe that KCC will also play an important role in those fields.

3.2 Related Work

One of the earliest works is the synthetic discriminant filter (SDF) that describes a signal as a linear combination of orthonormal basis function expansion, so that a single averaged spatial filter can be obtained from a weighted linear combination of these functions [34]. The unified synthetic discriminant filters (USDF) provide a general basis function and hyperspace description for SDFs and show the generality of the correlation matrix observation space [35]. Later, it is proved that SDF is only suitable for processing white noises and a minimum-variance synthetic discriminant function (MVSDF) is designed for colored noises [36]. To obtain sharper correlation peaks, the minimum average correlation energy (MACE) filter is proposed to minimize the energy of the correlation output which facilitates the translational target detection [37]. The optimal trade-off filter (OTOF) is designed to ensure the sharpness of the output against noise robustness and higher efficiency [38].

Afterwards, researchers' interests have been focused on distortion performance. For instance, the unconstrained correlation filter (UCF) removes the usual hard

constraints on the output of SDF [39]. This exhibits superior distortion tolerance while still keeps the attractive features of MACE and MVSDF. The distance-classifier correlation filter (DCCF) enables the multi-class recognition by introducing a distance-classifier scheme [40]. By specifying the desired response at every location, the average synthetic exact filter (ASEF) generalizes across entire training set by averaging multiple exact filters [41]. To overcome the overfitting problem of ASEF, the minimum output sum of squared error (MOSSE) filter adds a regularization term and introduces it into visual tracking [31]. Its superior speed and robustness ignite the boom and development of CF-based tracking.

Kernelized correlation filter (KCF) brings circulant training structure into kernel ridge regression [30]. This enables learning with element-wise operation instead of costly matrix inversion, providing much more robustness while still with reasonable learning speed. Multi-kernel correlation filter (MKCF) [42] extends KCF to multiple kernel case, which further improves the accuracy. To alleviate the boundary effect of CFs, zero aliasing correlation filter (ZACF) [43] introduces the zero-aliasing constraints and provides both closed-form and iterative proximal solutions by ensuring that the optimization criterion for a given CF corresponds to a linear correlation rather than a circular correlation. However, it requires heavy computation and is not suitable for real-time applications. Discriminative scale space tracking (DSST) [33] is proposed to learn multiple MOSSE on different scales, enabling estimation of both translation and scale variants, which is at the cost of repeated calculations of MOSSE.

3.3 Formulation of Kernel Cross-Correlator

In this section, a new formulation for non-linear cross-correlator based on kernel trick is presented. For the sake of simplicity, signals will be denoted as column vectors $\mathbf{z}, \mathbf{x} \in \mathbb{R}^n$. The derived results can be easily extended to matrix case.

For a non-linear feature mapping function $\varphi : \mathbb{R}^n \mapsto \mathbb{R}^d$, $d \gg n$, the kernel trick is to find the inner product of feature mapping without calculating the high dimension features explicitly. Denote the kernel function as $\kappa : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$, such that $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$. Given a signal $\mathbf{x} \in \mathbb{R}^n$ and desired output $\mathbf{g} \in \mathbb{R}^m$, the

kernel cross-correlator is defined as:

$$\hat{\mathbf{g}} = \hat{\boldsymbol{\kappa}}_{\mathbf{z}}(\mathbf{x}) \odot \hat{\mathbf{h}}^*, \quad (3.1)$$

where $\boldsymbol{\kappa}_{\mathbf{z}}(\mathbf{x})$ is the kernel vector. For single kernel case, $\boldsymbol{\kappa}_{\mathbf{z}}(\mathbf{x}) = [\cdots \kappa(\mathbf{x}, \mathbf{z}_i) \cdots]^T$, where $\mathbf{z}_i = \mathcal{T}_i(\mathbf{z}) \in \mathcal{T}(\mathbf{z}) \in \mathbb{R}^n$ is generated from the training sample $\mathbf{z}$. The transform function $\mathcal{T}(\cdot)$ is predefined to imply the training objective.

For multi-kernel case, each element of the kernel vector $\boldsymbol{\kappa}_{\mathbf{z}}(\mathbf{x})$ is a linear combination of a set of kernels κ_i , where $i = 1, 2, \cdots l$. The k_{th} element of the kernel vector is defined as $\mathbf{d}^T \mathbf{k}_k$, where $\mathbf{d} \in \mathbb{R}^l$ is the weights vector subject to $\|\mathbf{d}\|_1 = 1$, and the vector stack $\mathbf{k}_k = [\kappa_1(\mathbf{x}, \mathbf{z}_k), \cdots, \kappa_l(\mathbf{x}, \mathbf{z}_k)]^T$.

Note that the length of $\mathbf{h}$ and the target $\mathbf{g}$ are related to the number of sample-based vectors $\mathbf{z}_i$, i.e. $\mathbf{h}, \mathbf{g} \in \mathbb{R}^m$ and normally $n \neq m$. The desired output $\mathbf{g}$ can take any shape, e.g. a Gaussian function with the peak centered at the training signal. In the training stage, $\mathbf{z}_i$ can take any affine transform of $\mathbf{z}$. The corresponding pattern will be encoded into $\mathbf{h}$, so that the pattern of a test sample $\mathbf{x}$ can be predicted through the correlation output $\mathbf{g}(\mathbf{x})$. For example, if the desired output $\mathbf{g}$ is a binary vector and the single peak is located at the first element, the location of the maximum response in the correlation output $\mathbf{g}(\mathbf{x})$ indicates the test sample's pattern associated with that location in the kernel vector $\boldsymbol{\kappa}_{\mathbf{z}}(\mathbf{x})$.

Denote $\mathbf{z}^i$ as the i_{th} training sample. Note that $\mathbf{z}^i$ is different from the sample-based vector $\mathbf{z}_i$ generated from the training sample $\mathbf{z}$. To be efficient, training is conducted in the Fourier domain to take advantage of simple element-wise operation. The training objective is to find $\mathbf{h}$ that maps $\mathbf{z}^i$ to the desired output $\mathbf{g}^i$ and minimizes the sum of squared error (SSE) between the output and desired output:

$$\min_{\hat{\mathbf{h}}^*} \sum_{i=1}^s \|\hat{\boldsymbol{\kappa}}_{\mathbf{z}^i}(\mathbf{z}^i) \odot \hat{\mathbf{h}}^* - \hat{\mathbf{g}}^i\|^2 + \lambda \|\hat{\mathbf{h}}^*\|^2, \quad (3.2)$$

where s is the number of training samples. For single kernel case, $\hat{\boldsymbol{\kappa}}_{\mathbf{z}^i}(\mathbf{z}^i) = [\kappa(\mathbf{z}^i, \mathbf{z}_1^i), \cdots, \kappa(\mathbf{z}^i, \mathbf{z}_m^i)]^T$ and $\mathbf{z}_j^i$ is generated from $\mathbf{z}^i$. The second term in (3.2) is a regularization to prevent overfitting.

Theorem 3.3.1 (Kernel Cross-Correlator). The problem (3.2) has a closed form solution

$$\hat{\mathbf{h}}^* = \frac{\sum_{i=1}^s \hat{\mathbf{g}}^i \odot \hat{\boldsymbol{\kappa}}_{\mathbf{z}^i}^*(\mathbf{z}^i)}{\sum_{i=1}^s \hat{\boldsymbol{\kappa}}_{\mathbf{z}^i}^*(\mathbf{z}^i) \odot \hat{\boldsymbol{\kappa}}_{\mathbf{z}^i}(\mathbf{z}^i) + \lambda}, \quad (3.3)$$

where the operator $\dot{\div}$ denotes the element-wise division.

Proof. Solving the optimization problem (3.2) requires setting the first derivative of $\hat{\mathbf{h}}^*$ to zero, i.e.,

$$\frac{\partial}{\partial \hat{\mathbf{h}}^*} \left(\sum_{i=1}^s \|\hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i) \odot \hat{\mathbf{h}}^* - \hat{\mathbf{g}}^i\|^2 + \lambda \|\hat{\mathbf{h}}^*\|^2 \right) = 0. \quad (3.4)$$

Then by expanding the element-wise complex square ($\|a\|^2 = aa^*$), we have

$$\begin{aligned} \frac{\partial}{\partial \hat{\mathbf{h}}^*} \left[\sum_{i=0}^{N-1} \left(\hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i) \odot \hat{\mathbf{h}}^* \odot \hat{\kappa}_{\mathbf{z}^i}^*(\mathbf{z}^i) \odot \hat{\mathbf{h}} - \hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i) \odot \hat{\mathbf{h}}^* \odot \hat{\mathbf{g}}_i^* - \right. \right. \\ \left. \left. \hat{\mathbf{g}}_i \odot \hat{\kappa}_{\mathbf{z}^i}^*(\mathbf{z}^i) \odot \hat{\mathbf{h}} + \hat{\mathbf{g}}_i \odot \hat{\mathbf{g}}_i^* \right) + \lambda \hat{\mathbf{h}}^* \odot \hat{\mathbf{h}} \right] = 0 \end{aligned} \quad (3.5)$$

The derivative is obtained by taking $\hat{\mathbf{h}}$ as a variable that is independent of $\hat{\mathbf{h}}^*$, thus

$$0 = \sum_{i=0}^{N-1} \left(\hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i) \odot \hat{\kappa}_{\mathbf{z}^i}^*(\mathbf{z}^i) \odot \hat{\mathbf{h}} - \hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i) \odot \hat{\mathbf{g}}_i^* \right) + \lambda \hat{\mathbf{h}} \quad (3.6)$$

Then we can solve for $\hat{\mathbf{h}}$ as

$$\hat{\mathbf{h}} = \frac{\sum_{i=1}^s \hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i) \odot \hat{\mathbf{g}}_i^*}{\sum_{i=1}^s \hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i) \odot \hat{\kappa}_{\mathbf{z}^i}^*(\mathbf{z}^i) + \lambda}. \quad (3.7)$$

Due to the properties of complex conjugate, i.e. $a^* \cdot b^* = (a \cdot b)^*$, $a^*/b^* = (a/b)^*$, and $a^* + b^* = (a + b)^*$, (3.3) can be easily obtained from (3.7) by taking the conjugate on both sides, which completes the proof. $\square$

To further simplify the solution, we update the objective as in (3.8), which adds a weighted regularization to (3.2).

$$\min_{\hat{\mathbf{h}}^*} \sum_{i=1}^s \|\hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i) \odot \hat{\mathbf{h}}^* - \hat{\mathbf{g}}^i\|^2 + \lambda \|\sqrt{\hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i)} \odot \mathbf{h}\|^2. \quad (3.8)$$

Theorem 3.3.2 (Kernel Cross-Correlator, weighted regularization). The problem (3.8) has a closed form solution (3.9), if only one sample is available, i.e. $s = 1$.

$$\hat{\mathbf{h}}^* = \frac{\hat{\mathbf{g}}}{\hat{\kappa}_{\mathbf{z}}(\mathbf{z}) + \lambda}. \quad (3.9)$$

Proof. The solution to (3.8) can be obtained by updating the regularization of (3.7), hence, we have

$$\begin{aligned}\hat{\mathbf{h}} &= \frac{\sum_{i=1}^s \hat{\mathbf{g}}^{i*} \odot \hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i)}{\sum_{i=1}^s \hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i) \odot \hat{\kappa}_{\mathbf{z}^i}^*(\mathbf{z}^i) + \lambda \hat{\kappa}_{\mathbf{z}^i}(\mathbf{z}^i)}, \\ &= \frac{\hat{\mathbf{g}}^*}{\hat{\kappa}_{\mathbf{z}}^*(\mathbf{z}) + \lambda}.\end{aligned}\tag{3.10}$$

Then (3.9) can be obtained by taking the conjugate on both sides. $\square$

Theorem 3.3.3 (Equivariance). Kernel Cross-Correlator (3.1) is equivariant to affine transform $\mathcal{T}_i(\cdot)$ in spatial domain, i.e.

$$\mathbf{g}(\mathcal{T}_i(\mathbf{x})) = \kappa_{\mathbf{z}}(\mathcal{T}_i(\mathbf{x})) * \mathbf{h} = (\kappa_{\mathbf{z}}(\mathbf{x}) * \mathbf{h})_{(i)} = \mathbf{g}_{(i)}(\mathbf{x}),\tag{3.11}$$

if $\mathcal{T}_i(\cdot)$ is periodical.

Proof. Assume $\mathbf{g}' = \kappa_{\mathbf{z}}(\mathcal{T}_j(\mathbf{z})) * \mathbf{h}$, we have

$$\mathbf{g}' = [\cdots, \kappa(\mathcal{T}_j(\mathbf{z}), \mathbf{z}_i), \cdots] * \mathbf{h}.\tag{3.12}$$

Since $\mathcal{T}_j(\cdot)$ is periodical and $\mathbf{z}_i = \mathcal{T}_i(\mathbf{z})$, then $\kappa_{\mathbf{z}}(\mathcal{T}_j(\mathbf{z})) = (\kappa_{\mathbf{z}}(\mathbf{z}))_{(j)}$. Therefore, according to (2.28), we have

$$\kappa_{\mathbf{z}}(\mathcal{T}_j(\mathbf{z})) * \mathbf{h} = (\kappa_{\mathbf{z}}(\mathbf{z}))_{(j)} * \mathbf{h} = (\kappa_{\mathbf{z}}(\mathbf{z}) * \mathbf{h})_{(j)},\tag{3.13}$$

which completes the proof. $\square$

Since the kernel functions and $\mathcal{T}_j$ are not specified in the training process, any kernel and training data can be applied. This generalizes the existing CFs fundamentally, e.g. MOSSE [31], KCF [30], MKCF [42], and DSST [33], which will be presented in Section 3.5. More specifically, the properties of KCC can be customized, if the transform function $\mathcal{T}(\cdot)$ and the kernel functions $\kappa(\cdot, \cdot)$ are specified, which will be presented in the next section.

3.4 Special Cases of KCC

This section presents three examples using the translation transform $\mathcal{T}_T(\cdot)$, rotation transform $\mathcal{T}_R(\cdot)$, and scale transform $\mathcal{T}_S(\cdot)$, resulting in three special cases of KCC, i.e. kernel translation correlator (KTC), kernel rotation correlator (KRC), and kernel scale correlator (KSC), respectively. Without loss of generality, the single radial basis function (RBF) kernel (3.14) is used, the cases for other kernels can be derived similarly.

$$\kappa(\mathbf{x}, \mathbf{z}_i) = h(\|\mathbf{x} - \mathbf{z}_i\|^2). \quad (3.14)$$

3.4.1 Kernel Translation Correlator

For a training sample $\mathbf{z} \in \mathbb{R}^n$ with length n , the number of all possible translational shifts is n , i.e. $|\mathcal{T}_T(\mathbf{z})| = n$, where the operator $|\cdot|$ returns the number of elements in a set. In this sense, the length of $\mathbf{h}$ and $\kappa_{\mathbf{z}}(\mathbf{x})$ equals the length of input signal, i.e. $m = n$. More intuitively, we have $\mathbf{z}_i = \mathbf{z}_{(i)}$, where the subscript $\cdot_{(m)}$ denotes the left circular rotation of a vector by m elements, as defined in (2.22). In spatial domain, the complexity for calculating a kernel (3.14) is $\mathcal{O}(n)$, hence, the complexity for computing a kernel vector $\kappa_{\mathbf{z}}(\mathbf{x})$ is $\mathcal{O}(n^2)$. Fortunately, it will be shown that the calculation can be done in Fourier domain with $\mathcal{O}(n \log n)$ complexity. We have

$$h(\|\mathbf{x} - \mathbf{z}_i\|^2) = h(\|\mathbf{x}\|^2 + \|\mathbf{z}_i\|^2 - 2\mathbf{x}^T \mathbf{z}_i). \quad (3.15)$$

Since $\|\mathbf{x}\|^2$ and $\|\mathbf{z}_i\|^2$ are constants, then

$$\kappa_{\mathbf{z}}(\mathbf{x}) = [h(\|\mathbf{x} - \mathbf{z}_1\|^2), \dots, h(\|\mathbf{x} - \mathbf{z}_n\|^2)]^T \quad (3.16a)$$

$$= h(\|\mathbf{x}\|^2 + \|\mathbf{z}\|^2 - 2[\mathbf{x}^T \mathbf{z}_0, \dots, \mathbf{x}^T \mathbf{z}_{n-1}])^T. \quad (3.16b)$$

From the cross-correlation theory, we know that $\mathbf{x} * \mathbf{z} = [\mathbf{x}^T \mathbf{z}_0, \dots, \mathbf{x}^T \mathbf{z}_{n-1}]^T$. Substitute it into (3.16b), then

$$\kappa_{\mathbf{z}}(\mathbf{x}) = h(\|\mathbf{x}\|^2 + \|\mathbf{z}\|^2 - 2 \cdot \mathbf{x} * \mathbf{z}) \quad (3.17a)$$

$$= h(\|\mathbf{x}\|^2 + \|\mathbf{z}\|^2 - 2 \cdot \mathcal{F}^{-1}(\hat{\mathbf{x}} \odot \hat{\mathbf{z}}^*)). \quad (3.17b)$$

Since the bottle-neck of (3.17b) is the forward and backward FFTs, the kernel vector can be calculated in complexity $\mathcal{O}(n \log n)$. For implementation purpose, the vector norm in (3.17b) is obtained in frequency domain using Parseval's theorem [44], so that the original signals don't need to be stored.

$$\kappa_{\mathbf{z}}(\mathbf{x}) = h\left(\frac{1}{n}\|\hat{\mathbf{x}}\|^2 + \frac{1}{n}\|\hat{\mathbf{z}}\|^2 - 2 \cdot \mathcal{F}^{-1}(\hat{\mathbf{x}} \odot \hat{\mathbf{z}}^*)\right). \quad (3.18)$$

Therefore, based on (3.17b) or (3.18), KCC can be used to predict the translation transform of a signal. This results in the kernel translation correlator (KTC).

3.4.2 Kernel Rotation Correlator

The kernel rotation correlator (KRC) takes the rotation transform $\mathcal{T}_R(\mathbf{z})$ to generate the sample-based vectors $\mathbf{z}_i$. In applications, the rotation transforms are usually discretized with specific resolution ϵ , hence, the number of transforms $|\mathcal{T}_R(\mathbf{z})| = 2\pi/\epsilon = m$ is a constant and is independent of signal length n . Therefore, the kernel vector can be calculated in spatial domain with complexity $\mathcal{O}(n)$.

Because of the periodic structure in the rotation space, i.e. $\phi = \phi + 2\pi$, where ϕ is a rotation angle, the kernel vector $\kappa_{\mathbf{z}}(\mathbf{x})$ is cyclic when it is extended to more than one circle. Hence, there is no boundary effect for the rotation correlator, making it more accurate than the others. In some applications, the rotation angle between two consecutive frames is small, hence, it is not necessary to calculate the full kernel vector $\kappa_{\mathbf{z}}(\mathbf{x})$ at all positions. Instead, we only need to compute them within a specified range in (3.19), while setting all the others to 0. This further reduces the computational requirements, while still keeping the periodic property.

$$\kappa_{\mathbf{z}}(\mathbf{x}) = [0, \dots, \kappa(\mathbf{x}, \mathbf{z}_i), \dots, 0]^T, \quad (3.19)$$

where $\mathbf{z}_i \in \mathcal{T}'_R(\mathbf{z})$ only contains the possible rotations. To the best of our knowledge, this is the first non-linear rotation correlator, which would be useful in many applications, such as the rotation prediction in visual tracking.

3.4.3 Kernel Scale Correlator

Similarly, the sample-based vectors $\mathbf{z}_i$ are obtained from the scale transforms, and only a small set needs to be calculated. However, different from the rotation, the periodic property is not kept in the scale space. Therefore, the boundary effects may have a negative impact on the performance. It is possible to use the existing strategies to eliminate this effect [43, 45]. One of the simplest methods is to add zero padding to the kernel vector,

$$\boldsymbol{\kappa}_{\mathbf{z}}(\mathbf{x}) = [\kappa(\mathbf{x}, \mathbf{z}_1), \dots, \kappa(\mathbf{x}, \mathbf{z}_m), 0, \dots, 0]^T. \quad (3.20)$$

In this case, the length of the kernel vector will be doubled, i.e., $\boldsymbol{\kappa}_{\mathbf{z}}(\mathbf{x}), \mathbf{h} \in \mathbb{R}^{2m}$. Some correlation filters, e.g. DSST and LCT, are also able to predict scale transforms. However, it is achieved by repeatedly calculating the translation correlators at different scales, and the best scale transform is obtained by finding the maximal translation response among all scales. Therefore, they are not scale correlators in the strict sense. More details about this will be presented in the next section. To the best of our knowledge, KST is the first non-linear scale correlator. Note that zero padding in (3.20) is not compulsory. Chapter 4 shows that the KST-based visual tracker outperforms DSST [33] and Staple [46] even without zero padding.

3.5 Theoretical Analysis

The theoretical connection to the existing works, such as MOSSE [31], KCF [30], MKCF [42], DSST [33] will be presented, respectively.

3.5.1 Connection to MOSSE

The MOSSE filter also minimizes SSE and obtains the solution in Fourier domain. The difference is that MOSSE only finds a linear correlation filter, while the objective (3.2) of KCC is to find a non-linear correlator based on the kernel trick at a reasonable computational cost. More specifically, MOSSE is the linear case of KCC and KCC becomes MOSSE when $\boldsymbol{\kappa}_{\mathbf{z}}(\mathbf{z}) = \mathbf{z}$ in (3.1) and (3.3), so that the

solution (3.21) to MOSSE is obtained,

$$\hat{\mathbf{h}}^* = \frac{\sum_{i=1}^s \hat{\mathbf{g}}^i \odot \hat{\mathbf{z}}^{i*}}{\sum_{i=1}^s \hat{\mathbf{z}}^i \odot \hat{\mathbf{z}}^{i*} + \lambda}. \quad (3.21)$$

Note that the solution (3.21) to MOSSE can only predict the translation of the signal $\mathbf{z}$ and the lengths of the filter and signals are always the same, i.e. $n = m$.

3.5.2 Connection to KCF

As shown in Section 2.3.1, KCF is only applicable for circulant data structure and non-weighted kernel functions, which excludes its usage in many other applications. The derivation of KCF is also quite different. First, starting from the kernel regression model, the target output of KCF is a scalar, while KCC is based on the kernel cross-correlation, where target can be a vector or matrix. Second, KCF is derived from the minimization of regression error in time domain, the solution of which in Fourier domain is obtained by diagonalizing the circulant matrix with DFT. On the contrary, the solution (3.9) to KCC is obtained by the minimization of correlation outputs error in Fourier domain naturally. This makes KCC applicable to arbitrary training samples. Assume only one training sample $\mathbf{z} \in \mathbb{R}^n$ is available, then the solution (3.9) becomes (3.22):

$$\hat{\mathbf{h}}^* = \frac{\hat{\mathbf{g}}}{\hat{\kappa}_{\mathbf{z}}(\mathbf{z}) + \lambda}. \quad (3.22)$$

If the sample-based vectors are the circular shifts of the training sample $\mathbf{z}$, i.e. $\mathbf{z}_i = \mathbf{z}_{(i)} \in \mathcal{T}_T(\mathbf{z})$ in (2.44), thus $m = n$, and the solution (3.22) is degraded to KTC with single training sample. It appears similar to KCF in (2.43) given below

$$\hat{\mathbf{a}} = \frac{\hat{\mathbf{y}}}{\hat{\kappa}_{\mathbf{x}}(\mathbf{x}) + \lambda}, \quad (3.23)$$

where $\kappa_{\mathbf{z}}(\mathbf{z})$ is replaced by the first row of a circulant kernel matrix and $\mathbf{g}$ is the stack vector of the ridge regression targets.

Another problem of KCF is that it requires the kernel functions satisfy $\kappa(\mathbf{x}, \mathbf{z}) = \kappa(M\mathbf{x}, M\mathbf{z})$ for any permutation matrix M , which is described in (2.45). However, KCC can be applied to any kernel function, which means that the properties of KCC are customizable with different kernel functions. This is useful for applications

where some signal positions are more important than others. For example, a kernel function can be designed to give high weighting to those pixels near the image center, so that the central part is focused. Chapter 5 presents that such weighted kernel functions are useful for joint scale-rotation correlators.

In summary, KCF is a special case of KCC, when

- only one training sample is available;
- the sample-based vectors are the circular translational shifts of each other;
- only single kernel function is available;
- the kernel function treats all dimensions of data equally.

3.5.3 Connection to MKCF, DSST

MKCF extends KCF to multi-kernel case. It has been shown that KCC is degraded to the multi-kernel case when the k_{th} element of the kernel vector is $\mathbf{d}^T \mathbf{k}_k$, where the vector $\mathbf{d}$ is the combination coefficients. Since MKCF is derived from KCF, all the limitations of KCF are imposed on MKCF. This limits its usages in other applications. Note that all the base kernels κ_i for MKCF need to follow the circulant property mentioned in last section, which is not necessary for KCC.

DSST learns multiple MOSSE for scale tracking. In the detection step, several patches at different scales are sampled around the previous target. Then the scale transform of target is obtained by finding the best maximum response among all patches. In this sense, DSST extends MOSSE to multi-scale case at the price of repeatedly calculating the forward and backward FFTs at different scales. However, the proposed KSC can estimate the scale transform with single calculation of FFT, which reduces the complexity dramatically.

3.6 Experiments

To show the flexibility of KCC, its capability on human activity recognition using wearable devices is demonstrated. Interests in such devices have grown exponentially recently [47, 48]. Despite this popularity, barriers to adoption still remain.

FIGURE 3.1: The color map examples of the 25 dimensional raw data from the wearable sensors. The horizontal axis is the time instant which is recorded with update rate of 30Hz. The data are normalized for better visualization.

3.6.1 Dataset

The public wearable action recognition database (WARD) [49] is chosen. It is comprised of 20 human subjects (7 females and 13 males) with ages from 19 to 75 and 13 daily action categories, including rest at standing (ReSt), rest at sitting (ReSi), rest at lying (ReLi), walk forward (WaFo), walk forward left-circle (WaLe), walk forward right-circle (WaRi), turn left (TuLe), turn right (TuRi), go upstairs (Up), go downstairs (Down), jog (Jog), jump (Jump), and push wheelchair (Push). Five sensors, each of which consists of a triaxial accelerometer and a biaxial gyroscope, are located at the waist, left and right forearms, left and right ankles. Therefore, each sensor produces 5 dimensional data stream and totally 25 dimensional data is available. Each subject performs five trails for activity, thus the database totally contains $20 \times 13 \times 5$ data stream that lasts more than 10 seconds.

3.6.2 Implementation Details

Since our aim is to demonstrate the robustness and flexibility of KCC, we remove the necessary data pre-processing procedures [47, 48], e.g. noise removal, feature extraction, activity start time estimation, and principle component analysis (PCA). Instead, KCC is directly applied to the raw data stream. Figure 3.1 shows some of

FIGURE 3.2: The average confusion matrix of KCC and DTW over all the subjects. There are 5 trials for each of the 13 activities, resulting in a 65×65 confusion matrix. It is obvious that KCC is able to better distinguish human activities and produce less false positives.

the data examples. KCC similarity $\mathcal{S}(\mathbf{x}, \mathbf{y})$ is defined as the maximal value of KCC output in (3.24), that takes the signals $\mathbf{x}$ and $\mathbf{y}$ as the test and training samples, respectively.

$$\mathcal{S}(\mathbf{x}, \mathbf{y}) = \max \mathcal{F}^{-1} \left(\frac{\hat{\kappa}_{\mathbf{y}}(\mathbf{x}) \odot \hat{\mathbf{g}}}{\hat{\kappa}_{\mathbf{y}}(\mathbf{y}) + \lambda} \right) \quad (3.24)$$

It is intuitive that the higher KCC similarity is, the more similar the two signals are. Therefore, KCC similarity can be applied for activity recognition based on the highest similarity value. In the experiments, short duration segments (50 instants, about 1.67 s) are randomly selected from the database for each trail. Then the cross-validation with **single training sample** (one segment) is performed among all 13 activities. This process is repeated until each sample is taken as the training sample exactly one time, resulting in a 65×65 confusion matrix, as there are in total 13×5 activity instances. Note that this process is more difficult than the 5-fold cross-validation, which takes 4 subsets in 5 as the training data. In the experiments, we set the regularization term $\lambda = 0.0015$ in (3.2) with Gaussian kernel $\sigma = 1$. The desired output is a vector with its first element being 1 and the rest being 0, i.e. $\mathbf{g} = [1, 0, 0, 0, \dots]$. Note that the sequence is randomly cropped, hence the activity start time is not available. However, KCC can predict the translation between two signals, which demonstrates its flexibility.

FIGURE 3.3: The performance comparison on WARD dataset. The left and right vertical axes indicate the recognition accuracy (%) and average running time (μs), respectively.

3.6.3 Performance

Since the activity recognition is a non-linear task, the linear correlator, MOSSE, is not compared in this section. Instead, we compare KCC with another popular algorithm, dynamic time warping (DTW), which performs very well on non-linear tasks. It is one of the most famous algorithms for measuring similarity between two temporal sequences which may vary in speed [50]. DTW might be one of the most suitable methods for comparison, since it has been widely used in sequence matching, e.g. speech recognition and data mining, etc [51]. The classification is based on the shortest DTW distances and highest KCC response among all the activities. Figure 3.2 shows the confusion matrix for the two methods. It can be found that KCC performs much better in distinguishing different activities, and produces higher similarity measurements within classes. In summary, KCC achieves 90.7% overall classification accuracy with average running time of 116.4 μs , while DTW achieves 78.8% accuracy with average running time of 156.9 μs . This means that KCC improves 15.1% accuracy, while still saves 25.8% computational time. More details about the comparison for all 13 activities are shown in Figure 3.3. It is noticed that both KCC and DTW cannot perform well on the first three activities, i.e. rest at standing, sitting, and lying. The possible reason is that the signals of the three activities are relatively flat and similar due to the static posture.

