Homography Estimation: From Geometry to Deep Learning

Rui Zeng

M.Eng., M.Sc., B.Sc.

PhD Thesis

submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

SAIVT Research Laboratory

School of Electrical Engineering and Computer Science

Science and Engineering Faculty

Queensland University of Technology

2019

Keywords

Computer vision, machine learning, deep learning, multiple view geometry, camera calibration, image registration, image processing, image stitching, panorama generation, homography, perspective transformation, projective geometry, feature extraction, feature fusion, perspective feature, saliency map, convolutional neural network, fully convolutional network, PFNet, autoencoder, perspective field, image classification, fine-grained classification, cuboid detection, region of interest pooling, object recognition, object detection, object verification.

Abstract

The homography is one of the most fundamental concepts in computer vision. It provides a geometric relationship for any two images of the same planar surface in space. Therefore, estimating homographies correctly among images is of great importance for understanding scene geometry, which can significantly improve the performance of many vision tasks.

Methods for homography estimation can be categorized into two types: geometric and deep learning based methods. Geometric methods aim to find geometrically-meaningful correspondences (e.g. points, lines, circles, etc.) across visual data and then match them to compute homographies. This kind of methods perform well when correspondences can be clearly detected and matched. However, searching for correspondences across different viewpoints can be time consuming, expensive, and problematic. The common pipeline used for geometric methods has a vast number of modules, which significantly increases the computational complexity. Furthermore, different methods need to be designed for different scenes in which the types of correspondences, viewpoints, illumination, and image properties are different. Last but not least, when visual data is severely contaminated, the performance of such methods degrade rapidly. For these reasons, deep learning methods for homography estimation have attracted interests in recent years.

Homography estimation using deep learning techniques inherits all the advan-

tages from deep models while developing its own wisdom regarding learning this geometry-based task. Existing methods for predicting homographies using deep models have many favourable characteristics such as: end-to-end learning, correspondence free, less computational complexity, etc. As a new research direction, to date numerous efforts have been made to develop more accurate and geometrically explainable models for homography estimation.

This thesis focuses on studying homography estimation from both geometric and deep learning perspectives. Therefore, the contributions in this thesis consist of two major areas of work. The first focuses on geometric homography estimation; and develops a more accurate and effective homography estimation system for sports scenes. The second area involves finding a more appropriate deep learning homography parameterization, which is geometrically interpretable; and investigating homographies in feature maps generated by CNNs.

Several novel techniques which can be used in the geometric homography estimation pipeline have been developed. A method is presented for efficiently computing the focal length for each frame within a video captured by a PTZ camera. An algorithm for saliency map generation has been proposed for sports scenes. A high quality panorama generation method is proposed for broadcast sports videos. A unified framework for camera calibration in broadcast sports videos has been proposed. In contrast to previous camera calibration algorithms which locally align each sports frame, the proposed system calibrates each frame globally by aligning them with the generated panorama.

Existing homography parameterizations for deep learning models typically use either a four-point vector or eight degrees of freedom directly as the network output. In this thesis, it is shown that the proposed perspective field (PF) significantly outperforms the aforementioned two parameterizations, while also having greater geometric meaning. This new parameterization has reduced computational

complexity and can be applied to any existing fully convolutional neural network.

This thesis takes a further step, using homographies with feature maps generated from deep learning models, to propose a geometry-constrained car recognition system. In contrast to previous works which focus on independent 2D views, the proposed system predicts the 3D bounding box of a car and generates its 3D feature representation using region of interest perspective (RoIPers) pooling layers, which process the 3D bounding box using homographies in feature space. Furthermore, the 3D bounding box generation is constrained through a novel vanishing point loss, which ensures that the bounding box is geometrically correct.

Contents

List of Tables

List of Figures

List of Abbreviations

2D	Two-dimensional
3D	Three-dimensional
3DPN	3D Perspective Network
MACE	Mean average corner error
CQ	Cuboid Quality
CNN	Convolutional neural network
DNN	Deep neural network
FFN	Feature Fusion Network
GN	Global Network
LSTM	Long Short-Term Memory
PCK	Percentage of Correct Keypoints
PF	Perspective Field
PFNet	Perspective Field Network
RANSAC	Random Sample Consensus
RMS	Root mean square
SIFT	Shift invariant
STN	Spatial Transformer Network
SURF	Speeded up robust feature

Statement of Original Authorship

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signed: QUT Verfied Signature

Date: 01/11/2019

Acknowledgments

First of all, I would like to express my appreciation to my family for their love and support throughout my life. They have made it possible for me to pursue a career path that is well suited to my interests, ambitions, and aptitude.

I would like to give big thanks to my principal supervisor, Prof. Sridha Sridharan, for offering me the opportunity to pursue a PhD degree in the Speech, Audio, Image, and Video Technology (SAIVT) research group at Queensland University of Technology (QUT). During the process of pursuing my PhD, Prof. Sridha has given me tons of kind advice and support, which has been crucially important for my PhD study and career development. In addition, Prof. Sridha offered strong support for visiting the world-class Australian Centre for Robotic Vision (ACRV) at Monash University, Melbourne, Australia.

I would like to express gratitude to my associate supervisors, Dr. Simon Denman and Prof. Clinton Fookes, for their expert advice and encouragement throughout the hard times of my PhD. Without their careful supervision, the finalization of this thesis would not be possible. Furthermore, please allow me to thank Dr. Simon Denman again, who continuously supports me by offering kind suggestions throughout my PhD. In addition, it is my honor to be co-supervised by Dr. Ruan Lakemond for some of the research projects. I would also like to thank Prof. Stuart Morgan for the opportunity this PhD program provided to me.

I would like to gratefully acknowledge the scholarships and sponsorships supported by Queensland University of Technology (QUT) and Australia Institute of Sport (AIS). I spent lots of unforgettable and memorable moments in QUT during my PhD study. Without the support from AIS, I could not finish my PhD in a timely manner.

During my PhD study, I obtained a valuable opportunity to visit the ACRV at Monash University for collaborate with their researchers. I want to thank Prof. Tom Drummond, Prof. Paul Bonnington, and Dr. Zongyuan Ge, who provided me this splendid opportunity, and gave me financial support to stay in Melbourne. I would like to especially acknowledge my appreciation for my friend and mentor Dr. Zongyuan Ge, for his ongoing support of my academic career and encouragement throughout my PhD.

I would like to thank panel members of my confirmation, final seminar, and thesis examiners for their efforts in providing helpful and constructive feedback to improve the quality of my research and thesis.

I am so fortune to be a member in SAIVT research group, which has so many kind and lovely colleagues. Also, I am happy to be a member of the school of electronic engineering and computer science (EECS), and I have enjoyed a lot of fun times in the social hub at level 12.

Rui Zeng

Queensland University of Technology

2019

Chapter 1

Introduction

Computer vision aims to teach computers how to observe and understand the world like a human being. One of the main tasks in computer vision is to gain a high-level understanding from scenes. Extracting meaningful information regarding three-dimensional structures and geometric relationships is critically helpful for achieving scene understanding. As one of the most fundamental concepts in computer vision geometry, the homography plays a key role in extracting basic geometry relationships across different viewpoints. Thus, a vast collection of computer vision research focuses on estimating homographies from vision scenes, and using them to improve the performance of other methods.

In order to estimate homographies properly from multiple viewpoints, numerous efforts have been made to propose more robust and accurate methods. These methods can be typically divided into two categories according to what techniques they are reliant on, i.e., geometric and deep learning based methods. The former aims to search for geometric correspondences (e.g. points, lines, circles, and other geometric patterns) across different viewpoints in visual data, and then estimate homographies. This kind of method is time consuming and problematic due to the

inevitable use of a correspondence detection module. This is because detecting correspondences between images is prone to errors due to variations in illumination and viewpoint. As a result, deep learning methods for homography estimation have been vigorously researched in recent years.

Homography estimation using deep learning techniques skips the correspondence detection module by using end-to-end architectures. While various deep learning models have been designed for more accurate homography prediction and reduced computational complexity, the homography parameterization used in existing deep learning models lacks a geometrically explainable meaning. If an interpretable geometry parameterization could be gleaned from deep learning models, it could be used to significantly improve the performance of models.

This PhD project aims to fill the gap between geometry and deep learning homography estimation methods. To address the aforementioned issues, this thesis proposes efficient and effective algorithms to estimate homographies using both geometric and deep learning techniques. In addition, the works done in this project take a further step in introducing projective geometry into feature maps generated from CNNs in a "geometry meets deep learning" style.

1.1 Motivation

The basic geometric relationships provided by homographies within scenes are crucial to achieve full scene understanding, which benefits many computer vision tasks significantly. Therefore, the main motivation of this thesis is that researching homography estimation deeply and developing specific algorithms for various scenes, could significantly benefit the computer vision community.

In the past decade, the dominant homography estimation approach is to match

geometric correspondences across different viewpoints and then derive homographies using projective geometry knowledge. Geometry-based methods have many merits such as being deterministic, explainable, high accuracy (assuming good correspondences), and easy error analysis. This type of methods performs well if correspondences can be clearly detected and matched. However, they are prone to problems in some domains such as sports videos, where every frames contain a significant number of dynamic pixels (e.g. moving players, spectator areas, moving objects, etc.) that impede correspondence detection. Therefore, solving homography estimation for frames with a large number of high dynamic pixels is one of the thesis motivations.

Deep learning based methods have a big advantage over geometry methods because that the use of end-to-end architectures removes the need for correspondence detection. Therefore, deep learning models for homography estimation have more robustness when being applied to images that are difficult to extract correspondences from. Deep learning models implicitly encode latent knowledge of projective geometry, though do not explicitly produce geometrically meaningful outputs. If geometry explainability could be gleaned from deep learning models, it can be used to significantly improve the measurement and accuracy of computed homographies. Therefore, studying geometrically explainable models for homography estimation is another motivation in this thesis.

So far, deep learning homography estimation and projective geometry have been studied separately and orthogonally. The methods combining the benefits of both are yet well explored. The final motivation of this thesis is to fill the gap between geometry and deep learning regarding homography estimation, i.e., develop a model to jointly use projective geometry and deep learning correctly.

1.2 Research Objectives and Scope

1.2.1 Objectives

The primary research objectives for this thesis are to:

1. Develop a novel geometry-based homography estimation and decomposition method that can be used on video clips captured in sports scenes.
2. Develop a more appropriate homography parameterization, which is geometrically explainable and can be elegantly applied to any existing fully convolutional neural networks.
3. Develop a deep learning model, which combines the benefits of both deep learning and projective geometry.

The first objective is motivated by the need to calibrate cameras in scenes capturing sporting events. Most sports videos are captured by PTZ cameras in noisy environments, which contain a large number of factors that adversely affect correspondence detection. These factors include illumination changes, viewpoint differences, scarce overlapping regions across frames, dynamic pixels introduced by moving players and spectator areas, etc. Thus, existing geometric homography estimation methods work poorly due to the lack of consideration of these factors in sports videos.

The second objective is motivated by the urgent requirement for geometrically explainable deep learning models. Existing widely used homography parameterizations for deep learning models do not consider the geometry properties of the homography at all, and as such it is hard to further improve performance of these models.

The third objective is motivated by the existing gap between projective geometry and deep learning techniques. They are studied separately and orthogonality. Developing an algorithm which leverages all the information regarding both geometry and deep learning is invariably helpful for the computer vision community.

1.2.2 Scope

Existing geometry methods for homography estimation do not take into account the properties of sports videos and are prone to poor performance when applied them in sports scenes. The challenges in sports videos largely come from variations of illumination, dynamic pixels, blur, motion, etc. To overcome these limitations, this thesis focuses on homography estimation for sports videos. Chapter 4 presents a homography estimation system for pitch-based sports videos, which lack reliable correspondences for detection. Based on this work, Chapter 5 calibrates each frame in a broadcast sports video using homography decomposition.

It is expected that deep learning models could learn projective geometry explicitly. To explain deep learning based homography estimation properly, a novel homography parameterization, which models the essence of the homography - pixel-to-pixel bijection, is proposed. This work is outlined in Chapter 6

To combine the benefits of both projective geometry and deep learning, a car recognition system has been developed. It predicts the 3D bounding box of a car using a novel network, and then homographies are applied on the predicted 3D bounding box to generate normalized CNN feature maps, which are shown to enhance car feature representation. This work can be found in Chapter 7

1.3 Thesis Structure

Chapter 2

This chapter introduces the basic concepts in projective geometry and presents preliminaries for homography estimation. The 2D projective linear group, which is composed of isometric, similarity, affine, and projective transformation, is introduced in a hierarchical manner. Subsequently, the pinhole camera model is presented to establish relationships between the 2D image plane and a real world coordinate system. Afterwards, three different kinds of homography that arise from three situations are described. This chapter aims to introduce what a homography is, and why homographies are important in computer vision.

Chapter 3

This chapter reviews literature regarding both geometric and deep learning homography estimation algorithms. In addition, the advantages and disadvantages of them are discussed in detail.

Chapter 4

Homography estimation is a preliminary step in sports analytics which enables us to transform player positions to standard playing area coordinates. While many homography estimation systems work well when the visual content contains sufficient clues, such as a key frame; calibrating without such information, as may be needed when processing footage captured by a coach from the sidelines or stands, is challenging. In this chapter an innovative automatic camera calibration system, which does not make use of any key frames, is presented for sports

analytics. Apart from this, a camera vertical axis normalization system, which is used to rotate a camera to a viewpoint which fits to human visual perspective, is proposed.

Chapter 5

This chapter takes a further step towards camera calibration using homography decomposition. In this chapter, an innovative method is proposed to calibrate broadcast basketball videos using key frames generated from the video itself instead of searching for them. This chapter showcases the proposed method using NBA broadcast basketball videos, where many video clips lack key frames and specific geometric features due to a highly-customized playing area.

Chapter 6

This chapter proposes a conceptually simple, reliable, and general framework for homography estimation. In contrast to existing works, this method formulates the problem as predicting a perspective field (PF) which models the essence of the homography - pixel-to-pixel bijection. The PF is naturally learned by the proposed fully convolutional residual network, PFNet, to keep the spatial order of each pixel. Moreover, since every pixels' displacement can be obtained from the PF, it enables robust homography estimation using dense correspondences.

Chapter 7

This chapter presents a novel learning framework for vehicle recognition from a single RGB image. The proposed unified framework learns a joint representation

of the 2D global texture and 3D-bounding-box in a mutually correlated and reinforced way. These two kinds of feature representations are combined by a novel fusion network, which predicts the vehicle's category. A 2D global feature is extracted using an off-the-shelf detection network, where the estimated 2D bounding box assists in finding the region of interest (RoI). With the assistance of the RoI, the 3D bounding box and its corresponding features are generated in a geometrically correct way using a novel 3D perspective Network (3DPN). The 3DPN consists of a convolutional neural network (CNN), a vanishing point loss, and RoI perspective layers. The CNN regresses the 3D bounding box under the guidance of the proposed vanishing point loss, which provides a perspective geometry constraint. Thanks to the proposed RoI perspective layer, the variation caused by viewpoint changes is corrected via the estimated geometry, enhancing the feature representation.

Chapter 8

The final chapter concludes the thesis, and considers possible future research directions.

1.4 Original Contributions of Thesis

The presented thesis is based on the original contributions from the works listed below, arranged according to the order of the chapters:

Preliminary

A comprehensive review of homography estimation from both the geometric and deep learning perspective is presented. The review first introduces basic concepts and preliminaries in projective geometry. Then three different types of homography are derived by relating image and real world coordinates in three different ways using a pinhole camera model. Furthermore, the common algorithmic framework for homography estimation using deep learning is reviewed. We analyzed the main disadvantages and challenges faced by existing state-of-the-art methods in details and propose a efficient and elegant solution for deep-learning-based homography estimation task.

Calibrating Cameras in Poor-conditioned Pitch-based Sports Games

A novel homography estimation methodology for pitch-based sports video is presented. Without relying on searching for key frames, our method generates the panorama of the given sports video, which serves as the key frame used to estimate homographies among other frames.

Key Frame Generation for Camera Calibration in Broadcast Sports Videos

A novel camera calibration system based on homography estimation and decomposition is presented. To this end, the following contributions are made:

1. A unified framework for camera calibration in broadcast sports video. Instead of directly searching for key frames for registration, the proposed system generates them from highly dynamic frames using the proposed sports

panorama technique.

2. A novel camera calibration method for sports video frames captured by PTZ cameras. Camera intrinsic, extrinsic, and distortion parameters can be estimated and jointly optimized by solving the proposed optimization function, which is designed under the constraints of sports scenes.

3. A novel image saliency algorithm for sports video frames is introduced by taking into account sports video properties. Our qualitative results show that the proposed method significantly outperforms existing state-of-the-art methods in terms of sports scenes.

Rethinking Planar Homography Using Perspective Fields

To study geometrically explainable deep learning models for homography estimation, the thesis proposes a perspective field, which encodes the essence of the homography elegantly and geometrically.

1. A new parameterization for homography estimation is proposed. The new parameterization is geometrically interperetable, unlike existing parameterizations. Furthermore, it can be easily applied to any existing FCN to further improve accuracy.

2. The Perspective field network (PFNet) is proposed for homography estimation. Endowed with the proposed novel deconvolution blocks, PFNet allows for upsampling and predicting homographies efficiently. Thanks to the proposed fully convolutional architecture, the network has fewer hyperparameters while greatly boosting the performance compared with existing state-of-the-art methods.

Geometry-constrained Car Recognition Using a 3D Perspective Network

To fill the gap between geometry and deep learning, an end-to-end network is proposed for car recognition. The contributions of this work are summarized as follows:

1. A unified network architecture for vehicle recognition which takes full advantage of the 2D and 3D representations is presented. To the best of our knowledge, our method is the first work which extracts a meaningful feature representation from the 3D bounding box to enhance vehicle recognition.