3.7 Conclusion

A general formulation for non-linear correlator is proposed, which unifies the existing solutions. The advantages of KCC are its ability to be customized by specific kernels and not limited to circulant data structure, hence, it can predict the affine transforms, especially the scale, rotation, and translation. The theoretical analysis of the KCC is conducted to demonstrate its generality and it is compared with the existing solutions to illustrate its flexibility. Extensive experimental results on public datasets demonstrate that the proposed KCC performs favorably against the state-of-the-art methods in terms of efficiency, accuracy and robustness.

Chapter 4

Visual Object Tracking

4.1 Introduction

As shown in Section 3.5, KCC generalizes the existing correlators, including MOSSE, KCF, MKCF, and DSST, which have achieved the state-of-the-art performance in visual tracking. One of the main advantages of KCC is that it can predict affine transforms with high robustness. Therefore, it is straightforward to construct a KCC-based tracker, which may be able to achieve better performance by taking advantage of the prediction on affine transforms. Leveraging on the KTC and KSC, we propose a KCC-based tracker to predict the object scale variance and translational movement simultaneously.

4.2 Related Work

As one of the most fundamental tasks in visual perception, object tracking has received an increasing attentions due to their potential applications in video surveillance, human computer interaction, etc. Based on the appearance model, the existing visual tracking methods can be generally classified in two categories: generative and discriminative.

4.2.1 Generative Trackers

Generative tracker formulates the target object as an appearance model, whose assumption is that the target objects can be represented by the generative progresses. These methods can be categorized as template matching [52–54], subspace learning [55–57], and sparse representation [58, 59]. To be adaptive to the target appearance variations, the tracking model needs to be updated dynamically.

In template matching, the expectation maximization is one the most widely used methods to renew the object model and update the tracking models. For example, it was adopted in [60] for the particle filter based tracking framework. To reduce the drifting problem, an update method was proposed in [54], where the template can be updated in every frame and yet still stays firmly attached to the original object. Represented by multiple image fragments or patches, [52] proposed the ‘Frag-Track’ algorithm for object tracking in a video sequence. The tracking method [53] for nonrigid objects used a feature histogram-based representation, which successfully coped with camera motion, partial occlusions, clutter, and target scale variations.

Comparing to template matching, subspace learning is more compact, thus it usually takes less computational cost. The tracking model proposed in [55] is decomposed into multiple basic observation models, each of which covers a specific appearance of the object. Based on the Log-Euclidean Riemannian metric, [56] developed a tracking framework, in which the covariance matrices of image features in the five modes are used to represent object appearance, whose changes is adaptively updated by incrementally learning a low-order eigenspace representation. To efficiently update the object representation, [57] proposed an incremental 3D-DCT (discrete cosine transform) algorithm which decomposes the 3D-DCT into successive operations of the 2D-DCT and 1D-DCT on the input video data. As a result, the incremental 3D-DCT algorithm only needs to compute the 2D-DCT for newly added frames as well as the 1D-DCT along the third dimension, which significantly reduces the computational complexity.

The goal of sparse coding in visual tracking is to find succinct representation of stimuli, which is to discover the basic vectors that are capable of representing high-level features of data. The candidates with lowest reconstruction error are selected to represent the target appearance based on the L_1 norm based optimization. The sparse coding methods are usually combined with other algorithms. For example,

the sparse representation schemes proposed in [58] are combined with the principal component analysis (PCA) to learn effective appearance models. The authors in [59] formulate object tracking as a structured multi-task sparse learning problem in a particle filter framework, which is the linear combinations of dictionary templates that are updated dynamically. By employing popular sparsity-inducing mixed norms, the representation problem is regularized to enforce joint sparsity and particle representations are learned together.

4.2.2 Discriminative Trackers

Discriminative trackers take tracking as an on-line classification problem which distinguishes the target from background. Many advanced off-the-shelf machine learning algorithms have been applied to discriminative trackers, e.g. multiple-instance boosting [61], kernelized structured SVM [62], graph learning [63], self-supervised learning [64], and dictionary based trackers [65], several ensemble methods such as boosting [66] and random forest [67]. Discriminative trackers utilize the information from both the object and background and can be categorized in model learning, metric learning, deep learning, and correlation filter trackers [61, 68].

The model learning trackers apply the on-line model updating techniques to adapt to the object appearance changes. For example, the ensemble method in [68] takes tracking as a binary classification problem, where an ensemble of weak classifiers using AdaBoost is trained online to distinguish between the object and the background. This results in a combined strong classifier, which is then used to label pixels in the next frame as either belonging to the object or the background. An on-line semi-supervised boosting method proposed in [69] addressed the drifting problem with appearance changes in tracking applications. It formulates the updating process in a semi-supervised fashion as combined decision of a given prior and an on-line classifier, which comes without any parameter tuning. Due to the error accumulation, slight inaccuracies in the tracker can therefore lead to incorrectly labeled training examples, which degrade the classifier and can cause drift. To solve this problem, [61] used multiple instance learning (MIL) instead of supervised learning to avoid this drifting problem and can therefore lead to a more robust tracker with fewer parameter tweaks. In [70], the authors show that integrating prior information of instance labels and the most correct positive instance (i.e.,

the tracking result in the current frame) into a supervised learning algorithm can handle visual drift more effectively and efficiently than the existing MIL tracker.

Metric learning trackers exploit the distance metrics for searching the best representation of targets. The metric learning tracker framework presented in [71] seeks for a discriminative embedding for appearance modeling. It performs both on-line appearance modeling and visual matching by efficient gradient based optimization, hence the algorithm is time consuming. Collecting a set of supervised training samples on-the-fly in the observed video, [72] incorporates adaptive metric learning into the tracking framework, in which the optimal distance metric for accurate matching is automatically learned. This design ensures that the closest match is more likely to be the true target of interest based on the supervised training. The reliable patch trackers (RPT) [73] presents a metric to measure the reliability of a tracked patch, which improves the tracking effectively.

The deep learning based discriminative trackers have also been extensively studied. In [74], the authors make the first attempt to transfer rich feature hierarchies from a large auxiliary dataset based on an auto-encoder. The work in [75] addresses the problem of online training and finetuning deep convolutional neural network for visual tracking. [76] formulates a convolutional neural network as ensemble learning and shows its effectiveness in visual tracking. Tracking by convolutional neural network without training can be found in [77, 78]. As aforementioned, the recent trend towards deep learning based discriminative trackers mitigates tracking error to an extent, but the tracking speed may still be low. This challenge opens space for learning techniques that aim to achieve higher accuracy and yield faster training. To this end, the correlation filters based tracking [31] has been proposed with extremely high computational efficiency.

Different from the other methods, correlation filter trackers predict the object translational movement directly, thus improving the tracking efficiency dramatically. It has received increasing attentions due to its robustness to occlusion and object distortion. They have been proven to be competitive with far more complicated tracking methods [30] while operating at real-time speed. Except for the forward and backward FFT, which bounds the cost at a nearly-linear ($\mathcal{O}(n \log n)$, n is the dimensionality of the features), the cost of correlation filters is the element-wise operation with $\mathcal{O}(n)$ complexity, which is much faster than the conventional method with a complexity of $\mathcal{O}(n^3)$. The authors in [31] propose to seek a linear

correlation filter in the context of minimum output sum of squared error (MOSSE) in frequency domain, where the element-wise operation can be utilized, resulting in a very efficient tracking framework. The work in [45] addresses the boundary effects of correlation filters and are able to implicitly exploit possible patches densely extracted from training examples during learning process. Based on the kernel trick, KCF [30] applied the circular data constraint on the ridge regression, resulting in a kernelized correlation filter, which is more robust than the linear correlation filters.

More recent work shows that the correlation filter based trackers, such as DSST and Staple [79], outperform other trackers in this family. For example, the part-based KCF [80] is proposed to address the problem of occlusion. To improve the accuracy and robustness, the KCF is extended to multiple kernel case in [42]. A random fern classifier is implemented for object re-detection, which results in the long-term correlation tracking (LCT) [81]. Discriminative scale space tracking (DSST) [33] learns separate CFs on different scales, enabling estimation of both the translation and scale variants. A biology-inspired framework [82] is proposed for correlation filter based trackers, where both short and long-term memory are integrated. In [83], the authors hierarchically infer the maximum response of ensemble of correlation filters trained on the features extracted from different layers of CNN. The correlation filter based trackers fused with different features can be found in [84].

However, the drawbacks of correlation filters are also obvious, i.e. only the translational movements are predictable. This opens research space for KCC-based tracker, which is able to predict any affine transforms.

4.3 Methodology

In order to show the superiority of KCC over the existing ones, *Staple* (Sum of Template And Pixel-wise LEarners) [79] is selected as baseline, whose accuracy is higher than MOSSE, KCF, DSST, etc. In order to estimate the scale variations, Staple follows DSST and learns a distinct, multi-scale template for scale search using multiple correlation filter, at the price of repeatedly calculating the forward and backward FFTs at different scales. Here shows that this is not necessary and can be naturally solved by KSC. Specifically, KTC and KSC based sub-trackers are built which are used to generate an inherently more robust tracker.

4.3.1 Update Rules

During tracking, a target can often change appearance by changing its position, scale, pose, lighting conditions, and even undergoing non-rigid deformation. Therefore, correlators need to quickly adapt in order to follow objects. According to (3.3), we rewrite the solution for single training sample, i.e.

$$\hat{\mathbf{h}}^* = \frac{\hat{\mathbf{g}} \odot \hat{\boldsymbol{\kappa}}_{\mathbf{z}}^*(\mathbf{z})}{\hat{\boldsymbol{\kappa}}_{\mathbf{z}}(\mathbf{z}) \odot \hat{\boldsymbol{\kappa}}_{\mathbf{z}}^*(\mathbf{z}) + \lambda}. \quad (4.1)$$

Running average (4.2) is adopted for KTC to adapt the appearance changes.

$$\begin{aligned} \hat{\mathbf{h}}^* &= \frac{A_i}{B_i + \lambda}, \\ A_i &= \eta_t \cdot \hat{\mathbf{g}} \odot \hat{\boldsymbol{\kappa}}_{\mathbf{z}}^*(\mathbf{z}) + (1 - \eta_t) \cdot A_{i-1}, \\ B_i &= \eta_t \cdot \hat{\boldsymbol{\kappa}}_{\mathbf{z}}(\mathbf{z}) \odot \hat{\boldsymbol{\kappa}}_{\mathbf{z}}^*(\mathbf{z}) + (1 - \eta_t) \cdot B_{i-1}, \end{aligned} \quad (4.2)$$

where the subscript i denotes the i^{th} frame and $\eta_t = 0.01$ is the learning rate for KTC. Parameters A_i and B_i are initialized as the numerator and denominator of $\hat{\mathbf{h}}^*$ computed based on the tracking window centered on the object in the first frame. The kernel vector for KTC in the update rule (4.2) is calculated following (3.18).

To save the computational power for KSC, we apply the objective with weighted regularization in (3.8). It has a simpler solution (3.3), which is rewritten in (4.3) with single training sample.

$$\hat{\mathbf{h}}^* = \frac{\hat{\mathbf{g}}}{\hat{\boldsymbol{\kappa}}_{\mathbf{z}}(\mathbf{z}) + \lambda}, \quad (4.3)$$

where $\hat{\mathbf{g}}$ only needs to be calculated once when the program starts.

For the scale transform, we select m (m is odd) scale factors with step s around the previous estimated position, i.e the scale factor list $\mathbf{s}$ is

$$\mathbf{s} = (s^{-\lfloor m/2 \rfloor}, s^{-\lfloor m/2 \rfloor + 1}, \dots, s^{\lfloor m/2 \rfloor}). \quad (4.4)$$

We follow (3.20) to calculate the kernel vector without zero padding ($m = 11$, $s = 1.03$), resulting in an m -length scale kernel vector $\boldsymbol{\kappa}(\mathbf{Z}(\mathbf{s}))$, where $\mathbf{Z}(\mathbf{s}) = [\mathbf{z}_1, \dots, \mathbf{z}_m]$ is generated from the scale factors $\mathbf{s}$. Note that the scaling center is the image center and we crop the image after scaling to make sure that $\mathbf{z}_i$ has the same size as

z. Due to the simplicity of (4.3), the update rule (4.2) is not suitable. Instead, we propose another update rule for KSC.

Each time the scale and translation transform of the object is predicted, we crop and resize that patch to a predefined size and generate the scale patches $\mathbf{Z}(\mathbf{s})$. Running average (4.5) is adopted for KSC to adapt the scale appearance changes.

$$\begin{aligned}\hat{\mathbf{h}}^* &= \frac{\hat{\mathbf{g}}}{\hat{\kappa}(\mathbf{Z}_i(\mathbf{s})) + \lambda}, \\ \mathbf{Z}_i(\mathbf{s}) &= \eta_s \cdot \mathbf{Z}_i(\mathbf{s}) + (1 - \eta_s) \cdot \mathbf{Z}_{i-1}(\mathbf{s}),\end{aligned}\tag{4.5}$$

where the subscript i denotes the i^{th} frame and η_s is the scale learning rate. In the experiments, we set $\eta_s = 0.008$. The $\mathbf{Z}_i(\mathbf{s})$ is initially computed based on the tracking window centered on the object in the first frame.

4.3.2 Prediction

Since KTC and KSC are equivariant to translation and scale transform, respectively, we may resort to the correlation output to predict the position and scale transform in the new frame. According to Theorem 3.3.3, if the correlation output $\mathbf{g}$ is predefined as a mask with single peak value, the translation of peak value in the output will be corresponding to the transform of input. Because of noises and occlusion, it is impossible to get a perfect translational version of $\mathbf{g}$. Instead, the translation of the maximum value in the correlation output is regarded as the output translation. This simple strategy makes the prediction (4.6) very fast to compute, since only the inverse FFT is needed.

$$\arg \max_i \mathcal{F}^{-1}(\hat{\kappa}_{\mathbf{z}}(\mathbf{x}) \odot \hat{\mathbf{h}}^*)[i].\tag{4.6}$$

In experiments, the KTC target $\mathbf{g}$ is set as a smooth 2-D Gaussian mask, which makes the prediction robust to tiny translational disturbance, e.g. occlusion. In contrast, the KSC target $\mathbf{g}$ is set as a binary vector, in which the only 1 is located at the first element. This simple strategy makes the scale prediction more accurate.

FIGURE 4.1: The tracking examples of KCC in terms of number of frames (#).

FIGURE 4.2: Success plot of state-of-the-art methods on the OTB dataset.

4.4 Experiments

4.4.1 Dataset and Evaluation Metric

The well-known OTB dataset that contains 51 sequences [85] is selected. It has been widely used for demonstration and comparison purpose. More importantly, it covers a variety of challenging scenarios. To better evaluate and analyze the strength and weakness of tracking approaches, the videos are categorized with 11 attributes based on different challenging factors including low resolution, in-plane and out-of-plane rotation, scale variation, occlusion, deformation, background clutters, illumination variation, motion blur, fast motion, and out-of-view.

The standard criterion of success plot curve is adopted as the performance criterion. To measure the performance, the number of successful tracking frames whose overlap S in (4.7) is larger than the given threshold t_o is counted.

$$S = \left| \frac{r_t \cap r_g}{r_t \cup r_g} \right|, \quad (4.7)$$

where $\cap$ and $\cup$ stand for the intersection and union of two regions, respectively. r_t and r_g denote the bounding box of ground truth and tracking results, respectively.

FIGURE 4.3: Success plot of state-of-the-art methods on different categories (I).

FIGURE 4.4: Success plot of state-of-the-art methods on different categories (II).

FIGURE 4.5: Success plot of state-of-the-art methods on scale variation.

The success plot shows the ratios of successful frames at the thresholds varied from 0 to 1. The standard protocol to use the area under curve (AUC) of each success plot is followed to rank the tracking algorithms.

4.4.2 Comparison

In both KTC and KSC, the regularization term λ in (3.2) is set as 0.001 and the Gaussian kernel is used with sigma of 0.1. In order to demonstrate the performance of KCC, it is compared with all state-of-the-art methods in [85], KCF, and several

recent advanced methods including Random Forest [67] as well as Staple. In order to have a fair comparison, we only evaluate tracker with handcrafted features. Applying the proposed method on learning representation such as CNN features, however, is beyond the scope of this chapter, since the CNN-based methods require pre-training on a large data-set, while the objective of this chapter is the single object model-free tracking, in which only the bounding box of the object in the first frame is known. However, we expect that KCC can achieve better accuracy if CNN-based features are involved.

Figure 4.1 presents some tracking examples in the dataset, where the number on the left-up corner of each figure indicates the number of frames in the selected sequence, respectively. The blue dashed bounding box shows the image patch for training and testing, while the green bounding box shows the object tracked. This is to make sure that the training patch covers the object all the time, especially when the object moves very fast. It can be seen that the KCC-based visual tracker is quite robust to the image noise, object distortion, and occlusion.

Figure 4.2 shows the success plot of different methods on all the 51 datasets under one pass evaluation (OPE) [85]. In this case, a tracker is evaluated on a test sequence with initialization from the ground truth position in the first frame. As aforementioned, AUC is used to rank each method and for simplicity, only top 10 methods are illustrated. KCC performs significantly better than all the methods reported in [85], with an average relative improvement of 20% with respect to the best tracker evaluated in the original benchmark (SCM [86]). Moreover, KCC also outperforms recent trackers such as Random Forest [67] and well studied diagnose framework [87]. Most importantly, KCC achieves better performance than Staple while running at roughly the same frame rate (40 FPS), which clearly demonstrates the superiority.

KCC performs well under a variety of challenging scenarios such as deformation, illumination variations, occlusions and so on. The performance on different categories are shown in Figure 4.3 and Figure 4.4, where the number in the caption indicates how many sequences belong to this scenario.

The performance on the scale variance is illustrated in Figure 4.5, in which there is a large margin between KCC and other methods. This indicates that KSC plays an important role for scale adaption and further verifies its robustness and flexibility.

4.5 Conclusion

This chapter proposes a KCC-based visual tracker, which consists of KTC and KSC for translation and scale prediction, respectively. To quickly adapt the appearance change, an on-line joint update rule is proposed to fit the object model. Compared to the existing scale trackers that are basically multiple translational trackers on different scales, our KSC-based tracker can predict the scale transform directly. The extensive experiments on OTB dataset show that the KCC improves the performance dramatically. Compared with the state-of-the-art methods, KCC-based visual tracker is more robust to noise and distortion, and can quickly adapt to the scale changes.

Chapter 5

Correlation Flow

In Chapter 4, it has been shown that the KSC can predict the scale transform robustly. Intuitively, if both rotation and scale transforms are carried out on an image, a KRC is needed. However, in this case, the computational complexity is doubled in order to calculate the rotation correlator. Recall that one of the main advantages of KCC shown in Chapter 3 is that the weighted kernel functions can be applied. In this chapter, we show that by applying a weighted kernel function, the KRC and KSC can be calculated jointly, resulting in a lower computational complexity. To show its efficiency, we apply it to micro quad-copters for velocity estimation by computing the optical flow, which is named as correlation flow.

5.1 Introduction

To safely fly in cluttered environments, insects rely on optical flow (OF), which is generated by their own displacement relative to the surroundings [88]. Inspired by this, the optical flow based flight capabilities for unmanned aerial vehicles (UAV) have received increasing attentions, including obstacle avoidance, speed maintenance, odometry estimation, altitude regulation, wall following and corridor centering, orientation control, and landing [88–91]. For example, based on optical flow, quadcopters can achieve autonomous navigation and collision avoidance in urban or indoor environments [91]. Optical flow has also been combined with simultaneous localization and mapping (SLAM) algorithms to estimate distances from surrounding environment and stabilize the drone [90]. In recent years, some compact and

FIGURE 5.1: The framework for camera velocity estimation based on the proposed correlation flow. A downward facing camera mounted on a quadcopter moves arbitrarily in a 3-D space. The horizontal and altitude velocities and the yaw angular rate are estimated by the kernel translation correlator (KTC) and the kernel scale-rotation correlator (KSRC), respectively.

low-power optical flow sensors have been reported [92–94] for micro drones. This dramatically reduces the requirements for on-board energy, sensing, and processing capabilities.

However, the existing algorithms heavily rely on features extracted from the input image, which may be noisy or challenging to extract in flight scenarios, especially at high speed. Therefore, a more accurate and robust method for computing optical flow is needed. Inspired by the fact that local flows are averaged to obtain stable velocity estimation [92, 93, 95], we argue that it may be faster and more robust to predict global flow directly based on learning techniques. To this end, we propose a new optical flow method for velocity estimation based on kernel cross-correlators, which is shown in Figure 5.1. Specifically, we propose to learn the kernel translation correlator (KTC) to estimate the translation flow. Moreover, we also develop a joint kernel scale-rotation correlator (KSRC) to efficiently estimate the scale and rotation flow, resulting in altitude and yaw velocity estimates, which are useful for robust orientation control and landing. Compared with existing methods, correlation flow achieves higher accuracy while still possesses a similar level of computational efficiency. Experiments on autonomous flight of a quadcopter demonstrate the superiority and robustness of correlation flow.

5.2 Related Work

One of the earliest optical flow methods is the Horn-Schunck algorithm, that assumes the apparent velocity of the brightness pattern to vary smoothly almost everywhere in the image [96]. By approximating each neighborhood of two consecutive frames using quadratic polynomials, Farneback method estimates the displacement fields from the polynomial expansion coefficients [97]. Using phase correlations, [98] proposes to compute optical flow by block matching, followed by an additional optimization procedure to find smoother motion fields among several candidates. Although these methods yield a high density of flow vectors, they require complex calculation and are sensitive to noise.

To mitigate noise effect, several methods based on feature tracking, e.g. Shi-Tomasi [99], FAST [100], and Lucas-Kanade [101], have been widely used to compute optical flow. Nevertheless, those methods still cannot satisfy the real-time requirements for micro drones [93]. Leveraging on the efficiency of parallel computing, [95] presents an FPGA-based platform for computing the metric optical flow. PX4Flow [92] is an open source and open hardware optical flow sensor using a CMOS camera. It is based on the sum of absolute differences (SAD) block matching algorithm, where the position of the best match in the search area is selected as the resulting flow value. This optical flow sensor is easy to use but sensitive to illumination and motion blur. In [93], the local translation flow is calculated by matching the edge histograms that are obtained by the summation of image gradients in two orthogonal directions. Based on this work, [94] proposes to combine stereo vision and optical flow to estimate velocity and depth for pocket drones.

Deep learning based optical flow methods have also been extensively studied. For example, a thresholded loss for Siamese networks is proposed in [102], where the robustness of trained features for patch matching for optical flow is evaluated. An end-to-end learning strategy is demonstrated for optical flow estimation in [103]. By combining a classical spatial-pyramid method with deep learning, SPyNet trains one deep network per level to compute the optical flow [104]. Although the recent trend towards deep learning based methods mitigates estimation error to an extent, the computation speed is still too low and the cost of training data acquisition is too high for real-time robotic applications, especially micro drones. This opens space for learning techniques that achieve higher accuracy and yield faster training.

5.3 2-D KCC

The 1-D KCC has been introduced in Chapter 3, in this section, we briefly present the 2-D KCC, which is useful for developing the KSRC. For simplicity, only the single kernel and single training sample case will be considered. Denote the signals as 2-D matrices or images, i.e. $\mathbf{z}, \mathbf{x} \in \mathbb{R}^{n_x \times n_y}$, where $\mathbf{z}$ and $\mathbf{x}$ are regarded as the previous and current frame, respectively. Denote the 2-D fast Fourier transform (FFT) $\mathcal{F} : \mathbb{C}^{n_x \times n_y} \mapsto \mathbb{C}^{n_x \times n_y}$ as $\hat{\cdot}$, so that the cross-correlation of two images $\mathbf{g} = \mathbf{x} * \mathbf{h}$ is equivalent to $\hat{\mathbf{g}} = \hat{\mathbf{x}} \odot \hat{\mathbf{h}}^*$, where the operator $\odot$ and superscript $*$ denote the element-wise multiplication and complex conjugate, respectively.

The correlation output $\hat{\mathbf{g}}$ can be transformed back into spatial domain using the inverse FFT. Denote the kernel function as $\kappa(\cdot, \cdot)$, such that $\kappa(\mathbf{x}, \mathbf{z}) \in \mathbb{R}$. Given a desired output $\mathbf{g} \in \mathbb{R}^{m_x \times m_y}$, the 2-D kernel cross-correlator is:

$$\hat{\mathbf{g}} = \hat{\kappa}_{\mathbf{z}}(\mathbf{x}) \odot \hat{\mathbf{h}}^*, \quad (5.1)$$

where $\kappa_{\mathbf{z}}(\mathbf{x}) \in \mathbb{R}^{m_x \times m_y}$ is a kernel matrix, with element in the i^{th} row and j^{th} column as $\kappa(\mathbf{x}, \mathbf{z}_{ij})$, where $\mathbf{z}_{ij} \in \mathcal{T}(\mathbf{z}) \in \mathbb{R}^{n_x \times n_y}$ is generated from the previous frame $\mathbf{z}$. The transform function $\mathcal{T}(\cdot)$ is predefined for different training objectives.

The correlator $\mathbf{h}$ that maps $\mathbf{z}$ to the desired output $\mathbf{g}$ is to be trained by minimizing the sum of squared errors (SSE) between the correlation output and the desired target. To be efficient, we conduct the training in Fourier domain to take advantage of the simple element-wise operation:

$$\min_{\hat{\mathbf{h}}^*} \|\hat{\kappa}_{\mathbf{z}}(\mathbf{z}) \odot \hat{\mathbf{h}}^* - \hat{\mathbf{g}}\|^2 + \lambda \|\sqrt{\hat{\kappa}_{\mathbf{z}}(\mathbf{z})} \odot \hat{\mathbf{h}}^*\|^2, \quad (5.2)$$

where the second term is a regularization to prevent overfitting. Note that the regularization term in (5.2) is weighted by the FFT of kernel matrix, which is different from (3.2) but similar to (3.8). We will see that this results in a simpler solution, since the squared norm of kernel matrix's FFT is bypassed, making the system more efficient. To solve it we set the first derivative to zero, i.e.,

$$\frac{\partial}{\partial \hat{\mathbf{h}}^*} \left(\|\hat{\kappa}_{\mathbf{z}}(\mathbf{z}) \odot \hat{\mathbf{h}}^* - \hat{\mathbf{g}}\|^2 + \lambda \|\sqrt{\hat{\kappa}_{\mathbf{z}}(\mathbf{z})} \odot \hat{\mathbf{h}}^*\|^2 \right) = 0. \quad (5.3)$$

FIGURE 5.2: The computation structure of correlation flow. The position of the maximum response indicates the transformation of images. Except for the FFT, all the operations are element-wise, resulting in efficient computations. Due to the introduction of kernels, correlation flow is robust to noises, motion blur, and image distortions.

Similar to the Theorem 3.3.2, we can obtain a closed-form solution (5.4) for 2-D KCC, since all the operations in (5.3) are performed in an element-wise manner.

$$\hat{\mathbf{h}}^* = \frac{\hat{\mathbf{g}}}{\hat{\kappa}_{\mathbf{z}}(\mathbf{z}) + \lambda}. \quad (5.4)$$

The advantages of 1-D KCC is kept for 2-D KCC, including that any training data $\mathbf{z}$, affine transformation function $\mathcal{T}(\cdot)$, and kernel function $\kappa(\cdot, \cdot)$ can be applied, so that the 2-D KCC is customizable for specific applications.

5.4 Correlation Flow

In this section we demonstrate that the translation, scale-rotation flow can be computed by specifying the function $\mathcal{T}(\cdot)$ as translation, scale-rotation transforms, respectively.

5.4.1 Translation Flow

Figure 5.2 illustrates the computation structure, in which each prediction takes the previous and current frame as the training and test sample, respectively. To predict the translation flow, the translation transform $\mathcal{T}_T(\cdot)$ on 2-D matrix is applied to generate $\mathbf{z}_{ij}$. Since $\mathbf{z} \in \mathbb{R}^{n_x \times n_y}$, the number of all possible translational shifts $|\mathcal{T}_T(\mathbf{z})| = n_x n_y$, where the operator $|\cdot|$ returns the number of element in a set, and

$\mathcal{T}_T(\mathbf{z})$ is the set consisting of all translational shifts of $\mathbf{z}$. Therefore, the size of $\mathbf{h}$ and the kernel matrix $\kappa_{\mathbf{z}}(\mathbf{x})$ equals the size of the image, i.e. $n_x = m_x, n_y = m_y$.

Without loss of generality, we consider the 2-D RBF kernel as in (3.14). Since the complexity of calculating (3.14) is $\mathcal{O}(N)$, where $N = n_x n_y$, the complexity of computing a kernel matrix $\kappa_{\mathbf{z}}(\mathbf{x})$ is $\mathcal{O}(N^2)$, which might be infeasible for embedded systems. Fortunately, the kernel matrix can be computed in frequency domain with complexity $\mathcal{O}(N \log N)$ as in Section 3.4.1. Expand the norm in (3.14), thus

$$\kappa(\mathbf{x}, \mathbf{z}_{ij}) = h \left(\|\mathbf{x}\|^2 + \|\mathbf{z}_{ij}\|^2 - 2 \cdot \text{Tr}(\mathbf{x}^T \mathbf{z}_{ij}) \right), \quad (5.5)$$

where the operator $\text{Tr}(\cdot)$ returns the trace of a square matrix. Since $\|\mathbf{x}\|^2$ and $\|\mathbf{z}_{ij}\|^2$ are constants, the kernel matrix can be expressed as:

$$\kappa_{\mathbf{z}}(\mathbf{x}) = h \left(\|\mathbf{x}\|^2 + \|\mathbf{z}\|^2 - 2 \left[\text{Tr}(\mathbf{x}^T \mathbf{z}_{ij}) \right]_{n_x n_y} \right), \quad (5.6)$$

where the trace matrix $\left[\text{Tr}(\mathbf{x}^T \mathbf{z}_{ij}) \right]_{n_x n_y}$ is defined as:

$$\left[\text{Tr}(\mathbf{x}^T \mathbf{z}_{ij}) \right]_{n_x n_y} := \begin{bmatrix} \text{Tr}(\mathbf{x}^T \mathbf{z}_{11}) & \cdots & \text{Tr}(\mathbf{x}^T \mathbf{z}_{1n_y}) \\ \vdots & \ddots & \vdots \\ \text{Tr}(\mathbf{x}^T \mathbf{z}_{n_x 1}) & \cdots & \text{Tr}(\mathbf{x}^T \mathbf{z}_{n_x n_y}) \end{bmatrix}. \quad (5.7)$$

From the 2-D correlation theory, $\mathbf{x} * \mathbf{z} = \left[\text{Tr}(\mathbf{x}^T \mathbf{z}_{ij}) \right]_{n_x n_y}$. Substituting it into (5.6),

$$\kappa_{\mathbf{z}}(\mathbf{x}) = h \left(\|\mathbf{x}\|^2 + \|\mathbf{z}\|^2 - 2 \cdot \mathbf{x} * \mathbf{z} \right) \quad (5.8a)$$

$$= h \left(\|\mathbf{x}\|^2 + \|\mathbf{z}\|^2 - 2 \cdot \mathcal{F}^{-1}(\hat{\mathbf{x}} \odot \hat{\mathbf{z}}^*) \right). \quad (5.8b)$$

The bottleneck of (5.8b) is the forward and backward FFTs, so that the kernel matrix can be calculated in complexity $\mathcal{O}(N \log N)$. For implementation purpose, the matrix norm in (5.8b) can be obtained in frequency domain using the Parseval's theorem, so that there is no need to store the original signals.

$$\kappa_{\mathbf{z}}(\mathbf{x}) = h \left((\|\hat{\mathbf{x}}\|^2 + \|\hat{\mathbf{z}}\|^2)/N - 2\mathcal{F}^{-1}(\hat{\mathbf{x}} \odot \hat{\mathbf{z}}^*) \right). \quad (5.9)$$

Based on (5.9), it is not necessary to generate the sample-based matrices $\mathbf{z}_{ij}$ explicitly, which decreases both space and time complexity dramatically.