2. A 3D bounding box predictor, i.e. 3DPN, is proposed for vehicle recognition to use 3D information in a meaningful and correct manner, without the challenge of requiring a 3D CAD model.

3. A new parameterized CNN pooling layer termed RoI Perspective pooling is proposed. It grants the CNN the adaptive transformation modeling capability to normalize vehicle viewpoints in the feature space given the 3D bounding box.

4. A geometrically interpretable loss (vanishing point loss) is introduced. It elegantly enforces the consistency of the predicted 3D bounding box to improve regression accuracy.

1.5 Publications

The following publications are the outcomes of the PhD research:

1.5.1 Journals

1. **Rui Zeng**, Simon Denman, Ruan Lakemond, Sridha Sridharan, Clinton Fookes, Stuart Morgan, "Key Frame Generation for Camera Calibration in Broadcast Basketball Video," submitted to IEEE Transaction on Multimedia.

1.5.2 Conferences

2. **Rui Zeng**, Zongyuan Ge, Simon Denman, Sridha Sridharan, Clinton Fookes, "Geometry-constrained Car Recognition Using a 3D Perspective Network," submitted to *AAAI Conference on Artificial Intelligence (AAAI), 2020.*

3. **Rui Zeng**, Simon Denman, Sridha Sridharan, Clinton Fookes, "Rethinking Planar Homography ," in *Asian Conference on Computer Vision (ACCV), 2018.*

4. **Rui Zeng**, Ruan Lakemond, Simon Denman, Sridha Sridharan, Clinton Fookes, "Calibrating cameras in poor-conditioned pitch-based sports games," in *International Conference on Acoustic, Speech, and Signal Processing (ICASSP), 2018.*

5. **Rui Zeng**, Ruan Lakemond, Simon Denman, Sridha Sridharan, Clinton Fookes, Stuart Morgan, "Vertical axis detection for sport video analytics," in *Digital Image Computing: Techniques and Applications (DICTA), 2016.*

6. Jie Xie, **Rui Zeng**, Changliang Xu, Jinglan Zhang, Paul Roe, "Multi-Label Classification of Frog Species via Deep Learning," in *International Conference on E-Science (ICES), 2017.*

7. Osman Tursun, **Rui Zeng**, Simon Denman, Sabesan Sivipalan, Sridha Sridharan, Clinton Fookes, "MTRNet: A Generic Scene Text Eraser", in

International Conference on Document Analysis and Recognition (ICDAR), 2019.

8. Dung Nguyen, Sridha Sridharan, Simon Denman, **Rui Zeng**, Son Tran, Clinton Fookes, "Deep Cross-Modal Learning For Emotion Recognition," submitted to *AAAI Conference on Artificial Intelligence (AAAI), 2020.*

9. Yunyan Xing, Zongyuan Ge, **Rui Zeng**, Dwarikanath Mahapatra, Jarrel Seah, Meng Law, Tom Drummond, "Data Augmentation for Pulmonary Pathology Localisation with Pairwise Adversarial Learning," in *Medical Image Computing and Computer Assisted Interventions (MICCAI), 2019.*

Chapter 2

Homography

In the field of computer vision, the homography is one of the most fundamental concepts, which encompasses camera geometry (assuming a pinhole camera model) and formations of visual data through a 3D to 2D projection.

This chapter introduces the preliminaries of projective geometry that are required to understand the works related to homography estimation covered in this thesis. The algebra of projective geometry introduced in this chapter serves to formally express the concept that any two images of the same planar surface in space are related by a homography. The subject will be treated in a manner that provides a computer vision perspective to formulate and solve the problem of homography estimation in camera calibration and feature learning representations. For a greater and more detailed interpretation of homographies, interested readers are referred to [54].

Section 2.1 introduces the geometric ideas and notation of the homogeneous coordinate system, which is widely used in projective geometry. Subsequently, several geometric transformations, including *isometric*, *similarity*, *affine*, and

homography, are described in a hierarchical manner in Section 2.2. Section 2.3 defines a pinhole camera model. In Section 2.4, three situations in which a homography arise are described. The chapter is concluded in Section 2.5.

2.1 Homogeneous Coordinate System

Homogeneous coordinates are a system of coordinates used in projective geometry to introduce points at infinity when compared to Euclidean geometry. Therefore, the real projective plane is able to be thought of as the Euclidean plane with additional points at infinity added.

Thus, given a 2D point $\mathbf{p}$ on the Euclidean plane, for any non-zero real number k, the set of homogeneous coordinates for $\mathbf{p}$ is represented as a tuple:

$$\mathbf{p} = (kx_{\mathbf{p}}, ky_{\mathbf{p}}, k). \tag{2.1}$$

In particular, when $k = 1$, $(x_{\mathbf{p}}, y_{\mathbf{p}}, 1)$ is the homogeneous coordinate for $\mathbf{p}$ itself. When $k = 0$, $(x_{\mathbf{p}}, y_{\mathbf{p}}, 0)$ represents the infinity point corresponding to a specific direction. The original 2D point on the Euclidean plane can be recovered simply by dividing the first two elements by k (assuming that $k \neq 0$), i.e., $(\frac{x_{\mathbf{p}}}{1}, \frac{y_{\mathbf{p}}}{1})$. More generally, $(\frac{x_{\mathbf{p}}}{0}, \frac{y_{\mathbf{p}}}{0})$ can be obtained when $k = 0$. While it has no real meaning in reality, this form can be used meaningfully in algebra tools and is critical for camera calibration.

A standard Euclidean form of a line equation can be represented as:

$$ax + by + c = 0, \tag{2.2}$$

where a, b, and c are the coefficients of the line equation respectively. By observing Equation 2.2, one can find that the line equation is invariant to scaling and hence

it can be rewritten as:

$$akx_{\mathbf{p}} + bky_{\mathbf{p}} + ck = 0. \tag{2.3}$$

To represent the line equation in projective geometry, Equation 2.3 can be further represented as,

$$\mathbf{l}^{\top}\mathbf{p} = 0, \tag{2.4}$$

where $\mathbf{l} = (a, b, c)$ is the line equation, $\mathbf{p} = (kx_{\mathbf{p}}, ky_{\mathbf{p}}, k)$ is a set of homogeneous coordinates. In projective geometry, a line representation is the same as that of the homogeneous coordinate of a point. Given any line on a projective plane, there exists infinite number of points at infinity that lie on this line. All the points at infinity lie on a line, which is also called as the line at infinity. A line at infinity is treated the same as other lines mathematically.

The homogeneous coordinates can be easily extended from 2D projective geometry $\mathbb{P}^2$ to the 3D $\mathbb{P}^3$ by adding an extra component to a 3D Cartesian coordinate system. The homogeneous coordinates of a 3D point $\mathbf{p}'$ are therefore represented as $\mathbf{p}' = (kx_{\mathbf{p}'}, ky_{\mathbf{p}'}, kz_{\mathbf{p}'}, k)$. Similarly, a line equation in $\mathbb{P}^3$ can be represented as $\mathbf{l} = (a, b, c, d)$.

Homogeneous coordinates have a plenty of convenient characteristics:

1. Consistent matrix operation. The transformations in projective geometry can be represented as a simple matrix multiplication between the transformations and the homogeneous coordinates of the point. For example, in homogeneous coordinates, a similarity transformation performed on a point $\mathbf{p} = (x_{\mathbf{p}}, y_{\mathbf{p}})$ is presented as $\mathbf{p} = s\mathbf{p} + \mathbf{t}$. However, a similarity transformation is able to formulated in a more concise and consistent way than that in Equation 2.6.

2. Points at infinity and lines at infinity are able to be represented simply by adding an extra coordinate. Eculidean coordinate systems cannot represent points and lines at inifinity in a meaningful way.

The linear transformations in projective geometry will be discussed in the following section in a hierarchical way.

2.2 Transformations

2.2.1 Isometry

Isometries are transformation of the plane $\mathbb{R}^2$ that preserve Euclidean distance, i.e., length and area remain the same after *isometry* transformation. An Isometry is only composed of a 2D rotation and a translation vector, and hence only has three degrees of freedom. To define an isometry transformation, three parameters must be specified. Considering each point correspondence provides two equations, we need at least two point correspondences to compute the isometry transformation. An isometry is able to represented as:

$$\mathbf{p}' = \mathbf{H}_{\mathrm{I}}\mathbf{p} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{p}, \tag{2.5}$$

where $\mathbf{R}$ is a 2×2 rotation matrix, $\mathbf{t}$ is a translation vector and $\mathbf{0}^\top$ is a row of 2 zeros.

2.2.2 Similarity

A *similarity* transformation is nothing but an isometry composed with an isotropic scaling. By entangling a scaling factor into an isometry, the similarity has the matrix representation

$$\mathbf{p}' = \mathbf{H}_{\mathrm{S}}\mathbf{p} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{p}, \tag{2.6}$$

where s is a scalar and represents the isotropic scaling factor. This factor adds an additional degree of freedom and hence a similarity transformation has 4 degrees of freedom. Due to scaling along each axis isotropically, length and area are no longer invariant. However, the ratio of distance, angles, and the circular points (please refer to [54] for details) are preserved under similarity transformations, which is also known as an "shape" preservation transformation. To compute a similarity transform correctly, we need at least two point correspondences.

2.2.3 Affine

Unlike a similarity transformation which is composed of a 2×2 rotation and a isotropic scaling, an *affine* transformation replaces them with a general 2×2 non singular matrix $\mathbf{A}$. Therefore, an affine transformation can be written as:

$$\mathbf{p}' = \mathbf{H}_{\mathrm{A}}\mathbf{p} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^{\top} & 1 \end{bmatrix} \mathbf{p}. \tag{2.7}$$

Comparing with a similarity transform, an affine transformation has six degrees of freedom corresponding to six matrix elements, including four in $\mathbf{A}$ and two in $\mathbf{t}$. Since each point correspondence contributes to solve two degrees of freedom, three point correspondences are needed to compute an affine transformation.

A geometric way to understand the role of $\mathbf{A}$ in an affine transformation is to decompose it as a chain multiplication of several transformations, i.e.,

$$\mathbf{A} = \mathbf{R}(\theta)\mathbf{R}(-\phi)\mathbf{D}\mathbf{R}(\theta), \tag{2.8}$$

where $\mathbf{R}(\theta)$ and $\mathbf{R}(\phi)$ are rotations by θ and ϕ respectively, and $\mathbf{D}$ is a 2×2 diagonal matrix which represents a non-isotropic scaling. $\mathbf{D}$ can be written in a block form:

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \tag{2.9}$$

where λ_1 and λ_2 are the two non-isotropic scaling factors. By observing 2.8, an affine transformation could be understood as the concatenation of four independent operations: 1. a rotation by θ; 2. a scaling by λ_1 and λ_2 along x and y axis respectively; 3. a rotation back by ϕ; 4. a final rotation by θ. Compared to a similarity transform, the only difference is the non-isotropic scaling introduced by $\mathbf{D}$. One important thing worth noting is that an affine transform only scales axes in orthogonal directions, orientated at a particular angle. In an affine transformation, parallelism, ratio of areas, ratio of lengths on collinear or parallel lines, linear combinations of vectors, and the line at infinity are preserved.

2.2.4 Homography

A homography is the most general formulation of linear transformations in projective geometry, and is also known as a projective transformation. The term "homography" and "projective transformation" are interchangeable throughout the entire thesis. It is a general non-singular linear transformation of homogeneous coordinates, which generalizes an affine transformation by adding two extra two degrees of freedom. The block form of a homography can be represented as:

$$\mathbf{p}' = \mathbf{H}\mathbf{p} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^\top & v \end{bmatrix} \mathbf{p}, \tag{2.10}$$

where the vector $\mathbf{v} = (v_1, v_2)^\top$. Although a homography has nine elements, it can be defined by only eight parameters because the last element v is an arbitrary non-zero scale. A homography between two planes can be computed using four point correspondences, with no three coordinates collinear on either plane.

A helpful way to understand the geometric effects of a homography is to decompose

it into a chain of transformations in a low-to-high hierarchical manner, i.e.,

$$\mathbf{H} = \mathbf{H}_{\mathrm{S}}\mathbf{H}_{\mathrm{A}}\mathbf{H}_{\mathrm{P}} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^\top & v \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^\top & v \end{bmatrix}, \quad (2.11)$$

where the matrices $\mathbf{H}_{\mathrm{S}}$, $\mathbf{H}_{\mathrm{A}}$, and $\mathbf{H}_{\mathrm{P}}$ are a similarity, affine, and projective transformation (As indicated by the subscripts S, A, P) respectively. $\mathbf{A}$ is an non-singular matrix given by $\mathbf{A} = s\mathbf{R}\mathbf{K} + \mathbf{t}\mathbf{v}^\top$, where $\mathbf{K}$ is an upper-triangular matrix normalized as $|\mathbf{K}| = 1$. Please note that this decomposition is valid only if $\mathbf{v}$ does not equal 0, and is unique if $s > 0$. Even if a homography generalizes an affine transform, it does not retain any invariants which are preserved by the affine transformation. Despite this, a homography preserves special invariant properties, such as concurrency, collinearity, order of contact, inflections, tangent discontinuities and ratio of lengths.

2.3 Pinhole Camera model and Perspective Projection

Up to this point, plane to plane transformations have been introduced in the hierarchy. However, all of these transformations are typically performed on a 2D image and do not relate image planes with a 3D Cartesian coordinate system in the real world. In this section, we first introduce the pinhole camera model to present the relationship between a 3D point in the real world and its corresponding 2D point in the image plane. Afterwards, we bridge the gap between a homography and a real world coordinate system.

Cameras project rays of light from the 3D world onto a 2D image plane within the camera, which is known as the process of camera projection. The most common model used to formulate this process is the pinhole camera model, where the

camera aperture is described as a point. All projection rays pass through this camera aperture, i.e., the camera center, and generate the corresponding *image*s of the rays on the image plane. In reality, cameras have a significantly larger aperture and a lens that suffers from distortion. The distortion introduced by the lens changes the direction of the rays of lights by a non-linear mapping, and therefore images captured from camera usually use distorted coordinates. However, a high quality lens may produce an image that is very similar to an equivalent pinhole camera.

A ideal pinhole camera can be represented as:

$$\mathbf{p} = \mathbf{K}\,[\mathbf{R}|\mathbf{T}]\,\mathbf{p}', \tag{2.12}$$

where $\mathbf{p}' = (X_{\mathbf{p}'}, Y_{\mathbf{p}'}, Z_{\mathbf{p}'}, 1)^\top$ is a 3D point represented in the homogeneous coordinate system of the real world, $\mathbf{K}$ is the 3×3 camera intrinsic matrix, $\mathbf{R}$ is a 3×3 3D rotation matrix, $\mathbf{T}$ is a translation vector in the 3D real world coordinate system. $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{T}$ are respectively of the forms:

$$\mathbf{K} = \begin{bmatrix} f_x m_x & s & o_x \\ & f_y m_y & o_y \\ & & 1 \end{bmatrix}, \tag{2.13}$$

$$\mathbf{R} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix}, \tag{2.14}$$

$$\mathbf{T} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^\top, \tag{2.15}$$

where f_x and f_y are the focal length in terms of the x and y axis respectively, o_x and o_y are the camera center, which is also known as the principal point. In most cases, the pixels in the CCD are square such that $m_x = m_y$ where m_x and m_y are the number of pixels per unit distance in image coordinates. To make the notation more convenient, the pairs of numbers f_x, f_y and m_x, m_y are merged

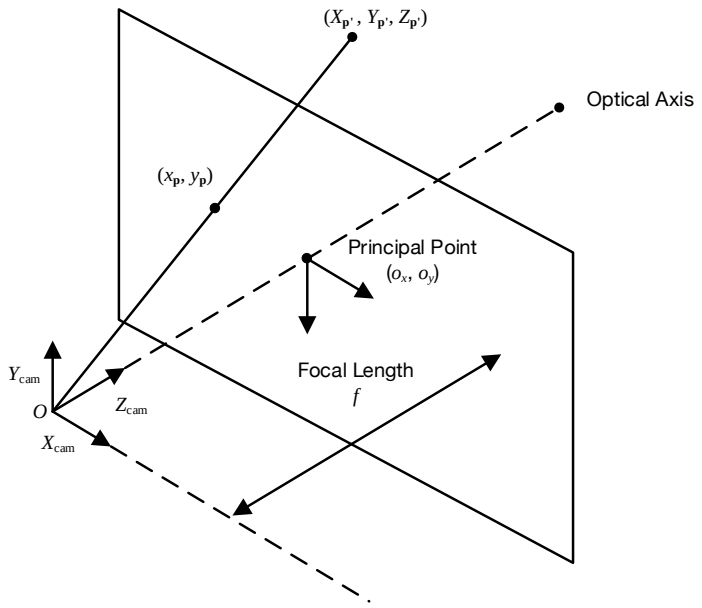


Figure 2.1: An example of the pinhole camera model.

into one number and we use f_* and m_* to represent them, where $*$ stands for one of x and y. s is the skew parameter and is very small value (typical 10^{-4}) for most cameras.

The formulation of an image, which projects a 3D ray into the pinhole camera model and generates a 2D image point is also known as perspective projection. Figure 2.1 illustrates the pinhole camera model where a 3D point in the real world is imaged as a 2D point in the image plane.

2.4 Three Types of Homographies

In this section, we introduce three common situations which arise homography estimation tasks.