FIGURE 5.3: An example of the KSRC. (a) previous image captured at 30Hz; (b) current image; (c) the response of KSRC. The location of the maximum response relative to the center indicates the transformation pattern. In this example, the image is rotated by 9 degrees and scaled with factor 1.2516.

The 2-D kernel translation correlator (KTC) can then be obtained using (5.4) and (5.9). In the experiments, only the center of the desired output $\mathbf{g}$ is set as 1, while all the other positions are set to 0. Intuitively, due to image noise and distortion, it is not possible to obtain an exact single peak for the test sample. Instead, the position of the maximum value in the output is used to find the translation of the test sample. Specifically, the translation (x_n, y_n) of the current frame relative to the training frame is obtained in (5.10), which is the position of the maximum value in the correlation output relative to the image center $(\frac{n_x}{2}, \frac{n_y}{2})$.

$$(x_n, y_n) = \arg \max_{(i,j)} \mathcal{F}_{(i,j)}^{-1} \left(\hat{\kappa}_{\mathbf{z}}(\mathbf{x}) \odot \hat{\mathbf{h}}^* \right) - \left(\frac{n_x}{2}, \frac{n_y}{2} \right), \quad (5.10)$$

where $\mathcal{F}_{(i,j)}^{-1}(\cdot)$ is an element of the inverse FFT with index (i, j) . Therefore, the estimated horizontal metric velocity (v_x, v_y) can be calculated as:

$$(v_x, v_y) = -\frac{h_a}{\Delta t} \left(\frac{x_n}{f_x}, \frac{y_n}{f_y} \right), \quad (5.11)$$

where h_a is the height measurement that can be obtained from an altimeter, and Δt is the time instant difference between the previous and the current image. f_x and f_y are the focal lengths in x and y direction, respectively.

5.4.2 Scale and Rotation Flow

As shown in Figure 5.1, the altitude velocity v_z and yaw rate ω_z can be measured by the image scale and rotation transformations, respectively, using a downward facing camera. Similar to the 2-D KTC, it is possible to estimate the scale and rotation using KCC by defining the function $\mathcal{T}(\cdot)$ as scale and rotation transformations, respectively. However, we will show that the complexity is too high for separately calculating the scale and rotation correlators.

In applications, rotation and scale transforms are usually discretized with specific resolution. Let $\mathbf{z}_{ij}$ be the transform of $\mathbf{z}$ with specific scale factor $s_i \in \Omega_s$ and rotation angle $\theta_j \in \Omega_\theta$, where Ω_s and Ω_θ are the sets of scale factors and rotation angles, respectively. Let $m_x = |\Omega_s|$ and $m_y = |\Omega_\theta|$, then calculating the scale and rotation kernel vectors are of complexity $\mathcal{O}(m_x N)$ and $\mathcal{O}(m_y N)$, respectively. Therefore, the complexity of (5.4) is $\mathcal{O}(N \log N + m_{x,y} N)$, which is still bounded by the FFT, especially when m_x or m_y is small. However, if scale and rotation are both presented in the image, the complexity of (5.4) becomes $\mathcal{O}(N \log N + MN)$, where $M = m_x m_y$. This means that the calculation is bounded by the complexity of the kernel matrix with $\mathcal{O}(MN)$, which is very difficult to compute in real-time.

To solve this problem, we propose the kernel scale-rotation correlator (KSRC), as a faster method to calculate the kernel matrix. Since pixels on image boundary are often meaningless and discarded due to the scale and rotation transforms, the pixels near to the image center should account for a greater proportion in the kernel function. Without loss of generality, consider a weighted RBF kernel in (5.12):

$$\kappa(\mathbf{x}, \mathbf{z}_{ij}) = h(\|\mathbf{w} \odot (\mathbf{x} - \mathbf{z}_{ij})\|^2), \quad (5.12)$$

where $\mathbf{w} \in \mathbb{R}_+^{n_x \times n_y}$ with $\|\mathbf{w}\|_1 = n_x n_y$ is the weight matrix. For the weighted kernel in (5.12), the trace matrix cannot be replaced by the cross-correlation as used in (5.8). Therefore, the complexity of calculating the kernel matrix $\kappa_{\mathbf{z}}(\mathbf{x})$ is dominated by the trace matrix, which is still $\mathcal{O}(MN)$. However, we find that this can be solved efficiently using a mapping function, which is defined as $\mathcal{M} : \mathbb{R}^{n_x \times n_y} \mapsto \mathbb{R}^{m_x \times m_y}$ in (5.13). We denote the mapping function $\mathcal{M}$ as $\tilde{\cdot}$, so that

$$\tilde{\mathbf{x}} = \mathcal{M}(\mathbf{x}), \text{ s.t. } \text{Tr}(\tilde{\mathbf{x}}^T \tilde{\mathbf{y}}) = \text{Tr}((\mathbf{w} \odot \mathbf{x})^T (\mathbf{w} \odot \mathbf{y})), \quad (5.13)$$

FIGURE 5.4: One example of the velocity estimation from the first 60 seconds of a flight test. It is obvious that the estimation from correlation flow is smoother and more accurate than PX4Flow.

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_x \times n_y}$. Since $\|\mathbf{x}\|^2 = \text{Tr}(\mathbf{x}^T \mathbf{x})$, $\|\tilde{\mathbf{x}}\|^2 = \|\mathbf{w} \odot \mathbf{x}\|^2$, substituting (5.13) into (5.12), we have

$$\kappa(\mathbf{x}, \mathbf{z}_{ij}) = h \left(\|\tilde{\mathbf{x}}\|^2 + \|\tilde{\mathbf{z}}_{ij}\|^2 - 2 \cdot \text{Tr}(\tilde{\mathbf{x}}^T \tilde{\mathbf{z}}_{ij}) \right), \quad (5.14)$$

where $\tilde{\mathbf{z}}_{ij} = \mathcal{M}(\mathbf{z}_{ij})$. Therefore, the kernel matrix can be calculated as:

$$\kappa_{\mathbf{z}}(\mathbf{x}) = h \left(\|\tilde{\mathbf{x}}\|^2 + \|\tilde{\mathbf{z}}\|^2 - 2 \left[\text{Tr}(\tilde{\mathbf{x}}^T \tilde{\mathbf{z}}_{ij}) \right]_{m_x m_y} \right), \quad (5.15)$$

where the trace matrix $[\text{Tr}(\tilde{\mathbf{x}}^T \tilde{\mathbf{z}}_{ij})]_{m_x m_y}$ is defined similarly to (5.7). Inspired by the Fourier-Mellin transform [105], the trace matrix can be converted to cross-correlation, if the element of the weight matrix $\mathbf{w}$ is set as (5.16):

$$\mathbf{w}_{[x,y]} \propto \left| \left\{ (i,j) \begin{cases} x = \langle \exp \xi_i \cos \theta_j \rangle \\ y = \langle \exp \xi_i \sin \theta_j \rangle \end{cases} \right\} \right|, \quad (5.16)$$

where (x, y) are image coordinates relative to the image center and the operator $\langle \cdot \rangle$ returns the nearest integer of a real number. In this sense, $\mathcal{M}(\cdot)$ becomes

the log-polar transform with coordinates (ξ, θ) , where $\xi = \log \sqrt{x^2 + y^2}$ and $\theta = \text{atan2}(y, x)$. Therefore, the image mapping $\tilde{\mathbf{z}}$ in the log-polar plane satisfies (5.17), which means that $\mathbf{z}_{ij}$ is the transformation of $\mathbf{z}$ with scale factor $s_i = \exp \xi_i$ and rotation angle θ_j .

$$\tilde{\mathbf{z}}_{ij}(\xi, \theta) = \tilde{\mathbf{z}}(\xi - \xi_i, \theta - \theta_j). \quad (5.17)$$

It is easy to verify that the weight matrix $\mathbf{w}$ defined in (5.16) satisfies the intuitive idea that pixels near to the center play more important roles than those near to the boundary. Substituting (5.17) into (5.15) and ignoring the boundary effect, we can approximate the trace matrix in (5.15) by cross-correlation in (5.18a) and element-wise multiplication in (5.18b).

$$\kappa_{\mathbf{z}}(\mathbf{x}) = h \left(\|\tilde{\mathbf{x}}\|^2 + \|\tilde{\mathbf{z}}\|^2 - 2 \cdot \tilde{\mathbf{x}} * \tilde{\mathbf{z}} \right) \quad (5.18a)$$

$$= h \left((\|\hat{\tilde{\mathbf{x}}}\|^2 + \|\hat{\tilde{\mathbf{z}}}\|^2)/M - 2\mathcal{F}^{-1}(\hat{\tilde{\mathbf{x}}} \odot \hat{\tilde{\mathbf{z}}}) \right). \quad (5.18b)$$

Dominated by the forward and backward FFTs in (5.18b), the complexity of calculating the kernel matrix is reduced to $\mathcal{O}(M \log M + M)$, which is much smaller than $\mathcal{O}(N \log N + MN)$. An example of KSRC is shown in Figure 5.3.

Assuming that (x_m, y_m) is the translation of the maximum value in the correlation output, which is obtained similarly to (5.10). Then the altitude velocity v_z and yaw rate ω_z can be calculated as [92]:

$$v_z = \frac{(s-1)h}{\Delta t}, \quad \omega_z = \frac{2\pi y_m}{m_y \Delta t}, \quad (5.19)$$

where s is the estimated scale factor:

$$s = \exp \left(\frac{\log(m_y/2)}{m_x} x_m \right). \quad (5.20)$$

Note that we can also obtain the altitude velocity by differentiating the altimeter measurements, but it is very noisy. One possible solution is to fuse the two sources of information, however, it is out of the scope of this chapter.

TABLE 5.1: Comparison on horizontal velocity with PX4Flow. (m/s)

Test	Correlation Flow		PX4Flow	
	RMSE	MAE	RMSE	MAE
01	0.069	0.054	0.145	0.117
02	0.072	0.058	0.167	0.138
03	0.074	0.057	0.148	0.123
04	0.069	0.053	0.140	0.110
05	0.076	0.064	0.132	0.100
Mean	0.072	0.057	0.146	0.118

5.5 Experiments

Successful autonomous navigation of drones depends on a robust optical flow system, which is used to provide accurate velocity estimation, and hence improving the position estimation [89, 90, 92, 93]. In this section, extensive experiments on velocity estimation, autonomous flight, and battery life hovering test demonstrate the superior performance of the proposed correlation flow system.

5.5.1 Implementation

5.5.1.1 Software

Since $\mathbf{g}$ is not changed during the training stage, $\hat{\mathbf{g}}$ only needs to be calculated once when starting the program. The regularization parameter λ in (5.4) is set as 0.1 to prevent overfitting. The Gaussian kernel (5.21), which is proved to be robust to noise and distortion is used:

$$\kappa(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2}\right), \quad (5.21)$$

where σ is set as 0.2. In the experiments, we found that λ and σ are not sensitive to the test environments, since the results are not much affected by different choices of these parameters in different test scenarios. Note that to obtain a higher update rate, the optical flow systems in [92] and [93] only process gray scale images with size 64×64 and 128×96 , respectively. However, because of the high efficiency of FFT and element-wise operation, our correlation flow is able to process images

FIGURE 5.5: The estimated yaw rate and altitude velocity compared with ground truth from Vicon system.

with size 320×240 , resulting in much higher flow resolution, yet still with high update rate (real-time) on an ultra-low-power processor. We implement and test our framework on Ubuntu with robot operating system (ROS). The source codes are released at https://github.com/wang-chen/correlation_flow.

5.5.1.2 Platform

Limited by the payloads and power consumption, we choose a credit card-sized computing board, UP, that is equipped with an ultra-low-power processor x5-Z8350 with scenario design power of only 2W. Tests conducted on this computing board show that correlation flow leaves enough computational resources for other tasks, such as localization, path planning, graph optimization [106], and non-iterative SLAM [107]. Together with the computing board, an industrial IDS uEye UV-1551LE CMOS camera is mounted ventrally on a micro-quadcopter as shown in Figure 5.6. All the experiments are performed in a Vicon-equipped room, that can provide very accurate pose and velocity estimation at 50Hz as the ground truth. The experimental results are recorded on-board during fully autonomous flight.

FIGURE 5.6: The flight environment and the control scheme.

5.5.2 Velocity Estimation

The velocity estimation is compared with one of the state-of-the-art methods, PX4Flow [92], which integrates a gyroscope and a sonar altimeter. It might be one of the most suitable methods to compare, since it has been commercialized, fully tuned and tested by the robotic community. To be fair, it is mounted on the same platform mentioned in Section 5.5.1.2. Limited by the bandwidth of serial port, the images from PX4Flow cannot be recorded with full rate. Hence, only the estimation results are saved for comparison. Five on-board flight tests, each of which lasts more than 2 minutes, are performed in order to cover distinct traveling distances, speeds, dynamics, and illumination conditions. Both PX4Flow and correlation flow have an update rate of 30Hz. Table 5.1 shows the accuracy comparison in terms of root mean squared error (RMSE) and median absolute error (MAE). It is obvious that correlation flow outperforms PX4Flow in all the tests, resulting in the accuracy (RMSE) improvement of more than 100%. Figure 5.4 shows the plot of velocity estimation from one of the tests. It can be seen that correlation flow is able to provide more accurate and smoother velocity estimation.

Since most of the existing methods, including [92] and [93], are unable to provide the estimation on altitude and yaw change, we only compare the scale and rotation flow with the ground truth from the Vicon system. Figure 5.5 presents an example of the estimation results of the scale and rotation flow. Note that the ground truth of yaw rate is obtained by differentiating the attitude estimation from Vicon since

FIGURE 5.7: The position estimation by integrating the estimated velocity during fully autonomous flight. The drone tries to fly along a rectangle.

it cannot estimate yaw rate directly. The altitude and yaw velocity estimation is crucial for robust orientation control and landing, but is out of the scope of the thesis. One example using correlation flow to improve the control stability and localization accuracy is presented in [108].

5.5.3 Autonomous Flight

The main objective of this section is to show the potential of correlation flow for fully autonomous flight. The estimation from correlation flow can be fused by the state estimator in the flight controller based on an extended Kalman filter. Hence, we can estimate the position by integrating the estimated velocity. Figure 5.6 illustrates this simple control scheme. The velocity command is sent to the controller, for which the quadcopter tries to follow a simple rectangular trajectory. Figure 5.7 presents the estimated trajectory which is obtained in real-time flight lasting about 127s. It can be seen that fully autonomous flight is enabled without other positional device, except for an inertial measurement unit (IMU) in the controller. The mean trajectory estimation error is about 0.085m with standard deviation of 0.051m, which is acceptable for most of the flight applications. This demonstrates the feasibility of correlation flow for autonomous flight.

5.5.4 Hovering Test

As the same as all the other optical flow methods, the position estimation of correlation flow will drift due to long-term integration. This section presents the limit test for drifting, in which the maximum hovering time indicates the drifting speed and is measured for demonstration. The timer is stopped if the quadcopter drifts too much when the autonomous mode is switched back to manual control. Testing environment is shown in Figure 5.6, which is a screen-shot during the flight. Limited by the payloads including the sensors, the battery life of the platform mentioned in Section 5.5.1.2 is about 5 min. The quadcopter can hover within the flight area during the whole battery life, which further demonstrates the robustness of correlation flow.

5.6 Conclusion

In this chapter we propose a robust and computationally efficient optical flow method, called correlation flow for robot velocity estimation using a monocular camera. We introduce a kernel translation correlator and a kernel scale-rotation correlator for the camera motion prediction. Due to the high efficiency of fast Fourier transform, our method is able to run in real-time on an ultra-low-power processor. Experiments on velocity estimation show that correlation flow provides more reliable results than PX4Flow. Autonomous flight and hovering tests demonstrate that correlation flow is able to provide robust trajectory estimation at very low computational cost. The source codes are released for research purpose. To obtain higher efficiency, we have limited the image resolution in the experiments. We expect that correlation flow is able to achieve higher accuracy by simply improving the image resolution or fusing with other methods, when the computational cost is not such an important constraint.

Chapter 6

Non-iterative RGB-D-inertial Odometry

The theory of KCC has been developed in Chapter 3 to Chapter 5. From this chapter, the applications of KCC in visual simultaneous localization and mapping will be presented.

6.1 Introduction

Simultaneous localization and mapping (SLAM) is one of the most basic capabilities of autonomous robots, and has received increasing attention in tandem with the development of hardware, such as smarter sensors and faster processors [109]. However, visual dense mapping algorithms are still difficult to be applied directly to micro-devices, e.g. head-mounted augmented reality (AR) devices or micro unmanned aerial vehicles (MUAV). This is because micro-devices are only able to provide limited computational resources, such as ultra-low power embedded processors, due to payload and power limitations. Moreover, dense maps, which are crucial for higher level applications, such as collision-free motion planning, object detection and scene understanding, require more computational resources to produce [110]. This chapter aims to develop a lightweight dense mapping system that can be carried by micro-devices, and can achieve faster computation with sufficient accuracy. Leveraging on inertial fusion and KCC, this chapter proposes a non-iterative front-end (odometry) for RGB-D-inertial SLAM systems.

Odometry system, which is in charge of processing sensor information to generate observations to be fed to an estimation machinery [111], is the most important and time consuming part of a SLAM system. From the following analysis, we find that iterative solutions dominate the existing visual or visual-inertial odometry systems. First, iterative closest point (ICP) is one of the most widely used algorithms for point cloud matching [112]. The problem of ICP and its variants lies in the high complexity, so that many modern algorithms such as those in [113–117] need powerful GPU to process the large amount of data. Second, feature-based [110, 118–121] and direct methods [120, 122–126] have been proposed for monocular, stereo cameras, and RGB-D cameras [122, 127]. However, they also need iterative methods, e.g. Gauss-Newton, to minimize the reprojection or photometric error, so that the camera motion can be estimated. Third, the tightly-coupled inertial fusion SLAM algorithms using monocular camera [128], stereo camera [129], and RGB-D camera [130] also need iterative numerical solutions to solve an associated non-linear optimization problem. Fourth, to remove outliers, iterative methods like RANSAC [131] are also widely used [121, 127]. Last, the recent trend towards deep learning based-methods [132–136] stimulates a new wave of research, but the real-time performance may still be low, especially the iterative training process (back-propagation) can only be performed off-line. This challenge opens space for on-line learning techniques.

The above discussion shows that iterative solutions, including but not limited to the abovementioned, may be the main computational burden of an odometry system. To reduce the computational complexity, we have the following observations:

(i) A closed-form solution to odometry system may be able to reduce the computational complexity dramatically. Moreover, a closed-form solution doesn't require well-designed initialization to prevent convergence to a local minimum. However, the objective functions of traditional methods, e.g. tightly-coupled inertial fusion, feature-based and direct methods, are highly non-linear and difficult to find feasible closed-form solutions.

(ii) In many cases, inertial fusion typically introduces additional complexity due to additional filtering or optimization processes for inertial measurements. Is it possible to reduce the complexity based on inertial fusion? One of our intuitions is to determine the camera translation by visual techniques, and determine the attitude by rotation estimation from an inertial sensor, e.g. the attitude and heading

reference system (AHRS). However, the magnetic drift is inevitable, the rotation estimation sometimes needs corrections from visual measurements.

To reduce the complexity while keeping sufficient accuracy, we propose to combine a non-iterative front-end (odometry) with an iterative back-end (local and global loop closure) for the RGB-D-inertial SLAM system. To this end, we reformulate the problem of point cloud matching by first leveraging on the inertial fusion to decouple the point clouds. Then the KCC is applied to obtain a pose estimation with a closed-form solution. For the back-end, we leverage on the feature-based method that jointly optimizes key-frame poses with landmark features over a pose graph. This results in a lightweight key-frame-only based loop closure system that aims to correct both odometry and inertial drifts. It will be shown that the proposed framework provides a good compromise between efficiency and accuracy.

The main contribution of this work lies in

- A novel framework for RGB-D-inertial SLAM, especially the non-iterative front-end odometry system. It demonstrates that, for the first time, a non-iterative solution is feasible to the RGB-D-inertial odometry system.
- A set of algorithms is proposed to improve the stability and reduce the complexity, including the orthogonal reprojection, dynamic resolution, and the element-wise map fusion methods.
- It is demonstrated that the proposed framework is able to reduce the complexity of RGB-D SLAM dramatically. This results in a lightweight RGB-D-inertial SLAM system that can be carried by micro-robot systems.

6.2 Related Work

This chapter focuses on odometry system, dense mapping, and inertial fusion. The works on monocular/stereo camera will also be reviewed.

6.2.1 Visual Odometry and Map Fusion

Many dense mapping systems resort to ICP and its variants to align point clouds with respect to photometric or geometric constraints. KinectFusion [113] is one of the most famous methods, where the pose transformation is obtained by tracking the live depth frame relative to the global model using a coarse-to-fine ICP algorithm with geometric constraints. It permits dense volumetric reconstruction of complex room-sized scenes. Following this work, Kintinuous [114] enables dense mesh-based mapping for extended scale environments and implements a triangular mesh generation module for the map representation. ElasticFusion [115] combines dense geometric constraints with photometric constraints to achieve robust pose estimation in more challenging scenes. Furthermore, [137] brings Kintinuous and ElasticFusion together, and adds a method for improving camera-frustum overlap for a greater reconstruction range. The pose estimation is obtained by a dense every-frame volumetric fusion front-end, and the dense surface is corrected by a non-rigid map deformation back-end. DynamicFusion [117] generalizes the truncated signed distance function to nonrigid case, so that dynamic scenes can be reconstructed and a volumetric 6-D motion field can be estimated. BundleFusion [138] applies a local-to-global pose alignment framework and a paralleled sparse-to-dense optimization for sparse feature correspondence and dense geometric and photometric matching. Since the above methods heavily rely on ICP and require powerful GPU to process large amount of data, they are not applicable to systems with low computational power. Instead of matching points directly, DIFODO [139] applies a depth flow constraint on the motion model to derive the linear and angular velocities in a rigid environment.

To decrease the complexity, dense visual odometry (DVO) [123] proposes a faster, robust method based on a t-distribution-based photometric error model that can be obtained by an iteratively re-weighted least square (IRLS) algorithm. An improved version of DVO [122] extends the t-distribution model to depth error and proposes an entropy-based similarity measurement for key-frame selection and loop closure detection. The proposed t-distribution model reduces to the standard least square minimization when error residual is assumed to be normally distributed.

Feature-based method is one of the most widely-used methods in visual odometry. It leverages on an iterative optimization process to minimize reprojection error. The basic idea is that the transformation between camera poses can be recovered

by matched features. During the feature-detection step, salient key features that are likely to be matched well with other images are selected. Basically, there are two main approaches to find the corresponding features [140]. One is to extract features by local search techniques [120] and match them between the latest image and key-frames [121]. It is suitable for images taken from nearby viewpoints. The other one is to extract features independently and match them based on similarity descriptors [141], which can also be used for large motion between viewpoints.

Based on feature tracking, PTAM [119] first proposes to track the camera poses parallel to the mapping thread in a bundle adjustment framework. This enables real-time landmarks tracking and pose estimation at frame-rate. ORB-SLAM [121] carries forward this work by designing a robust system that uses the same features for all SLAM tasks: tracking, mapping, relocalization, and loop closing. This leads to a more reliable and complete solution. One of the earliest dense feature-based SLAM systems is RGB-D mapping [110] where Generalized-ICP and RANSAC are combined to optimize the pose graph. RGB-D-SLAM [127] exploits feature correspondences by sparse bundle adjustment and has become one of the most successful feature-based methods in RGB-D SLAM family.

In contrast with feature-based methods, direct methods match raw image pixels directly. Since there is no descriptor, only local search techniques can be used to find the corresponding pixels, hence, it needs well-designed initialization for large motion between two viewpoints. DTAM [126] develops a real-time camera tracking and reconstruction system where pose estimation is from the minimization of photometric error on every pixel. Within a depth probabilistic framework, SVO [120] achieves semi-dense maps with accurate motion estimation by minimizing photometric error outside non-negligible gradients regions. LSD-SLAM [125] proposes to build consistent large-scale maps by aligning image directly based on photoconsistency and estimating a filtering-based semi-dense depth map. By incorporating disparity sources from fixed-baseline, Stereo-LSD [142] avoids the scale-drift problem and builds a consistent map for large scale environments using stereo cameras. DSO [143] proposes a direct sparse odometry system that minimizes the photometric error on sampled pixels with large intensity gradients.

The deep learning based visual odometry methods have also been studied. In [132], the authors explore the convolutional neural networks (CNN) to learn visual features for ego-motion estimation. DeepVO [133] proposes a CNN-based architecture

for estimating the object’s pose under a known environment. CNN-SLAM [134] combines the direct methods with CNN-predicted depth maps to overcome the absolute scale problem of monocular SLAM. [136] proposes to train two CNNs, one of which operates on single image and extracts salient 2-D points, while another one operates on pairs of the points and estimates the homography.

6.2.2 Inertial Fusion

Inertial sensors provide additional constraints for pose estimation and help improve the performance of visual odometry systems. The most computationally efficient inertial fusion approach may be the loosely coupled methods, that process inertial and visual measurements separately. [144] presents an extended Kalman filter (EKF)-based constant complexity framework that treats the visual odometry system as a black box. This leads to a relatively low computational cost for additional processing. MSF-EKF [145] presents a generic and modular multi-sensor fusion framework to deal with delayed absolute or relative measurements. Alternatively, attitude estimation can be extracted and fused into visual estimation algorithms. [146] shows that angular precision of long term visual odometry can be improved a lot by loosely coupled fusion.

In contrast to loosely coupled methods, tightly-coupled methods estimate the states jointly. MSCKF [147] derives a measurement model that is able to express geometric constraints from static image features. It jointly estimates the IMU states and a sliding window of camera poses by an EKF estimator, resulting in a linear complexity in the number of features. MSCKF 2.0 [148] improved MSCKF and corrected the inconsistent linearized system model that has incorrect observability properties. OKVIS [129] constructs a joint probabilistic cost function that combines visual reprojection errors and inertial measurements to observe landmarks using monocular or stereo cameras. [128] proposes a joint non-linear optimization system using direct method, resulting in a semi-dense map. [149] develops a preintegration theory for IMU on the manifold structure of rotation group that is able to integrate the large number of inertial measurements between key-frames as a single geometric constraint. [150] combines inertial tracking in a dense SLAM framework for reconstructing large scale out-door scenes. [130] proposes the first tightly-coupled

FIGURE 6.1: The proposed framework of non-iterative RGB-D-inertial odometry system. The 3-D movements are estimated in decoupled subspaces separately. This process is of complexity $\mathcal{O}(n \log(n))$ where n is the number of image points.

dense RGB-D-inertial SLAM system for map deformations. Tightly-coupled methods have shown a potential to improve the accuracy, while it introduces additional optimization complexity due to the large numbers of measurements.

The deep learning based inertial fusion is also studied. Based on recurrent and convolutional networks, VINet [135] presents an on-manifold sequence-to-sequence learning approach to inertial fusion.

6.3 System Architecture

This section recalls the existing iterative methods and proposes a non-iterative framework for RGB-D-inertial odometry system. Iterative methods are defined as the odometry algorithms that need iterative solutions.

6.3.1 Iterative Framework

The SLAM algorithms for RGB-D cameras, such as [113–117, 137, 138] that heavily depend on ICP and its variants are the most representative examples of iterative methods. Let $\{\mathcal{M}, \mathcal{S}\}$ be two finite point sets with both position and color information, the objective of ICP is to find a transform $\mathcal{T}(\cdot)$ applied to $\mathcal{M}$, such that the distance D in (6.1) between $\mathcal{T}(\mathcal{M})$ and $\mathcal{S}$ is minimized [112].

$$D(\mathcal{T}(\mathcal{M}), \mathcal{S}) := \sum_{\mathbf{m} \in \mathcal{T}(\mathcal{M})} \sum_{\mathbf{s} \in \mathcal{S}} \rho(\mathbf{m} - \mathbf{s}), \quad (6.1)$$

where $\rho(\cdot)$ is a general objective function, and can be the square of the Euclidean distance combined with the photometric error. ICP and its variants are the most widely used point set registration algorithms, although they cannot guarantee the global minimum and are sensitive to initialization [130].

Feature-based and direct methods are also the examples of iterative methods. They minimize the reprojection or photometric error to recover pose transforms, respectively. Assume the perspective model $\pi_p(\cdot)$ projects 3-D point $\mathbf{p}_i$ to image point $\mathbf{u}_i$, so that $\mathbf{u}_i = \pi_p(\mathbf{E}_{CW}\mathbf{p}_i)$ where $\mathbf{E}_{CW}$ is the transform matrix from the world to camera frame. The camera pose $\mathbf{E}'_{CW}$ is expressed by left-multiplication of an exponential mapping, i.e. $\mathbf{E}'_{CW} = \exp(\mu)\mathbf{E}_{CW}$, where μ is a vector with six degrees of freedom (DoF).

In feature-based methods, feature points $\mathbf{u}_i$ in one image are reprojected to $\mathbf{u}'_i$ in another image, the pose transformation vector μ is estimated by minimizing the reprojection error $\mathcal{R}(\mu)$.

$$\mathcal{R} := \sum_i \rho(u'_i - \pi_p(\mathbf{E}'_{CW}\pi_p^{-1}(u_i, \mathbf{E}_{CW}))). \quad (6.2)$$

In [119], ρ is the Tukey biweight function; whereas in [110, 120], ρ is taken as the L_2 -norm. For monocular camera [119, 121], an initial process to obtain the point depth is needed, while for stereo/RGB-D camera [151, 152], this process can be skipped.

In direct methods, new images are aligned with several key frames I_r by minimizing the photometric error $\mathcal{H}(\mu)$:

$$\mathcal{H} := \sum_{i,r} \rho(I_r(u'_i) - I(\pi_p(\mathbf{E}'_{CW}\pi_p^{-1}(u_i, \mathbf{E}_{CW}))). \quad (6.3)$$

Similar expressions can be found in [120, 122, 125, 126, 153]. Note that the direct RGB-D SLAM system [122] also minimizes depth error over pixels.

The non-robustness to illumination change is one of the potential problems to match pixel intensities directly [126]. Since both (6.2) and (6.3) are highly non-linear, iterative numerical solutions, e.g. the gradient descent methods, are needed.

Iterative solutions also play a significant role in tightly-coupled inertial fusion with the above methods. The inertial residual is imposed into the objective function

FIGURE 6.2: Differences between perspective and orthogonal projections. The perspective projection (a) projects points on the principle point of the camera, while orthogonal projection (b) projects points on the image plane. A quadrangular frustum pyramid can be a rectangle when the vertex coincides with the principle point. While for orthogonal image, the size ratio of the two rectangles will not change. This simple property is applied for data decoupling.

of ICP (6.1), reprojection error (6.2), and photometric error (6.3), resulting in the RGB-D-inertial [130], reprojection-inertial [129], and direct-inertial [128] optimization problems, respectively. Similarly, iterative numerical solutions are needed.

6.3.2 Non-iterative Framework

Our proposed non-iterative framework for the front-end shown in Figure 6.1 consists of several blocks and will be explained in the Section 6.4 to 6.9. A procedure of decoupling-estimation-recoupling is proposed as follows: 3-D point clouds are first decoupled into several subspaces, where data can be matched independently with lower complexity; then the camera motion is obtained from recoupling the estimation in the subspaces. Based on the single key-frame training using kernel cross-correlator (KCC), the translation of the camera pose is predicted by finding the maximum response of the correlation output. This enables the fast motion estimation on ultra-low power processors. Finally, 3-D maps are fused with non-key-frames by a moving average, so that the missing information of key-frames is complemented with complexity $\mathcal{O}(n)$. Unless otherwise stated, the complexity analysis in this chapter is in terms of the number of points n in an image.

6.4 Data Decoupling

This section introduces the data decoupling, from 6 DoF to 2 DoF. As shown in Figure 6.1, we are introducing a loop where the first block depends on the output of the last block, i.e. the rotation estimation from the last cycle is needed for data decoupling. At the beginning, the camera is assumed to be located at the origin.

6.4.1 6 DoF to 3 DoF

It has been shown that even the low-cost, low-precision inertial measurement unit (IMU) can significantly increase performance of the attitude estimation [146]. To avoid the traditional optimization problem and decouple the rotational and translational movement, the attitude estimation from Section 6.9 is applied directly. It is obtained from fusion of back-end and the attitude and heading reference system (AHRS)¹. A point cloud is a set of points with pose $\mathbf{G} \in \mathbf{SE}(3)$ and is defined with a specified color to represent an external surface of an object or environment. Assume the k_{th} key-frame is denoted as $\mathcal{K}_k(\mathbf{G}_k)$, a point cloud $\mathcal{P}_i(\mathbf{G}_i)$ can be rotated to align with the key-frame by its orientation $\mathbf{R} \in \mathbf{SO}(3)$. The aligned point cloud $\tilde{\mathcal{P}}_i$ is obtained through

$$\tilde{\mathcal{P}}_i(\mathbf{t}_i) = \mathbf{R}_k(\mathbf{R}_i)^{-1}\mathcal{P}_i(\mathbf{G}_i), \quad (6.4)$$

where

$$\mathbf{G}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{bmatrix} \text{ with } \mathbf{R}_i \in SO(3) \text{ and } \mathbf{t}_i \in \mathbb{R}^3. \quad (6.5)$$

In this sense, the original 6 DoF $\mathbf{G}_i$ is reduced to 3 translational DoF. From now on, we only need to estimate the translation between point clouds $\mathcal{P}_i$ and $\mathcal{K}_k$.

The idea of estimation after decoupling is proved to be feasible in [154], which leverages on orientation histograms for rough attitude estimation. However, it requires a fine alignment in the final step by running ICP and the translation estimation is based on linear correlation of two point clouds via 3-D Fourier transform with complexity $\mathcal{O}(n_x n_y n_z \log(n_x n_y n_z))$, where n_x, n_y, n_z are the number of voxels in each dimension, respectively. It will be shown that our method is only of complexity $\mathcal{O}(n \log n)$, where $n = n_x n_y$ is the number of points in the image.

¹<https://en.wikipedia.org/wiki/AHRS>

FIGURE 6.3: The point cloud obtained by a depth camera in Figure (a) is rectified by the attitude estimation from the back-end. Its perspective image (b) is then reprojected to orthogonal color image (c) and depth image (d). Since the missing points in (b) are also projected, there exists black holes in (c) and (d).

One of the potential problems to decouple point clouds by rotation estimation is that the distorted inertial measurements may have a negative influence on translation estimation. For example, magnetic drift or attitude bias can suddenly change in severe situations. In this work, we assume that they are short-term stable, i.e. the bias and drift estimation can roughly follow their real values between two consecutive key-frames. The accumulated long-term drift is corrected by the visual local and global loop closure. Since they are running in the back-end and only performed on key-frames, the system complexity is still dominated by the odometry part. It will be shown that this provides a good compromise between performance and computational complexity, since the non-iterative solution to the odometry system reduces the complexity dramatically yet still with comparable accuracy.

6.4.2 3 DoF to 2 DoF

Computation requirements can be further reduced if the 3-D translational movements are also decoupled. The principle is that the geometry properties in the 3

axes must be kept. We propose to apply orthogonal projection instead of respective projection on the aligned point cloud $\tilde{\mathcal{P}}_i(\mathbf{t}_i)$ to get orthogonal color and depth images $\mathbf{x}^C, \mathbf{x}^D \in \mathbb{R}^{n_x \times n_y}$. A 3-D point $\mathbf{p} = (x, y, z)^T \in \mathcal{P}$ on the visible surface is mapped to image coordinates $\mathbf{u} = (u, v)^T \in \Omega$ via the orthogonal projection model $\pi_a : \mathbb{R}^3 \mapsto \mathbb{R}^2$, which is defined as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{r_k} \begin{bmatrix} x \\ y \end{bmatrix}_{\tilde{c}}, \quad (6.6)$$

where $r_k \in \mathbb{R}^+$ is the projection resolution that can be set adaptively. The subscript $\tilde{c}$ denotes that the point coordinates are expressed in the aligned camera frame. The orthogonal depth image is defined as:

$$\mathbf{x}^D(\mathbf{u}) = [z]_{\tilde{c}}. \quad (6.7)$$

It stores the point distances to the orthogonal plane versus the perspective depth image that stores the distances to the camera center. This slight difference is illustrated in Figure 6.2. The perspective projection cannot keep the geometric relationship, since an object becomes smaller as its distance from the viewpoint increases. While in orthogonal projection, the size ratio of objects is kept the same, regardless their distances. This simplifies the estimation process, since we can now estimate the 2-D translation of orthogonal images followed by 1-D depth translation estimation perpendicular to the plane.

Before that, one thing that needs to be considered is the projection resolution r_k which is physically the size of covered space by each pixel. The field of view of orthogonal projection is a cuboid space with the central axis coinciding with the camera optical axis. The covered space is determined by the projection resolution r_k and the image size $n_x \times n_y$. Since points outside of the cuboid will be cropped, r_k is to be set large enough to make the cuboid cover most of the points. Meanwhile, there will be a large black margin on the image, if r_k is set too large, since not all the pixels can be covered by the projected points. Moreover, the minimum cuboid envelope of a point cloud may change dramatically because of the movements.

To be robust to changing scenes, we implement a searching procedure to find the resolution r_k , so that the cuboid space covers 80% points in the cloud (set empirically, most of the boundary points are outliers). To be efficient, it traverses

the cloud by sampling one point in every 25 points. Note that in this process, we keep the central axis of the cuboid envelope coincided with the camera's optical axis. Although the sampling is not precise, we find that the approximation is acceptable and its complexity is only $\mathcal{O}(n/25)$, so that the computational time can even be ignored compared to the runtime in the following sections. To further reduce repeated computations, the resolution r_k is updated only when a key-frame is created. Another advantage of adaptive resolution is that when the camera is near to the scene surfaces, more details can be preserved with smaller r_k .

Since the point clouds are generated by pinhole cameras, when they are reprojected on the orthogonal plane, some black holes may appear inside the image. This is because some missing points for pinhole cameras are also projected on the orthogonal plane. We will show that they can be filled by the map fusion process presented in Section 6.8. Figure 6.3 shows an example that a 6 DoF point cloud is rectified by rotation estimation and reprojected to orthogonal images.

6.5 Key-frame Training

We propose to apply the KTC to predict the image translation. The superscript $\mathcal{C}, \mathcal{D}$ of orthogonal key-frame and test images $\mathbf{z}^{\mathcal{C}, \mathcal{D}}, \mathbf{x}^{\mathcal{C}, \mathcal{D}} \in \mathbb{R}^{n_x \times n_y}$ will be left out in the absence of ambiguity. Assume $\mathbf{z}_{ij} = \mathcal{S}_{(i,j)}(\mathbf{z})$ is the circular translation of the key-frame $\mathbf{z}$ with (i, j) elements. Given a desired output $\mathbf{g} \in \mathbb{R}^{n_x \times n_y}$, the KTC is

$$\hat{\mathbf{g}} = \hat{\kappa}_{\mathbf{z}}(\mathbf{x}) \odot \hat{\mathbf{h}}^*, \quad (6.8)$$

where $\kappa_{\mathbf{z}}(\mathbf{x})$ is the kernel matrix, with element in i_{th} row and j_{th} column as $\kappa(\mathbf{x}, \mathbf{z}_{ij})$. The bottleneck of computation is still the forward and backward FFTs, hence, the complexity of the key-frame training has an upper bound $\mathcal{O}(n \log n)$, where $n = n_x \times n_y$ is the number of elements in the image.

The training objective is to find a filter $\mathbf{h}$ ($\mathbf{h} \in \mathbb{C}^{n_x \times n_y}$) maps $\mathbf{x}$ to $\mathbf{g}$ with least error. Similar to (3.8), training is conducted in the Fourier domain to take advantage of the simple element-wise operation, i.e. the sum of squared error $F(\mathbf{h}^*)$ in (6.9) between the correlation output and the desired output is minimized, whereas

objective (6.9) only regards the single key-frame $\mathbf{z}$ as the training sample.

$$F(\mathbf{h}^*) = \|\hat{\kappa}_{\mathbf{z}}(\mathbf{z}) \odot \hat{\mathbf{h}}^* - \hat{\mathbf{g}}\|^2 + \lambda \|\sqrt{\hat{\kappa}_{\mathbf{z}}(\mathbf{z})} \odot \hat{\mathbf{h}}^*\|^2, \quad (6.9)$$

where the second term is a regularization to prevent overfitting and $\sqrt{\cdot}$ is element-wise square root operation. The solution to (6.9) can be derived from (3.9), which is rewritten in (6.10).

$$\hat{\mathbf{h}}^* = \frac{\hat{\mathbf{g}}}{\hat{\kappa}_{\mathbf{z}}(\mathbf{z}) + \lambda}, \quad (6.10)$$

where the operator $\div$ denotes the element-wise division.

In this sense, the pattern of translational motion of the key-frame $\mathbf{z}$ is encoded in the filter $\mathbf{h}$. In the experiments, the radial basis function kernel (6.11) is applied.

$$\kappa(\mathbf{x}, \mathbf{z}_{ij}) = h(\|\mathbf{x} - \mathbf{z}_{ij}\|^2). \quad (6.11)$$

The kernel matrix $\kappa_{\mathbf{z}}(\mathbf{x})$ is computed following (5.9), which is rewritten in (6.12).

$$\kappa_{\mathbf{z}}(\mathbf{x}) = h\left(\frac{1}{n}\|\hat{\mathbf{x}}\|^2 + \frac{1}{n}\|\hat{\mathbf{z}}\|^2 - 2 \cdot \mathcal{F}^{-1}(\hat{\mathbf{x}} \odot \hat{\mathbf{z}}^*)\right). \quad (6.12)$$

Therefore, the key-frame training process (6.10) and (6.12) is of complexity $\mathcal{O}(n \log n)$, which is dominated by the forward and backward FFTs.

6.6 Translation Estimation

Based on KTC, the translational pattern of an orthogonal image can be estimated via (6.8) by regarding it as a test sample.

6.6.1 Image Translation Estimation

To make the correlation output distinctive, we simply set the first element of $\mathbf{g}$ as 1, while all the others as 0, i.e. $g[0, 0] = 1$, where the bracket $[\cdot]$ is for element indexing (starting from 0). The explanation is that the shift of value 1 in correlation output corresponds to the shift of test image (Theorem 3.3.3). Because of noises and occlusion, it is impossible to get the exact peak value 1. Instead, the location

FIGURE 6.4: The output from KCC on an example sequence. (a) is the first key-frame; the top-left figure of (b), (c), and (d) are selected from its subsequent frames. Their KCC outputs are shown on the top-right of their corresponding figures whose PSR is 161, 71, and 41, respectively. The outputs are normalized and shown in 3-D space for better visualization. It is assumed that the KCC outputs can be approximated by 2-D Gaussian functions with the center on the peak, which is useful for fast computing of estimated covariance in (6.15).

of the maximum value is used to find the translation:

$$[\tilde{x}, \tilde{y}] = \arg \max_{i,j} \underbrace{\mathcal{F}_{[i,j]}^{-1}(\hat{\kappa}_{\mathbf{z}}(\mathbf{x}) \odot \hat{\mathbf{h}}^*)}_{\mathbf{g}_{[i,j]}(\mathbf{x})} \quad (6.13)$$

Then the predicted translation $[\mathbf{t}_x, \mathbf{t}_y]^T$ of orthogonal image is obtained:

$$[\mathbf{t}_x, \mathbf{t}_y]^T = r_k \cdot [\tilde{x}, \tilde{y}]^T. \quad (6.14)$$

The complexity of (6.13) is bounded by the kernel matrix and inverse FFT, hence, the process of image translation estimation is of complexity $\mathcal{O}(n \log(n))$.

We next present the estimation of covariance $\sigma_{\mathbf{t}_x}^2$ and $\sigma_{\mathbf{t}_y}^2$. Intuitively, the value of each position in the correlation output indicates the estimation confidence of the corresponding translation, thus the estimation covariance can be computed as the covariance of relative weighted translation to the peak. In the experiments, we find that the normalized correlation output $\mathbf{g}(\mathbf{x}) / \sum \mathbf{g}(\mathbf{x})$ can be approximated

FIGURE 6.5: The image (a) is an example of orthogonal depth image from Figure 6.3. The corresponding well-matched points are shown in (b) by pixel intensities. The higher the brightness is, the more confidence the matches have.

by a Gaussian distribution with the center located at the peak, which is shown in Figure 6.4. Therefore, we approximate the covariance by using the peak value of a standard 2-D Gaussian function, which is $1/(2\pi\sigma^2)$, so that

$$\sigma_{\mathbf{t}_x}^2 = \sigma_{\mathbf{t}_y}^2 = \frac{r_k^2 \cdot \sum \mathbf{g}(\mathbf{x})}{2\pi \cdot \max \mathbf{g}(\mathbf{x})}. \quad (6.15)$$

6.6.2 Depth Translation Estimation

The overlap of orthogonal image $\mathbf{x}$ can be matched with key-frame $\mathbf{z}$, if $\mathbf{x}$ is shifted back with $[\tilde{x}, \tilde{y}]$ elements. Inspired by this, the camera motion in the depth direction can be estimated in (6.16a) which averages the depth differences of the matched pixels. To eliminate the influence of dynamic points, it only takes the well-matched pixels that is defined by the set $\mathcal{W}$ in (6.16b).

$$\mathbf{t}_z = \frac{1}{|\mathcal{W}|} \sum_{(i,j) \in \mathcal{W}} (\mathcal{S}_{(-[\tilde{x}, \tilde{y}])}(\mathbf{x}_{[i,j]}^{\mathcal{D}}) - \mathbf{z}_{[i,j]}^{\mathcal{D}}), \quad (6.16a)$$

$$\mathcal{W} = \left\{ i, j \mid \rho \left(\mathcal{S}_{(-[\tilde{x}, \tilde{y}])}(\mathbf{x}_{[i,j]}^{\mathcal{C}, \mathcal{D}}) - \mathbf{z}_{[i,j]}^{\mathcal{C}, \mathcal{D}} \right) < T_{c,d} \right\}, \quad (6.16b)$$

where the operator $|\cdot|$ returns the number of elements in a set and $\rho(\cdot)$ is a general objective function (L_1 -norm in our tests). The moving objects naturally cannot be matched with shifted image $\mathcal{S}_{(-[\tilde{x}, \tilde{y}])}(\mathbf{x})$, thus the parameters T_c and T_d are designed to control the elimination of dynamic points.

One of the advantages of (6.16) is that it only requires the average differences of depth image which is extremely fast to compute and all the well-matched points

are able to contribute to the estimation, resulting in the robustness to depth noises. The estimated variance can be obtained via direct calculation:

$$\sigma_{\mathbf{t}_z}^2 = \text{Var}_{(i,j) \in \mathcal{W}} (\mathcal{S}_{(-[\tilde{x}, \tilde{y}])}(\mathbf{x}_{[i,j]}^{\mathcal{D}}) - \mathbf{z}_{[i,j]}^{\mathcal{D}}). \quad (6.17)$$

Hence, the complexity of depth translation estimation is $\mathcal{O}(n)$. Figure 6.5 presents an example for orthogonal depth image and the corresponding well-matched points. Until now, the translation of a point cloud relative to the key-frame is obtained in all three directions.

6.7 Key-frame Selection

As the camera moves, the overlap between current frame and key-frame may decrease. Hence, the peak value of the correlation output may not be distinct enough to determine the translation. In this case, new key-frames need to be created for better matching quality. Since the new frames are only matched with the nearest key-frames, estimation error will be accumulated. Therefore, we need to be very careful to create key-frames. This section presents the conditions to select key-frames. Section 6.9 shows that the accumulated error can be corrected by the back-end relocalization and loop closure.

Key-frame selection has been studied. For example, DVO [122] uses the logarithmic determinant of the error distribution covariance matrix to determine key-frames. PTAM [119] creates several conditions: the distance to the last key-frame is large enough; at least twenty key-frames are passed. ORB-SLAM [121] adds the condition: a minimum percentage of tracked features must be reached. However, these conditions are designed in an ad-hoc way and is not suitable for our scenario, since we use different techniques.

We argue that the condition for creating key-frames should be able to represent the matching quality. However, the overlap of two frames is not suitable, since it may not be proportional to the matching quality because of the existence of dynamic objects. Considering the computational cost, we use a very simple criterion in (6.18), i.e. peak to sidelobe ratio (PSR), which is a measurement of peak strength [31]. It splits the correlation output $\mathbf{g}(\mathbf{x})$ into the peak and the sidelobe which

FIGURE 6.6: The first 200 frames of a sequence are extracted to show the changes of PSR with respect to frame number. Only when the PSR of the correlation output goes below T_K , a new key-frame will be created. The point clouds will be fused with the key-frame when PSR is higher than T_M .

contains the rest of pixels excluding the peak.

$$P_{sr}(\mathbf{x}) = \frac{\max \mathbf{g}(\mathbf{x}) - \mu_s}{\sigma_s} < T_K, \quad (6.18)$$

where μ_s and σ_s are the mean and standard deviation of the sidelobe. A straightforward explanation is that the peak strength of the correlation output indicates the matching confidence level. In this sense, the desired output with single peak has infinite peak strength. $P_{sr}(\mathbf{x})$ is regarded as a trigger to insert a new key-frame into the map when it is smaller than T_K . The criterion (6.18) is not only able to control the minimum confidence of each matching, but also save computational time, especially when the camera is kept still or moving slowly since there is no new training data is required. Only the calculation of mean and standard deviation of the sidelobe are performed in (6.18), hence, the complexity for key-frame selection is $\mathcal{O}(n)$.

Another important capability of (6.18) is to control the inertial drift compensation. In case of sudden magnetic interference or dynamic inertial biases, (6.18) will result in creation of more key-frames; (6.15) and (6.17) will result in larger estimation covariances. As a result, the back-end optimization will have more information for finer relocalization, which has a greater weight in pose graph, resulting in higher chance of recovering the true inertial states. More details about back-end optimization and loop closure will be presented in Section 6.9.

FIGURE 6.7: (a) is a new key-frame. (b) is the refined key-frame. It is shown that the ‘black holes’ in the original key-frame can be filled by subsequent matched frames based on (6.19). (c) and (d) are the same part of (a) and (b), respectively. Note that the keyboard is well complemented. The moving average in (6.19) will not make it blurred, as long as the image translation is predicted correctly.

6.8 Map Refinement and Fusion

To reduce the complexity, only the key-frames are selected to represent a map. As mentioned earlier, not all the pixels of orthogonal images can be reprojected by valid points, and some black holes may appear. A very simple idea is to complement the missing information of the key-frame $\mathbf{z}$ by the matched non-key-frames $\mathbf{x}_k$. This is useful when the criterion (6.18) is not satisfied, i.e. the camera keeps still or moves slowly.

We first define a weight matrix $\mathbf{w} \in \mathbb{R}^{n_x \times n_y}$ for each frame. The element of $\mathbf{w}$ presents the weight of the corresponding pixel to be fused. Each time a frame $\mathbf{x}_k$ is acquired, its corresponding weight matrix $\mathbf{w}_k^{\mathbf{x}}$ is initialized as a binary matrix, i.e. $\mathbf{w}_k^{\mathbf{x}} \leftarrow \{0, 1\}^n$ where 1 or 0 indicates whether that pixel can be fused or not.

The weight matrix $\mathbf{w}^{\mathbf{z}}$ for the key-frame is the element-wise summation of $\mathbf{w}_k^{\mathbf{x}}$ that has been fused. Initially, $\mathbf{w}^{\mathbf{z}}$ is simply set as $\mathbf{w}_k^{\mathbf{x}}$ when $\mathbf{x}_k$ is selected as a new key-frame. Therefore, the map fusion process can be expressed as a moving average in

(6.19) which is performed in both color and depth images.

$$\mathbf{s}_k \leftarrow \mathbf{w}^z + \mathcal{S}_{(-[\tilde{x}, \tilde{y}])}(\mathbf{w}_k^x) + e, \quad (6.19a)$$

$$\mathbf{z}^c \leftarrow \frac{(\mathbf{w}^z \odot \mathbf{z}^c + \mathcal{S}_{(-[\tilde{x}, \tilde{y}])}(\mathbf{w}_k^x \odot \mathbf{x}_k^c))}{\mathbf{s}_k}, \quad (6.19b)$$

$$\mathbf{z}^D \leftarrow \frac{(\mathbf{w}^z \odot \mathbf{z}^D + \mathcal{S}_{(-[\tilde{x}, \tilde{y}])}(\mathbf{w}_k^x \odot (\mathbf{x}_k^D - \mathbf{t}_z)))}{\mathbf{s}_k}, \quad (6.19c)$$

$$\mathbf{w}^z \leftarrow \mathbf{w}^z + \mathcal{S}_{(-[\tilde{x}, \tilde{y}])}(\mathbf{w}_k^x), \quad (6.19d)$$

where e is a small scalar (10^{-7}) to prevent division by 0.

Because of the dynamic points and outliers, some valid points in $\mathbf{x}_k$ can not be matched with the key-frame. Hence, we remove the unmatched points before (6.19) is performed. The elements of $\mathbf{w}_k^x$ corresponding to the unmatched points are set to 0 as in (6.20), so that they cannot be fused into the key-frames.

$$\mathbf{w}_{k[i,j]}^x \leftarrow 0, \text{ if } (i, j) \notin \mathcal{W}, \quad (6.20)$$

where the set $\mathcal{W}$ defined in (6.16b) contains the indexes of all matched points. To obtain a higher quality map, we perform (6.19) only when $P_{sr}(\mathbf{x}_k) > T_{\mathcal{M}}$ where $T_{\mathcal{M}}$ is a parameter to control the confidence of map fusion and $T_{\mathcal{M}} > T_{\mathcal{K}}$.

The PSR value plays an important role in our algorithm. Figure 6.6 presents an example for the changes of PSR in terms of frames. This sequence is extracted from the first 200 frames of freiburg1_desk from TUM dataset [155]. It can be seen that when the matching confidence is lower than $T_{\mathcal{K}}$, a new key-frame will be created, so that the matching confidence returns to a high value.

Since all the operations in (6.19) are element-wise, the complexity of map fusion process is $\mathcal{O}(n)$. Figure 6.7 gives an example where a key-frame is refined by non-key-frames. It demonstrates that the missing information of key-frames can be complemented, so that more details are preserved.

6.9 Visual Loop Closure

6.9.1 Relocalization

The relocalization is checked among key-frames. To “memorize” the visual appearances, we extract and save 1000 ORB features [156] for every key-frames. Since ORB feature descriptor is binary, it allows quick comparison by direct Hamming distance. Moreover, in order to perform quick image retrieval for relocalization, Bag-of-Words model [157] is applied to re-represent ORB key-frame features by visual words. Moreover, these visual words are taken from a pretrained dictionary that is similar to the one deployed in ORB-SLAM2. The vocabulary, pretrained with Bovisa 2008-09-01 dataset [158], contains 1 million words, arranged in 6 levels, each with 10 clusters [159]. The relocalization mechanism is implemented based on the excellent work of DBoW2. Firstly, potential match pairs $\langle b_t, b_{t-j} \rangle$, where b_t is the bag of key-frame t , are identified by a normalized similarity score [160].

$$\eta(b_t, b_{t-j}) = \frac{s(b_t, b_{t-j})}{s(b_t, b_{t-1})}, \quad (6.21)$$

where s is L1-score between two bags:

$$s(b_i, b_j) = 1 - \frac{1}{2} \left| \frac{b_i}{|b_i|} - \frac{b_j}{|b_j|} \right|. \quad (6.22)$$

These candidates then undergo a second test to determine if their relative transformation and PSR value are within a threshold.

6.9.2 Pose Graph Optimization

To correct accumulated error, we optimize a pose graph upon loop closure detection. A pose graph, with key-frames as $\mathbf{SE}(3)$ nodes interconnected by the relative transform and covariance estimation, is incrementally constructed as new key-frames are inserted. Once confident relocalization is available, we minimize the sum of squared error over the global key-frame trajectory using the Lavenberg-Marquardt algorithm. Our pose-graph optimization scheme leverages on the existing implementation of g²o [161]. Different from the preliminary version [107],

where the pose is recoupled by loosely coupled fusion, the back-end pose graph can correct both inertial and odometry drift. Moreover, since only key-frames are involved, the runtime is dominated by the odometry part.

In the experiments, we find that such feature-based method is sometimes sensitive to the pseudo-features which are produced by light reflections via pieces of glass, resulting in wrong trajectory optimization. To overcome this problem, we implement a simple false relocalization detection by comparing with the front-end odometry. If the results of graph optimization is too far away from the odometry, we simply reject such relocalization.

6.10 Experiments

Throughout this chapter, a set of non-iterative algorithms have been designed to reduce the complexity of the front-end of a RGB-D-inertial SLAM system. We will evaluate the system both quantitatively and qualitatively in terms of dense mapping, trajectory estimation and runtime efficiency. We also test the performance on micro-devices.

6.10.1 Implementation

The framework is implemented using FFTW3 [162] and Eigen [163] libraries for FFT and matrix calculation. The system runs in ROS [164] and is pre-calibrated using the Kalibr toolbox [165]. The inertial and visual measurements are simply synchronized using the ROS timestamp. Due to the communication channel delay, this will introduce a short time difference. However, we find that the performance is acceptable. We expect that a hardware triggering synchronization produces better performance.

In the experiments, the Gaussian kernel is applied in (3.14) with a standard deviation of 0.2 pixels. The regularization term λ in solution (6.10), the matching difference parameters T_c and T_d in (6.16b) are all set to 0.1. It is found that these parameters are not sensitive to the test environments, since different scenarios and sensors are tested and the results are not much affected by different choices of these parameters. The PSR thresholds T_K and T_M are set to 50 and 100, respectively.

TABLE 6.1: Simulation performance of ATE RMSE, mean, and median error on TUM dataset. ‘Dynamic’ means whether dynamic objects are contained in the corresponding sequence.