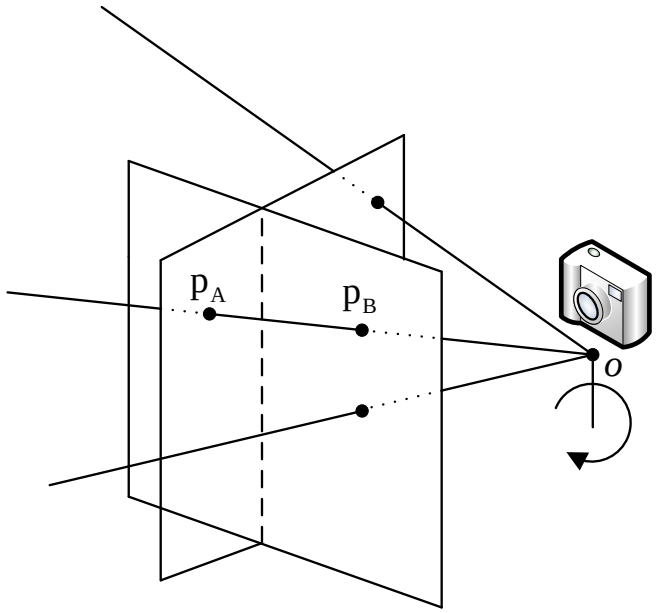


Figure 2.2: An example of the homography generated from different planes in a rotated camera.

2.4.1 A Rotated-camera with a Fixed Camera Center

In the first example we consider a rotating camera, and the correspondences fall on planes which are obtained by the camera rotation. These planes can also be related using a homography. Suppose that there are two planes A and B, which are obtained while the camera o is rotating. A ray projected from o goes through A and B and generates the points $\mathbf{p}_\mathrm{A}$ and $\mathbf{p}_\mathrm{B}$ respectively. To simply further analysis, the real world coordinate system is set to the same as the camera coordinate system. Since the camera o only rotates around a fixed axis and there is no translation, the points $\mathbf{p}_\mathrm{A}$ and $\mathbf{p}_\mathrm{B}$ can be represented as:

$$\mathbf{p}_\mathrm{A} = \mathbf{K}\,[\mathbf{R}_\mathrm{A}|\mathbf{0}]\,\mathbf{p}' \tag{2.16}$$

and

$$\mathbf{p}_\mathrm{B} = \mathbf{K}\,[\mathbf{R}_\mathrm{B}|\mathbf{0}]\,\mathbf{p}', \tag{2.17}$$

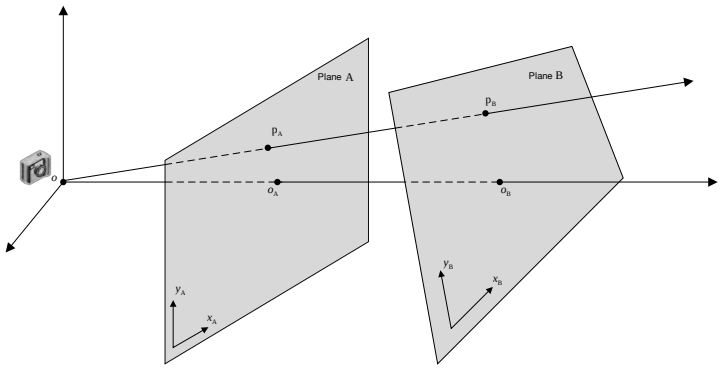


Figure 2.3: The illustration of a homography generated by projecting different planes into a fixed camera. The points $\mathbf{p}_\mathrm{A}$ and $\mathbf{p}_\mathrm{B}$, which are projected using the same line from the camera center o, can be related using the homography $\mathbf{H}$.

respectively. $\mathbf{R}_\mathrm{A}$ and $\mathbf{R}_\mathrm{B}$ are the rotation matrices of camera o in two different poses. $\mathbf{0}$ is a translation vector filled with zeros. Therefore, the homography $\mathbf{H}$ which maps $\mathbf{p}_\mathrm{A}$ to $\mathbf{p}_\mathrm{B}$ can be represented by:

$$\mathbf{p}_\mathrm{B} = \mathbf{H}\mathbf{p}_\mathrm{A}, \tag{2.18}$$

where $\mathbf{H} = \mathbf{K}\mathbf{R}_\mathrm{B}\mathbf{R}_\mathrm{A}^{-1}\mathbf{K}^{-1}$. Figure 2.2 shows a situation where a homography is generated from different planes captured by a rotating camera.

2.4.2 Different Planes Projected in a Fixed Camera

One common situation that results in a homography is when different planes are projected into a fixed camera. Suppose that two different planes, A and B, are placed in the front of the camera, where the camera center is denoted by o. A ray goes through o and projects two points, $\mathbf{p}_\mathrm{A}$ and $\mathbf{p}_\mathrm{B}$ on A and B respectively. Figure 2.3 illustrates the process of homography generation in this situation. A fix camera with two different planes can be regarded as a pan-tilt-zoom camera situation which is described in Section 2.4.1.

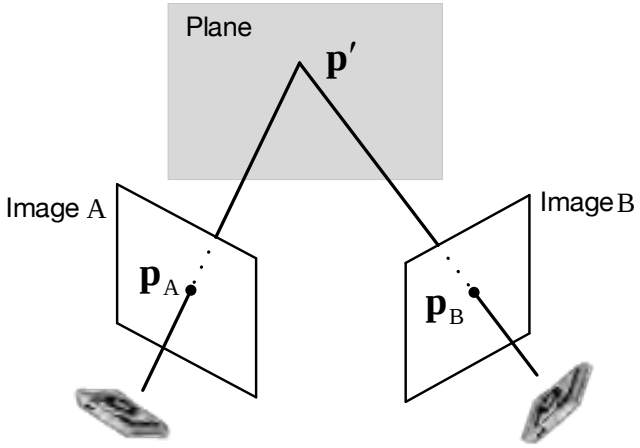


Figure 2.4: An example of the homography obtained from one plane projected by two different cameras.

2.4.3 The Same Plane Captured from Different Cameras

Another common situation which introduces the homography estimation problem is shown in Figure 2.4. One can see that a plane in the real-world coordinate system is captured by two different cameras. The point $\mathbf{p}'$ falls on the plane is projected onto two different imaging points $\mathbf{x}_{\mathrm{A}}$ and $\mathbf{x}_{\mathrm{B}}$ which lie on the image planes A and B respectively. By setting the xy plane of the real-world coordinate system equal to the plane where $\mathbf{p}'$ falls, $\mathbf{p}_{\mathrm{A}}$ and $\mathbf{p}_{\mathrm{B}}$ can be represented as:

$$\mathbf{p}_{\mathrm{A}} = \mathbf{K}_{\mathrm{A}} \left[\mathbf{R}_{\mathrm{A}} | \mathbf{T}_{\mathrm{A}}\right] \begin{bmatrix} X_{\mathbf{p}'} \\ Y_{\mathbf{p}'} \\ 0 \\ 1 \end{bmatrix}, \tag{2.19}$$

and

$$\mathbf{p}_{\mathrm{B}} = \mathbf{K}_{\mathrm{B}} \left[\mathbf{R}_{\mathrm{B}} | \mathbf{T}_{\mathrm{B}}\right] \begin{bmatrix} X_{\mathbf{p}'} \\ Y_{\mathbf{p}'} \\ 0 \\ 1 \end{bmatrix}, \tag{2.20}$$

respectively. Since the third element of $\mathbf{p}'$ is zero in the new real-world coordinates, Equation 2.19 and 2.19 can be further expressed as:

$$\mathbf{p}_{\mathrm{A}} = \mathbf{K}_{\mathrm{A}} \begin{bmatrix} r_{00,\mathrm{A}} & r_{01,\mathrm{A}} & t_{x,\mathrm{A}} \\ r_{10,\mathrm{A}} & r_{11,\mathrm{A}} & t_{y,\mathrm{A}} \\ r_{20,\mathrm{A}} & r_{21,\mathrm{A}} & t_{z,\mathrm{A}} \end{bmatrix} \begin{bmatrix} X_{\mathbf{p}'} \\ Y_{\mathbf{p}'} \\ 1 \end{bmatrix}, \tag{2.21}$$

and

$$\mathbf{p}_{\mathrm{B}} = \mathbf{K}_{\mathrm{B}} \begin{bmatrix} r_{00,\mathrm{B}} & r_{01,\mathrm{B}} & t_{x,\mathrm{B}} \\ r_{10,\mathrm{B}} & r_{11,\mathrm{B}} & t_{y,\mathrm{B}} \\ r_{20,\mathrm{B}} & r_{21,\mathrm{B}} & t_{z,\mathrm{B}} \end{bmatrix} \begin{bmatrix} X_{\mathbf{p}'} \\ Y_{\mathbf{p}'} \\ 1 \end{bmatrix}, \tag{2.22}$$

respectively. Therefore the homography between $\mathbf{p}_{\mathrm{A}}$ and $\mathbf{p}_{\mathrm{B}}$ can be represented as:

$$\mathbf{p}_{\mathrm{B}} = \mathbf{H}\mathbf{p}_{\mathrm{A}}, \tag{2.23}$$

where,

$$\mathbf{H} = \mathbf{K}_{\mathrm{B}} \begin{bmatrix} r_{00,\mathrm{B}} & r_{01,\mathrm{B}} & t_{x,\mathrm{B}} \\ r_{10,\mathrm{B}} & r_{11,\mathrm{B}} & t_{y,\mathrm{B}} \\ r_{20,\mathrm{B}} & r_{21,\mathrm{B}} & t_{z,\mathrm{B}} \end{bmatrix} \begin{bmatrix} r_{00,\mathrm{A}} & r_{01,\mathrm{A}} & t_{x,\mathrm{A}} \\ r_{10,\mathrm{A}} & r_{11,\mathrm{A}} & t_{y,\mathrm{A}} \\ r_{20,\mathrm{A}} & r_{21,\mathrm{A}} & t_{z,\mathrm{A}} \end{bmatrix}^{-1} \mathbf{K}_{\mathrm{A}}^{-1} \tag{2.24}$$

2.5 Chapter summary

This chapter presents basic algebra and projective geometry tools for understanding homography completely, i.e., what a homography is; how a homography arises; and why homography estimation is important. Specifically, this chapter first introduces the concept of a homogeneous coordinate system to provide a fundamental algebraic

tool for constructing projective geometry transformations. Then a pinhole camera model is introduced, and used to relate real world points with points on the imaging plane. Subsequently, three different kinds of homographies arising from three situations are derived and explained mathematically on the basis of aforementioned pinhole camera model. This chapter also serves to form the basis of understanding the problems and solutions of homography estimation, which are described in detail in the next chapter.

Chapter 3

Homography Estimation

Computer vision aims to sense three-dimensional structure from images with a goal of achieving full scene understanding. Homography estimation, which is able to compute geometric transformations between visual data, therefore is one of the most fundamental problems in computer vision and image processing. Knowing geometric transformations between sets of visual information is an essential step for any system that attempts to understand the semantics provided by spatial relationships. This chapter examines the challenging problems in homography estimation from two perspectives: geometry and deep learning.

Section 3.1 introduces a general pipeline for homography estimation using a geometric approach. Afterwards, three practical methods, including direct linear transformation (DLT), singular value decomposition (SVD), and optimization, are presented to compute a homography from a set of correspondences. In Section 3.2, a recently proposed method, which approaches homography estimation as a learning task, is introduced. Section 3.3 concludes this chapter.

3.1 Homography Estimation from a Geometric Perspective

While an image is worth a thousand words, it can only provide a fractional observation for real scene understanding due to the limited 2D perspective, which lacks the 3D structure of the scene. Different images of the same scene may appear to have significant variations simply because that they are captured from different viewpoints. Therefore, to fully recover the 3D structure of the scene in detail, informative visual data captured from different viewpoints is required.

One feasible way for 3D scene recovery is to combine information from multiple viewpoints using the homography, which encodes the geometry model of the scene. According to the preliminaries introduced in Chapter 2, the homography between two different planes is related using correspondences across the two different images. Therefore, the task of estimating the homography is transformed into correspondence detection, which aims to find local invariant features for matching. The performance of correspondence detection directly affects homography estimation accuracy. Correspondence detection is a very challenging task largely because the following aspects:

1. The properties of the correspondences may have significant variance. Since a homography does not maintain invariants such as length, area, parallelism, etc., correspondences across the same scene may appear vastly different. Figure 3.1 illustrates this situation (sourced from http://www.cvg.rdg.ac.uk/PETS2006/data.html). One can see that these two images only share small overlapping regions. Due to the distance between these two viewpoints, even the same patterns (brown squares) on the floor are difficult to match across images. The matching of correspondences

between widely separate views is known as the wide baseline matching problem, which is out of the scope of this thesis. Interested readers are referred to [78].

2. Correspondences are only coplanar locally. As mentioned in Section 2.4, three kinds of homographies all arise by related correspondences which fall on the same plane. In reality, the detected correspondences may not fall on the same plane and as such the estimated homography may have a great error. Figure 3.2 illustrates the detected correspondences across two images captured from different viewpoints. One can see that all correspondences only approximately fall on the same plane when ignoring distance along the Z axis in a camera coordinate system.

3. An ideal homography cannot model the geometric relationship among images captured in a real scene. This situation is often a result of the inevitable lens distortion. Moreover, some specific kinds of lenses, such as wide-angle and fish eye lenses, have severe distortion, which adversely affects the accuracy of the homograpy model. Figure 3.3 shows the effect of lens distortion in fish eye and wide angle lenses respectively. The severe distortion adversely affects the accuracy of the coordinates of the correspondences and hence the computed homography has a significant deviation from the true mapping.

In practice, there is no prior knowledge about the aforementioned issues. Thus, homographies are often estimated approximately under the following practical assumption:

1. Texture or context information does not change significantly under the homography transformation. Many correspondence detectors and descriptor have been proposed according to this principle over the past two decades. The features extracted from the key points using a proposed descriptor are

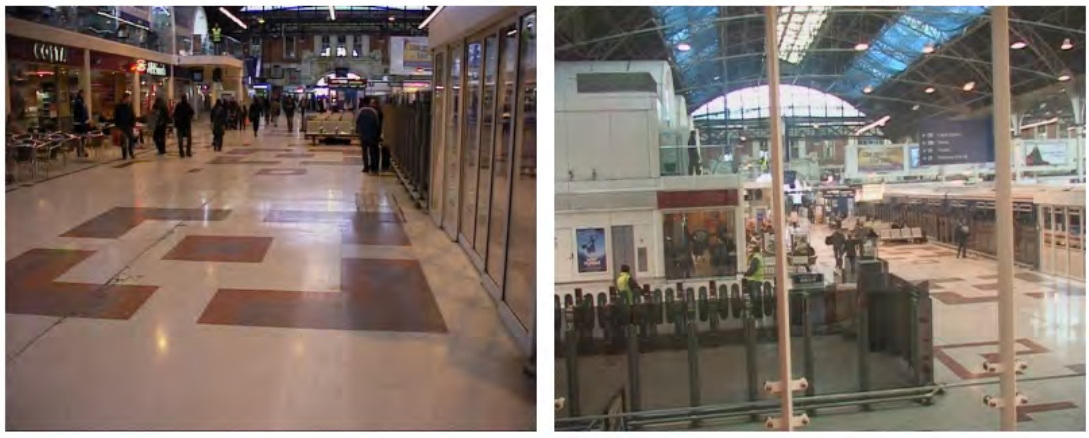


Figure 3.1: Images captured from two remotely placed surveillance cameras in the PETS2006 dataset. The two viewpoints are widely separated and hence overlapping areas are small. The correspondences on both two images are very difficult to detect and match. This is referred to as the wide baseline matching problem.

Figure 3.2: The detected correspondences do not strictly fall on the same plane in the real world coordinate system, i.e., they are only locally planar. The pictures are sourced from [13]

Figure 3.3: The effects of lens distortion that arise from fish eye and wide angle lenses respectively. The severe distortion adversely affects the accuracy of the coordinates of the correspondences and hence the computed homography has a significant deviation from the true one. These two images are captured by Geoffrey Morrison.

usually quite similar to the feature obtained from the correspondence in the other viewpoint.

2. Surfaces are approximately globally planar. One important computer vision application of homography estimation is panorama generation. When taking photos in a manner shown in Figure 3.2, the correspondences are approximately on the same plane. This is because the distance between the correspondences and the image plane is sufficiently large such that the difference can be ignored.

3. The distortion of camera lens is locally negligible (except for wide-angle lenses and fish eye lenses, these lenses are not considered in this thesis.)

4. The relative order between correspondences is preserved across different viewpoints.

Given these assumptions, it is possible to compute an appropriate geometric model related multiple views. The general pipeline for homography estimation in a geometric perspective is composed of two steps:

1. Correspondences detection. Any object which can be identified uniquely across views is able to be used as a correspondence. Therefore, correspondences detection consists of three components: (1) A *detector* which is able to detect prominent objects or regions which can potentially be used as correspondences; (2) A *descriptor* is required to describe these potential correspondences by encoding their local and/or global context information, such that they can be identified uniquely across images; (3) A *matcher* which aims to remove outliers in a putative correspondence set. The putative correspondence set formed by the first two steps is usually contaminated by incorrect matches. Thus, the next step, i.e., correspondence matching, is to filter out outliers and reduce fitting error as much as possible. Two

popular distances are used to evaluate the most unique correspondences. KNN matching is used to match feature vector extracted from descriptor. To ensure matches remain unique, we only select the first match's distance only if it is significant larger than the second one closest distance. Another widely used technique to filter out outliers is random sample consensus (RANSAC), which is a robust technique and can be applied to a set of points which contains a lot of outliers.

2. Homography Estimation. Up to this point, a set of correspondences with a maximum confidence score has been found. The homography among these correspondences can be computed using a specific techniques according to correspondence types.