Dataset	RMSE (m)	Mean (m)	Median (m)	Std (m)	$\bar{v}$ (m/s)	$\bar{\omega}$ (°/s)	Dynamic
freiburg1_rpy	0.062	0.054	0.049	0.031	0.062	50.15	
freiburg1_xyz	0.011	0.010	0.008	0.006	0.244	8.92	
freiburg1_desk	0.025	0.024	0.022	0.009	0.413	23.32	
freiburg1_desk2	0.055	0.048	0.041	0.028	0.426	29.31	
freiburg1_plant	0.103	0.091	0.079	0.049	0.365	27.89	
freiburg1_room	0.128	0.112	0.098	0.062	0.334	29.88	
freiburg2_rpy	0.025	0.021	0.018	0.013	0.014	5.77	
freiburg2_xyz	0.012	0.011	0.010	0.005	0.058	1.72	
freiburg3_cabinet	0.078	0.076	0.076	0.021	0.216	10.25	
freiburg3_large_cabinet	0.106	0.096	0.091	0.066	0.362	8.75	
freiburg3_sitting_rpy	0.036	0.027	0.021	0.023	0.042	23.84	✓
freiburg3_sitting_xyz	0.031	0.029	0.028	0.012	0.132	3.56	✓
freiburg3_sitting_static	0.006	0.005	0.006	0.003	0.011	1.70	✓
freiburg3_walking_xyz	0.056	0.044	0.037	0.035	0.208	5.49	✓
freiburg3_walking_static	0.024	0.018	0.014	0.016	0.012	1.39	✓
freiburg3_walking_halfsphere	0.115	0.094	0.078	0.066	0.221	8.27	✓
freiburg3_long_office_household	0.033	0.031	0.030	0.011	0.249	10.19	
freiburg3_structure_texture_near	0.019	0.018	0.017	0.006	0.141	7.68	
freiburg3_structure_texture_far	0.014	0.013	0.012	0.005	0.193	4.32

FIGURE 6.8: Figure (a) and (b) are the dense reconstruction of our lab. The panorama of this room is given in Figure (c). This environment is quite challenging for feature-based methods, since there are many feature-less area and light reflections produced by pieces of glass.

6.10.2 Simulation on TUM dataset

We first evaluate the performance of trajectory estimation using the TUM RGB-D benchmark [155] that provides synchronized ground truth from a motion capture system, which is also used to simulate inertial measurements in the tests. To demonstrate the effectiveness of our proposed non-iterative front-end, which mainly contributes to the translation estimation, we evaluate the system based on position error instead of rotation. The position accuracy in terms of absolute trajectory (ATE) root-mean-square error (RMSE), mean error, and median error are listed in Table 6.1, where the standard deviation of RMSE, maximum error, average velocity and angular rate are also given. More details about the calculation of these metrics can be found in [155]. In comparison, all estimated camera poses are associated with the ground truth by their nearest time stamp.

Comparable performance on this dataset can be found in state-of-the-art works, e.g. Volumetric Fusion [137], RGB-D SLAM2 [127], ElasticFusion [116], and ORB-SLAM2 (RGB-D) [121, 159]. Although we cannot see a significant improvement on accuracy (we get better results in some sequences, but the numbers are still on the same magnitude), our framework can run much faster ($2\times-4\times$), and perform

FIGURE 6.9: The examples of color (first row) and depth (second row) images, in which lots of specular reflections and featureless regions can be seen.

well on some scenarios which are challenging for other methods, as will be seen in the following sections.

6.10.3 Dense Mapping

In this section, we present the qualitative performance of in-door dense reconstruction. Figure 6.8 presents the 3-D map of our lab, that is generated in real-time and shown from different point of view in (a) and (b), respectively. The full view of this room is shown in Figure 6.8 (c) where large featureless area and visual pseudo-features produced by pieces of glass can be seen. This environment is quite challenging for feature-based methods, since the pseudo-features will move along with the camera movement, which will produce wrong pose estimation. For our method, the kernel cross-correlator matches the point cloud as a whole, regardless the distinct features. Therefore, our method is insensitive to the visual corner outliers and work better in such situations. Note that the back-end of our system sometimes also produces false relocalization due to the feature outliers, while most of them are rejected by the front-end odometry stated in Section 6.9.2.

In Chapter 7, we will also construct the dense map for the same environment, but requires the aids from extern ultra-wideband (UWB) anchors. The performance of other methods in such environment is presented in next section.

TABLE 6.2: Root mean square error (RMSE) comparison on the benchmark. Only instant pose estimation is accepted.

Dataset	ours		ORB-SLAM2		ElasticFusion		RGB-D SLAM2	
	RMSE	std.	RMSE	std.	RMSE	std.	RMSE	std.
01	0.106	0.038	0.323	0.095	0.439	0.165	0.122	0.060
02	0.373	0.230	-	-	0.602	0.185	0.575	0.417
03	0.123	0.055	0.142	0.079	0.444	0.171	0.191	0.057
04	0.076	0.037	0.058	0.029	0.123	0.058	0.124	0.055
05	0.093	0.040	0.513	0.219	0.281	0.121	-	-
06	0.113	0.045	0.128	0.057	0.355	0.139	0.543	0.155
07	0.021	0.009	0.025	0.015	0.024	0.012	0.031	0.022

TABLE 6.3: Mean error and maximum error comparison on the benchmark. Only instant pose estimation is accepted.

Dataset	ours		ORB-SLAM2		ElasticFusion		RGB-D SLAM2	
	MEAN	MAX	MEAN	MAX	MEAN	MAX	MEAN	MAX
01	0.099	0.261	0.308	0.508	0.407	0.688	0.106	0.248
02	0.294	1.020	-	-	0.572	1.067	0.397	1.527
03	0.110	0.300	0.118	0.380	0.410	0.778	0.182	0.389
04	0.066	0.211	0.051	0.150	0.109	0.256	0.111	0.470
05	0.084	0.183	0.464	1.003	0.254	0.557	-	-
06	0.103	0.209	0.114	0.352	0.327	0.591	0.520	0.840
07	0.019	0.053	0.021	0.065	0.021	0.050	0.022	0.186

6.10.4 Performance on Benchmark

6.10.4.1 Dataset and Platform

The experimental benchmark² contains 7 sequences that are totally 39.7 GB and recorded in an indoor environment. The ground truth is obtained from a highly

²https://wangchen.online/ni_slam

TABLE 6.4: Real-time efficiency comparison on Kinect dataset with PC platform. All the data are give by average update rate (Hz).

Dataset	ours	ORB-SLAM2	RGB-D SLAM2
01	73.4	41.1	18.7
02	82.2	-	20.1
03	81.6	41.5	18.3
04	80.7	45.6	20.9
05	77.2	42.7	21.1
06	77.8	46.6	21.6
07	85.1	50.0	23.3
overall	79.7	44.6	20.6

accurate motion capture system (Vicon). Except for the ground truth, this benchmark include measurements from Kinect at 30Hz, inertial measurements from myAHRS+³ at 100Hz, which is a low cost high performance inertial sensor containing a 16-bit 3-axis gyroscope, a 16-bit 3-axis accelerometer, and a 13-bit 3-axis magnetometer. Some of the image examples are shown in Figure 6.9. They are quite challenging for vision-based methods because there are lots of specular reflections and featureless regions. All the experiments are conducted on a standard PC running Ubuntu with an Intel Core i7-4700 CPU and 8G RAM.

6.10.4.2 Baseline Algorithms

The proposed odometry system will be compared with some state-of-the-art methods, i.e. RGB-D SLAM2 [127], ElasticFusion [116], and ORB-SLAM2 (RGB-D) [121, 159]. Their performance is obtained using the same platform mentioned in Section 6.10.4.1 based on their open source implementation.

Although these methods do not include inertial fusion, they are still excellent baseline and have been extensively tested by the robotic community. The implementation of RGB-D-inertial odometry method, e.g. [130] is unavailable, hence we are unable to compare. The traditional tightly-coupled or loosely-coupled inertial fusion methods typically introduce additional complexity due to the additional filtering or optimization process for the large number of inertial measurements. However, we will show that the proposed non-iterative inertial fusion method reduces the complexity of pure visual system dramatically. For accuracy, it will

³https://wangchen.online/myahrs_driver

be shown that the proposed method doesn't reduce the performance of pure visual methods and even performs better in some scenarios that are challenging for traditional methods.

Since ElasticFusion has no ROS interface, a simple ROS wrapper is implemented to read images through ROS communication protocol, while the runtime measurement will not include the communication time. Some of the above methods optimize the whole trajectory in the back-end, and output the optimized trajectory only after the program is terminated. This is not suitable for low-latency robotic systems. To evaluate and compare real-time performance, we only accept the instant pose estimation.

6.10.4.3 Accuracy Comparison

To demonstrate the effectiveness of the proposed framework, especially the front-end translation estimation, the accuracy is evaluated in terms of ATE RMSE and absolute mean error (MEAN). The comparison is reported in Table 6.2 and Table 6.3. It can be seen that the proposed method achieves similar accuracy with all the other methods. We notice that the accuracy of feature-based methods, ORB-SLAM2 and RGB-D SLAM2 vary a lot in different datasets, this may be due to the existence of specular reflection and feature-less regions. This phenomenon indicates that the feature-based methods are sensitive to feature outliers. In contrast, ElasticFusion performs similarly for different datasets. The results of some sequences that are obviously wrong are not reported, this is because we find those methods fail to give a well optimization due to feature outliers.

It is noticed that the accuracy of ORB-SLAM2 (RGB-D) is higher when true visual local loop closure is available frequently, while our method works better in feature-less and pseudo-feature environments. One of the reasons may be that the KCC used in our method is more robust to object distortion and occlusion. In this sense, the proposed non-iterative framework is a good alternative solution to the front-end of RGB-D-inertial SLAM system.

FIGURE 6.10: The trajectory estimation an ultra-low power CPU Atom x5-Z8350 with Scenario Design Power (SDP) of 2W.

FIGURE 6.11: A dense mapping of office cubicles. It is constructed in real-time on the ultra-low power CPU Atom x5-Z8350. The map is sent to a remote server in real-time only for visualization purpose (no post-processing).

6.10.4.4 Efficiency Comparison

The performance in terms of efficiency is evaluated in terms of update rate (runtime), which is reported in Table 6.4. The proposed framework outperforms all the other methods. Note that the reported runtime of our method is the summation of both tracking and mapping process; while that of the other methods only contains the tracking time. This is because the tracking thread is independent of the mapping in other methods, while they are processed sequentially in ours. The runtime of our method varies for different sequences according to the number of trainings. If landmarks change rapidly, our method has to train new models, resulting in a little bit increasing of runtime.

TABLE 6.5: Average runtime (ms) on the ultra low power platform.†

Processor	Method	Tracking	Mapping	Update
x5-Z8350	ours	26.9	5.4	32.3
	ORB2	167	499	≥ 167

†All tests are using the RGB-D image with size 640×480 . The mapping process of ORB-SLAM2 is parallel to tracking.

Note that we don't report the efficiency of ElasticFusion in Table 6.4, since it provides only GPU implementation, while the other methods are running on pure CPU. The platform mentioned in Section 6.10.4.1 contains a NVIDIA Quadro 2000 GPU, in which the average update rate for ElasticFusion is about 5.1 Hz. We believe that ElasticFusion is able to run faster if more powerful GPU is provided, while the efficiency comparison is not meaningful.

6.10.5 Performance on Micro-device

6.10.5.1 Platform

The real-time performance on low power device is crucial for micro-robot systems. As shown before, the proposed system requires the least computational resources, thus we are able to test it on micro-devices. We choose the Intel robotic development kit, that includes RealSense R200 RGB-D camera and a credit card sized computational board equipped with an ultra-low power processor Atom x5-Z8350. We also integrate it with an inertial sensor myAHRS+ which is mentioned in Section 6.10.4.1. Running at 1.44GHz with 2G RAM, this platform is very difficult for most of the state-of-the-art dense mapping algorithms to run in real-time (30Hz).

6.10.5.2 Performance

Limited by the on-board memory, we cannot evaluate the performance using benchmarks. Instead, we will test the real-time performance in terms of average runtime, trajectory estimation, and dense mapping.

- (i) We list the average tracking and mapping runtime in Table 6.5, where the performance of ORB-SLAM2 (RBG-D) is also given for comparison. It can be seen that our system requires much less runtime compared with ORB-SLAM2. The tracking and mapping process of ORB-SLAM is parallel to each other and cannot be simply summed up.
- (ii) Some examples of the overhead trajectory estimation is shown in Figure 6.10, where the ground truth is from the Vicon system. The estimated poses are associated with the truth values by the red lines according to the ROS timestamp. We cannot report the accuracy comparison with ORB-SLAM2, since it cannot work in real-time (a lot of dropped frames).
- (iii) An example for real-time dense mapping of our working area (cubicles) is shown in Figure 6.11. The map is constructed and fused real-time by the on-board processor. Meanwhile, the key-frames (orthogonal images) are sent to a remote server for visualization without post-processing. Some black holes can be seen inside the map, the reasons could be that the IR sensors seem to have no response to some specific dark material, resulting in no depth information. Since the key-frame fusion algorithm in (6.19) requires the complementary information from non-key-frames, it cannot work in such situation.

6.11 Conclusion

In this chapter, a novel framework for the RGB-D-inertial SLAM system is proposed. Combining with non-iterative front-end and a lightweight iterative back-end, our system dramatically reduces the computational complexity. It was demonstrated that the odometry system can be accelerated based on an on-line training process. To the best of our knowledge, the proposed framework may be the first non-iterative solution to the front-end of a dense mapping system. We also demonstrated that it is able to achieve much faster speed and comparable accuracy with the state-of-the-art methods.

Chapter 7

UWB Aided NI-SLAM

The back-end of the non-iterative RGB-D-inertial SLAM system presented in Chapter 6 still leverages on the feature detection and matching, which is sometime sensitive to the environments. In this chapter, an alternative solution to the back-end using ultra-wide band (UWB) communication technology is proposed.

7.1 Introduction

As pointed out in Chapter 6, the researchers have been paying increasing attentions on the real-time performance of VSLAM, especially on low power devices, such as the micro unmanned aerial vehicles (MUAV). The non-iterative solution to the front-end of RGB-D-inertial SLAM system in Chapter 6 has push the front line forward in research of the odometry system. However, the back-end of the SLAM system still leverages on the matching of visual features, which are sometimes sensitive to environments. Moreover, the map produced from the traditional methods, e.g., [121, 143], only consists of sparse points instead of a full depth map. Other existing methods capable of producing a real-time dense map are computationally heavy [113, 127] and are not suitable for devices with limited computational resources. Therefore, we believe that an alternative solution to loop closure is needed for the NI-SLAM system, as well as all the other traditional vision-based methods.

So far most of the works on loop closure detection are still based on visual techniques which require additional computational resources or pretrained data [166,

167]. Other works, such as [168] that combines VO and laser range finder, still rely on the geometry information of the environments and only reduce instead of removing visual drifts. To solve this problem, the ultra-wideband (UWB) communication technology is investigated to bypass the complexity of visual loop closure detection. Another motivation for fusing with UWB is to overcome the shortcomings suffered by pure visual methods. Recall the Figure 6.9, the scenarios with reflective or featureless areas will obviously raise issues for visual techniques, but have much less impact on the UWB communication.

UWB is robust to multipath and non-line-of-sight (NLOS) effects, thus it can measure the distance by time-of-flight between the UWB modules. In this sense, it is a localization technology by its own, and can achieve a cm-to-dm accuracy when fused with other sensors [169, 170]. Based on the extended Kalman filter (EKF), [171] proposes a robot localization system using one-way communication with fixed-position UWB anchors. The tracking system proposed in [172] only requires instrumentation of the target with a single UWB transceiver. However, these works on UWB mostly rely on EKF, which is sensitive to measurement outliers, especially for highly non-linear systems.

To solve the aforementioned problems, a graph-optimization-based UWB aided SLAM framework is proposed for global drift-free trajectory estimation. First, we propose to add the smooth and range constraints for the UWB localization system, which forms a drift-free but coarse position estimation. Then the incremental pose estimation from the non-iterative odometry (NIO) in Chapter 6 is introduced into the graph, so that dense maps can be registered in real-time without accumulated error, resulting in a super lightweight dense mapping system. The front-end NIO is substituted by other SLAM methods, which demonstrates the flexibility of the proposed framework. The experiments show it that the proposed framework is able to improve the accuracy and robustness of a vision-based SLAM system significantly, yet still with feasible complexity. In summary, the main contributions are

- a pose graph based framework is proposed for UWB aided dense mapping system;
- robust objective functions for dense SLAM are designed to reduce the drift;
- real-time dense mapping is demonstrated on an ultra-low power processor even in featureless environments.

FIGURE 7.1: The proposed framework of the UWB aided dense mapping system consists of three parallel threads. The UWB localization thread is to provide global drift-free loop closure. The thread of RGB-D-inertial odometry is served for local accurate trajectory estimation. The key-frames from the odometry thread together with the loop closure from UWB thread are sent to the mapping thread where an optimization is performed to generate live dense drift-free maps.

7.2 System Architecture

The system proposed in this chapter consists of three parallel threads, namely UWB localization, odometry, and map optimization. The UWB localization thread serves to provide drift-free global position estimation. The odometry thread is to create local trajectory estimation and send the key-frames to mapping thread for optimization. Given the global drift-free coarse constraints from UWB and incremental pose constraints from the odometry, the mapping thread directly integrates them into the pose graph where an optimization is performed to create live dense drift-free maps. The proposed system architecture is illustrated in Figure 7.1, where each block will be introduced in the Section 7.2.1, 7.2.2, and 7.2.3, respectively.

7.2.1 UWB Localization

7.2.1.1 Cost Function

By applying a peer-to-peer two-way time-of-flight measurement [169], the robot at position $\mathbf{p}_k \in \mathbb{R}^3$ is able to range to the i^{th} UWB anchor at position $\mathbf{p}_i^{\text{uwb}} \in \mathbb{R}^3$, so that distance $d_i \in \mathbb{R}$ can be obtained by the multiplication of light speed c and

the measurement of time of flight.

$$d_i = c \frac{Q_{M_1}^{Rx} - Q_{M_0}^{Tx} - \sigma_i}{2} + \eta_i, \quad (7.1a)$$

$$= \|\mathbf{p}_i^{\text{uwb}} - \mathbf{p}_k\| + \eta_i, \quad (7.1b)$$

where $Q_{M_1}^{Rx}$ and $Q_{M_0}^{Tx}$ are the time stamps when the UWB ranging radio is sent and received relative to the robot's clock respectively and σ_i is the processing time delay by the i^{th} anchor. It is assumed that $\eta_i \sim \mathcal{N}(0, \Omega_i)$ is a zero mean Gaussian noise. The proposed scheme shown in Figure 7.2 is based on continuous optimization of the range error over a sliding window of recent poses, taking into account a constraint motion model for smoothing the estimated trajectory. The range error E_r of a trajectory of points is defined as the sum of weighted residue r_k^r ,

$$E_k^r(\mathbf{p}) := \sum_{i,k} \omega_{i,k}^r \cdot \rho(\underbrace{\|\mathbf{p}_i^{\text{uwb}} - \mathbf{p}_k\| - d_i^k}_{r_k^r}), \quad (7.2)$$

where k is the time index associated with the optimizable poses $\mathbf{p}_k$ in a sliding window and $\rho(\cdot)$ is the Pseudo-Huber loss function defined as $\rho(e) = \delta^2(\sqrt{1 + (e/\delta)^2} - 1)$ that is designed to approximate quadratic function $e^2/2$ for small values of e and linear function δe for large values of e . This ensures that the derivatives are continuous for all degrees. In addition, a measurement covariance based penalty term $\omega_{i,k}^r$ is given by

$$\omega_{i,k}^r = \frac{\gamma^2}{\|\Omega_i^k\|^2 + \gamma^2}, \quad (7.3)$$

which down-weights the measurements with high covariance and γ is a free-parameter. The range error (7.2) only gives geometric constraints on the estimated trajectory according to the range measurements, but fails to form a smooth trajectory. To overcome this problem, smooth error E_s is defined among consecutive poses:

$$E_k^s(\mathbf{p}) := \sum_k \omega_k^s \cdot \rho(\underbrace{\|\mathbf{p}_{k+1} - \mathbf{p}_k\|}_{r_k^s}), \quad (7.4)$$

where $\rho(\cdot)$ is the Pseudo-Huber norm. Assume the maximum linear velocity of robot is $\mathbf{v}_{max} \in \mathbb{R}^3$, the penalty term $\omega_{i,k}^s$ is defined as:

$$\omega_k^s = \frac{9\gamma^2}{\|\mathbf{v}_{max}\|^2 (T_{k+1} - T_k)^2 + 9\gamma^2}, \quad (7.5)$$

FIGURE 7.2: The proposed optimization scheme for UWB ranging localization. The pose graph is updated by inserting a new and removing the oldest pose in a sliding window, when a new ranging measurement is acquired. The trajectory in the window is optimized based on the range and smooth constraints. Note that only the latest optimized pose in the window is reported to detect the odometry drift, resulting in a real-time and dynamic system.

where T_k is the time stamp of pose x_k . It can be interpreted that the probabilistic distance between two consecutive poses is under the $3\text{-}\sigma$ rule and the robot's velocity follows a normal distribution. Combining the range and smooth constraint (7.2), (7.4), a well-defined trajectory $\mathbf{p}^*$ can be obtained through the minimization

$$\mathbf{p}^* = \arg \min_{\mathbf{p} \in \mathcal{N}_p} \underbrace{E_k^r(\mathbf{p}_k) + E_k^s(\mathbf{p}_k)}_{E_k(\mathbf{p})}, \quad (7.6)$$

where $\mathcal{N}_p$ is the set of positions in the sliding window.

7.2.1.2 Optimization

A numerical solution of (7.6) can be obtained by using the gradient descent algorithm. The residual is computed at the current stacked state $\hat{\mathbf{p}}$ by first order Taylor expansion, i.e.

$$r_k(\hat{\mathbf{p}} + \Delta\mathbf{p}) = r_k^r(\hat{\mathbf{p}} + \Delta\mathbf{p}) + r_k^s(\hat{\mathbf{p}} + \Delta\mathbf{p}), \quad (7.7a)$$

$$= r_k(\hat{\mathbf{p}}) + \mathbf{J}_k \Delta\mathbf{p}, \quad (7.7b)$$

where $\mathbf{J}_k$ is the Jacobian of $r_k(\mathbf{p})$ at $\hat{\mathbf{p}}$ which is the sum of the range Jacobian $\mathbf{J}_k^r$ and smooth Jacobian $\mathbf{J}_k^s$.

$$\mathbf{J}_k^r = \frac{\partial \|\mathbf{p}_i^{\text{uwb}} - \mathbf{p}_k\|}{\partial \mathbf{p}}, \quad \mathbf{J}_k^s = \frac{\partial \|\mathbf{p}_{k+1} - \mathbf{p}_k\|}{\partial \mathbf{p}}. \quad (7.8)$$

For the sake of computational efficiency, a technique used in [173] is to find a re-weighted residual. The re-weighted smooth residual is shown in (7.9) and a similar expression can also be found for re-weighted range residual.

$$(\omega_k^{s*} r_k^s)^T \omega_k^s (\omega_k^{s*} r_k^s) = \rho \left(\sqrt{r_k^{sT} \cdot \omega_k^s \cdot r_k^s} \right), \quad (7.9)$$

where

$$\omega_k^{s*} = \frac{\sqrt{\rho(\|r_k^s\|_{\omega_k^s})}}{\|r_k^s\|_{\omega_k^s}} \quad \text{with} \quad \|r_k^s\|_{\omega_k^s} := \sqrt{r_k^{sT} \cdot \omega_k^s \cdot r_k^s}. \quad (7.10)$$

For simplicity of notation, the range and smooth errors are rewritten as combined residual r_k^* relating to position $\mathbf{p}_k$. Therefore, the combined summation error $E(\mathbf{p})$ can be rewritten as,

$$E^*(\mathbf{p}) := \sum_k r_k^*(\mathbf{p}_k)^T \Omega_k^* r_k^*(\mathbf{p}_k), \quad (7.11)$$

where Ω_k^* is the combined weights associated with the position $\mathbf{p}_k$. Starting from a good initial value $\hat{\mathbf{p}}$, the combined range-smooth error (7.11) can be calculated when a small disturbance $\Delta \mathbf{p}$ is applied.

$$E_k(\hat{\mathbf{p}} + \Delta \mathbf{p}) = \sum_k r_k(\hat{\mathbf{p}} + \Delta \mathbf{p})^T \Omega_k r_k(\hat{\mathbf{p}} + \Delta \mathbf{p}). \quad (7.12)$$

Substitute (7.7b) into (7.12) and denote $r_k(\hat{\mathbf{p}})$ as $\hat{r}_k$:

$$E_k(\hat{\mathbf{p}} + \Delta \mathbf{p}) = \sum_k \hat{r}_k^T \Omega_k \hat{r}_k + 2 \underbrace{\hat{r}_k^T \Omega_k \mathbf{J}_k}_{\mathbf{b}_k} \Delta \mathbf{p} + \Delta \mathbf{p}^T \underbrace{\mathbf{J}_k^T \Omega_k \mathbf{J}_k}_{\mathbf{H}_k} \Delta \mathbf{p}. \quad (7.13)$$

The trajectory $\mathbf{p}$ is updated by $\mathbf{p}_{(n+1)} = \mathbf{p}_{(n)} + \Delta \mathbf{p}$ in each iteration and $\Delta \mathbf{p}$ can be found via Levenberg-Marquardt algorithm [161].

$$(\mathbf{H} + \lambda \mathbf{I}) \Delta \mathbf{p}^* = -\mathbf{b}, \quad (7.14)$$

where $\mathbf{b} = \sum b_k$, $\mathbf{H} = \sum \mathbf{H}_k$, $\mathbf{H} \in (\mathbb{R}^{3 \times 3})^{n \times n}$, and $\lambda > 0$.

When a new ultra-wideband radio measurement is received, the associated pose will be added to the trajectory and the two kinds of constraints will be imposed to (7.6). Since pose $\mathbf{p}_k$ in the trajectory will only be restricted by one range error and two smooth errors, the $\mathbf{H}$ matrix is sparse and the number of non-zero blocks in $\mathbf{H}$ is $3n - 1$, where n is the number of poses. This allows to solve (7.14) efficiently with sparse Cholesky factorization [174].

7.2.2 RGB-D-Inertial Odometry

In this part, we employ the NI-SLAM introduced in Chapter 6 directly. To fuse the UWB and NI-SLAM, we need to impose the tracking constraint with estimation covariance of the odometry system.

Compared to UWB localization, the odometry system is able to provide 3-D incremental pose estimation, which has three additional rotational DoF in the non-Euclidean space where optimization is difficult. A proper solution is to denote the transforms on a manifold, so that they can be easily optimized in Lie algebra space. Denote the poses in the trajectory as $\mathbf{z}_k \in \mathbf{SE}(3)$ and the transformation from $\mathbf{z}_i$ to $\mathbf{z}_j$ as $\mathbf{z}_{ij} \in \mathbf{SE}(3)$, their associated representation in Lie algebra are 6-D vectors $\xi_k, \xi_{ij} \in \mathfrak{se}(3)$, for which $\xi = [\mathbf{p}^T, \phi^T]^T$, where $\mathbf{p}$ is the translational part and ϕ is a minimum presentation of rotation. Define the right operator $\oplus$ which applies the incremental transformation ξ_{ij} on the Manifold space, i.e. $\xi_j = \xi_i \oplus \xi_{ij}$. Hence, the exponential mapping from $\mathfrak{se}(3)$ to $\mathbf{SE}(3)$ is $\mathbf{x}_i = \exp_{\mathbf{SE}(3)} \xi_i^\wedge$. Assume the pose measurement from the odometry thread is denoted as $\mathbf{z}_{ij} \in \mathbf{SE}(3)$, we have

$$\mathbf{z}_{ij} = \exp_{\mathbf{SE}(3)}(\xi_{ij} \oplus \Delta\xi)^\wedge, \quad (7.15)$$

where $\Delta\xi$ is a small disturbance around ξ_{ij} and $\Delta\xi \sim \mathcal{N}(\mathbf{0}, \Omega_{ij})$ with $\Omega_{ij} \in \mathbb{R}^{6 \times 6}$. Since the 3-D translational movement is decoupled from the rotational ones and it is assumed that the movements are independent, then $\Omega_{ij} = \text{diag}(\sigma_{\mathbf{t}_x}^2, \sigma_{\mathbf{t}_y}^2, \sigma_{\mathbf{t}_z}^2, \Omega_\phi)$ where $\sigma_{\mathbf{t}_x}^2, \sigma_{\mathbf{t}_y}^2, \sigma_{\mathbf{t}_z}^2$ denote the translational covariance, respectively, $\Omega_\phi \in \mathbb{R}^{3 \times 3}$ is the rotational part and can be acquired from the inertial sensor. Recall that we have derived the estimation covariance in (6.15) and (6.17), which is written as

$$\sigma_{\mathbf{t}_x}^2 = \sigma_{\mathbf{t}_y}^2 = \frac{r_k^2 \cdot \sum \mathbf{g}(\mathbf{x})}{2\pi \cdot \max \mathbf{g}(\mathbf{x})}, \quad (7.16)$$

$$\sigma_{\mathbf{t}_z}^2 = \text{Var}_{(i,j) \in \mathcal{W}} (\mathcal{S}_{(-[\tilde{x}, \tilde{y}])}(\mathbf{x}_{[i,j]}^{\mathcal{D}}) - \mathbf{z}_{[i,j]}^{\mathcal{D}}), \quad (7.17)$$

where $\mathbf{g}(\mathbf{x})$ and $\mathcal{S}_{(-[\tilde{x}, \tilde{y}])}(\cdot)$ are the correlation output and image translation operator, respectively.

7.2.3 UWB Aided Odometry

The minimization (7.6) only outputs 3-D position $\mathbf{p} \in \mathbb{R}^3$, while the visual-inertial odometry system is able to provide pose constraints in Euclidean space $\mathbf{SE}(3)$. Assume the poses in the windowed trajectory is denoted as $\xi = [\xi_0^T, \xi_1^T, \dots]^T$, so that the pose tracking constraint from the odometry system is defined as:

$$E^t(\xi) := \sum_{i,j} \omega_{ij}^t \cdot \rho(\underbrace{\|\ln_{\mathfrak{se}(3)}(\mathbf{z}_{ij}^{-1} \cdot \mathbf{x}_{ij})^\vee\|}_{r_{ij}^t}), \quad (7.18)$$

where $^\vee$ is the inverse operator of $^\wedge$. The weights ω_{ij}^t is defined as:

$$\omega_{ij}^t = \frac{\gamma^2}{\|\Omega_{ij}\|^2 + \gamma^2}. \quad (7.19)$$

The inevitable drift in (7.18) can be corrected by the range constraints from UWB ranging measurements. To this end, we propose to fuse the constraints based on trajectory optimization with either of the following objective functions.