In the following, the literature regarding correspond detection are reviewed. Then homography estiamtion solutions are introduced in 3.1.2

3.1.1 Correspondence Detection

In the past three decades, a significant number of local feature detectors and descriptors have been proposed for various applications. A detailed survey regarding local features has been published in [135]. Among the array of available feature extractors, only those that are affine invariant are sufficiently robust to match across images. The point correspondences are the most widely used form in correspondence detection because that the shape of points is preserved during homography transformation. In addition, point correspondences exist over a wide range of images and as such they are easy to detect. This thesis only focuses on point correspondence unless explicitly stated otherwise. Other types of correspondence, such as line, circle, region, etc., are not considered in this thesis.

Point Correspondence Detection

This thesis uses SIFT and SURF as the feature extractors, and as such only these extractors are reviewed. Interested readers are referred to [109] for an overview of other detectors.

The scale invariant feature transform (SIFT) is a classical feature detection algorithm proposed by Lowe [92], and has been successfully applied to a large number of computer vision applications such as object recognition [39, 92, 124, 136], simultaneous localization and mapping (SLAM) [42, 104, 111], image stitching [13, 68, 82], 3D modeling [10, 113], video tracking [64, 117, 157], etc. SIFT describes an image using a large collection of feature vectors, each of which is generated by a *key point*. More specifically, the process of SIFT consists of four steps:

1. Scale-space extrema detection, which detects extrema in a difference of Gaussian (DoG) computed by a multi-scale image pyramid.
2. Key point Localization. The objective of this step is to select key points based on a measure of stability.
3. Orientation Assignment. Each preserved key point is assigned to one or more orientations based on local image gradient directions to enable rotation invariance.
4. Key point Description, which uses local image gradients at the selected scale and rotations to describe each key point region.

A SIFT feature extractor has two noteworthy aspects:

1. It is invariant to a similarity transformation (e.g. uniform scaling, rota-

tion, and translation), which is an important component of a homography. Therefore, SIFT is partially invariant to homography transforms and able to be used in correspondence detection across views transformed by the homography.

2. It provides a complete pipeline for the process of both detection and description, unlike the previously proposed Harris corner method [52] which can only serve as a detector.

Apart from the aforementioned general advantages, SIFT also has many merits regarding practical applications:

1. Locality: The feature vector generated by SIFT is from the local region and as such it is robust to occlusion and clutter.

2. Distinctive: The dimensionality of SIFT is 128, which contains the properties such as angle and scale, and the feature vector has small intra class but large inter class distances. Therefore, it can be easily matched in a large database of objects.

3. Quantity: The feature generation is based on key points rather than lines, curves, region, etc. A pixel is the most basic element for an image, thus a large numbers of keypoints can be detected.

4. Efficiency: The computational complexity is such that SIFT achieves close to real-time performance. In recent years, the GPU implementation of SIFT enabled real-time or faster performance.

5. Extensibility: The SIFT feature can be easily extended to tasks which require domain knowledge by combining domain knowledge into the SIFT process [2, 70, 90].

Speeded up robust features (SURF) is a widely used local feature detector and descriptor, which is three times faster than SIFT. SURF makes a significant modification to almost every step of the process used by SIFT to decrease the computational complexity. Unlike SIFT which approximates the Laplacian of Gaussian (LoG) with DoG, SURF uses an integer approximation of the determinant of the Hessian blob detector to detect key points. The use of integral images and convolution with a box filter enables parallel computing for different scales. Regarding the orientation assignment and the feature descriptor, SURF uses Haar wavelet responses of both horizontal and vertical directions, which can be computed once for all descriptors. Moreover, SURF lowers the dimensionality of the feature vector to 64, further improving the speed of computation and matching. In a practical implementation, the OpenCV [12] library extends the 64 dimension feature vector to 128 to improve its distinctiveness. Thanks to the orientation assignment and approximation of LoG, SURF is invariant to rotation and scale. Another big advantage SURF has is that it exhibits greater robustness against blurring and rotation, which often interfere with point correspondence detection for other types of feature extractors. However, SURF performs poorly when presented with viewpoint and illumination changes.

SIFT and SURF can both locate points of interest and extract features from them simultaneously. Aside from these feature extractors, local point descriptors which aim to generate discriminative features from points of interest efficiently have also been studied extensively. DAISY [132] is one of the most representative local image descriptors, which is used in Chapters 4 and Chapter 5 to generate features from detected point correspondences. DAISY is robust against many types of photometric changes and geometric transformations. A DAISY descriptor can be computed efficiently at every pixel and as such that it can be used to generate dense correspondences, which can help make correspondence matching more robust and reliable.

Once points of interest are detected and described using the aforementioned extractors, a putative point correspondence set is built up using approximate nearest neighbor where may contains many outliers. Random sample consensus (RANSAC) method is employed in this thesis to filter out outliers. Regarding feature matcher and RANSAC, interested readers are referred to [109] for an overview of point correspondence matching and filtering process.

3.1.2 Homography Estimation using a set of Point Correspondences

Once the point correspondences have been detected and matched across images, the computing of the homography can commence. In the following sections, several ways to determine a homography are introduced.

Direct Linear Transformation

Suppose that we have two overlapping images I_{A} and I_{B}, the matching points across these two images is written as $\mathbf{p}_{\mathrm{A}} = [x_{\mathbf{p}_{\mathrm{A}}}, y_{\mathbf{p}_{\mathrm{A}}}, 1]^{\top}$ and $\mathbf{p}_{\mathrm{B}} = [x_{\mathbf{p}_{\mathrm{B}}}, y_{\mathbf{p}_{\mathrm{B}}}, 1]^{\top}$. The homography of these two images can be written as:

$$\mathbf{p}_{\mathrm{A}} = \mathbf{H}\mathbf{p}_{\mathrm{B}}. \tag{3.1}$$

$$\begin{pmatrix} x_{\mathbf{p}_A} \\ y_{\mathbf{p}_A} \\ 1 \end{pmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{pmatrix} x_{\mathbf{p}_B} \\ y_{\mathbf{p}_B} \\ 1 \end{pmatrix}. \tag{3.2}$$

By observation Equation 3.2, we can find that the homography can be rewritten

in in homogeneous form as:

$$x_{\mathbf{p}_A} = \frac{h_{00}x_{\mathbf{p}_B} + h_{01}y_{\mathbf{p}_B} + h_{02}}{h_{20}x_{\mathbf{p}_B} + h_{21}y_{\mathbf{p}_B} + h_{22}}, \tag{3.3}$$

and

$$y_{\mathbf{p}_A} = \frac{h_{10}x_{\mathbf{p}_B} + h_{11}y_{\mathbf{p}_B} + h_{12}}{h_{20}x_{\mathbf{p}_B} + h_{21}y_{\mathbf{p}_B} + h_{22}}. \tag{3.4}$$

By reformatting above two equations, we can obtain:

$$x_{\mathbf{p}_B}h_{00} + y_{\mathbf{p}_B}h_{01} + h_{02} - x_{\mathbf{p}_B}x_{\mathbf{p}_A}h_{20} - y_{\mathbf{p}_B}x_{\mathbf{p}_A}h_{21} - x_{\mathbf{p}_A}h_{22} = 0, \tag{3.5}$$

and

$$x_{\mathbf{p}_B}h_{10} + y_{\mathbf{p}_B}h_{11} + h_{12} - x_{\mathbf{p}_B}y_{\mathbf{p}_A}h_{20} - y_{\mathbf{p}_B}y_{\mathbf{p}_A}h_{21} - y_{\mathbf{p}_A}h_{22}. \tag{3.6}$$

Therefore, each point correspondence contributes two equations for solving $\mathbf{H}$. The direct linear transformation (DLT) is a basic method which aims to estimate the homography between I_A and I_B from a set of noisy point correspondences $\{\mathbf{p}_A^i, \mathbf{p}_B^i\}$. For a point correspondence i, we can further formulate 3.5 and 3.6 in matrix form as:

$$\mathbf{A}_i\mathbf{h} = \mathbf{0}, \tag{3.7}$$

where

$$\mathbf{A}_i = \begin{bmatrix} x_{\mathbf{p}_\mathrm{B}} & x_{\mathbf{p}_\mathrm{B}} & 1 & 0 & 0 & 0 & -x_{\mathbf{p}_\mathrm{A}}x_{\mathbf{p}_\mathrm{B}} & -x_{\mathbf{p}_\mathrm{A}}y_{\mathbf{p}_\mathrm{B}} & -x_{\mathbf{p}_\mathrm{A}} \\ 0 & 0 & 0 & x_{\mathbf{p}_\mathrm{B}} & y_{\mathbf{p}_\mathrm{B}} & 1 & -y_{\mathbf{p}_\mathrm{A}}x_{\mathbf{p}_\mathrm{B}} & -y_{\mathbf{p}_\mathrm{A}}y_{\mathbf{p}_\mathrm{B}} & -y_{\mathbf{p}_\mathrm{A}} \end{bmatrix}, \tag{3.8}$$

$\mathbf{h}$ is the vector form of $\mathbf{H}$. Regarding all point correspondences in $\{\mathbf{p}_\mathrm{A}^i, \mathbf{p}_\mathrm{B}^i\}$, $\mathbf{A} \in \mathbb{R}^{2i\times 9}$ is formed by stacking $\mathbf{A}_i$ vertically, i.e.,

$$\mathbf{A}\mathbf{h} = \mathbf{0} \tag{3.9}$$

Since $\mathbf{H}$ is up to an arbitrary scale, the eight degrees of freedom can be solved by four point correspondences. To solve the homography, the requirement for these four point correspondences is that no three points can be collinear. In four point correspondences case, $\mathbf{A}$ is a 8×9 matrix and hence the null space of $\mathbf{A}$ is the solution space for $\mathbf{h}$.

To avoid issues of numerical precision, the point correspondences need to be first normalized before applying DLT as instructed by [54].

Singular Value Decomposition

Another way to solve the homography is to reformulate 3.9 as a minimization problem usingan algebraic distance cost function, i.e.:

$$\arg\min_{\mathbf{h}} \|\mathbf{A}\mathbf{h}\|^2 . \tag{3.10}$$

Since $\mathbf{h}$ is up to an arbitrary scale, we constrain $\|\mathbf{h}\| = 1$ to eliminate the ambiguity. So 3.10 can be rewritten as:

$$\arg\min_{\mathbf{h}} \mathbf{h}^\top \mathbf{A}^\top \mathbf{A}\mathbf{h}, \quad \text{subject to } \|\mathbf{h}\| = 1. \tag{3.11}$$

By introducing a Lagrange multiplier λ, Equation 3.11 is equivalent to minimizing:

$$\arg\min_{\mathbf{h}} \mathbf{h}^\top \mathbf{A}^\top \mathbf{A}\mathbf{h} - \lambda\left(\mathbf{h}^\top \mathbf{h} - 1\right), \quad \text{subject to } \|\mathbf{h}\| = 1. \tag{3.12}$$

Since the function is of the second order, it's global minimum can be reached when it's partial derivatives are zero. Let $L(\mathbf{h}) = \mathbf{h}^\top \mathbf{A}^\top \mathbf{A}\mathbf{h} - \lambda\left(\mathbf{h}^\top \mathbf{h} - 1\right)$. By taking the derivative with respect to $\mathbf{h}$ and setting the equation to zero, we can obtain:

$$\frac{\partial L(\mathbf{h})}{\partial \mathbf{h}} = \mathbf{A}^\top \mathbf{A}\mathbf{h} - \lambda\mathbf{h} = 0. \tag{3.13}$$

By observing Equation 3.13, one can see that the solution of $\mathbf{h}$ is one of the eigenvectors of $\mathbf{A}^\top \mathbf{A}$. Considering the objective of 3.11 is to minimize $L(\mathbf{h})$, the minimum value is obtained when $\mathbf{h}$ is the eigenvector corresponding to the minimum eigenvalue, $L(\mathbf{h}) = \lambda_{\min}$.

Therefore, after constructing $\mathbf{A}$ completely, singular value decomposition is employed to get the eigenvector corresponding to the minimum eigenvalue of $\mathbf{A}^\top \mathbf{A}$; and as such it is the solution of the homography.

Optimizing the Homography Using Geometrically Meaningful Errors

Homography estimation tasks have natural geometric meaning as mentioned in Chapter 2 and 3. Therefore, a homography estimation task can be reformulated as a optimization task using different geometrically meaningful cost functions. Commonly used cost functions include Euclidean error, reprojection error, and sampson error. The objective of Euclidean error is to minimize the distance between $\mathbf{p}_\mathrm{A}$ and $\mathbf{p}_\mathrm{B}$. Reprojection error is a cost function that makes a correction for each correspondence by considering the distance between the original points and their reprojections. Sampson error aims to minimize the first order approximation of geometric distance between point correspondences. Interested readers are referred to as [54] for more details.

3.2 Homography Estimation Using Deep Learning

While solving the homography estimation task from geometric perspective has achieved success to some extent, the aforementioned methods may still fail due to the fact that they are heavily reliant on accurate correspondence [1] detection and matching. In practice, an accurate and general correspondence detector is difficult to formulate because any variations in illumination, perspective, position, etc. may adversely affect it's performance.

With the rise of deep learning, homography estimation, i.e., a classical task in geometric computer vision, had gained new life. Generally speaking, deep learning models have an ability to represent input data as a more abstract and semantic

[1] Note that only point correspondences are considered in this thesis. This assumption also holds for other geometric correspondences, such as lines, circle, squares, etc.

compound in a hierarchic manner. Latent variables within a deep learning model are able to encode all information regarding tasks used for learning. Well-trained deep learning models implicitly master the knowledge used for handling tasks. Hence, there is no need to solve these tasks in a step-by-step way explicitly. This ability enables solving homography estimation in an end-to-end fashion, without using any correspondence detection and matching processes.

This section reviews literature in terms of homography estimation from a deep learning perspective. In the first subsection, a plain geometric transformation parameterization for deep learning models is introduced for explaining how to reformulate the deterministic geometric transformation task as a learning task. Afterwards, several homography estimation methods that operate over a pair of images are introduced as well a new four-corner homography corner parameter set. In the last part of this section, the advantages and disadvantages of these methods are discussed.

3.2.1 Plain Geometric Transformation Parameterization for Deep Learning Models

Learning spatial transformations between visual data is an active research topic in computer vision. This on-going interest arises from the fact that one of the main obstacles for image recognition, classification, pose estimation, etc. is variances between visual data originated from viewpoints, i.e. the effects of the spatial transformation.

A significant amount of literature has studied estimating spatial transformations using deep learning. These methods can be generally classified into two categories [66]:

1. learning and analyzing transformation-invariant representation; and

2. feature selection using spatial transformations.

[14, 24, 40, 69, 81, 119] aim to use CNNs to generate features of images, which are invariant to spatial transformations, i.e, isometry, affine, homography, etc. Regarding feature selection methods, [8, 31, 43, 48, 66, 112] proposed to generate distinctive features using attention mechanisms, which are built on spatial transformations. Specifically, through a series of spatial transformations, distinctive parts of images are enlarged and emphasized and as such the extracted features are more robust and informative.

Notably, [66] proposed a modular subnetwork, termed a "spatial transformer network" (STN). STN can be added directly to almost all existing CNNs to extract more informative features using self-learned transformations. Specifically, given a proper input (typically an image or feature maps), STN automatically computes an appropriate spatial transformation in an explicit way, i.e., compute the value of each entry in the spatial transformation matrix, such as a similarity, affine, homography matrix, etc. The computed transformation is applied to the input to locate and sample the distinctive region. Therefore, the processed input possesses more informative features, which are helpful for further tasks, such as classification, identification, segmentation, etc.

Although STN has achieved success to some extent, it still has many drawbacks when applying it to some specific tasks. For example, the parameterization of spatial transformations used in STN is not suitable for homography estimation, because each parameter in a homography matrix may have a large value variation. A homography is typically composed of a rotation, translation, scaling, shear, etc. components and hence different parameters may have significantly different value ranges. In neural network training, parameters which possess low values have

less impact than those that have larger values; and as such low value parameters cannot be trained appropriately. Moreover, this parameterization only uses eight parameters and therefore it is not robust. If any one of the predictions fails, it may result in a significant error in the overall prediction of the homography.

3.2.2 Four-points Homography Parameterization

To alleviate the instability of the parameterization proposed in STN [66] when applied to homography estimation, [28] propose a new architecture and parameterization to address this issue. Specifically, [28] presents a VGG19-like CNN architecture to predict the four corners offsets of a pair of images. Afterwards, the homography between this pair of images is recovered from these four corner offsets. This parameterization is termed a "four-points" homography. In a CNN model which employs a four-points homography parameterization, the last layer of the model is an eight node dense layer which uses a linear activation. Each neuron in the last layer predicts an x or y offset for one of the four corners.

Inspired by [28], many researchers [32, 100] take a step further by studying more advanced architectures which can extract more useful information from a pair of images. Erlik *et al.* [32] proposed a hierarchy of twin convolutional regression networks to estimate the homography between a pair of images. In the model proposed by [32], the modular networks are stacked to reduce error bounds of the estimate in a sequential manner. In the final process, all outputs are merged together to estimate the four-point homography between images. Nguyen *et al.* [100] presented an unsupervised deep learning algorithm to estimate planar homography between a pair of images in an end-to-end fashion. [100] first estimates the four corner offsets in the middle stage of the network and then uses the initial homography recovered from these offsets to generate the warped image. In the

final stage, the network minimizes a pixel-wise intensity error metric between the warped and original image and as such it does not need ground truth data.