7.2.3.1 Cost function 1

The most straightforward way is to apply the range constraint (7.2) and smooth constraint (7.4) together with the tracking constraints (7.18) directly.

$$\xi^* = \arg \min E^t(\xi) + E^r(\mathbf{p}) + E^s(\mathbf{p}). \quad (7.20)$$

To save computational resources, the constraints (7.2) and (7.4) are applied only when a loop closure is needed. Figure 7.3 shows the basic structure of this method.

FIGURE 7.3: The UWB aided mapping framework using the objective (7.20). The range-smooth constraints are connected to the nearest tracking constraint.

FIGURE 7.4: The UWB aided mapping framework using the objective (7.23). The virtual range constraints r_i^{vr} are connected to tracking constraints directly.

7.2.3.2 Cost function 2

Smooth error (7.4) provides a constraint on two consecutive poses associated with two range measurements. However, when other measurements are available between the two range constraints, it is not necessary to apply this smooth constraint.

Alternatively, we apply a virtual range constraint (7.21) instead of the separated constraints (7.2) and (7.4), which is shown in Figure 7.4. In this case, we only need

to optimize the poses associated with the tracking measurements, so that a lot of calculations can be avoided.

$$E^{vr}(\mathbf{p}) := \sum_{i,k} \omega_{i,k}^{vr} \cdot \rho(\|\mathbf{p}_i^{\text{uwb}} - \mathbf{p}'_k\| - d_i^k). \quad (7.21)$$

Different from (7.2), the virtual range error (7.21) applies the range constraint on position $\mathbf{p}'_k$ which is the nearest neighbour of $\mathbf{p}_k$ associated with the tracking measurement. The weights $\omega_{i,k}^{vr}$ is defined in (7.22), where $\Delta T'_k$ is the time interval between the nearest ranging and tracking measurements.

$$\omega_{i,k}^{vr} = \frac{9\gamma^2}{\|\mathbf{v}_{max}\|^2 (\Delta T'_k)^2 + 9\|\Omega_i^k\|^2 + 9\gamma^2}. \quad (7.22)$$

By combining the tracking error (7.18) with virtual range error (7.21), the optimized trajectory ξ^* can be obtained,

$$\xi^* = \arg \min E^t(\xi) + E^{vr}(\mathbf{p}). \quad (7.23)$$

7.2.3.3 Optimization

Similar to (7.12), the optimal solution ξ^* of (7.20) and (7.23) can be found from a re-weighted residual. Starting from a initial guess $\hat{\xi}$,

$$E(\hat{\xi} \oplus \Delta\xi) = \sum_k r_k(\hat{\xi} \oplus \Delta\xi)^T \Omega_\xi r_k(\hat{\xi} \oplus \Delta\xi), \quad (7.24)$$

$$= E(\hat{\xi}) + J(\hat{\xi})\Delta\xi, \quad (7.25)$$

where r_k is the re-weighted residual of (7.21). The trajectory ξ is updated by $\xi_{(n+1)} = \xi_{(n)} \oplus \Delta\xi_{(n)}$ in each iteration and $\Delta\xi_{(n)}$ can be found by solving

$$(\mathbf{J}^T \Omega_\xi \mathbf{J} + \lambda \mathbf{I}) \Delta\xi_{(n)} = -r(\xi_{(n)})^T \Omega_\xi \mathbf{J}, \quad (7.26)$$

where $\mathbf{J}$ is the derivative of the stacked residual vector $\mathbf{r} = [\dots, r_i, \dots]^T$ and

$$\mathbf{J} = \left. \frac{\partial \mathbf{r}(\xi_{(n)} \oplus \Delta\xi)}{\partial \Delta\xi} \right|_{\Delta\xi=\mathbf{0}}. \quad (7.27)$$

7.3 Experiments

In this section we present extensive experimental outcomes to demonstrate the superior performance of the proposed system. At first we evaluate the proposed UWB aided localization and mapping system in terms of average accuracy, dense mapping improvement, as well as the efficiency on both standard laptop and ultra-low power computing platforms. In addition, to benchmark the integration of UWB with NI-SLAM against other SLAM methods in the framework, we replace NI-SLAM with another state-of-the-art method, ORB-SLAM2 (RGB-D version) [159] and compare it with the original performance. The experimental results demonstrate the competitiveness of the proposed system as well as the versatile nature of the fusion framework.

7.3.1 Implementation

The dense mapping procedure is implemented following Section 6.8, which is quite fast even in low power processors. The experiment datasets were recorded by a hand-held Kinect RGB-D camera in an indoor environment. The data include color and depth images captured at 30Hz, UWB ranging measurements at 40Hz, IMU measurements at 100Hz, and 6-D ground truth obtained from a Vicon motion capture system. In order to cover distinct situations, we collected 17 datasets with different travelling distances, speeds, dynamics, and illumination conditions in a 6m×6m area equipped with a Vicon system. The UWB signals are detected sequentially, thus any number of UWB modules can be deployed in theory, but four UWB anchors with known positions are used in our experiments.

It should be highlighted that the datasets are quite challenging for pure vision-based approaches because in the scene there are large black and white regions which are featureless (Figure 6.9). In addition, the several pieces of glass shown in Figure 7.7 produce a lot of *pseudo landmarks* which do not follow perspective principals. This makes the traditional feature-based methods easy to lose tracking.

To test the performance on micro-robots with ultra-low power processors and compare with the state-of-the-art algorithms, we will use two different platforms. One is a credit-card sized mobile-phone-level UPboard[®] with an ultra-low power processor Atom x5-Z8350 whose scenario design power is only 2W. Running at 1.44

TABLE 7.1: The improved performance of the UWB aided NI-SLAM (NI-SLAM + UWB) system in terms of ATE RMSE and MAE (Unit: m)

Dataset	NI-SLAM+UWB		NI-SLAM	
	RMSE	MAE	RMSE	MAE
01_circle	0.060	0.054	0.150	0.140
02_rectangle	0.062	0.059	0.112	0.108
03_person1	0.045	0.035	0.070	0.065
04_person2	0.057	0.048	0.124	0.109
05_person3	0.054	0.048	0.120	0.108
06_person4	0.050	0.045	0.138	0.129
07_double_circle1	0.060	0.056	0.148	0.128
08_double_circle2	0.103	0.089	0.337	0.305
09_double_rectangle1	0.070	0.059	0.225	0.216
10_double_rectangle2	0.059	0.053	0.228	0.197
11_double_rectangle3	0.071	0.064	0.209	0.165
12_double_semicircle	0.115	0.103	0.487	0.444
13_long_translation	0.059	0.056	0.068	0.065
14_infinite-shape	0.094	0.090	0.314	0.262
15_shake	0.032	0.022	0.021	0.018
16_fast_rotation1	0.088	0.069	0.321	0.321
17_fast_rotation2	0.090	0.078	0.414	0.385
Mean	0.069	0.060	0.205	0.186

GHz with 2GB RAM, this platform is very difficult for most of the state-of-the-art algorithms to run in real-time. Therefore, for comparison and visualization purpose, we also test the proposed framework on a standard laptop running Ubuntu 16.04 with 8G RAM and an Intel Core i7-6700HQ CPU at 2.6 GHz. Limited by the payloads and power consumption, we choose Intel RealSense[®] camera to work with the lower power platform, and Microsoft Kinect[®] to work with the standard laptop. The IMU used in the experiments is myAHRS+[®], which is a low cost high performance attitude and heading reference system (AHRS) containing 3-axis 16-bit gyroscope, a 3-axis 16-bit accelerometer, and a 3-axis 13-bit magnetometer. To obtain the best performance, NI-SLAM [107] runs with axonometric image size 360×480 for laptop and 240×360 for Up-Board[®].

7.3.2 Accuracy Evaluation

An extensive experimental validation is performed in terms of absolute trajectory error (ATE) through root mean squared error (RMSE) and median absolute error

FIGURE 7.5: Some examples of trajectory estimation w/wo UWB aids. The estimation with UWB aids is obviously much closer to the ground truth.

TABLE 7.2: Average runtime of the UWB aided NI-SLAM (NI+UWB) system.

Platform	Method	Tracking	Mapping	Total
Laptop	NI+UWB	6.7ms	9.1ms	15.8ms
	NI-SLAM	6.5ms	9.2ms	15.7ms
Up-Board [®]	NI+UWB	27ms	5ms	32ms
	NI-SLAM	27ms	5ms	32ms

(MAE) over the entire trajectory. As shown in Table 7.1, the proposed UWB aided system surpasses the pure vision-based approach. Figure 7.5 illustrates the plot of overhead 2-D trajectory from the No. 07 and No. 14 dataset. It can be seen that the trajectory estimation from the UWB aided system is much closer to the ground truth and no drift is presented. Because of the UWB measurement errors, it is noticed that the results without UWB aids in the No. 15 *shake* dataset performs a little better. The explanation is that the UWB constraints are better at eliminating drift in long travel rather than small tracking errors in tiny back-and-forth movements during rapid shaking. To avoid the problem, one of the solutions is to adapt the UWB aiding strategy by inserting UWB constraints only after having traveled a certain distance.

7.3.3 Efficiency Evaluation

We evaluate the efficiency of the proposed system on both tracking and mapping over all datasets. The average running time on two platforms is given separately

in Table 7.2. Note that the UWB process is actually independent of the SLAM thread, hence with the aid of UWB measurement, the time usage nearly does not change because of the concurrent multiple threads. Therefore, the proposed UWB aided approach does not corrupt the instantaneity of the original method. We also note that the runtime varies for different datasets. This is because NI-SLAM has to create new key-frames and train new models frequently, if the camera moves rapidly, thus increasing a little bit of running time.

7.3.4 Dense Mapping

In this section, the performance of dense mapping is presented qualitatively. All the maps are created online and displayed without any post-processing. Note that different from the existing dense reconstruction methods [127, 137, 138], our system does not need any GPU to process large number of mapping data. In Figure 7.6, the dense map of the testing area is presented. It is obvious that the point clouds align pretty well and the 3-D map depicts the environment faithfully, although there are lots of pseudo features and large featureless area in this site.

7.3.5 System Generalization

To show the generality and robustness, performance on other SLAM methods is demonstrated in this section. One of the most popular visual SLAM methods, ORB-SLAM [159] is substituted into the proposed UWB aided system. Note that NI-SLAM is designed as a real-time visual odometry method while ORB-SLAM works as a full SLAM system with local mapping adjustment and loop-closure detection. Hence, the finalized motion estimation can only be obtained after the program is terminated. To establish a low-latency system, the UWB aided system only accepts the instant pose estimation. While the loop closure detection of ORB-SLAM is still enabled to make sure the previous estimation can be corrected when a new visual loop closure is detected.

The example of feature extraction from ORB-SLAM is illustrated in Figure 7.7. This shows the great challenge for ORB-SLAM since the mirror reflections taken as features actually provide highly misleading information, thus reduce localization accuracy. However, we find that the proposed framework of UWB aids successfully

FIGURE 7.6: The live dense mapping for the IoT (Internet of Things) lab in NTU. The first two figures show the maps from two points of view that is generated in real-time, under which the panoramic view is presented for comparison.

constraint ORB-SLAM estimates from deviating to far, according to the data in Table 7.3. It is highlighted that NI-SLAM and ORB-SLAM are based on different front-end systems, one using on-line learning tracking and the other using ORB feature tracking. However, regardless of the difference in structure, the proposed approach can always work to provide global correction for long term errors. These experiments demonstrate the effectiveness and competitiveness of the proposed

FIGURE 7.7: The features that are extracted by ORB-SLAM. The specular reflections obviously raise issues for pure visual methods.

TABLE 7.3: The improved performance for UWB aided ORB-SLAM in terms of efficiency and accuracy.

Dataset	ORB-SLAM+UWB		ORB-SLAM	
	RMSE	Runtime	RMSE	Runtime
01_circle	0.065m	26ms	0.257m	26ms
02_rectangle	0.091m	27ms	0.147m	27ms
03_person1	0.054m	24ms	0.088m	24ms
04_person2	0.058m	48ms	0.119m	48ms
05_door	0.112m	29ms	0.432m	29ms
Mean	0.076m	33ms	0.209m	33ms

framework for localization and dense mapping.

7.4 Conclusion

An UWB aided localization and mapping system is proposed in this chapter. By taking advantage of different technologies, multiple capabilities are achieved simultaneously. First, accurate and drift-free localization are enabled through a novel

graph optimization technique incorporating multiple constraints from different sensors. Second, the proposed approach greatly reduces the complexity, so that dense maps are able to be generated in real-time even for featureless environments, while only running on an ultra-low power processor. Finally, the framework is demonstrated to be versatile, robust, and can be easily adapted for other methods.

Chapter 8

Kervolutional Neural Networks

The theories and applications of KCC, that is constructed based on cross-correlator, have been developed in Chapter 3 to Chapter 5 and Section 6 to Section 7, respectively. In this part, the similar operator of cross-correlator, convolution introduced in Section 2.2.1 will also be generalized. Since convolution has been applied to a large number of applications, we expect the extended operation will also play important roles in those fields.

8.1 Introduction

Convolutional neural networks (CNNs) have been tremendously successful in computer vision, e.g. image recognition [8, 15] and object detection [175, 176]. The core operator, convolution, was partially inspired by the animal visual cortex where different neurons respond to stimuli in a restricted and partially overlapped region known as the receptive field [177, 178]. Convolution leverages on its equivariance to translation to improve the performance of a machine learning system [20]. Its efficiency lies in that the learnable parameters are sparse and shared across the entire input (receptive field). Nonetheless, convolution still has certain limitations, which will be analyzed below. To address them, this chapter introduces kervolution to generalize convolution via the kernel trick. The artificial neural networks containing kervolutional layers are named as kervolutional neural networks (KNN).

There is circumstantial evidence that suggests most cells inside striate cortex¹ can be categorized as simple, complex, and hypercomplex, with specific response properties [178]. However, the convolutional layers are linear and designed to mimic the behavior of simple cells in human visual cortex [179]. Therefore, they are not able to express the non-linear behaviors of the complex and hypercomplex cells inside the striate cortex. It has also been demonstrated that higher order non-linear feature maps are able to make subsequent linear classifiers more discriminative [180–182]. However, the non-linearity of existing artificial neural networks mostly comes from the activation layers, e.g. sigmoid and rectified linear unit (ReLU), which can only provide point-wise non-linearity. Inspired by this, we argue that CNN may perform better if convolution can be generalized to patch-wise non-linear operations via kernel trick. Because of the increased expressibility and model capacity, we may be able to obtain better model generalization ability.

Non-linear generalization is simple in mathematics, however, it is generally difficult to retain the advantages of convolution, i.e. (i) sharing weights (weight sparsity) and (ii) low computational complexity. There exists several works towards non-linear generalization. The non-linear convolutional networks [179] implement a quadratic convolution at the expense of additional $n(n+1)/2$ parameters, where n is the size of the receptive field. However, the quadratic form of convolution loses the property of "weight sparsity", since the number of additional parameters of the non-linear terms increases exponentially with the polynomial order, which dramatically increases the training complexity. Another strategy to introduce high order features is to explore the pooling layers. The kernel pooling method in [182] directly concatenates the non-linear terms. However, this requires explicit calculation of the non-linear terms, also resulting in a much higher computational complexity.

To address the above problems, kervolution is introduced in this chapter to extend convolution to kernel space while keeping the aforementioned advantages of linear convolutions. Since convolution has been applied to many fields, e.g. image and signal processing, we expect kervolution will also play an important role in those applications. However, in this chapter, we focus on its applications in artificial neural networks. The contributions of this chapter include: (i) via kernel trick, the convolutional operator is generalized to kervolution, which retains the advantages

¹The striate cortex is the part of the visual cortex that is involved in processing visual information.

of convolution and brings new features, including the increased model capacity, translational equivariance, stronger expressibility, and better generalization; (ii) we provide explanations for kervolutional layers from feature view and show that it is a powerful tool for network construction; (iii) it is demonstrated that KNN achieves better accuracy and surpasses the baseline CNN.

8.2 Related Work

As the name indicates, CNN [14] employs convolution as the main operation, which is modeled to mimic the behavior of simple cells found in the *primary visual cortex* known as V1 [177, 178]. It have been tremendously successful in numerous applications [8, 15, 20, 183]. Many strategies have been applied to improve the capability of model generalization.

AlexNet [8] proves that the ensemble method "dropout" is very effective for reducing over-fitting of convolutional networks. The non-saturated rectified linear unit (ReLU) dramatically improves the convergence speed [8] and becomes a standard component of CNN. Network in network (NIN) [184] establishes micro networks for local patches within the receptive field, each of which consists of multiple fully connected layers followed by non-linear activation functions. This improves the model capacity at the expense of additional calculation and complex structures. GoogLeNet [185] increases both depth and width of CNN by introducing the Inception model, which further improves the performance. VGG [186] shows that deep CNN with small convolution filters (3×3) is able to bring about significant improvement.

ResNet [15] addresses the training problem of deeper CNN and proposes to learn the residual functions with reference to the layer input. This strategy makes CNN easier to optimize and improves regression accuracy with an increased depth, enabling great success of many applications, e.g. AlphaGo Zero [17]. DenseNet [187] proposes to connect each layer to every other layer in a feed-forward fashion, which further mitigates the problem of vanishing-gradient. The ResNeXt [188] is constructed by repeating a building block that aggregates a set of transformations with the same topology, resulting in a homogeneous, multi-branch architecture. It

demonstrates the essence of a new dimension, which is the size of the set of transformations. In order to increase the representation power, SENet [189] focuses on channels and adaptively recalibrates channel-wise feature responses by explicitly modeling the interdependencies channels.

In recent years, researchers have been paying much attentions on the extension of convolution. To enable the expressibility of convolution for complex cells, the non-linear convolutional network [179] extends convolution to non-linear space by directly introducing high order terms. However, as indicated before, this introduces a large number additional parameters and increases the training complexity exponentially. To be invariant to spatial transformations of input data, the spatial transformer network [190] inserts learnable modules to CNN for manipulating transformed data. For the same purpose, deformable convolutional network [191] adds 2-D learnable offsets to regular grid sampling locations for standard convolution, which enables the learning of affine transforms; while [192] applies a simple two-parameter image warp before a convolution. CapsNet [193] proposes to replace convolution by representing the instantiation parameters of a specific type of entity as activity vectors via a capsule structure. This opens a new research space for artificial neural networks, although its performance on large dataset is still relatively weak. Decoupled network [194] interprets convolution as the product of norm and cosine angle of the weight and input vectors, resulting in explicit geometrical modeling of the intra-class and extra-class variation. To process graph inputs, Spline-CNN [195] extends convolution by using continuous B-spline basis, which is parametrized by a constant number of trainable control values. To decrease the storage requirement, Modulated CNN [196] extends the convolution operator to binary filters, resulting in easier deployment on computationally limited devices.

The kernel technique in this chapter was applied to create non-linear classifiers in the context of optimal margin [6], which was later recognized as support vector machines (SVM) [7]. While recently it has also been applied to artificial neural networks to improve the model performance. The convolutional kernel network (CKN) [197] proposes to learn transform invariance by kernel approximation, where the kernel is used as a tool for learning CNN. Nevertheless, the aim of CKN is not to extract non-linear features and it is only different from CNN in the cost functions. The SimNets [198] proposes to insert kernel similarity layers under convolutional layer. However, both the similarity templates and filters are needed to be trained

and require a pre-training process for initialization, which dramatically increases the complexity. To capture higher order interactions of features, the kernel pooling [182] is proposed in a parameter-free manner. This is motivated by the aforementioned idea that higher dimensional feature map produced by kernel functions is able to make subsequent linear classifier more discriminative [181]. However, the kernel extension in the pooling stage is not able to extract non-linear features in a patch-wise way. Moreover, the additional higher order features are still need to be calculated explicitly, which also dramatically improves the complexity. To solve these problems, kervolution is defined to generalize convolution via the kernel trick.

8.3 Kervolution

We start from a convolution with output $\mathbf{f}(\mathbf{x})$, i.e.

$$\mathbf{f}(\mathbf{x}) = \mathbf{x} \oplus \mathbf{w}, \quad (8.1)$$

where $\oplus$ is the convolutional operator and $\mathbf{x} \in \mathbb{R}^n$ is a vectorized input and $\mathbf{w} \in \mathbb{R}^n$ is the filter. Specifically, the i_{th} element of the convolution output $\mathbf{f}(\mathbf{x})$ is calculated as:

$$\mathbf{f}_i(\mathbf{x}) = \langle \mathbf{x}_{(i)}, \mathbf{w} \rangle, \quad (8.2)$$

where $\langle \cdot, \cdot \rangle$ is the inner product of two vectors and $\mathbf{x}_{(i)}$ is the circular shift of $\mathbf{x}$ by i elements. For simplicity, the index i always starts from 0 in this chapter.

The kervolution output $\mathbf{g}(\mathbf{x})$ is defined as

$$\mathbf{g}(\mathbf{x}) = \mathbf{x} \otimes \mathbf{w}, \quad (8.3)$$

where $\otimes$ is the kervolutional operator. Specifically, the i_{th} element of $\mathbf{g}(\mathbf{x})$ is defined as:

$$\mathbf{g}_i(\mathbf{x}) = \langle \varphi(\mathbf{x}_{(i)}), \varphi(\mathbf{w}) \rangle, \quad (8.4)$$

where $\varphi(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^d$ ($d \gg n$) is a non-linear mapping function. The definition (8.4) enables us to extract features in a high dimensional space, while its computational complexity is also much higher than (8.2). Fortunately, we are able to bypass the explicit calculation of the high dimensional features $\varphi(\mathbf{x}_{(i)})$ via the kernel trick

[7], since

$$\langle \varphi(\mathbf{x}_{(i)}), \varphi(\mathbf{w}) \rangle = \sum_j c_j (\mathbf{x}_{(i)}^T \mathbf{w})^j = \kappa(\mathbf{x}_{(i)}, \mathbf{w}), \quad (8.5)$$

where $\kappa(\cdot, \cdot) : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ is a kernel function, whose complexity is normally of $\mathcal{O}(n)$ as same as the inner product of convolution. The coefficient c_j can be determined by the mapping function $\varphi(\cdot)$ or a predefined kernel $\kappa(\cdot, \cdot)$, e.g. the Gaussian RBF kernel, in which the feature dimension d is infinite. Intuitively, the inner product (8.2) is a linear kernel, thus convolution is a linear case of kervolution.

The definition of kervolution (8.3) retains the advantages of convolution and brings new features: (i) sharing weights (Section 8.3.1); (ii) the equivariance to translation (Section 8.3.2); (iii) increased model capacity and new metrics for feature similarity (Section 8.3.3);

8.3.1 Sharing Weights

Sharing weights normally mean less trainable parameters and lower computational complexity. It is straightforward that the number of elements in filter $\mathbf{w}$ is not increased according to the definition of (8.5), thus kervolution keeps the sparse connectivity of convolution. As a comparison, we take the Volterra series-based non-linear convolution adopted in [179] as an example, the additional parameters of non-linear terms dramatically increase the training complexity, since the number of learnable parameters increases exponentially with the order of non-linearity. Even an quadratic expression $\mathbf{g}^v(\mathbf{x})$ in (8.6) from [179] is of complexity $\mathcal{O}(n^2)$:

$$\mathbf{g}_i^v(\mathbf{x}) = \mathbf{x}_{(i)}^T \mathbf{w}_2 \mathbf{x}_{(i)} + \mathbf{w}_1^T \mathbf{x}_{(i)}, \quad (8.6)$$

where $\mathbf{w}_1 \in \mathbb{R}^n$ and $\mathbf{w}_2 \in \mathbb{R}^{n \times n}$ are the linear and quadratic filters, respectively. The quadratic term in (8.6) introduces additional $n(n+1)/2$ parameters ($\mathbf{w}_2$ is an upper triangular matrix). Instead, a typical non-linear kernel is normally of complexity $\mathcal{O}(n)$, i.e. the Gaussian RBF kernel, which is the same with linear kernel (8.2), thus kervolution preserves the linear computational complexity $\mathcal{O}(n)$.

Another strategy to introduce higher order features is to explore the pooling layers. For example, the kernel pooling method proposed in [182] directly concatenates the non-linear terms $c_j (\mathbf{x}_{(i)}^T \mathbf{w})^j$ as in (8.5) to the pooling layers. However, this

requires explicit calculation of the non-linear terms up to p orders, although it can be approximated by applying the discrete Fourier transform (DFT) of p times, resulting in a computational complexity of $\mathcal{O}(p \cdot n \log n)$. Nevertheless, based on the kernel trick, kervolution can introduce any order of non-linear terms yet still with linear complexity.

8.3.2 Translational Equivariance

A crucial aspect of current architectures of deep learning is the encoding of invariances. One of the reasons is that the convolutional layers are naturally equivariant to image translation [20]. In this section, we show that kervolution preserves this important property. An operator is equivariant to a transform when the effect of the transform is detectable in the operator output [25]. Therefore, we have $\mathbf{f}_{(j)}(\mathbf{x}) = \mathbf{x}_{(j)} \oplus \mathbf{w}$, which means the input translation results in the output translation [20]. Similarly,

Theorem 8.3.1. kervolution (8.3) is equivariant to translation.

Proof. Assume $\mathbf{g}'(\mathbf{x}) = \mathbf{x}_{(j)} \otimes \mathbf{w}$, according to (8.5), we have

$$\mathbf{g}'_i(\mathbf{x}) = \kappa(\mathbf{x}_{(i+j)}, \mathbf{w}) = \mathbf{g}_{i+j}(\mathbf{x}). \quad (8.7)$$

Therefore, the i_{th} element of $\mathbf{x}_{(j)} \otimes \mathbf{w}$ is the $(i + j)_{\text{th}}$ element of $\mathbf{g}(\mathbf{x})$, hence we have

$$\mathbf{g}_{(j)}(\mathbf{x}) = \mathbf{x}_{(j)} \otimes \mathbf{w}, \quad (8.8)$$

which completes the proof. $\square$

Note that the translational invariance of CNN is achieved by concatenating pooling layers to convolutional layers [20], and the translational invariance of KNN can be achieved similarly. This property is crucial, since when invariances are present in the data, encoding them explicitly in an architecture provides an important source of regularization, which reduces the amount of training data required [192].

FIGURE 8.1: The comparison of learned filters on MNIST from the first layer (six channels and filter size of 5×5) of CNN and polynomial KNN. It is interesting that some of the learned filters (e.g. channel 4) from KNN are quite similar to CNN. This indicates that part of the kervolutional layer learns linear behavior, which is controlled by the linear part of the polynomial kernel.

Method	Convolution	L^1 -norm	L^2 -norm
None	99.17	99.12	99.11
FGSM	71.92	74.08	76.36

TABLE 8.1: Test Accuracy (%) of the white-box FGSM attack with L^p -norm kervolution on MNIST. 10K images are randomly selected.

8.3.3 Model Capacity and Features

It is straightforward that the kernel function (8.5) takes kervolution to non-linear space, thus the model capacity is increased without introducing extra parameters. Recall that CNN is a powerful approach to extract discriminative local descriptors. In particular, the linear kernel (8.2) of convolution measures the similarity of the input $\mathbf{x}$ and filter $\mathbf{w}$, i.e. the cosine of the angle θ between the two patches, since $\langle \mathbf{x}, \mathbf{w} \rangle = \cos(\theta) \cdot \|\mathbf{x}\| \|\mathbf{w}\|$. From this point of view, kervolution measures the similarity by match kernels, which are equivalent to extracting specific features [199]. We next discuss how to interpret the kernel functions and present a few instances $\kappa(\cdot, \cdot)$ of the kervolutional operator. One of the advantages of kervolution is that the non-linear properties can be customized without explicit calculation.

L^p -Norm Kervolution The L^1 -norm in (8.9a) and L^2 -norm in (8.9b) simply measures the Manhattan and Euclidean distances between input $\mathbf{x}$ and filter $\mathbf{w}$,

respectively.

$$\kappa_m(\mathbf{x}, \mathbf{w}) = \|\mathbf{x} - \mathbf{w}\|_1, \quad (8.9a)$$

$$\kappa_e(\mathbf{x}, \mathbf{w}) = \|\mathbf{x} - \mathbf{w}\|_2. \quad (8.9b)$$

Both "distances" of two points involves aggregating the distances between each element. If vectors are close on most elements, but more discrepant on one of them, Euclidean distance will reduce that discrepancy (elements are mostly smaller than 1 because of the normalization layers), being more influenced by the closeness of the other elements. Therefore, the Euclidean kervolution may be more robust to slight pixel perturbation. This hypothesis is verified by a simple simulation of adversary attack using the fast gradient sign method (FGSM) [200], shown in Table 8.1, where 'None' means the test accuracy on clean data.

Polynomial Kervolution Although the existing literatures have shown that the polynomial kernel (8.10) works well for the problem of natural language processing (NLP) when $d_p = 2$ using SVM [201], we find its performance is better when $d_p = 3$ in KNN for the problem of image recognition.

$$\kappa_p(\mathbf{x}, \mathbf{w}) = (\mathbf{x}^T \mathbf{w} + c_p)^{d_p} = \sum_{j=0}^{d_p} c_p^{d_p-j} (\mathbf{x}^T \mathbf{w})^j, \quad (8.10)$$

where d_p ($d_p \in \mathbb{Z}^+$) extends the feature space to d_p dimensions; c_p ($c_p \in \mathbb{R}^+$) is able to balance the non-linear orders (Intuitively, higher order terms play more important roles when $c_p < 1$). As a comparison, the kernel pooling strategy [182] concatenates the non-linear terms $c_j(\mathbf{x}^T \mathbf{w})^j$ directly, while they are finally linearly combined by subsequent fully connected layer, which dramatically increases the number of learnable parameters in the linear layer.