While the four-point homography parameterization has been widely used in recent work and has gained success in some specific tasks, it still has many drawbacks:

1. Spatial relationships between corners are broken. The Four-point homography uses the offsets of the four corners between a pair of images and these four offsets (i.e. typically eight elements) are merely embedded into a vector (i.e., a dense layer). Therefore the vector representation used in a four-points homography breaks the spatial relationship among these four corners.

2. Unstable parameterization. The four correspondences recovered from four corner offsets are the minimum to estimate the planar homography between a pair of images. Perturbation of any one predicted corner significantly affects the accuracy of the estimate adversely. In other words, a four-point homography is not robust. Therefore, the unstable four-point homography parameterization is the main obstacle for homography estimation based on deep learning models.

3. High computational complexity. Since the last layers in deep learning models used for four-point homography estimation are dense layers, fully-connected layers preceded the dense layers are frequently used to extract abstract features for the homography. The use of fully-connected layers significantly increases the computation requirement. For example, the fully-connected layers in networks such as VGGNet [116], AlexNet [75], etc., occupy nearly 20% (i.e., less representation capacity) of the weights, but require 80% of the computation.

Thus, designing a proper homography estimation parameterization to address these aforementioned issues becomes an urgent task. This thesis presents a novel

parameterization for homography estimation in a natural and elegant way. All three aforementioned drawbacks of the four-point homography parameterization have been addressed using the proposed parameterization. The details of this work are presented in Chapter 6.

3.3 Chapter summary

This chapter introduces homography estimation problems from two different perspectives: geometry and deep learning. In Section 3.1, one of the most commonly used pipelines, i.e., homography estimation based on point correspondence detection, is described. Afterwards, three types of solutions are characterized in a geometric way, and what situations they should be used in are outlined. Meanwhile, the drawbacks of geometric based homography estimation are discussed to motivate the development of deep learning models, which have been widely researched recently for homography estimation. Section 3.2 briefly reviews the literature on deep learning based homography estimation tasks. Two types of homography estimation parameterization are introduced. In addition, their advantages and disadvantages are discussed.

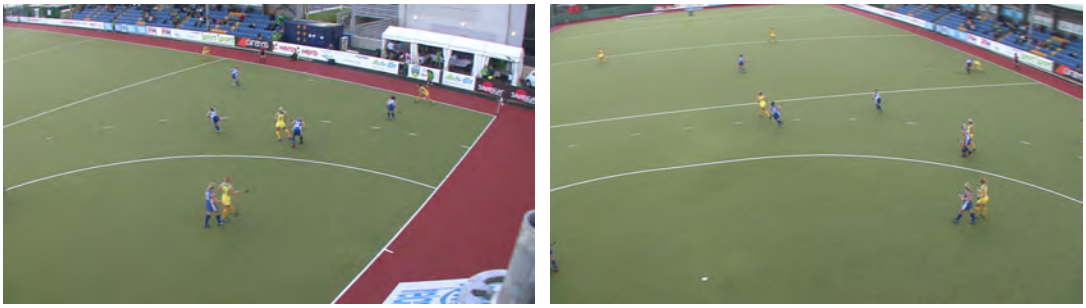
Chapter 4

Calibrating Cameras in Poor-conditioned Pitch-based Sports Games

4.1 Introduction

Camera calibration is a preliminary step in sports analytics, which enables us to transform player positions to standard playing area coordinates. While many camera calibration systems work well when the visual content contains sufficient clues, such as a key frame, calibrating without such information, which may be needed when processing footage captured by a coach from the sidelines or stands, is challenging. In this chapter, two developments that allow the better utilisation of sports videos are presented:

1. An innovative automatic camera calibration system, which does not make use of any key frames, is presented for sports analytics in Section 4.2.



(a) (b)

Figure 4.1: Two examples of challenging frames. (a) is a frame which only contains part of the playing area and the playing surface has limited information available for camera calibration. (b) represents a frame which fully covers the whole playing area. This kind of frames is usually set as the first frame in manual registration methods. One may see that several pitch lines are missing on the farthest side of the playing area and this makes calibration difficult. Moreover, the distortion of the camera increases the difficulty of camera calibration.

2. An algorithm to rectify a video, i.e. find the true vertical axis of the video is presented in Section 4.3.

4.2 Key-Frame Free Camera Calibration

Calibrating cameras in sports scenes is often vital for further analysis and sports applications such as tactical analysis, player tracking, camera calibration enables us to determine the real-world positions of players from their positions in the videos.

Designing a camera calibration system is a challenging task for two reasons: (1) Land marks such as pitch markings are sparse and frequently occluded by players, and are hard to accurately detect in the far field; (2) frames in sports videos are highly variable due to the changes in perspective, focal length, distortion, illumination, motion blur, the field of view.

Most existing camera calibration systems in the sports domain fall broadly into two categories. The approaches in the first category rely on multiple stationary cameras which are fixed at specific positions in a stadium. This kind of approach calibrates cameras by utilizing objects and land markings in the overlapped regions of each fixed camera. It is often considered to be a more reliable solution than a single camera as the information collected from multiple views is richer and more helpful for calibration. These approaches have an obvious drawback as placing multiple cameras around the playing area is often unrealistic, and hence such approaches do not generalize to all situations.

The second category determines camera parameters in a video clip through calibrating a key frame. Then the camera parameters of the remaining frames are derived from the relationship between themselves and the key frame. However, the approaches in this category fail when a suitable key frame cannot be found, calibrated, or manually registered. This can occur because pitch markings used for calibrating the key frame cannot be recognized due to the player occlusion, camera distortion, motion blur, illumination change and camera position. For example, in a close-up view, as shown in Figure 4.1a, an algorithm can only see a small portion of the pitch and a few line markings. The insufficient clues make camera calibration hard. In Figure 4.1b, one may see that the pitch lines near the farthest side cannot be seen due to the horizontal view angle. Such frames frequently appear in video clips captured by a single camera and hence there is no so-called key frame to be set as a reference. As far as we know, the problem of camera calibration for a single camera in sports scenes has yet to be solved.

In this section, we present an automatic camera calibration system for a single cameras in sports scenes to address the aforementioned issues. The proposed system can undistort and calibrate each frame simultaneously by using correspondences between pairs of consecutive frames. The system consists of three

components: a robust linear panorama module, a playing area estimation module, and a homography estimation module. The key idea of this method is to calibrate each frame in a holistic way, i.e., calibrating each frame through calibrating the panorama generated from every frame, rather than relying on a specific fame which contains the well-conditioned pitch structure. We only focus on field hockey in this work, but the technique is general and applicable to other team sports such as soccer, volleyball, football, etc.

4.2.1 Related works

Due to the large body of work concerning calibration, this literature review will focus only on calibration methods that have been applied to sports video tasks. The predominant approach for camera calibration in sports is to make use of the features on the playing area, such as line markings. Han *et al.* [51] detected straight lines on the tennis court using color and local texture constrains and subsequently computed camera parameters by matching the intersections of detected lines with a standard court model. Hu *et al.* [62] improved the above method by adding color dominant segmentation to robustly determine the boundary of the basketball playing area. Some systems calibrated key frames in a video first and then derived camera parameters in successive frames through Iterated Closest Point (ICP) [153]. For instance, Lu *et al.* [93] detected the boundaries of the basketball court using a Canny detector [16] to calibrate a key frame and applied ICP to calibrate the remaining frames. Okuma *et al.* [101] matched the detected feature points with a standard ice hockey model and the remaining calibration task in successive frames was performed with ICP. Sha *et.al* [114] manually located reference points at the beginning of a swimming race. Then the calibration of the remaining frames is done by finding correspondences between adjacent frames using a SIFT feature extractor.

Apart from the aforementioned single camera calibration works, multiple cameras in sports have also been exploited for calibration. For instance, Puwein *et al.* [105] presented a multi-view pan-tilt-zoom (PTZ) camera calibration system which leveraged the trajectories of the players. The lines on the playing area are used to further improve the result. Stein [123] made use of the trajectories of objects moving on a common ground plane to calibrate the extrinsic parameters in unsynchronized cameras with known intrinsic camera parameters. Meingast *et al.* [96] localized cameras to recover the orientation and position by utilizing objects' trajectories without assumptions that objects are moving on a ground plane. Our system is significantly different from past works since we neither rely on land markings on the playing area, nor use multiple cameras pre-mounted around the stadium. This makes our system able to generalize to any sport.

Our system is inspired by [59] and [93]. Hess *et al.* [59] registered a video sequence with a pre-defined static model which can be regarded as a panorama. The main drawback is that it needed users to provide that pre-defined panorama. Our approach directly generates such a panorama from video clips automatically. In the process of generating the panorama, each frame is undistorted by using correspondences in adjacent frames. [93] utilizes boundaries of the basketball court to do camera calibration and requires at least one individual frame containing the whole structure of the basketball court which is set to be a reference frame. In our work, we segment the playing area from the generated panorama using dominant color. This means that even though no frame contains enough information for calibration, the process of camera calibration still can be done by using the information carried by all frames.

4.2.2 Overview of the proposed system

Our camera calibration system can be described as a compound of three interacting, but clearly separate modules: robust linear panorama generation, playing area estimation, and homography estimation. Figure 4.2 depicts our system architecture with it's main functional units and the data-flow. The robust linear panorama generation module (see Section 4.2.3) extracts key points from each frame and then matches them in a linear way to get correspondences between each pair of adjacent frames. Subsequently, image undistortion using correspondences is employed here to remove the distortion in the original frames and improve their quality simultaneously. These undistorted frames are then used to generate the panorama. In the playing area estimation module (see Section 4.2.4), the boundary of the playing area is segmented from the generated panorama using color dominant segmentation. Later, a series of image processing techniques are imposed on this segmented binary mask to get the refined panorama. In the homography estimation module, the panorama is calibrated using the four corner points of the pitch. Then the camera calibration of each individual frame (see Section 4.2.5) can be derived from the calibrated panorama by making use of the relationship between the panorama and an individual frame.

4.2.3 Robust linear panorama generation

In order to calibrate each individual frame in a video clip, we first need to generate the panorama which contains the rough border of the playing area. This work utilizes a robust linear image stitching framework similar to [13, 129].

The first step in the panoramic playing area generation is to extract feature points in each frame. Speeded-Up Robust Feature (SURF) [9] is employed here to detect

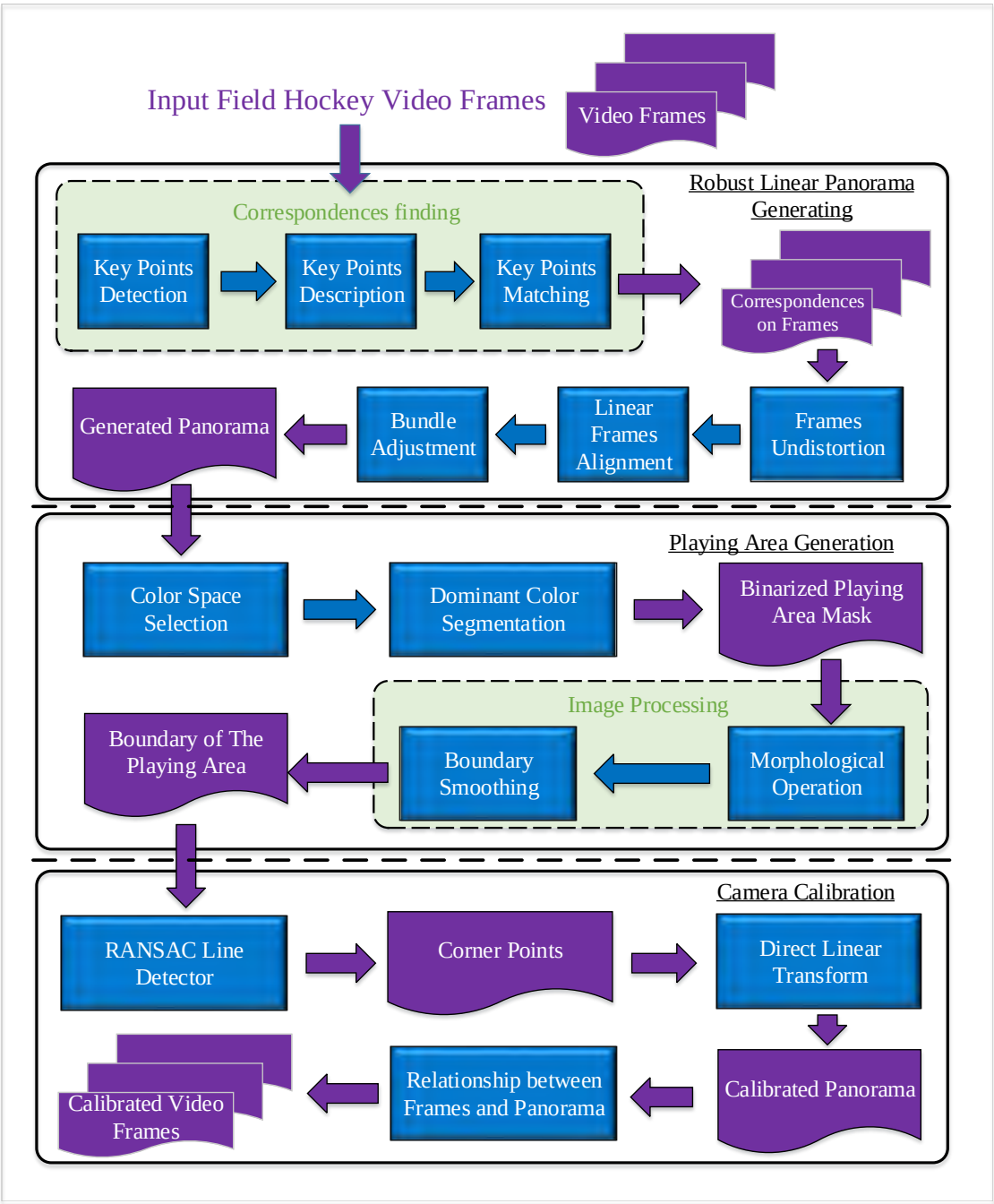


Figure 4.2: The architecture of the complete system. It consists of three modules: robust linear panorama generation, playing area extraction, and camera calibration.

feature points and then the DAISY [132] descriptor is used to describe dense information of the distribution around the detected points. The SURF detector

and DAISY descriptor are chosen by us for three reasons:

1. The SURF key point detector is stable in local patches and invariant to scale and affine transforms. The perspective transform between every pair of two adjacent frames is not significant and hence it can be regarded as an affine transform.

2. SURF is a speeded-up key point detector. As there are lots of frames in a video, the speed of a key point detector is very important.

3. The DAISY descriptor is a dense feature extractor that operates over a selected region. Since there are few clues on the playing surface, a dense descriptor which can produce distinguishable features is crucially important for getting good features and accurate correspondences.

Consider that what we are dealing with is a video sequence, such that consecutive frames have a clear temporal order. Thus a probability model [13] is not employed here to estimate the correlation of one frame with every other frame. A frame is only registered with its adjacent frame in a linear way, i.e. linear correspondence matching. Each feature in a frame is matched to its nearest neighbors in the feature space of the adjacent frames by using a K-D tree. In our experiments, the number of trees is set to 5.

Once the correspondences have been found in a linear way, the correlation of them can be formulated by utilizing a homography matrix $\mathbf{c}'_{i,j+1} = \mathbf{H}'_{j,j+1}\mathbf{c}'_{i,j}$:

$$\begin{bmatrix} x_{\mathbf{c}'_{i,j+1}} \\ y_{\mathbf{c}'_{i,j+1}} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{H}'^{11}_{j,j+1} & \mathbf{H}'^{12}_{j,j+1} & \mathbf{H}'^{13}_{j,j+1} \\ \mathbf{H}'^{21}_{j,j+1} & \mathbf{H}'^{22}_{j,j+1} & \mathbf{H}'^{23}_{j,j+1} \\ \mathbf{H}'^{31}_{j,j+1} & \mathbf{H}'^{32}_{j,j+1} & \mathbf{H}'^{33}_{j,j+1} \end{bmatrix} \begin{bmatrix} x_{\mathbf{c}'_{i,j}} \\ y_{\mathbf{c}'_{i,j}} \\ 1 \end{bmatrix} \tag{4.1}$$

where $\mathbf{c}'_{i,j}$ and $\mathbf{c}'_{i,j+1}$ are the homogeneous coordinates of the ith pair of correspondences that fall on two consecutive frames, j and $j+1$ respectively. Consider

that matched correspondences may contains outliers, and as such we make use of Random Sample Consensus (RANSAC) to estimate the homgoraphy $\mathbf{H}'_{j,j+1}$. After taking a set of random pairs of feature points into account, a score function is computed iteratively as follow:

$$\min_{\mathbf{H}_{j,j+1}} \sum_{i} \left\| \mathbf{H}'_{j,j+1} \mathbf{c}'_{i,j} - \mathbf{c}'_{i,j+1} \right\|^2. \tag{4.2}$$

Finally, the homography with the lowest score is selected as the best consensus. Then the inliers in a set of correspondences can be determined using a preset pixel threshold, $\sigma = 5$, as follow:

$$\mathcal{L}_{ij} = \begin{cases} 1 & \left\| \mathbf{H}'_{j,j+1} \mathbf{c}'_{ij} - \mathbf{c}'_{i,j+1} \right\| < \sigma \\ 0 & \text{other} \end{cases}. \tag{4.3}$$

Figure 4.3 shows some results obtained in the process of robust linear correspondence matching. One may see that most feature points are distributed near the pitch markings and the spectator. There are also some feature points that fall on players. Such feature points do adversely affect image matching since players are moving between frames. But after using RANSAC to filter feature points, we can observe that such feature points are discarded.

Since a single camera can often suffer from severe camera distortion (particularly when wide-angle zoom lenses are used), the homography that we obtained from the above process is not accurate. A camera undistortion algorithm is needed to refine the homography and enhance the quality of the frames. In order to remedy distortion, a standard approach is to make use of camera intrinsic parameters and extrinsic parameters to analytically undistort frames. However, in sports videos captured from a single camera, the aforementioned intrinsic parameters and extrinsic parameters cannot be obtained because camera parameters of each frame vary from each other (as the zoom varies), and the relative position between the playing area and the camera is unknown.

Figure 4.3: Example results obtained in the process of linear correspondence matching. In the first row, key points detected by the SURF detector are denoted by a coloured circle. Images in the second row show an initial correspondences matching completed by Fast Approximate Nearest Neighbor. The last row depicts the inliner correspondences obtained from RANSAC.

To tackle this problem, a correspondence-based approach inspired by [79, 122] is applied here to undistort each frame. Instead of estimating camera intrinsic parameters and extrinsic parameters, we correct lens distortion in each frame by using correspondences found on adjacent frames.

The lens distortion model of a single frame j is defined as:

$$\mathbf{c}_{ij} = \frac{1}{1 + \lambda_j r_{ij}^2} \mathbf{c}'_{ij}, \tag{4.4}$$

where λ_j is the parameter of the lens division model, r_{ij} is the distance of the feature

point i to the Center of Distortion (COD) squared, and $\mathbf{c}_{ij}$ is the undistorted coordinate of $\mathbf{c}'_{ij}$.

To simplify the lens distortion problem in a video sequence, we assume that the COD is in the center of the frame and the lens distortion parameter in two consecutive frames is the same (while this is not strictly correct, the change between two consecutive frames is typically very small). Then the undistorted homography $\mathbf{H}_{j,j+1}$ which relates the two undistorted frames $\mathbf{c}_j$ and $\mathbf{c}_{j+1}$ can be defined as:

$$(\mathbf{D}_1 + \lambda_{j,j+1}\mathbf{D}_2 + \lambda_{j,j+1}\mathbf{D}_3)\mathbf{h}_{j,j+1} = 0, \tag{4.5}$$

where $\mathbf{h}_{j,j+1}$ is the vectorized form of $\mathbf{H}_{j,j+1}$, $\lambda_{j,j+1}$ is the lens distortion parameter shared by the frame j and $j+1$, and $\mathbf{D}_1$, $\mathbf{D}_2$, and $\mathbf{D}_3$ are the factor matrices. Let us assume that

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_1 \\ & \mathbf{I} \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} -\mathbf{D}_2 & -\mathbf{D}_3 \\ \mathbf{I} & \end{bmatrix}, \ \mathbf{v} = \begin{bmatrix} \mathbf{h}_{j,j+1} \\ \lambda_{j,j+1}\mathbf{h}_{j,j+1} \end{bmatrix}, \tag{4.6}$$

so that the solution of $\lambda_{j,j+1}$ is found by iteratively solving $\mathbf{v}$ from

$$(\mathbf{A} - \lambda_{j,j+1}\mathbf{B})\mathbf{v} = 0, \tag{4.7}$$

and updating $\lambda_{j,j+1}$ by finding the smallest magnitude root of the scalar quadratic equation [11][79]:

$$\mathbf{v}^\top(\mathbf{B}^\top + \lambda_{j,j+1}\mathbf{A}^\top)(\mathbf{A} - \lambda_{j,j+1}\mathbf{B})\mathbf{v} = 0. \tag{4.8}$$

After $\lambda_{j,j+1}$ is obtained from the above process, the distorted correspondences $\mathbf{c}'_{i,j}$ and $\mathbf{c}'_{i,j+1}$ can then be undistorted using Equation 4.4.

In practice, a distortion parameter space $\{\lambda_{1,2}, \ldots, \lambda_{j,j+1}, \lambda_{n-1,n}\}$ can be obtained from a video sequence containing n frames. The frame j makes use of $\lambda_{j,j+1}$ to do undistortion. The refined homography $\mathbf{H}_{j,j+1}$ between the frame j and $j+1$ is obtained from $\lambda_{j,j+1}$ using Equation 4.5. Figure 4.4 shows undistorted frames and their corresponding original frames.

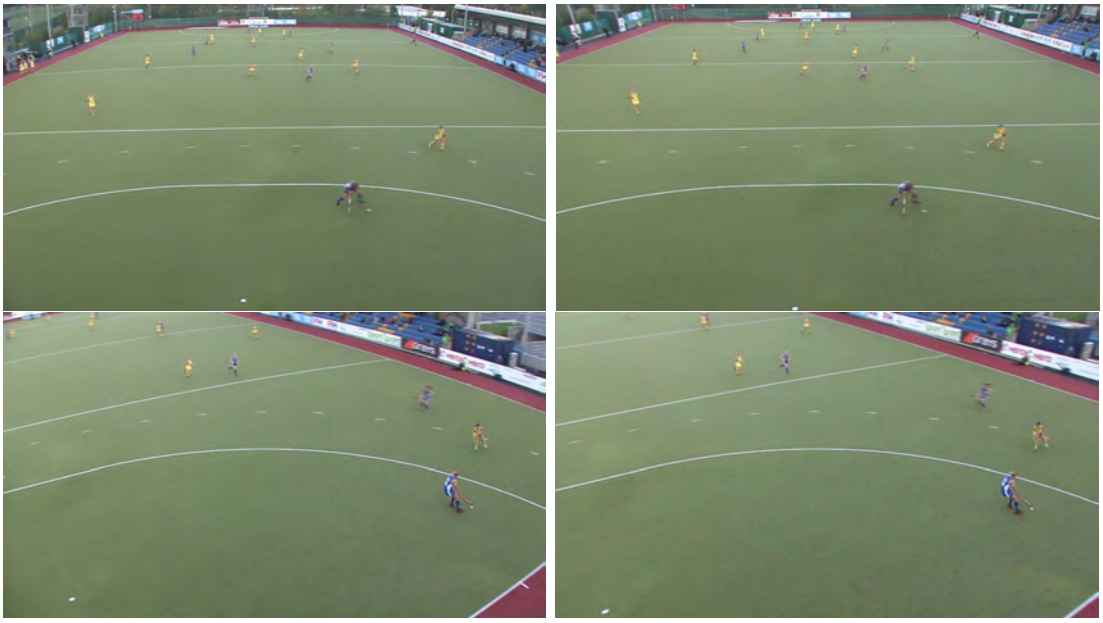


Figure 4.4: The left column is the original frames that have lens distortion. One may see that straight lines appear to be slightly curved. The right column shows their corresponding undistorted frames. The lens distortion in the two frames has been eliminated.

Once the refined homography between every two consecutive frames has been established, the homography between the anchor frame n and the jth frame can be computed as $H_{j,n} = H_{j,j+1}H_{j+1,j+2}\cdots H_{n-1,n}$. The coordinates in the anchor frame are selected as the coordinate system in the generated panorama. In practice, the anchor frame of the panorama is often selected as the last frame of a video clip.

To minimize the errors induced in linear matching, bundle adjustment is utilized here to refine all of the homographies jointly. This step is essential as the concatenation of pairwise homographies can cause accumulated errors and disregards multiple constraints between images. The bundle adjuster can be formulated by minimizing the following error projection function

$$e = \sum_j \sum_i \mathcal{L}_{ij} \left\| \mathbf{H}_{j+1,n}^{-1} \mathbf{H}_{j,n} \mathbf{c}_{ij} - \mathbf{c}_{i,j+1} \right\|_2^2. \tag{4.9}$$

Frames are added to the bundle adjuster one by one and then the parameters in homographies are updated using the Levenberg-Marquardt (LM) algorithm [54]. Each iteration step in LM is of the form

$$\Delta\theta = (\mathbf{J}^T\mathbf{J} + \lambda_l \mathbf{C}_P^{-1})^{-1}\mathbf{J}^T\mathbf{r}, \tag{4.10}$$

where $\Delta\theta$ is the gradient of all parameters, λ_l is the learning rate, $\mathbf{r}$ is the residuals of the correspondences and $\mathbf{J}$ is the derivatives of $\frac{\partial \mathbf{r}}{\partial \theta}$. The covariance matrix C_p encodes the variance of each parameter in it's diagonal. The derivatives $\mathbf{J}$ can be computed analytically via the chain rule.

Finally, the coordinate transformation between the generated panorama P and the frame I_j can be summarized as

$$\mathbf{c}_P = \mathbf{H}_{j,n}\mathbf{c}_{I_j}. \tag{4.11}$$

Figure 4.5 illustrates the panorama which is generated from a set of frames. It can be seen that there are many shadow players due to image blending, but the borders of the pitch are clear.

4.2.4 Playing area extraction

Up to this point, the panorama generated from a video clip is obtained from the above robust linear panorama system. The next step is to extract the playing area from this panorama.

Consider that the color of the playing area in pitch-like sports games is usually green and is typically much different from the surrounding area. Thus color information can be used as an efficient low-level feature to segment the playing area from the background. As implied in [18], a color image I can be represented

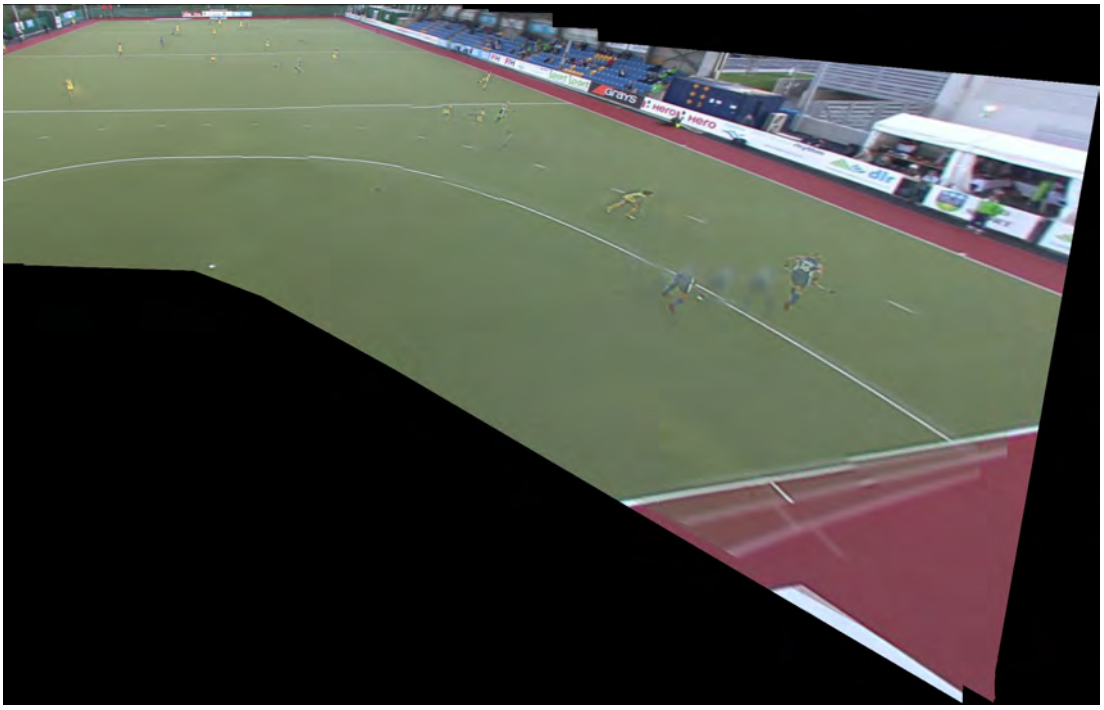


Figure 4.5: The panorama generated from a set of consecutive frames. This panorama roughly restores the borders of the playing area. The surrounding black area cannot be generated due to the viewing angle of the single moving camera which never covers these parts of the pitch.

as a set of dominant colors and the corresponding percentage of occurrence of each color:

$$I = \{(d_i, p_i) : \ i = 1, \ldots, N, \ p_i \in [0, 1]\}, \tag{4.12}$$

where d_i represents the ith dominant color in the image I of a selected color space (e.g. RGB, HSV, CMYK, etc.), and p_i is the corresponding percentage of d_i. Therefore the segmenation of the playing area can be done by searching for the dominant green color in the panorama. While there are a number of approaches for extracting dominant color [17, 76, 138], we build a dominant color segmentation model using a combination of the CIELAB color space and Kmeans clustering. First, the input panorama is converted into CIELAB color space which contains a luminosity layer 'L', and two chromaticity layers 'A' and 'B'. 'A' and 'B' indicate if the color falls along the red-green axis and blue-yellow axis respectively. The

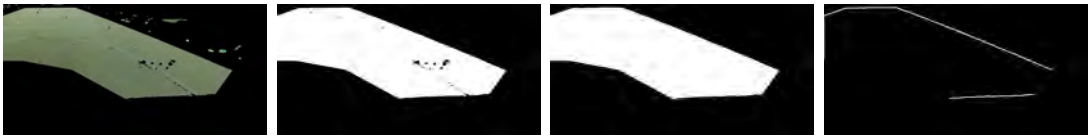


Figure 4.6: The leftmost picture depicts the segmentation results obtained from the dominant color segmentation using the layer 'A' of the CIELAB color space. The next picture shows the binarized playing area where small blobs are already filtered by an opening operation. The third picture shows the refined binary mask without small holes processed by a closing operation. The last image is the smoothed boundary line.

reason why we chose CIELAB color space is that the color of the playing area is usually green and the athletic track appears red. Thus, the chromaticity layer 'A' where the color falls along the red-green axis may be used to distinguish the green area from the red athletic track area. This work makes use of K-means clustering to group the pixels only in the chromaticity layer 'A'. In practical experiments, the number of the centroids is set to 5 to ensure the ability to segment the playing area.

To get a better result, the playing area obtained from above steps is further processed by making use of some image processing techniques. The rough segmented mask is first binarized to simplify subsequent operations. Then small blobs and holes in the image are filtered using morphological opening and closing operations respectively, as follows:

$$I \circ S = (I \ominus S) \oplus S, \tag{4.13}$$

$$I \bullet S = (I \oplus S) \ominus S, \tag{4.14}$$

where $\ominus$ and $\oplus$ represent erosion and dilation respectively and S is a structural element [47]. Finally, the contour of the playing area is extracted by the Moore-Neighbor tracing algorithm and then smoothed by using Savitzky-Golay filters [125]. Figure 4.6 depicts the intermediate and final segmentation results of the playing area.

4.2.5 Camera calibration

Knowing players' locations in relation to the entire playing area is necessary for many sports applications. The task of camera calibration is to compute a geometric transformation which maps a player location $\mathbf{p} = [x_{\mathbf{p}}, y_{\mathbf{p}}, 1]^\top$ in the image homogeneous coordinates to a point $\mathbf{p}' = [x_{\mathbf{p}'}, y_{\mathbf{p}'}, z_{\mathbf{p}'}, 1]^\top$ in the real-world homogeneous coordinates on the standard pitch plane P'. In order to achieve this, a pair of corresponding points can be related by a pinhole camera model [54]:

$$\mathbf{p} = \mathbf{K}[\mathbf{R}|\mathbf{T}]\mathbf{p}', \tag{4.15}$$

where $\mathbf{K}$ is a 3×3 matrix which contains five camera intrinsic parameters. $\mathbf{R}$ and $\mathbf{T}$ are the 3×3 rotation matrix and the 3×1 translation vector of the camera extrinsic parameters respectively. Unfortunately, the calibration approaches [133, 154] of this pinhole camera model are usually not applicable in the sports domain due to the use of single cameras and the lack of an auxiliary markings in the playing area. Thus, to simplify the calibration, we assume that the playing plane is on the plane $z_{\mathbf{p}'} = 0$, and only compute the homography transformation between the panorama and the playing plane instead of the pinhole camera model as in [51, 93]:

$$\mathbf{p} \propto \mathbf{H} \begin{bmatrix} x_{\mathbf{p}'} \\ y_{\mathbf{p}'} \\ 1 \end{bmatrix}. \tag{4.16}$$

Consider that the 3×3 matrix $\mathbf{H}$ has eight Degree of Freedom (DoF) and a pair of corresponding points $\mathbf{p}$ and $\mathbf{p}'$ can supply two equations, i.e. 2 DoF. Thus, four pairs of corresponding points is enough for estimating $\mathbf{H}$. In this system, the four pairs of corresponding points are obtained from four corner points intersected by four boundary lines.