To show the behavior of polynomial kervolution, the learned filters of LeNet-5 trained for MNIST are visualized in Figure 8.1, which contains all six channels of the first kervolutional layer using polynomial kernel ($d_p = 2, c_p = 0.5$). The optimization process is described in Section 8.4. For a comparison, the learned filters from CNN are also presented. It is interesting that some of the learned filters of KNN and CNN are quite similar, e.g. channel 4, which means that part of the capacity of KNN learns linear behavior as CNN. This verifies our understanding of

polynomial kernel, which is a combination of linear and higher order terms. This phenomenon also indicates that polynomial kervolution introduces higher order feature interaction in a more flexible and direct way than the existing methods.

Gaussian Kervolution The Gaussian RBF kernel (8.11) extends kervolution to infinite dimensions.

$$\kappa_g(\mathbf{x}, \mathbf{w}) = \exp(-\gamma_g \|\mathbf{x} - \mathbf{w}\|^2), \quad (8.11)$$

where γ_g ($\gamma_g \in \mathbb{R}^+$) is a hyperparameter to control the smoothness of decision boundary. It extends kervolution to infinite dimensions because of the i -degree terms in (8.12).

$$\kappa_g(\mathbf{x}, \mathbf{w}) = C \sum_{i=0}^{\infty} \frac{(\mathbf{x}^T \mathbf{w})^i}{i!}, \quad (8.12)$$

where $C = \exp\left(-\frac{1}{2}(\|\mathbf{x}\|^2 + \|\mathbf{w}\|^2)\right)$ if $\gamma_g = \frac{1}{2}$.

The expression (8.12) is helpful for our intuitive understanding, while the recent discovery reveals more information. It is shown in [202] that the Gaussian kernel and its variants are able to measure the similarity of gradient based patch-wise features, e.g. SIFT [203] and HOG [204]. This provides a unified way to generate a rich, diverse visual feature set [205]. However, instead of using the hand-crafted features as kernel SVM, with KNN, we are able to inherit the substantial achievements based on kernel trick while still taking advantage of the great generalization ability of neural networks.

8.3.4 Kervolutional Layers and Learnable Kernel

Similar to a convolutional layer, the operation of a kervolutional layer is slightly different from the standard definition (8.3) in which $\mathbf{x}_{(i)}$ becomes a 3-D patch in a sliding window on the input. To be compatible with existing works, we also implement all popular available structures of convolution in CNN library [206] for kervolution, including the input and output channels, input padding, bias, groups (to control connections between input and output), size, stride, and dilation of the sliding window. Therefore, the convolutional layers of all existing networks can be directly or partially replaced by kervolutional layers, which makes KNN inherit all the existing achievements of CNN, e.g. network architectures [8, 15] and their numerous applications [176].

With kervolution, we are able to extract specific type of features without paying attention to the weight parameters. However, as aforementioned, we still need to tune the hyperparameters for some specific kernels, e.g. the balancer c_p in polynomial kernel, the smoother γ_g in Gaussian RBF kernel. Although we noticed that the model performance is mostly insensitive to the kernel hyperparameters, which is presented in Section 8.4.2, it is sometimes troublesome when we have no knowledge about the kernel. Therefore, we also implement training the network with learnable kernel hyperparameters based on the back-propagation [207]. This slightly increases the training complexity theoretically, but in experiments we found that this brings more flexibility and the additional cost for training several kernel parameters is negligible, compared to learning millions of parameters in the network. Taking the Gaussian kervolution as an example, the gradients are computed as:

$$\frac{\partial}{\partial \mathbf{w}} \kappa_g(\mathbf{x}, \mathbf{w}) = 2\gamma_g(\mathbf{x} - \mathbf{w})\kappa_g(\mathbf{x}, \mathbf{w}), \quad (8.13a)$$

$$\frac{\partial}{\partial \gamma_g} \kappa_g(\mathbf{x}, \mathbf{w}) = -\|\mathbf{x} - \mathbf{w}\|^2 \kappa_g(\mathbf{x}, \mathbf{w}). \quad (8.13b)$$

Note that the polynomial order d_p is not trainable because of the integer limitation, since the real exponent may produce complex numbers, which makes the network complicated.

8.4 Ablation Study

This section explores the influences of the kernels, the hyperparameters, and combination of kervolutional layers using LeNet-5 and MNIST [18]. To eliminate the influence of other factors, all configurations are kept as the same. The accuracy of modern networks on MNIST has been saturated, thus we adopt the evaluation criteria proposed in DAWNBench [208] that jointly considers the computational efforts and precision. It measures the total training time to a target validation accuracy (98%), which is a trade-off between efficiency and accuracy. In all the experiments of this section, we apply the stochastic gradient descent (SGD) method for training, where a mini-batch size of 50, a momentum of 0.9, an initial learning rate of 0.003, a multiplicative factor of 0.1, and a maximum epoch of 20 with milestones of [10, 15] are adopted. Our algorithm is implemented based on the

FIGURE 8.2: The influences of kervolution on the convergence rate. The best validation accuracy (20 epochs) and training time to target accuracy (98%) are displayed in brackets within the legend of each figure, respectively. (a) and (b) demonstrate that the kernel functions have a significant impact on the convergence speed. (c) and (d) demonstrate that the hyperparameters of kervolutional layer play less important roles than kernels. (e) and (f) shows the effects of arrangement of kervolutional layers.

PyTorch library [206]. All tests are conducted on a single Nvidia GPU of GeForce GTX 1080Ti. The reported training time does not include the testing and checkpoint saving time.

8.4.1 Kernels

Following the ablation principle, we only replace the convolutional layers of LeNet-5 by kervolutional layer using three kernel functions, i.e. polynomial kernel of $d_p = 3, c_p = 1$ in (8.10), Gaussian kernel of $\gamma_g = 1$ in (8.11), and also sigmoid kernel $\kappa_s(\mathbf{x}, \mathbf{w}) = \tanh(\mathbf{x}^T \mathbf{w})$. As shown in Figure 8.2 (a) and (b), although the computational complexity of non-linear kernels is slightly higher than that of linear kernel (convolution), the polynomial and Gaussian KNN are still able to converge to a validation accuracy of 98% more than $2\times$ faster than the original CNN. However, the convergence speed of sigmoid KNN is $2\times$ slower than that of CNN, which indicates that the kernel functions are crucial and have a significant impact on performance. Thanks to the wealth of traditional methods, we have many other useful kernels [209], although we cannot test all of them in this chapter. The L^1 and L^2 -norm KNN achieve an accuracy of 99.05% and 99.19%, respectively, but

we omit them in Figure 8.2 (a) and (b) because they nearly coincide with the polynomial curve.

8.4.2 Hyperparameters

From the above analysis, we credit the significant improvements of convergence speed to the usage of different kernels. This part verifies this assumption and further explores the influences of kernel hyperparameters. The polynomial kervolution (8.10) with two hyperparameters, non-linear order d_p and balancer c_p , is selected. As shown in Figure 8.2 (c) and (d), the convergence speed and validation accuracy of polynomial KNN using different hyperparameters are pretty similar to that of Figure 8.2 (a) and (b), which indicates that KCC is less sensitive to the kernel hyperparameters.

It is also noticed that the KNN with learnable kernel parameters achieves the best precision (99.20%) in this group, although it slightly increases the training time compared to KNN ($d_p = 3, c_p = 1$). However, the cost is justifiable since it saves the hyperparameter tuning process and the convergence is still much faster than the baseline CNN.

8.4.3 Layer Arrangement

This part explores the influences of the placement of kervolutional layers. Thanks to the simplicity of LeNet-5 (two convolutional layers), we can test all possible configurations of layer arrangement, i.e. "conv-conv", "kerv-conv", "conv-kerv", and "kerv-kerv". As shown in Figure 8.2 (e) and (f), where the polynomial kernel ($d_p = 3, c_p = 1$) is adopted, KNN still brings faster convergence. One interesting phenomenon is that the architecture of "kerv-conv" achieves better precision but slower convergence than "conv-kerv" (we run multiple times and the results are similar). This indicates that the sequence of kervolutional layers has an impact on performance, although the model complexity is the same. One side effect is that we may need to make some efforts to adjust the layer sequence for deeper KNNs. It is also noticed that the architecture of 'kerv-kerv' achieves the fastest convergence but only with a comparable validation accuracy to CNN. We argue that this is caused by the over-fitting problem, since its final training loss is very close to others (≈ 0.01),

which means that the model capacity of double kervolutional layers together with the activation and max pooling layers is too large for the MNIST dataset.

8.4.4 Removing ReLU

As mentioned in Section 6.1, the non-linearity of CNN mainly comes from the activation (ReLU) and max pooling layers. Intuitively, KNN may be able to achieve same model performance without activation or max pooling layers. To this end, we simply remove all the activation layers of LeNet-5 and replace the max pooling by average pooling layers, which means that all the non-linearity comes from the kervolutional layers. Without surprise, the CNN only achieves an accuracy of 92.22%, which is far from the target accuracy of 98%, hence the training time comparison figure is omitted. However, the KNN of "gaussian-polynomial" and "polynomial-polynomial" both achieve an accuracy of 99.11%, which further verifies the effectiveness of kervolution. In another sense, the strategy of removing the activation layers is one of the solutions to the aforementioned over-fitting problem in Section 8.4.3, although we need more investigations to find the best architectures for KNN. Note that we do not encourage to remove all ReLU layers in deeper networks, since there may be gradient vanishing or exploding problems, and kervolution is not designed to overcome these problems.

8.5 Experiments

This section aims to demonstrate the effectiveness of deep KNN on larger datasets. In practice, the network architecture has a significant impact on performance. Since the modern networks are so deep and kervolution provides many possibilities via different kernels, we cannot perform exhaustive tests to find the best sequence of kervolutional layers. Hence, we construct KNN based on several existing architectures by only changing the first convolutional layers to kervolutional layers. Other factors, such as data augmentation and optimizers, are kept as their original configurations. As discovered in Section 8.4, this may not be the best configuration, while it can demonstrate the effectiveness of kervolution.

Network	CIFAR-10	CIFAR-100
CNN	13.63	44.74
KNN	10.85	37.12

TABLE 8.2: Validation error (%) of ResNets on CIFAR-10 and CIFAR-100 without data augmentation.

Architecture	CIFAR-10+		CIFAR-100+	
	CNN	KNN	CNN	KNN
GoogLeNet	13.37	5.16	26.65	20.84
ResNet	6.43	4.36	27.22	22.49
DenseNet	5.24	5.08	24.42	24.92

TABLE 8.3: Validation error (%) on CIFAR on different architectures with data augmentation.

The CIFAR experiments in this section are conducted in a single GPU of Nvidia GeForce GTX 1080Ti while we employ four Nvidia Tesla M40 in the ImageNet experiments. The polynomial kervolutional layer in this section adopts the learnable balancer c_p with power $d_p = 3$.

8.5.1 CIFAR

The CIFAR-10 and CIFAR-100 [210] datasets consist of colored natural images with 32×32 pixels in 10 and 100 classes, respectively. Each dataset contains 50k images for training and 10k for testing. In the testing procedure, only the single view of the original image is evaluated.

The proposed KNNs are first evaluated without data augmentation using the architecture of ResNet. We construct and train ResNet-34 following the original configurations of [15] with cross-entropy loss. The stochastic gradient descent (SGD) is adopted with momentum of 0.9. We train the networks for 200 epochs with a mini-batch size of 128. The learning rate decays by 0.1 for every 50 epochs; the weight decay stays at 0.0005. The validation error of KNN as well as the best performance of baseline CNN [211] are presented in Table 8.2. It is interesting that KNN outperforms CNN with a large margin of 2.8% and 7.63%, respectively.

Hyperparm	$d_p = 3$	$d_p = 5$
$c_p = 0$	4.78	5.42
$c_p = 1$	4.60	5.36
learnable c_p	4.76	4.73

TABLE 8.4: Validation error (%) of polynomial KNN using ResNet-32 on CIFAR-10+ using different hyper parameters.

Hyperparm	$d_p = 3$	$d_p = 5$
$c_p = 0$	0.83	1.41
$c_p = 1$	1.41	1.46
learnable c_p	0.86	0.79

TABLE 8.5: Training time (h) of polynomial KNN using ResNet-32 on CIFAR-10+ using different hyper parameters.

We perform more tests using the data augmentation techniques, and the datasets are denoted as ‘CIFAR-10+’ and ‘CIFAR-100+’, respectively. The KNNs are constructed following the architecture of GoogLeNet [185] and DenseNet-40-12 [187]. Data augmentation is applied following the configuration in ResNet [15], including horizontal flipping with a probability of 50%, reflection-padding by 4 pixels and taking a random crop of size 32×32 .

Different from ResNet and GoogLeNet, we train the DenseNet-40-12 following its original configuration [187], i.e. SGD with batch size 64 for 300 epochs. The initial learning rate is set to 0.1, and is divided by 10 at 50% and 75% of the total number of training epochs. Table 8.3 lists the validation error of KNN and the baselines from [15, 185, 187, 212]. In Table 8.3, the validation errors of GoogLeNet and ResNet are reduced by 8.21%/5.81% and 2.07%/4.73% on CIFAR-10/100, respectively. While for DenseNet, we cannot see such significant improvements. This indicates that the polynomial kernel layers may not be suitable for fully connected architectures.

We further demonstrate the sensitivity to the kernel hyperparameters. Table 8.4 lists the validation errors of KNN on CIFAR-10+ with polynomial kernel of $d_p = 3, 5$ and $c_p = 0, 1$ using the architecture of ResNet-32. The performance with learnable balancer of c_p is also given for comparison. As suggested by [208], their training time to an accuracy of 94% is measured and presented in Table 8.5. It

is interesting that, the configuration of $d_p = 3, c_p = 1$ achieves the best accuracy, while $d_p = 5$ with learnable c_p requires the least training time. The networks with learnable kernel achieve the best overall performance by jointly considering the training time and accuracy. This indicates that the learnable kernel technique can produce compromised performance without tuning parameters.

8.5.2 ImageNet

The ILSVRC 2012 classification dataset [13] is composed of 1.2M images for training and 50k images for validation in 1000 classes. For fair comparison, we apply the same data augmentation as described in [15, 213], where the single-crop and 10-crop at a size of 224×224 are applied for testing.

We select four versions of ResNet [15], including ResNet-18, ResNet-34, ResNet-50 and ResNet-101, as the baselines. The kervolutional layer is applied with a polynomial kernel ($d = 3, c_p = 2$). All the networks are trained using the stochastic gradient descent (SGD) method for 100 epochs with a batch size of 256. The learning rate is set to 0.1, and is reduced every 30 epochs. Also, a weight decay of 10^{-4} and a momentum of 0.9 without dampening are employed. In our experiments, the best performance of ResNet cannot be achieved in limited training time. To guarantee a fair comparison, the results of ResNet which have the best accuracy in [15, 187, 213, 214] are chosen. We report the single-crop and 10-crop validation error on ImageNet in Table 8.6, where the performance of KNN is the average of five running.

In Table 8.6, top-1 errors for single-crop and 10-crop are reduced by 0.29%-1.0% and 0.37%-0.83%, respectively. For top-5 errors, KNN-18/34/50/101 outperform corresponding ResNets by 0.43%, 0.24%, 0.23%, 0.28% in single-crop, and 0.39%, 0.68%, 0.74%, 0.87% in 10-crop, respectively. It is noticed that simple replacement of the convolutional layer in ResNet leads to obvious improvements. We believe that more customized network architecture as well as extensive hyperparameter searches can further improve the performance of KNN on ImageNet.

Network	Top-1	Top-5
ResNet-18	30.24 / 27.88	10.92 / 9.42
KNN-18	29.74 / 27.43	10.49 / 9.03
ResNet-34	26.70 / 25.03	8.58 / 7.76
KNN-34	26.29 / 24.28	8.34 / 7.08
ResNet-50	23.85 / 22.85	7.13 / 6.71
KNN-50	23.56 / 22.05	6.90 / 5.97
ResNet-101	22.63 / 21.75	6.44 / 6.05
KNN-101	21.63 / 20.92	6.16 / 5.18

TABLE 8.6: The top-1 and top-5 validation errors on ImageNet, with single-crop and 10-crop testing, respectively.

8.6 Discussion

Why Kervolution? Different from convolution, which can only extract linear features, kervolution is able to extract customized patch-wise non-linear features, which makes KNN much more flexible. It is demonstrated that the higher order terms make the subsequent linear classifier more discriminant, while this does not increase the computational complexity. However, we have only tested several kernels, e.g. polynomial and Gaussian. The kernel functions and their hyperparameters can be task-driven and more investigations are to be done.

Network Architecture In this chapter, we have applied kervolution to the existing architectures, e.g. GoogLeNet and ResNet, directly. It is obvious that the performance of kervolution is dependent on the architecture. One of the interesting challenges for future work is to investigate the relationship between the architecture and kervolution.

8.7 Conclusion

This chapter introduces the kervolution to generalize convolution to non-linear space, and extends convolutional neural networks to kervolutional neural networks. The capacity of KNN is dramatically increased by leveraging on the non-linearity

of kernel functions, which requires no extra parameters. The higher order terms also make the model more discriminative. Due to the large number of choices of kervolution, we cannot perform a brute force search for all the possibilities, while this opens a new space for the construction of deep networks. It has been demonstrated that, with careful kernel chosen, the performance of CNN can be significantly improved via directly replacing convolutional layers by kervolutional layers. We expect the introduction of kervolutional layers in more architectures and extensive hyperparameter searches can further improve the performance.

Chapter 9

Conclusion & Future Work

9.1 Conclusion

This thesis has developed a set of kernel learning algorithms for visual perceptual tasks, including object tracking, optical flow, localization, mapping, and image recognition. Both theoretical and practical contributions are made.

The formulation for non-linear correlator (KCC) proposed in Chapter 3 unifies the existing solutions. KCC is able to be customized by specific kernels and not limited to circulant data structure, hence it can predict the affine transforms, especially the scale, rotation, and translation. The theoretical analysis of the KCC was conducted to demonstrate its generality and the KCC was compared with the existing solutions to illustrate its flexibility. Extensive experimental results on public datasets demonstrated that the proposed KCC performs favorably against the state-of-the-art methods in terms of efficiency, accuracy and robustness.

Chapter 4 proposed a KCC-based visual tracker, which consists of KTC and KSC for translation and scale prediction, respectively. To quickly adapt the appearance change, an on-line joint update rule was proposed to fit the object model. Compared to the existing scale trackers that are basically multiple translational trackers on different scales, the KSC-based tracker can predict the scale transform directly. The extensive experiments on OTB dataset showed that the KCC improves the performance dramatically. KCC tracker is more robust to noise and distortion, and can quickly adapt the scale change, comparing with the state-of-the-art methods.

In Chapter 5, a robust and computationally efficient optical flow method, called correlation flow was proposed for velocity estimation using a monocular camera. To compute the translation and scale-rotation flow efficiently, the kernel translation correlator and joint kernel scale-rotation correlators were introduced. Due to the low computational cost, the proposed method can run in real-time on an ultra-low-power processor. Experiments on velocity estimation showed that correlation flow produces more reliable results than PX4Flow. Autonomous flight and hovering tests demonstrated that correlation flow can provide robust trajectory estimation at very low computational cost. The source codes were released for research purpose.

A novel framework for non-iterative RGB-D-inertial SLAM system was proposed in Chapter 6. It combines a non-iterative front-end and a iterative back-end, resulting in a lightweight and robust SLAM system. Based on KCC, the front-end odometry system can be accelerated by an on-line training process. To the best of my knowledge, the proposed framework is the first non-iterative solution to the front-end of a dense mapping system. It is also demonstrated that it is able to achieve much faster speed and comparable accuracy with the state-of-the-art methods.

An ultra-wide band aided localization and dense mapping system was proposed in Chapter 7. By taking advantage of the front-end of NI-SLAM and the UWB technologies, multiple capabilities are achieved simultaneously. First, accurate and drift-free localization are enabled through a novel graph optimization technique incorporating multiple constraints from different sensors. Second, the proposed approach greatly reduces the computational complexity, so that dense maps can be generated in real-time even for featureless environments, while only running on an ultra-low power processor. Finally, the framework was demonstrated to be versatile, robust, and can be easily adapted for other methods.

Chapter 8 extended the theoretical work in Chapter 3 to neural networks. It introduced the kervolution, which generalizes convolution to non-linear space, and extends CNN to KNN. The capacity of KNN is dramatically increased by leveraging on the non-linearity of kernel vector, which requires no extra parameters. Due to the large number of choices of kervolution, a brute force search for all the possibilities cannot be performed, while this opens a new space for the construction of deep networks. It has been demonstrated that, with careful kernel chosen, the performance of CNN can be significantly improved via directly replacing convolutional layers by kervolutional layers.

9.2 Future Work

The theoretical analysis and practical experiments presented in this thesis have demonstrated the great advantages of the proposed kernel learning algorithms. The successes of these methods inspire us to conduct more in-depth and extensive research. The possible future works are listed below.

- KCC has shown its great advantage on efficiency and robustness in human activity recognition, optical flow, visual tracking, and RGB-D-inertial odometry. We can expect its usages in other fields, especially in low power and real-time systems, such as the IoT (Internet of Things) devices.
- The KNN is another useful tool for deep learning, although only the application in image recognition is demonstrated. Considering the popular trends of CNN, we may expect its application in other fields, including but not limited to object detection, image segmentation, transfer learning, etc.

Except for the aforementioned topics, we believe the progresses in the following areas will benefit a lot directly or indirectly for the sustainable development of human beings.

9.2.1 Machine Learning

Multimodal Learning for Heterogeneous Data Deep learning is applied in various areas such as visual computing, speech recognition, text mining, etc. In each area, the learning model is trained with large amounts of data in isolation. A multimodal learning approach from heterogeneous data sources, including video, speech, text, etc. is necessary to achieve breakthrough in performance [215].

Learning with Fewer Labeled Data Labeling large amounts of training data is one of the most time-consuming steps in the deep learning pipeline, and is often not feasible. The aim is to achieve the same performance with $10\times$ fewer labeled training samples. This is a strong step in the direction of unsupervised deep learning, where representations are learnt with large amounts of unlabeled unstructured data [216]. Algorithms to move towards fewer training data will be fundamental progresses for various applications.

Learning with Smaller Models Current deep learning models have tens of millions of neurons, making it infeasible for deployment on mobile and embedded hardware. The aim is to achieve models that are much smaller, with little loss in performance. This capability will also be critical for emerging neuromorphic platforms, where the number of spiking neurons is orders of magnitude smaller than the current state-of-the-art deep learning models.

Online Learning with Real-time Model Updates State-of-the-art deep learning models still take long time to train on expensive GPU infrastructure. It is important to deploy an evolvable learning system to adapt new data and even new classes. Algorithms to move towards real-time model updates will be essential.

Interpretable and Contextually Informed Models Current deep learning models are largely black-boxes that are not informed by contextual relations and provide little intuition. Improved abilities to regulate the learning process, and interpretation of the predictions will be essential for the next wave of AI, especially for complex learning scenarios and applications involving critical decisions.

Learning with Knowledge Graphs Current deep learning systems have no real understanding of images, with output typically a bunch of bounding boxes, learning using a large number of label or sentence-level associations. Knowledge graphs will be useful for understanding higher-level semantics and relationships between entities, and improving classification accuracy.

9.2.2 Application

Internet of Things Generally speaking, Internet of Things (IoT) refers to the networked interconnection of everyday objects, which are often equipped with ubiquitous intelligence [217]. IoT is opening tremendous opportunities for a large number of novel applications that promise to improve the quality of our lives. Due to the physical size and power supply, the IoT devices normally can only provide very limited computational resources, hence requiring the software system to be efficient and intelligent to reduced human exertions. To promote the development

of IoT, researchers need to pay more attentions to efficient computing, awareness about the world and human intention, and the network scalability.

Mixed Reality Different from virtual reality (VR), which is a simulated environment generated by computers [218] and augmented reality (AR), which refers to the merging of real world objects into virtual worlds [219], mixed reality (MR) is the merging of real and virtual worlds to produce new environments and visualizations where physical and digital objects co-exist and interact in real time [220]. MR is the next evolution in human interaction with the world and is able to unlock possibilities that before now were restricted to our imaginations, including but not limited to entertainment, education, creativity, and business, etc. The investigations for MR including at least full body tracking, deformable reconstruction of the environments, and scene understanding will be highly active research areas.

Neuromorphic Computing The human brain is the most powerful supercomputer on Earth, and researchers are taking inspiration from the structure of the human brain to make better artificial intelligence systems. Neuromorphic computing is a rapidly evolving field that aims to develop brain-inspired intelligence by neural modelling and to implement such computing in neuromorphic circuits for applications such as robotics, IoT sensors and wearables [221]. One possible orientation is to investigate how existing learning structures can be efficiently computed, quantized, or compressed that is suitable for specific hardware implementation.

List of Author's Awards, Patents, and Publications¹

Awards

- **Best Paper Award in Robotic Planning**, “Non-Iterative SLAM,” in *2017 18th International Conference on Advanced Robotics (ICAR)*, Hong Kong, Jul. 2017.

Patents and Technical Disclosure

- **Chen Wang***, Lihua Xie*, and Junsong Yuan, “Simultaneous Localization and Mapping Methods and Apparatus,” *International/U.S. Patent, Application No.: PCT/SG2017/050446*, Sep. 2016.
- **Chen Wang***, Lihua Xie*, and Xu Fang, “A Novel Graph Optimization Approach to Localization with Application for Unmanned Systems,” *NTU Ref: TD/254/17, Sep. 2017*, Sep. 2017.
- **Chen Wang***, Lihua Xie*, Jianfei Yang, and Junsong Yuan, “Kervolutional Neural Networks,” *NTU Ref: PAT/344/17/18/SG PRV, Singapore Provisional Patent Application No.: 10201802135Y.*, Mar. 2018.

Publications

- **Chen Wang**, Junsong Yuan, and Lihua Xie, “Non-iterative SLAM,” in *International Conference on Advanced Robotics (ICAR)*, pp. 83-90, Jul. 2017.

¹The superscript * indicates joint first authors

- **Chen Wang**, Handuo Zhang, Thien-Minh Nguyen, and Lihua Xie, “Ultra-Wideband Aided Fast Localization and Mapping System,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1602-1609, Sep. 2017.
- **Chen Wang**, Le Zhang, Lihua Xie, and Junsong Yuan, “Kernel Cross-Correlator,” in *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4179-4186, Feb. 2018.
- **Chen Wang***, Tete Ji*, Thien-Minh Nguyen, and Lihua Xie, “Correlation Flow: Robust Optical Flow Using Kernel Cross-Correlators,” in *International Conference on Robotics and Automation (ICRA)*, May. 2018.
- **Chen Wang**, Jianfei Yang, Lihua Xie, and Junsong Yuan, “Kervolutional Neural Networks,” *2019 Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.
- **Chen Wang**, Minh-Chung Hoang, Lihua Xie, and Junsong Yuan, “Non-iterative RGB-D-inertial Odometry,” *arXiv preprint: 1710.05502*.