Taking the points on the detected contour of the playing area as input, a RANSAC-based line detector built on [51] is used here to parameterize all four boundary

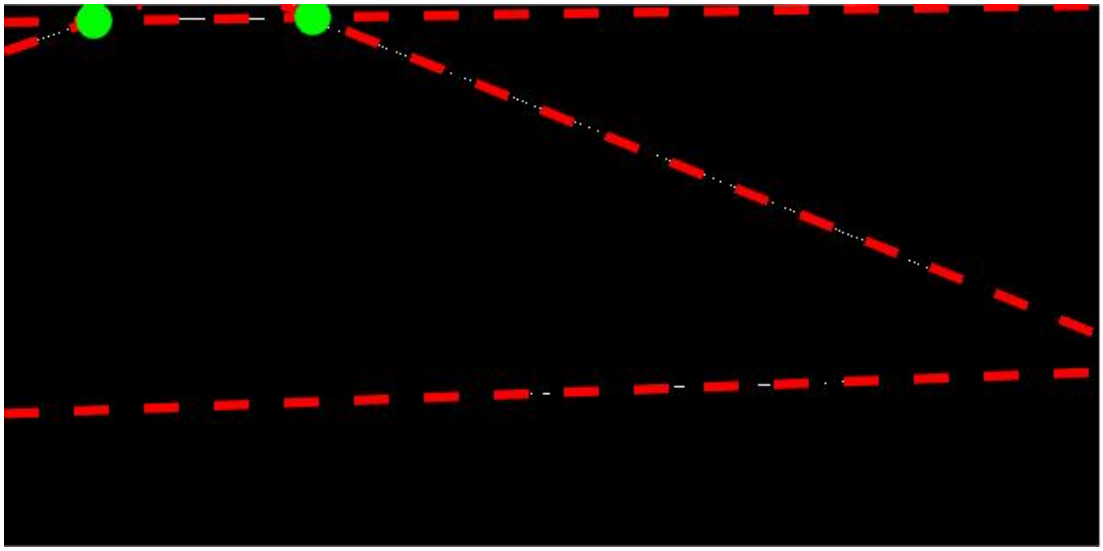


Figure 4.7: Four boundary lines are parameterized by a RANSAC-based line detector from the contour of the playing area. Green points are the intersections of the lines. Due to the size of the image, only two points are shown here.

lines. Specifically, a line $\mathbf{l}$ is hypothesized by randomly selecting two points $\mathbf{m} = [x_{\mathbf{m}}, y_{\mathbf{m}}, 1]^\top$ and $\mathbf{n} = [x_{\mathbf{n}}, y_{\mathbf{n}}, 1]^\top$ which lie on the boundary lines of the segmented playing area. The idea of this method is to minimize the distance from the points in the boundary point set to the randomly hypothesized line $\mathbf{l}$; i.e.,

$$\min_{\mathbf{l}} \sum_{[x,y,1]^\top \in \mathcal{P}} \mathbf{l} \times [x, y, 1]^\top. \quad \text{s.t.} \quad \mathbf{l} = \mathbf{m} \times \mathbf{n}, \quad \mathbf{m}, \mathbf{n} \in \mathcal{P}, \tag{4.17}$$

where $\mathcal{P}$ is the set of points $[x, y, 1]$ on the boundary lines, and $\times$ is the cross product of two vectors. After executing a sufficient number of iterations, the solution with minimum distance is selected as a boundary line. To estimate other potential boundary lines, the points whose distance to the detected dominant line is less than T are eliminated and then we repeat in the above process several times to get the remaining boundary lines. Figure 4.7 shows four detected lines and their intersections which bound the borders of the playing area.

Once four corner points in the panorama $\{\mathbf{p}_i\}_{\mathrm{i}=1}^4$ have been determined by the intersections of the detected lines, the next step is to map them to the correspondences $\{\mathbf{p}'_i\}_{\mathrm{i}=1}^4$ that are in a standard field hockey model. Here, a Direct Linear

Transformation (DLT) is applied to solve the homography from a set of equations:

$$\begin{bmatrix} \mathbf{0}^\top & -\mathbf{p'}_i^\top & y_{\mathbf{p}_i}\mathbf{p'}_i^\top \\ \mathbf{p'}_i^\top & \mathbf{0}^\top & -x_{\mathbf{p}_i}\mathbf{p'}_i^\top \end{bmatrix} \mathbf{h} = 0. \quad i = 1, \dots 4 \ , \tag{4.18}$$

where $\mathbf{h} \in \mathbb{R}^{9\times 1}$ is the vectorized homography $\mathbf{H}$. After mapping the panorama P to the standard pitch model P', the coordinates in frame I_j can be easily transformed to the standard pitch model using Equation 4.19:

$$\mathbf{c}_{P'} = \mathbf{H}\mathbf{H}_{j,n}\mathbf{c}_{I_j}. \tag{4.19}$$

4.2.6 Evaluations and Discussions

The proposed system was evaluated on four different field hockey video sequences. Each sequence was captured with a single camera at a resolution of 1920×1080, and undergoes a moderate range of rotations and zooms. The length of a video sequence varies between 10 and 18 seconds, i.e., between 250 and 450 frames. All the video sequences sweep the whole playing area to make sure the videos contain enough information for camera calibration. Due to the nature of the dataset (captured from a hand held camera) we do not have ground truth and the true camera parameters cannot be obtained. Furthermore, manually calibrating each frame is not realistic and so we evaluate our proposed approach qualitatively.

Table 4.1 summarizes the average number of the correspondences for each frame and the homography transfer error in terms of four video sequences used in our experiments. The average number of the correspondences for each frame in a video indicates the performance of the feature extraction method in the panorama generation. The homography transfer error is computed by Root Mean Square (RMS) error which reports the pixel-level distance between the initially matched correspondences and the correspondences obtained from bundle adjustment. In

Method	Metric	Video sequences			
		1	2	3	4
DAISY	points	264	288	281	272
	RMS	2.519	2.318	2.732	2.556
KLT	points	34	28	29	30
	RMS	3.807	4.231	3.247	3.334
SIFT	points	92	74	81	87
	RMS	4.986	5.223	5.031	5.274

Table 4.1: The homography transfer error and the average number of the correspondences for the tested video sequences.

addition, two commonly used feature extraction methods such as KLT [140] and SIFT are utilized to compare with DAISY. One may see that both RMS and the average number of the points in DAISY outperform those of the other two methods.

Figure 4.8 depicts the results of the calibrated panorama which is generated through the process of mapping the panorama P to the standard pitch model P' using four corner points. One may see that perspective is eliminated and the calibrated panorama is roughly aligned to the standard field hockey model. Thus the positions on the panorama can be properly transformed to that in the standard field hockey model.

Figure 4.9 shows some representative frames and the calibration results of these frames. The calibration of these frames can be derived from the relationship between them and the panorama. The calibration of these frames are shown in Figure 4.9.

we only compare the proposed system to several existing representative approaches like [51, 62, 93]. These works are of court games such as tennis, basketball, and volleyball, where enough land markings can be fully and clearly seen by a single

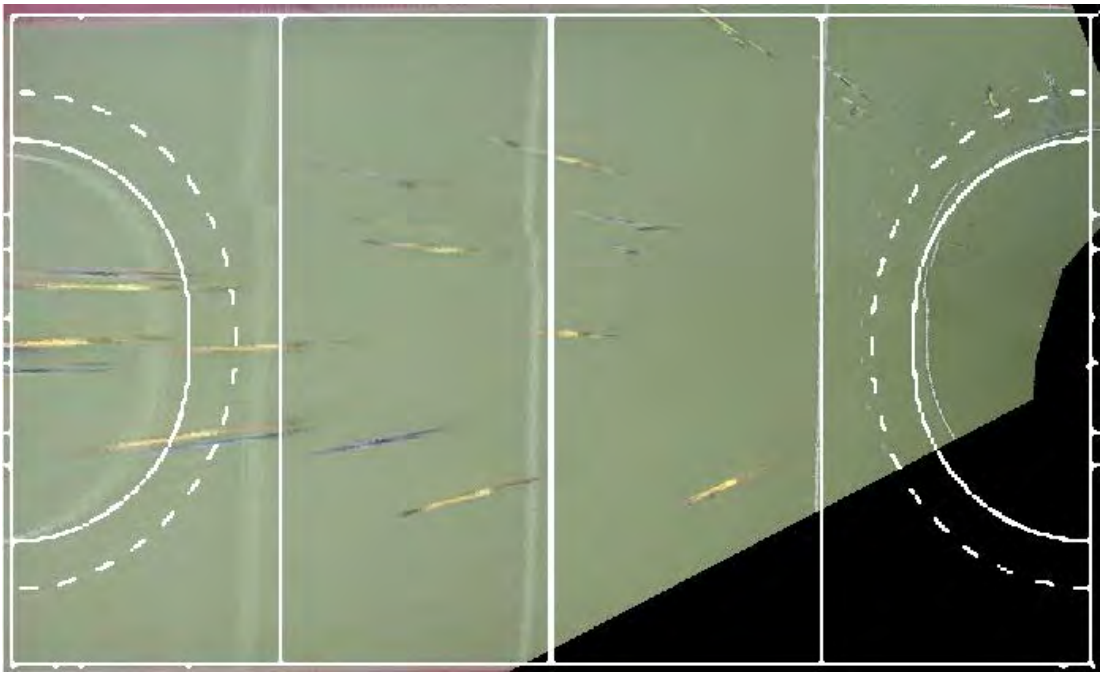


Figure 4.8: The camera calibration result computed from four pairs of corresponding points. Each corner in the real playing area is mapped to its equivalent in the standard field hockey pitch.

Figure 4.9: The calibrated results of representative frames in the panorama. Their corresponding regions are shown by a yellow, magenta, cyan, red, green, blue, white and black box in the leftmost image. Meanwhile, the playing areas of these frames are drawn by the same color in a standard field hockey model in the second image.

camera. However, each individual frame in our dataset does not contain enough clues for calibration due to the limited view coverage of the single camera. This means their systems do not work in our case. Even for some particular frames, where the whole structure of the pitch can be rarely seen, their methods cannot work due to the disappearance of land marks when they are far away from the camera.

The problems of the proposed camera calibration system may be encountered in video sequences where we cannot generate a panorama that contains four boundary lines. This can be solved by using the straight line in the middle of the playing area. Thus the possibility of this case occurring is very low. Moreover, the reason why we do not generate the whole structure of the playing area is that the field-of-view of the perspective transformation on a camera plane does not allow more than 180°.

It is worth mentioning that the accumulated error would increase significantly when the camera stays at the same position for a long time. This is because the small perturbation in pairs of consecutive frames in the same scene makes image alignment hard. To solve this problem, we plan to generate the panorama from several representative frames in corresponding video shots. Subsequently, the remaining frames in that video shot can be calibrated using that representative frame. Finally, this system is not robust to frames which have severe camera dithering. Thus, a technique for dealing with such frames is needed.

4.3 Vertical Axis Detection for Sport Video Analytics

Human detection is one of the fundamental challenges in human-centred vision systems [27, 144, 145]. While calibration is invariably helpful in this task, most uncalibrated methods assume that the image's vertical axis is roughly aligned with the true vertical axis of the scene, or the 'up' direction [71, 151]. However, this assumption may not be valid in real cases. For example, in most camera installations, maximizing camera coverage is often more important than obtaining a correct viewing orientation and as such cameras may have arbitrary rotations,

and the vertical axis of the videos captured may not correspond to the true vertical direction of the scene.

Cameras used for capturing outdoor sports scenes usually suffer from the above issues because they need to cover the entire playing area. Furthermore, camera calibrations information is usually not available in these situations to normalize the acquired video. The performance of human detectors may be degraded when applied to rotated cameras because the detection algorithms are mostly based on the assumption that the scene and the camera vertical axis are aligned.

Several efforts have been made to address the issues faced by sports videos with a non-normalized vertical axis. Feature-based methods [6, 121] have been proposed that can recover the vertical direction by making some assumptions about the scene, such as field markings, or geometric lines on playing area. Given that the cues in sports scenes are sparse, not unique, and susceptible to noise, player occlusion, and deformation, camera calibration usually fails. Modeling based methods counter an un-normalized vertical axis by fitting a human to a particular model, for example built by the uniform color, and histogram of gradients [75]. While these methods can be hand-crafted with success for some specific sport scenes and tasks, applying such approaches to a new scene or sport usually requires domain knowledge owing to the changes in the viewing angle, field markings, and illumination or color of players' uniform. Furthermore, the features visible in a camera view are often inadequate for any feature-based matching approach, or even for manual identification. Examples are shown in Figure 4.11, where the distinguishable surface features are absent or ambiguous.

In sports video analytics, recovering the true direction of a video will greatly reduce the difficulty of detecting and modelling players. To the best of our knowledge, there is no detailed study on recovering the vertical axis in outdoor videos so far. This section proposes an efficient way to achieve this task.

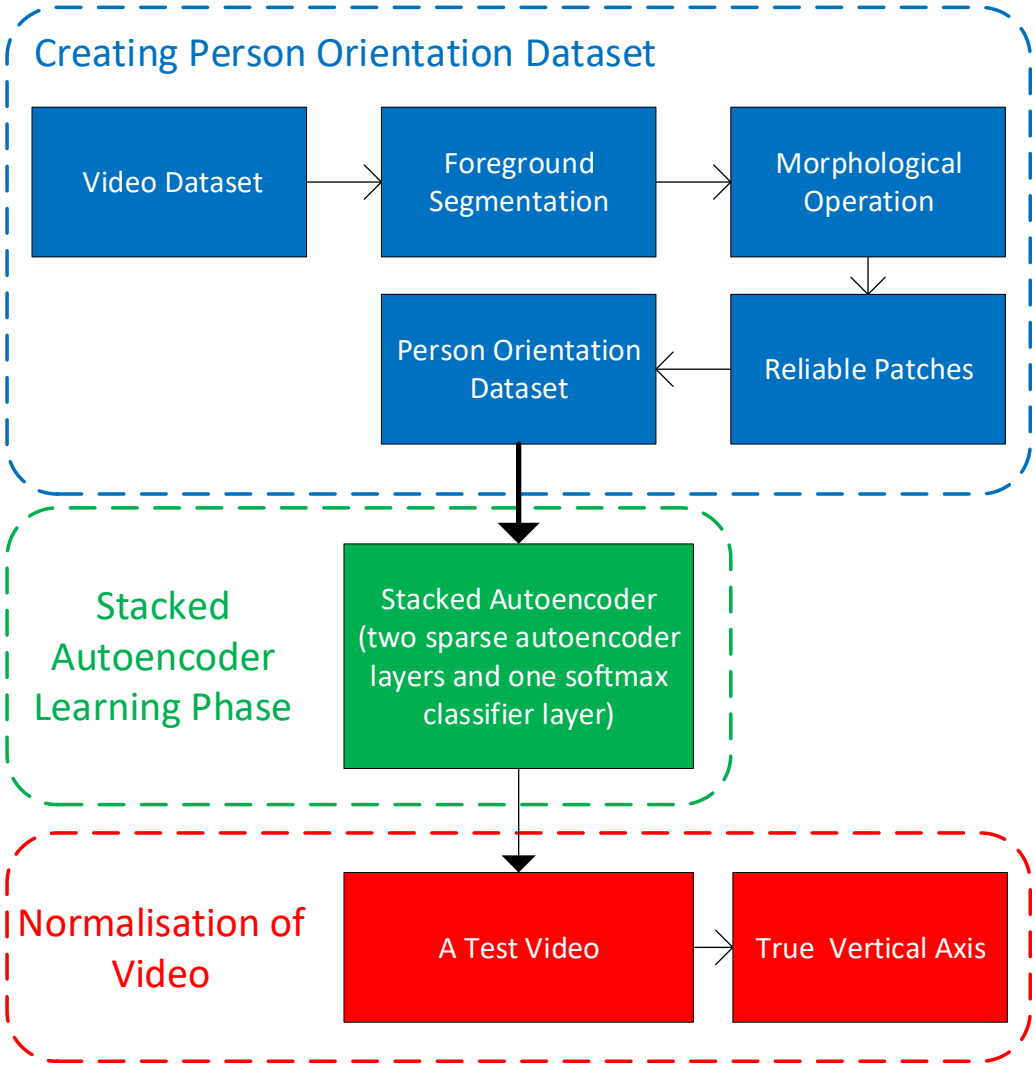


Figure 4.10: The overview of our approach for vertical axis detection

Figure 4.10 depicts a flow chart of our technique to determine the true vertical direction of video acquired using a camera with arbitrary rotation. The proposed system consists of three components. First, we automatically extract from a sports video, human images whose orientation corresponds to one of eight uniformly spaced directions (i.e.: 45 degrees apart) in the vertical plane. The true vertical direction is manually annotated (according to the eight 45 degree bins) such that this database can be used for training and testing a classifier. The annotation is based the assumption that when the players are in their upright position, the true orientation of each payer will be in the vertical direction. We call this data set the "person orientation dataset". Next, we design a stacked autoencoder which contains a softmax classifier layer and train the classifier using images from our person orientation dataset. Subsequently, testing images are employed to evaluate the performance of the classifier on the dataset and report the accuracy of the classification. Finally, we show how to normalize a given unseen video by estimating the true vertical direction using our classifier.

The main contribution of this work is an efficient way to recover the true vertical axis of an unseen video. Experiments conducted demonstrate that our proposed detector works well to effectively normalize the vertical axis of a video. Although we demonstrate the use of this approach for an outdoor sports scene, the technique is equally applicable to videos captured from other scenes.

4.3.1 The Person Orientation Dataset

In this section, we describe the process of creating the person orientation database by extracting image patches from a sports video.

The video data was collected by a sporting body and consists of eight field hockey videos captured from eight fixed cameras in the same match. Each camera captures

Figure 4.11: Example images from the field hockey video dataset.

data at 25 frames per second, at a resolution of 1888×1062, and with a static background, which contains only a few illumination variations and small amounts of noise. Representative frames from these eight views are shown in Figure 4.11. The selected frames highlight the variations among of the athlete's uniforms and angle of view. Five different kinds of uniforms appear in these videos. They are two teams' uniforms, a referee's uniform, and two goals keeper uniforms respectively. One may see that these eight fixed cameras have been placed in different corners around playing area, and the true vertical direction of videos are compromised by the need to provide maximum coverage of the playing filed.

The person orientation dataset created from the sporting videos consists of 14,704 person images. Each person image is automatically extracted from the sporting videos, to enable the evaluation of the true vertical direction, which is based on the assumption that players tend to move approximately upright. Figure 4.12 displays some representative images in this dataset. All images are transformed into gray scale and the size of all images is scaled to 50×50 pixels.