Bibliography

- [1] Hyeran Byun and Seong-Whan Lee. A survey on pattern recognition applications of support vector machines. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(03):459–486, 2003. [9](#)
- [2] Edgar Osuna, Robert Freund, and Federico Girosit. Training support vector machines: an application to face detection. In *Computer vision and pattern recognition, 1997. Proceedings., 1997 IEEE computer society conference on*, pages 130–136. IEEE, 1997. [10](#)
- [3] Massimiliano Pontil and Alessandro Verri. Support vector machines for 3d object recognition. *IEEE transactions on pattern analysis and machine intelligence*, 20(6):637–646, 1998. [10](#)
- [4] Samy Bengio and Johnny Mariéthoz. Learning the decision function for speaker verification. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 425–428. IEEE, 2001. [10](#)
- [5] Pengyu Hong, Qi Tian, and Thomas S Huang. Incorporate support vector machines to content-based image retrieval with relevant feedback. In *ICIP*, pages 750–753, 2000. [10](#)
- [6] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992. [10](#), [126](#)
- [7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. [10](#), [126](#), [128](#)
- [8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Conference on Neural Information Processing Systems*, 2012. [10](#), [123](#), [125](#), [132](#)
- [9] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. [11](#)
- [10] Herbert Robbins and Sutton Monro. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. Springer, 1985. [13](#)

- [11] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International conference on computational learning theory*, pages 416–426. Springer, 2001. [14](#)
- [12] Shai Shalev-Shwartz and Yoram Singer. *Online learning: Theory, algorithms, and applications*. PhD thesis, Senate of the Hebrew University, 2007. [15](#)
- [13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009. CVPR 2009. IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009. [15](#), [139](#)
- [14] Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, pages 143–155, 1989. [16](#), [125](#)
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016. [16](#), [123](#), [125](#), [132](#), [137](#), [138](#), [139](#)
- [16] Shaoqing Ren, Kaiming He, Ross B Girshick, and Jian Sun. Faster R-CNN - Towards Real-Time Object Detection with Region Proposal Networks. In *Conference on Neural Information Processing Systems*, 2015. [16](#)
- [17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7):354–359, October 2017. [16](#), [125](#)
- [18] Y Lecun, L Bottou, Y Bengio, and P Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. [16](#), [133](#)
- [19] Pejman Tahmasebi, Ardeshir Hezarkhani, and Muhammad Sahimi. Multiple-point geostatistical modeling based on the cross-correlation functions. *Computational Geosciences*, 16(3):779–797, 2012. [17](#)
- [20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. [18](#), [20](#), [123](#), [125](#), [129](#)
- [21] Ronald Newbold Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986. [19](#)
- [22] Leo Bluestein. A linear filtering approach to the computation of discrete fourier transform. *IEEE Transactions on Audio and Electroacoustics*, 18(4):451–455, 1970. [19](#)

- [23] Charles M Rader. Discrete fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56(6):1107–1108, 1968. [19](#)
- [24] L Gaunce Jr, J Peter May, Mark Steinberger, et al. *Equivariant stable homotopy theory*, volume 1213. Springer, 2006. [19](#), [20](#)
- [25] Taco Cohen and Max Welling. Group Equivariant Convolutional Networks. *ICML*, 2016. [20](#), [129](#)
- [26] Craig Saunders, Alexander Gammerman, and Volodya Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, 1998. [22](#)
- [27] M Bishop Christopher. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2016. [22](#)
- [28] Philip J Davis. *Circulant matrices*. American Mathematical Soc., 2012. [22](#)
- [29] Robert M Gray. Toeplitz and Circulant Matrices: A Review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239, 2005. [23](#)
- [30] J F Henriques, R Caseiro, and P Martins. High-Speed Tracking with Kernelized Correlation Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. [23](#), [27](#), [29](#), [32](#), [35](#), [46](#), [47](#)
- [31] Dav Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual Object Tracking Using Adaptive Correlation Filters. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2544–2550. IEEE, 2010. [25](#), [27](#), [29](#), [32](#), [35](#), [46](#), [89](#)
- [32] D Messerschmitt et al. Stationary points of a real-valued function of a complex variable. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-93*, 2006. [25](#)
- [33] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Discriminative Scale Space Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1561–1575, 2017. [27](#), [29](#), [32](#), [35](#), [47](#)
- [34] Charles F Hester and David Casasent. Multivariant technique for multiclass pattern recognition. *Applied Optics*, 19(1):1758–1761, June 1980. [28](#)
- [35] David Casasent. Unified synthetic discriminant function computational formulation. *Applied Optics*, 23(1):1620–1627, May 1984. [28](#)
- [36] B V K Vijaya Kumar. Minimum-variance synthetic discriminant functions. *Journal of the Optical Society of America A*, 3(10):1579–1584, October 1986. [28](#)

- [37] Abhijit Mahalanobis, B V K Vijaya Kumar, and David P Casasent. Minimum average correlation energy filters. *Applied Optics*, 26(1):3633–3640, September 1987. [28](#)
- [38] Philippe Refregier. Optimal trade-off filters for noise robustness, sharpness of the correlation peak, and Horner efficiency. *Optics Letters*, 16(1):829–832, June 1991. [28](#)
- [39] Abhijit Mahalanobis, B V K Vijaya Kumar, Sewoong Song, S R F Sims, and J F Epperson. Unconstrained correlation filters. *Applied Optics*, 33(1): 3751–3759, June 1994. [29](#)
- [40] A Mahalanobis, B V K Vijaya Kumar, and S R F Sims. Distance-classifier correlation filters for multiclass target recognition. *Applied Optics IP*, 35(1): 3127–, June 1996. [29](#)
- [41] D S Bolme, B A Draper, and J R Beveridge. Average of Synthetic Exact Filters. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 2105–2112. IEEE, 2009. [29](#)
- [42] Ming Tang and Jiayi Feng. Multi-kernel Correlation Filter for Visual Tracking. In *IEEE International Conference on Computer Vision*, pages 3038–3046. IEEE, 2015. [29](#), [32](#), [35](#), [47](#)
- [43] Joseph A Fernandez, Vishnu Naresh Boddeti, Andres Rodriguez, and B V K Vijaya Kumar. Zero-Aliasing Correlation Filters for Object Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8): 1702–1715, August 2015. [29](#), [35](#)
- [44] Lord Rayleigh. Liii. on the character of the complete radiation at a given temperature. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 27(169):460–469, 1889. [34](#)
- [45] Hamed Kiani Galoogahi, Terence Sim, and Simon Lucey. Correlation Filters With Limited Boundaries. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4630–4638. IEEE, 2015. [35](#), [47](#)
- [46] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip H S Torr. Staple: Complementary Learners for Real-Time Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1401–1409. IEEE, 2016. [35](#)
- [47] Oscar D Lara and Miguel A Labrador. A Survey on Human Activity Recognition using Wearable Sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, 2013. [37](#), [38](#)
- [48] Arsalan Mosenia, Susmita Sur-Kolay, Anand Raghunathan, and Niraj Jha. Wearable medical sensor-based system design: A survey. *IEEE Transactions on Multi-Scale Computing Systems*, 2017. [37](#), [38](#)

- [49] Allen Y Yang, Roozbeh Jafari, S Shankar Sastry, and Ruzena Bajcsy. Distributed recognition of human actions using wearable motion sensor networks. *Journal of Ambient Intelligence and Smart Environments*, 1(2):103–115, 2009. [38](#)
- [50] Diego F Silva and Gustavo EAPA Batista. Speeding up all-pairwise dynamic time warping matrix calculation. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 837–845. SIAM, 2016. [40](#)
- [51] Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855:1–23, 2008. [40](#)
- [52] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *Proceedings of the IEEE Conference on Computer vision and pattern recognition*, volume 1, pages 798–805. IEEE, 2006. [44](#)
- [53] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003. [44](#)
- [54] Iain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815, 2004. [44](#)
- [55] Junseok Kwon and Kyoung Mu Lee. Visual tracking decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1269–1276. IEEE, 2010. [44](#)
- [56] Xi Li, Weiming Hu, Zhongfei Zhang, Xiaoqin Zhang, Mingliang Zhu, and Jian Cheng. Visual tracking via incremental log-euclidean riemannian subspace learning. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. [44](#)
- [57] Xi Li, Anthony Dick, Chunhua Shen, Anton Van Den Hengel, and Hanzi Wang. Incremental learning of 3d-dct compact representations for robust visual tracking. *IEEE transactions on pattern analysis and machine intelligence*, 35(4):863–881, 2013. [44](#)
- [58] Dong Wang, Huchuan Lu, and Ming-Hsuan Yang. Online object tracking with sparse prototypes. *IEEE Transactions on Image Processing*, 22(1):314–325, 2013. [44](#), [45](#)
- [59] Tianzhu Zhang, Bernard Ghanem, Si Liu, and Narendra Ahuja. Robust visual tracking via structured multi-task sparse learning. *International journal of computer vision*, 101(2):367–383, 2013. [44](#), [45](#)

- [60] Shaohua Kevin Zhou, Rama Chellappa, and Baback Moghaddam. Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Transactions on Image Processing*, 13(11):1491–1506, 2004. [44](#)
- [61] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011. [45](#)
- [62] Sam Hare, Amir Saffari, and Philip HS Torr. Struck: Structured output tracking with kernels. In *2011 IEEE International Conference on Computer Vision*, pages 263–270. IEEE, 2011. [45](#)
- [63] Chenglong Li, Liang Lin, Wangmeng Zuo, and Jin Tang. Learning patch-based dynamic graph for visual tracking. In *AAAI*, pages 4126–4132, 2017. [45](#)
- [64] Ying Wu and Thomas S Huang. Self-supervised learning for visual tracking and recognition of human hand. In *AAAI/IAAI*, pages 243–248, 2000. [45](#)
- [65] Xue Mei, Haibin Ling, Yi Wu, Erik P Blasch, and Li Bai. Efficient minimum error bounded particle resampling l1 tracker with occlusion detection. *IEEE Transactions on Image Processing*, 22(7):2661–2675, 2013. [45](#)
- [66] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *Proceedings of British Machine Vision Conference*, volume 1, page 6, 2006. [45](#)
- [67] Le Zhang, Jagannadan Varadarajan, Ponnuthurai Nagaratnam Suganthan, Narendra Ahuja, and Pierre Moulin. Robust visual tracking using oblique random forests. In *IEEE International Conference on Computer Vision and Pattern Recognition*. IEEE, 2017. [45](#), [54](#)
- [68] Shai Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, 2007. [45](#)
- [69] Helmut Grabner, Christian Leistner, and Horst Bischof. Semi-supervised on-line boosting for robust tracking. In *Proceedings of the European Conference on Computer Vision*, pages 234–247. Springer, 2008. [45](#)
- [70] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time object tracking via online discriminative feature selection. *IEEE Transactions on Image Processing*, 22(12):4664–4677, 2013. [45](#)
- [71] Xiaoyu Wang, Gang Hua, and Tony X Han. Discriminative tracking by metric learning. In *European conference on computer vision*, pages 200–214. Springer, 2010. [46](#)
- [72] Nan Jiang, Wenyu Liu, and Ying Wu. Learning adaptive metric for robust visual tracking. *IEEE Transactions on Image Processing*, 20(8):2288–2300, 2011. [46](#)

- [73] Yang Li, Jianke Zhu, and Steven CH Hoi. Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 353–361, 2015. [46](#)
- [74] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In *Advances in Neural Information Processing Systems*, pages 809–817, 2013. [46](#)
- [75] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *IEEE International Conference on Computer Vision*, pages 3119–3127, 2015. [46](#)
- [76] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Stct: Sequentially training convolutional networks for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, 2016. [46](#)
- [77] K. Zhang, Q. Liu, Y. Wu, and M. H. Yang. Robust visual tracking via convolutional networks without training. *IEEE Transactions on Image Processing*, 25(4):1779–1792, April 2016. ISSN 1057-7149. [46](#)
- [78] L. Zhang and P. N. Suganthan. Visual tracking with convolutional random vector functional link neural network. *IEEE Transactions on Cybernetics*, 2016. doi: 10.1109/TCYB.2016.2588526. [46](#)
- [79] Luca Bertinetto, Jack Valmadre, Stuart Golodetz, Ondrej Miksik, and Philip HS Torr. Staple: Complementary learners for real-time tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1401–1409, 2016. [47](#)
- [80] Ting Liu, Gang Wang, and Qingxiong Yang. Real-time part-based visual tracking via adaptive correlation filters. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4902–4912, June 2015. [47](#)
- [81] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-Term Correlation Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5388–5396. IEEE, 2015. [47](#)
- [82] Zhibin Hong, Zhe Chen, Chaohui Wang, Xue Mei, Danil Prokhorov, and Dacheng Tao. Multi-Store Tracker (MUSTer): a cognitive psychology inspired approach to object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 749–758, 2015. [47](#)
- [83] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, December 2015. [47](#)

- [84] Le Zhang and Ponnuthurai Nagaratnam Suganthan. Robust visual tracking via co-trained kernelized correlation filters. *Pattern Recognition*, 69:82–93, 2017. [47](#)
- [85] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE Conference on Computer vision and pattern recognition*, pages 2411–2418. IEEE, 2013. [51](#), [53](#), [54](#)
- [86] Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang. Robust object tracking via sparse collaborative appearance model. *IEEE Transactions on Image Processing*, 23(5):2356–2368, 2014. [54](#)
- [87] Naiyan Wang, Jianping Shi, Dit-Yan Yeung, and Jiaya Jia. Understanding and diagnosing visual tracking systems. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3101–3109, 2015. [54](#)
- [88] Dario Floreano and Robert J Wood. Science, technology and the future of small autonomous drones. *Nature*, 521(7):460–466, May 2015. [57](#)
- [89] Daniel Watman and Hideaki Murayama. Design of a miniature, multi-directional optical flow sensor for micro aerial vehicles. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2986–2991. IEEE, 2011. [67](#)
- [90] Farid Kendoul, Isabelle Fantoni, and Kenzo Nonami. Optic flow-based vision system for autonomous 3d localization and control of small aerial vehicles. *Robotics and Autonomous Systems*, 57(6):591–602, 2009. [57](#), [67](#)
- [91] Simon Zingg, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. Mav navigation through indoor corridors using optical flow. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3361–3368. IEEE, 2010. [57](#)
- [92] Dominik Honegger, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1736–1741. IEEE, 2013. [58](#), [59](#), [66](#), [67](#), [69](#)
- [93] K McGuire, G de Croon, and C De Wagter. Local histogram matching for efficient optical flow computation applied to velocity estimation on pocket drones. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3255–3260. IEEE, 2016. [58](#), [59](#), [67](#), [69](#)
- [94] Kimberly McGuire, Guido de Croon, Christophe de Wagter, Karl Tuyls, and Hilbert Kappen. Efficient Optical Flow and Stereo Vision for Velocity Estimation and Obstacle Avoidance on an Autonomous Pocket Drone. *IEEE Robotics and Automation Letters*, 2(2):1070–1076, February 2017. [58](#), [59](#)

- [95] Dominik Honegger, Pierre Greisen, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Real-time velocity estimation based on optical flow and disparity matching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5177–5182. IEEE, 2012. [58](#), [59](#)
- [96] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. [59](#)
- [97] Gunnar Farneback. Two-frame motion estimation based on polynomial expansion. *Image analysis*, pages 363–370, 2003. [59](#)
- [98] Alfonso Alba, Edgar Arce-Santana, and Mariano Rivera. Optical flow estimation with prior models obtained from phase correlation. *Advances in Visual Computing*, pages 417–426, 2010. [59](#)
- [99] Jianbo Shi and Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600. IEEE Comput. Soc. Press, 1994. [59](#)
- [100] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1508–1515. IEEE, 2005. [59](#)
- [101] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001. [59](#)
- [102] Christian Bailer, Kiran Varanasi, and Didier Stricker. Cnn-based patch matching for optical flow with thresholded hinge embedding loss. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. [59](#)
- [103] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. [59](#)
- [104] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. [59](#)
- [105] Qin-Sheng Chen, M Defrise, and F Deconinck. Symmetric phase-only matched filtering of Fourier-Mellin transforms for image registration and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12):1156–1168, 1994. [65](#)
- [106] Chen Wang, Handuo Zhang, Thien-Minh Nguyen, and Lihua Xie. Ultra-Wideband Aided Fast Localization and Mapping System. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1602–1609, September 2017. [68](#)

- [107] Chen Wang, Junsong Yuan, and Lihua Xie. Non-Iterative SLAM. In *2017 18th International Conference on Advanced Robotics (ICAR)*, pages 83–90, Hong Kong, July 2017. IEEE. [68](#), [93](#), [116](#)
- [108] Thien-Minh Nguyen, Abdul Hanif Zaini, Chen Wang, Kexin Guo, and Lihua Xie. Robust Target-relative Localization with Ultra-Wideband Ranging and Communication. In *International Conference on Robotics and Automation (ICRA)*. IEEE, May 2018. [70](#)
- [109] T Bailey and H Durrant-Whyte. Simultaneous Localization and Mapping (SLAM): Part II. *Robotics & Automation Magazine*, 13(3):108–117, 2006. [73](#)
- [110] P Henry, M Krainin, E Herbst, X Ren, and D Fox. RGB-D Mapping: Using Kinect-Style Depth Cameras for Dense 3D Modeling of Indoor Environments. *International Journal of Robotics Research*, 31(5):647–663, April 2012. [73](#), [74](#), [77](#), [80](#)
- [111] Rafael Valencia and Juan Andrade-Cetto. *Mapping, planning and exploration with Pose SLAM*. Springer, 2018. [74](#)
- [112] Paul J Besl and Neil D McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:1–18, February 1992. [74](#), [79](#)
- [113] Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *2011 IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, 2011. [74](#), [76](#), [79](#), [105](#)
- [114] T Whelan, M Kaess, M Fallon, and H Johannsson. Kintinuous: Spatially extended kinectfusion. *CSAIL, MIT*, 2012. [76](#)
- [115] Thomas Whelan, Stefan Leutenegger, Renato Salas Moreno, Ben Glocker, and Andrew Davison. ElasticFusion: Dense SLAM Without A Pose Graph. In *Robotics: Science and Systems*. Robotics: Science and Systems Foundation, July 2015. [76](#)
- [116] T Whelan, R F Salas-Moreno, B Glocker, A J Davison, and S Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *International Journal of Robotics Research*, September 2016. [96](#), [99](#)
- [117] Richard A Newcombe, Dieter Fox, and Steven M Seitz. DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 343–352. IEEE, 2015. [74](#), [76](#), [79](#)
- [118] 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, April 2007. [74](#)

- [119] G Klein and D Murray. Parallel Tracking and Mapping for Small AR Workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007. [77](#), [80](#), [89](#)
- [120] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast Semi-Direct Monocular Visual Odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22. IEEE, 2014. [74](#), [77](#), [80](#)
- [121] Raul Mur-Artal, J M M Montiel, and Juan D Tardos. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. [74](#), [77](#), [80](#), [89](#), [96](#), [99](#), [105](#)
- [122] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Dense Visual SLAM for RGB-D Cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2100–2106. IEEE, 2013. [74](#), [76](#), [80](#), [89](#)
- [123] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Robust odometry estimation for RGB-D cameras. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3748–3754. IEEE, 2013. [76](#)
- [124] Daniel Gutierrez-Gomez, Walterio Mayol-Cuevas, and J J Guerrero. Dense RGB-D Visual Odometry Using Inverse Depth. *Robotics and Autonomous Systems*, 75:571–583, 2016.
- [125] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision*, pages 834–849, Switzerland, 2014. Springer International Publishing. [77](#), [80](#)
- [126] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. DTAM: Dense Tracking and Mapping in Real-Time. In *IEEE International Conference on Computer Vision*, pages 2320–2327. IEEE, 2011. [74](#), [77](#), [80](#)
- [127] Felix Endres, Jurgen Hess, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. 3-D Mapping With an RGB-D Camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2014. [74](#), [77](#), [96](#), [99](#), [105](#), [118](#)
- [128] Alejo Concha, Giuseppe Loianno, Vijay Kumar, and Javier Civera. Visual-Inertial Direct SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1331–1338. IEEE, 2016. [74](#), [78](#), [81](#)
- [129] S Leutenegger, S Lynen, M Bosse, R Siegwart, and P Furgale. Keyframe-Based Visual-Inertial Odometry Using Nonlinear Optimization. *International Journal of Robotics Research*, 34(3):314–334, February 2015. [74](#), [78](#), [81](#)
- [130] T Laidlow, M Bloesch, W Li, and S Leutenegger. Dense RGB-D-Inertial SLAM with Map Deformations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6741–6748, 2017. [74](#), [78](#), [80](#), [81](#), [99](#)

- [131] Rahul Raguram, Jan-Michael Frahm, and Marc Pollefeys. A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In *Computer Vision – ECCV 2008*, pages 500–513. Springer, Berlin, Heidelberg, Berlin, Heidelberg, October 2008. [74](#)
- [132] G Costante, M Mancini, and P Valigi. Exploring Representation Learning With CNNs for Frame-to-Frame Ego-Motion Estimation. *IEEE Robotics and Automation Letters*, 1(1):18–25, 2016. [74](#), [77](#)
- [133] Vikram Mohanty, Shubh Agrawal, Shaswat Datta, Arna Ghosh, Vishnu Dutt Sharma, and Debashish Chakravarty. DeepVO: A Deep Learning approach for Monocular Visual Odometry. *arXiv.org*, page arXiv:1611.06069, 2016. [77](#)
- [134] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. *arXiv.org*, page arXiv:1704.03489, 2017. [78](#)
- [135] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem. *arXiv.org*, 2017. [79](#)
- [136] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Toward Geometric Deep SLAM. *arXiv.org*, page arXiv:1707.07410, July 2017. [74](#), [78](#)
- [137] T Whelan, M Kaess, H Johannsson, M Fallon, J J Leonard, and J McDonald. Real-Time Large-Scale Dense RGB-D SLAM with Volumetric Fusion. *International Journal of Robotics Research*, 34(4-5):598–626, 2015. [76](#), [79](#), [96](#), [118](#)
- [138] A Dai, M Nießner, M Zollhöfer, and S Izadi. BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Reintegration. *ACM Transactions on Graphics*, 36(3):1–18, 2017. [76](#), [79](#), [118](#)
- [139] Mariano Jaimez and Javier Gonzalez-Jimenez. Fast Visual Odometry for 3-D Range Sensors. *IEEE Transactions on Robotics*, 31(4):809–822, July 2015. [76](#)
- [140] Friedrich Fraundorfer and Davide Scaramuzza. Visual Odometry: Part II: Matching, Robustness, Optimization, and Applications. *Robotics & Automation Magazine*, 19(2):78–90, May 2012. [77](#)
- [141] Sheraz Khan and Dirk Wollherr. IBuILD: Incremental bag of Binary words for appearance based loop closure detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5441–5447. IEEE, 2015. [77](#)
- [142] Jakob Engel, Jorg Stuckler, and Daniel Cremers. Large-Scale Direct SLAM with Stereo Cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942. IEEE, 2015. [77](#)

- [143] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, April 2017. 77, 105
- [144] Stephan Weiss and Roland Siegwart. Real-Time Metric State Estimation for Modular Vision-Inertial Systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4531–4537. IEEE, 2011. 78
- [145] Simon Lynen, Markus W Achtelik, Stephan Weiss, Margarita Chli, and Roland Yves Siegwart Ingénieur mécanicien. A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013. 78
- [146] Kurt Konolige, Motilal Agrawal, and Joan Solà. Large-Scale Visual Odometry for Rough Terrain. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. 78, 82
- [147] Anastasios I Mourikis and Stergios I Roumeliotis. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3565–3572. IEEE, 2015. 78
- [148] M Li and A I Mourikis. High-Precision, Consistent EKF-Based Visual-Inertial Odometry. *International Journal of Robotics Research*, 32(6):690–711, 2013. 78
- [149] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation. In *Robotics: Science and Systems*. Robotics: Science and Systems Foundation, July 2015. 78
- [150] L Ma, J M Falquez, S McGuire, and G Sibley. Large scale dense visual inertial slam. *Field and Service Robotics*, 2016. 78
- [151] Nikolas Engelhard, Felix Endres, Jurgen Hess, Jurgen Sturm, and Wolfram Burgard. Real-time 3D visual SLAM with a hand-held RGB-D camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, 2011. 80
- [152] Y Ji, A Yamashita, and H Asama. RGB-D SLAM Using Vanishing Point and Door Plate Information in Corridor Environment. *Intelligent Service Robotics*, 8(2):105–114, 2015. 80
- [153] Sammy Omari, Michael Bloesch, Pascal Gohl, and Roland Siegwart. Dense Visual-Inertial Navigation System for Mobile Robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2634–2640. IEEE, 2015. 80

- [154] A Makadia, A I Patterson, and K Daniilidis. Fully Automatic Registration of 3D Point Clouds. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR'06)*, pages 1297–1304. IEEE, 2006. [82](#)
- [155] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 573–580. IEEE, 2012. [92](#), [96](#)
- [156] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision*, pages 2564–2571. IEEE, 2011. [93](#)
- [157] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2161–2168. Ieee, 2006. [93](#)
- [158] Andrea Bonarini, Wolfram Burgard, Giulio Fontana, Matteo Matteucci, Domenico Giorgio Sorrenti, and Juan Domingo Tardos. Rawseeds: Robotics advancement through web-publishing of sensorial and elaborated extensive data sets. In *In proceedings of IROS*, volume 6, 2006. [93](#)
- [159] R Mur-Artal and J D Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, pages 1–8, 2017. [93](#), [96](#), [99](#), [115](#), [118](#)
- [160] Dorian Galvez-Lopez and Juan D Tardos. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012. [93](#)
- [161] Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2O: a General Framework for Graph Optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613. IEEE, 2011. [93](#), [110](#)
- [162] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on “Program Generation, Optimization, and Platform Adaptation”. [94](#)
- [163] Gaël Guennebaud and Benoît Jacob. Eigen v3. <http://eigen.tuxfamily.org>, 2010. [94](#)
- [164] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, pages 1–6. Kobe, Japan, 2009. [94](#)

- [165] Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1280–1286. IEEE, 2013. [94](#)
- [166] Stephanie Lowry, Niko Sunderhauf, Paul Newman, John J Leonard, David Cox, Peter Corke, and Michael J Milford. Visual Place Recognition: A Survey. *IEEE Transactions on Robotics*, 32(1):1–19, January 2016. [105](#)
- [167] Arren Glover, William Maddern, Michael Warren, Stephanie Reid, Michael Milford, and Gordon Wyeth. OpenFABMAP: an Open Source Toolbox for Appearance-Based Loop Closure Detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4730–4735. IEEE, May 2012. [106](#)
- [168] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: low-drift, robust, and fast. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181. IEEE, 2015. [106](#)
- [169] Mark W Mueller, Michael Hamer, and Raffaello D’Andrea. Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadcopter state estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1730–1736. IEEE, 2015. [106](#), [107](#)
- [170] T M Nguyen, A H Zaini, K Guo, and L Xie. An Ultra-Wideband-based Multi-UAV Localization System in GPS-denied environments. *imavs.org*, 2016. [106](#)
- [171] Anton Ledergerber, Michael Hamer, and Raffaello D’Andrea. A Robot Self-Localization System Using One-Way Ultra-Wideband Communication. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3131–3137. IEEE, October 2015. [106](#)
- [172] Benjamin Hepp, Tobias Nägeli, and Otmar Hilliges. Omni-directional person tracking on a flying robot using occlusion-robust ultra-wideband signals. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 189–194. IEEE, 2016. [106](#)
- [173] G Grisetti, R Kümmerle, H Strasdat, and K Konolige. g2o: A general Framework for (Hyper) Graph Optimization. Technical report, Technical report, 2011. [110](#)
- [174] Yanqing Chen, Timothy A Davis, William W Hager, and Sivasankaran Rajamanickam. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Transactions on Mathematical Software*, 35(3), October 2008. [111](#)
- [175] Ross Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision*, pages 1440–1448. IEEE, 2015. [123](#)

- [176] 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 and Machine Intelligence*, 39(6):1137–1149, 2017. [123](#), [132](#)
- [177] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex. *The Journal of physiology*, 160(1):106–154, 1962. [123](#), [125](#)
- [178] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968. [123](#), [124](#), [125](#)
- [179] Georgios Zoumpourlis, Alexandros Doumanoglou, Nicholas Vretos, and Petros Daras. Non-Linear Convolution Filters for CNN-Based Learning. In *IEEE International Conference on Computer Vision*, 2017. [124](#), [126](#), [128](#)
- [180] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear CNN models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457, 2015. [124](#)
- [181] Mathieu Blondel, Masakazu Ishihata, Akinori Fujino, and Naonori Ueda. Polynomial Networks and Factorization Machines: New Insights and Efficient Training Algorithms. In *International Conference on Machine Learning*, 2016. [127](#)
- [182] Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel Pooling for Convolutional Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3049–3058. IEEE, 2017. [124](#), [127](#), [128](#), [131](#)
- [183] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. [125](#)
- [184] Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. *arXiv.org*, December 2013. [125](#)
- [185] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9. IEEE, 2015. [125](#), [138](#)
- [186] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Research*, 2015. [125](#)

- [187] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely Connected Convolutional Networks. *CVPR*, pages 2261–2269, 2017. [125](#), [138](#), [139](#)
- [188] Saining Xie, Ross B Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. *CVPR*, pages 5987–5995, 2017. [125](#)
- [189] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. *arXiv.org*, page arXiv:1709.01507, September 2017. [126](#)
- [190] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial Transformer Networks. In *Conference on Neural Information Processing Systems*, 2015. [126](#)
- [191] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable Convolutional Networks. *ICCV*, 2017. [126](#)
- [192] João F Henriques and Andrea Vedaldi. Warped Convolutions: Efficient Invariance to Spatial Transformations. In *International Conference on Machine Learning*, 2017. [126](#), [129](#)
- [193] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic Routing Between Capsules. In *Conference on Neural Information Processing Systems*, December 2017. [126](#)
- [194] Weiyang Liu, Zhen Liu, Zhiding Yu, Bo Dai, Rongmei Lin, Yisen Wang, James M Rehg, and Le Song. Decoupled Networks. In *The Conference on Computer Vision and Pattern Recognition*, 2018. [126](#)
- [195] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018. [126](#)
- [196] Xiaodi Wang, Baochang Zhang, Ce Li, Rongrong Ji, Jungong Han, Xianbin Cao, and Jianzhuang Liu. Modulated convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 840–848, 2018. [126](#)
- [197] Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional Kernel Networks. In *Conference on Neural Information Processing Systems*, 2014. [126](#)
- [198] Nadav Cohen, Or Sharir, and Amnon Shashua. Deep SimNets. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4782–4791. IEEE, 2016. [126](#)
- [199] Liefeng Bo and Cristian Sminchisescu. Efficient Match Kernel between Sets of Features for Visual Recognition. In *Conference on Neural Information Processing Systems*, pages 135–143, 2009. [130](#)

- [200] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv.org*, December 2014. [131](#)
- [201] Yoav Goldberg and Michael Elhadad. splitsvm: fast, space-efficient, non-heuristic, polynomial kernel computation for nlp applications. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 237–240. Association for Computational Linguistics, 2008. [131](#)
- [202] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Kernel Descriptors for Visual Recognition. In *Conference on Neural Information Processing Systems*, 2010. [132](#)
- [203] David G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. [132](#)
- [204] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 886–893. IEEE, 2005. [132](#)
- [205] Peter Gehler and Sebastian Nowozin. On feature combination for multiclass object classification. In *2009 IEEE 12th International Conference on Computer Vision (ICCV)*, pages 221–228. IEEE, 2009. [132](#)
- [206] 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. In *NIPS Workshop*, 2017. [132](#), [134](#)
- [207] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988. [133](#)
- [208] Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Re, and Matei Zaharia. DAWNBench: An End-to-End Deep Learning Benchmark and Competition. *Training*, 100, 2017. [133](#), [138](#)
- [209] Alex J Smola and Bernhard Schölkopf. *Learning with kernels*, volume 4. Citeseer, 1998. [134](#)
- [210] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009. [137](#)
- [211] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep Networks with Stochastic Depth. In *European Conference on Computer Vision*, 2016. [137](#)
- [212] Abhimanyu Dubey, Otkrist Gupta, Pei Guo, Ramesh Raskar, Ryan Farrell, and Nikhil Naik. Improving fine-grained visual classification using pairwise confusion. *arXiv preprint arXiv:1705.08016*, 2017. [138](#)

- [213] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016. 139
- [214] Tong Xiao Sam Gross. fb.resnet.torch. <https://github.com/facebook/fb.resnet.torch>, 2016. 139
- [215] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011. 145
- [216] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009. 145
- [217] Feng Xia, Laurence T Yang, Lizhe Wang, and Alexey Vinel. Internet of things. *International Journal of Communication Systems*, 25(9):1101–1102, 2012. 146
- [218] C Burdea Grigore and P Coiffet. *Virtual reality technology*. London: Wiley-Interscience, 1994. 147
- [219] Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385, 1997. 147
- [220] Yuichi Ohta and Hideyuki Tamura. *Mixed reality: merging real and virtual worlds*. Springer Publishing Company, Incorporated, 2014. 147
- [221] Javier Andreu Perez, Fani Deligianni, Daniele Ravi, and Guang-Zhong Yang. Artificial intelligence and robotics. *arXiv preprint arXiv:1803.10813*, 2018. 147