Image Patch Extraction

To create the person orientation dataset, we first need to extract image patches of players with different orientations from the sport video data. A Gaussian mixture

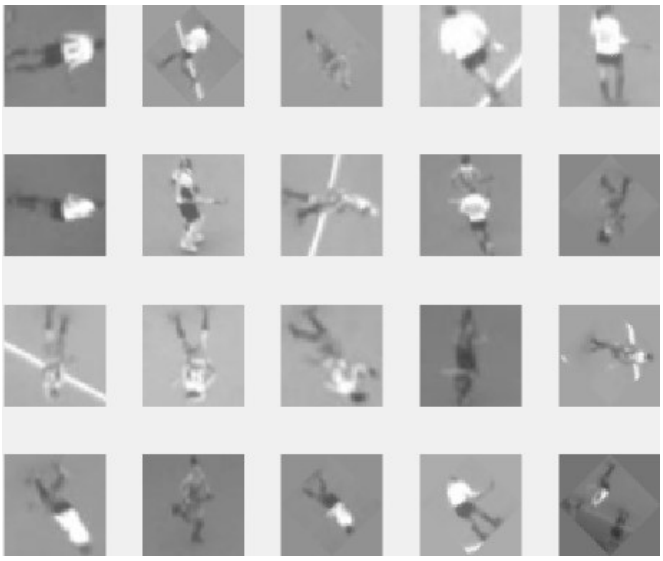
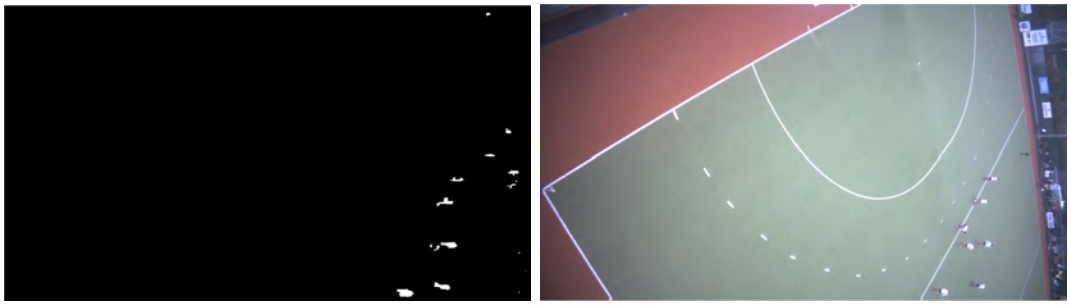


Figure 4.12: Representative images in the person orientation dataset.



(a) Foreground mask. (b) Original frame.

Figure 4.13: Representative frame showing the computed foreground mask (a) and the original frame (b)

model based method [121] is used to detect foreground in each frame. In the Gaussian mixture model, each pixel in a video is modeled by a mixture of K Gaussian distributions. The probability that a certain pixel has intensity x_t at time t is estimated using:

$$\Pr(x_t) = \sum_{j=1}^{K} \frac{w_j}{(2\pi)^{\frac{d}{2}}|\Sigma_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(x_t-\mu_j)^T\Sigma_j^{-1}(x_t-\mu_j)}, \tag{4.20}$$

where w_j, μ_j, and Σ_j are the weight, the mean, and covariance of the jth Gaussian distribution respectively. The first B distributions, which are used to model the background of the scene, are computed using:

$$B = \arg\min_b \left((\frac{\Sigma_{j=1}^{b} w_j}{\Sigma_{j=1}^{K} w_j}) > T \right), \tag{4.21}$$

where threshold T is the fraction of the total weight which is set for the background model. Subsequently, background subtraction is performed by marking any pixel that is more than 2.5 standard deviations away from any of the B distributions. A foreground mask with the coarse position of players is obtained for each frame in the videos. Figure 4.13 shows the binary foreground mask after background subtraction and the original frame.

Once a foreground mask has been generated for the estimation of player position, we analyze the blobs using morphological operations to remove noise and connect separated areas of player on foreground mask.

Since the detected blobs contain several holes due to the impact of noise, an eight-point square neighborhood is used to identify an individual player included in the foreground mask as one region. Then, the binarized foreground mask is processed by closing and opening operations to obtain a refined mask showing only the players in the scene. The closing and opening operators are respectively represented as:

$$A \bullet B = (A \oplus B) \ominus B, \tag{4.22}$$

$$A \circ B = (A \ominus B) \oplus B, \tag{4.23}$$

where A is a binary mask, B is structural element [6], and $\oplus$ $\ominus$ represent the dilation and erosion operations respectively.

In practice, not all blobs detected above are suitable for inclusion in the dataset, largely because there is some misdetection caused by noise. Rapid change in intensity over a few frames may lead to unstable backgrounds and blobs of noise. Thus, we need to filter out these erroneous blobs to prevent false candidates from being included in the dataset. To achieve this, each blob is localised using a bounding box around its centroid. Subsequently, bounding boxes whose sizes are too small or are too large are eliminated. Furthermore, bounding boxes with improper aspect ratio are also deleted from the binarized foreground mask. In our experimental setting, the proper size of bounding box is set to be between 900 and 4000 pixels to filter out small blobs of noise and the aspect ratio is constrained to the range [0.6 1.4]. In the final processing, some blobs only appear in a few discontinuous frames and these are removed as well. The remaining blobs are therefore reliable to use as the basis for the person orientation dataset.

Next, the image patches that contain reliable blobs are manually inspected and classified into one of eight directions. The diagram of the directions in our approach is shown in Figure 4.14. The angle between each adjacent direction pair is 45

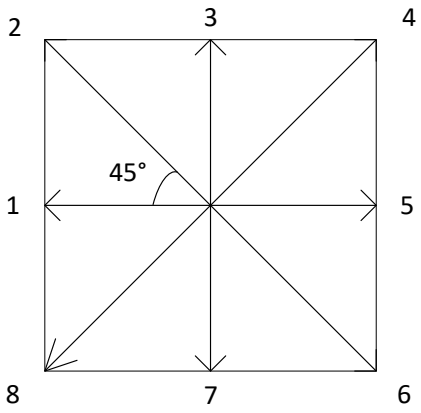


Figure 4.14: Eight vertical directions of players in our dataset.

degrees.

Dataset augmentation and labeling

Since the number of samples in each category is not the same, the distribution is not appropriate to train a robust and reliable classifier for estimating the true vertical direction of videos.

The easiest and most common method to address the issue is to artificially enlarge the dataset using a transformation [22, 23, 75]. The form of data augmentation used here is to generate image rotations. Each image is augmented such that seven extra images are generated, by applying seven rotations of 45 degrees, i.e. the transformation increases the size of our dataset by a factor of eight. Due to rotating, we cannot avoid losing pixels at the boundary of the images, and the pixels that are lost are replaced with the mean value of the image.

4.3.2 Learning a Stacked Sparse Auto-encoder

For classification, we use a sparse auto-encoder. In this section, we review the principles of a typical three-layer autoencoder and then describe the stacked sparse autoencoder used as the classifier. At the end of this section, the reasons for choosing the stacked sparse autoencoder are outlined.

Typical Sparse Autoencoder

A sparse autoencoder (AE) [147] is an unsupervised feature learning algorithm which aims to generate a sparse feature representation of high-dimensional input data. A simple, sparse autoencoder is usually represented as a neural network consisting of three layers, in which the number of neurons in the input layer is the same as that of the output layer, trained by a back-propagation algorithm. The cost function to be minimized can be written as:

$$\frac{1}{2N}\sum_{i=1}^{N}\|h_{W,b}(X_i)-X_i\|^2+\alpha\|W\|_2^2+\beta\sum_{j=1}^{l_h}\mathrm{KL}(\rho||\hat{\rho}_j). \tag{4.24}$$

The first term is an average sum-of-squares error term where N is the number of training images, X_i is the ith input image, and W and b represent weight and bias parameters of the whole sparse autoencoder respectively. Here, $h_{W,B}(X_i)$ is defined as the output of the sparse autoencoder, which may be obtained through a forward propagation of the neural network. The second term is a regularization term that tends to decrease the magnitude of the weights, and helps prevent overfitting. In the third term, l_h is the number of neurons in the hidden layer, and $\mathrm{KL}(\rho)$ is the Kullback-Leibler (KL) divergence between $\hat{\rho}_j$, i.e. the output value of the neuron j in the hidden layer and presetting sparsity parameter, ρ. α and β are the weight factors of the second term and the third term respectively.

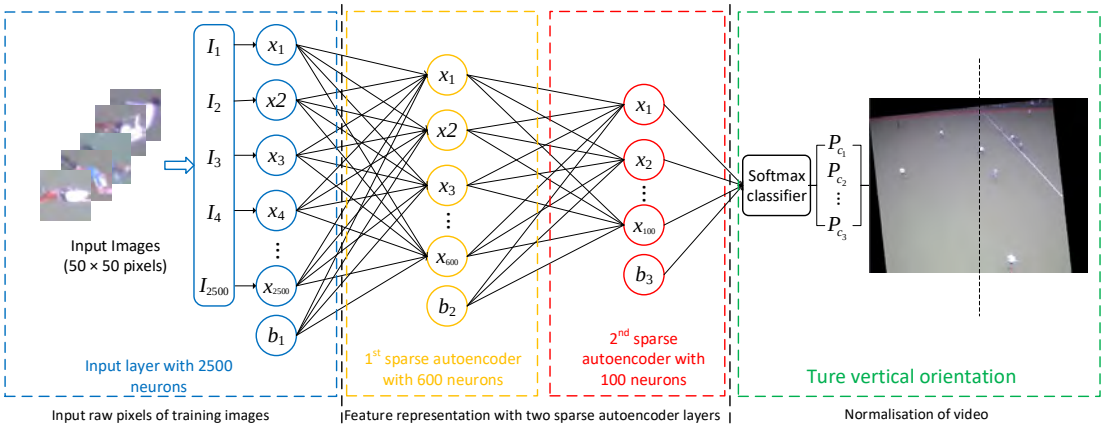


Figure 4.15: Two layer stacked autoencoder used for testing performance of our person orientation dataset. The rightmost picture is a frame which vertical axis is corrected.

The stacked autoencoder with softmax layer

A stacked autoencoder (SAE) [147] is a deep neural network consisting of multiple basic sparse autoencoders. In this work, a stacked autoencoder with two basic sparse autoencoder (AE) layers and a softmax layer is used for classification of a person's vertical orientation. Figure 4.15 illustrates the architecture of the SAE utilized.

The set of training images are denoted as $\{I_i\}_{i=1}^{N}$. For the first AE in the SAE system, the cost function shown in Equation 4.24 is minimized over parameters W^1 and b^1 via the image set $\{I_i\}_{i=1}^{N}$, where the superscript 1 of W^1 and b^1indicates that they are the parameters of the first SAE. We assume that $W_1^1$and b_1^1 are the weight and the bias between the input layer and the hidden layer respectively. Thus, the features of each image, F_i^1, extracted by the first SAE can be written as

$$F_i^1 = f(W_1^1 I_i + b_1^1), \quad i = 1, 2, \ldots N, \tag{4.25}$$

where f is defined by a sigmoid logistic function as $f(x) = 1/(1 + \exp(-x))$.

Almost repeating the same process as the first SAE, we can get a new feature set

$\{F_i^2\}_{i=1}^N$from the old feature set $\{F_i^1\}_{i=1}^N$ through the second SAE in our system. The new feature set is a further abstract representation of the original image set at which data has lower dimensionality and better classification performance. In the last layer, we link these two basic AEs with a softmax layer, which is an output layer used in classification tasks, to estimate the probability of each category.

There are several reasons for selecting such a stacked autoencoder. As shown in Figure 4.12, lots of patches are synthesized by a rotation operation and hence the difference between them is only the orientation of the person within them, and slight illumination variations. Traditional image classification methods which use a combination of hand-crafted features and a classifier largely depend on proper feature descriptor and knowledge of the specific tasks. In our case, it is difficult to choose a hand-crafted descriptor to obtain discriminative features, and more advanced classification algorithms based on unsupervised feature learning are needed to classify images in this dataset.

Since we have a large the number of samples in the dataset (14,704), it is possible for us to train a classifier with a complex structure. The autoencoder, which is a mature deep learning architecture, is easier to train and has a more concise structure than other deep learning methods (e.g. deep belief network, convolutional neural network, etc.). Thus, the autoencoder classifier utilized in our dataset is chosen to balance of performance and efficiency.

Another reason for choosing an autoencoder as a classifier is due to the fact that the last layer, which is a softmax layer, can output the probability of each category of one image. The next subsection will describe how to compute the angle of true vertical direction according to the probabilities of the reliable image patches.

Suppose that there is a reliable image patch, A, extracted from a test video to estimate the vertical direction. After being processed by the stacked autoencoder,

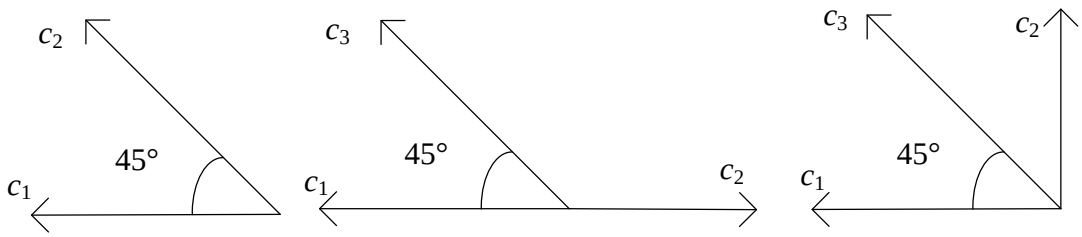


Figure 4.16: Computation of player's orientations. The first column: the two categories c_1 and c_2 are adjacent. The second column: c_1 and c_3 are in opposite directions. The third column: c_1 is orthogonal to c_2, and c_3 is in the middle of c_1 and c_2

eight probabilities $\{P_i\}_{i=1}^{8}$ in descending order which corresponding to categories $\{c_i\}_{i=1}^{8}$ may be obtained. In practice, three cases may occur when computing the angle, and are shown in Figure 4.16 and illustrated as follows.

1. Classes c_1 and c_2 are adjacent. This case is the simplest one when computing angle. One may compute the angle θ from c_1 to c_2 as follows:

$$\theta = 45^{\circ}(\frac{P_{c_2}}{P_{c_1}}). \tag{4.26}$$

2. The first two classes c_1 and c_2 are in opposite directions, or nearly opposite directions. The third largest probability c_3 is introduced to assist the angle computation. By comparing c_1 with c_3 and c_2 with c_3, the class whose direction is furthest away from the others is deleted and then we compute angle using the probabilities of remaining two classes like Equation 4.26.

3. c_1 is orthogonal to c_2. If the third class is in the middle of the first two, the angle from c_2 to c_1 may be computed as

$$\theta = 45^{\circ}(1 + \frac{P_{c_3}}{P_{c_1}})(\frac{P_{c_2}}{P_{c_1} + P_{c_3}}). \tag{4.27}$$

   For those cases where the third class is not in the middle of c_2 and c_1, we ignore them.

Finally, the true vertical axis of a video may be computed as the average of the angles of the reliable image patches.

4.3.3 Experimental Design and Results

The person orientation dataset created above contains 14,704 images which have 8 direction classes. For each class, 80 percent of the images are randomly selected for training and the remaining images are used for testing.

Since the size of the images is 50×50, the number of neurons dimensions in the input layer for the vectorized image is set to 2500. Two SAE layers for intermediate feature extraction contain 600 and 100 neurons respectively. In the last softmax layer, 8 neurons are employed to output the probability of each category. The architecture of the autoencoder is shown in Figure 4.15

We employ a greedy layer-wise approach for pre-training the SAE by training two AEs and one softmax layer in order. After pre-training is done, all three layers are combined together to form a compact SAE system. Finally, fine tuning is applied to the SAE system to get a classification model which has good accuracy. The classification results evaluated by the stacked autoencoder are shown in Figure 4.17 via a confusion matrix. The rows of this confusion matrix plot the predicted class of the vertical direction, and the columns show the true class. The diagonal cells display where the true class and predicted class match. Other cells in off-diagonal show instances where the classifier has made errors. One may see that the misclassified samples are concentrated around the true class. It should be noted that there is a small number of errors in the ground truth labels due to the difficulty of the manual annotation. The large variation in player pose, e.g. running, squatting, makes the data difficult to label. Thus the method should achieve a higher accuracy than its classification accuracy based on the manual

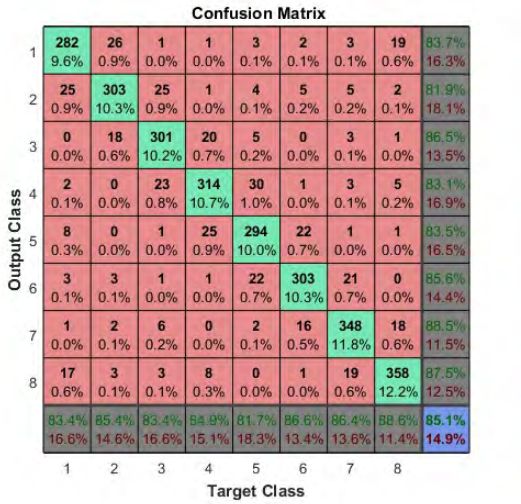


Figure 4.17: The confusion matrix of the classification results from the vertical direction dataset trained by the stacked autoencoder.

annotation. The right-most column shows the accuracy for each predicted class, while the row at the bottom of the plot shows the accuracy for each true class. The cell in the right-bottom of the plot shows the overall accuracy 85.1%. This accuracy indicates that the performance with the dataset is acceptable. It should be noted that there are still a few samples that are misclassified as the opposite direction. In a real case, their correct angle may be computed by the strategy proposed above.

The learned stacked autoencoder filters are shown in Figure 4.18. An intriguing pattern is observed in the filters. One may see that all the filters attempt to capture the directional pattern of the players and the details of uniform as well as background are ignored by the filters. When there is some background in images, several filters become low-pass filters, to suppress the responses from background regions.

The normalization of the vertical axis for any given test video is based on the classifier trained above. The normalization is based angles estimated by the

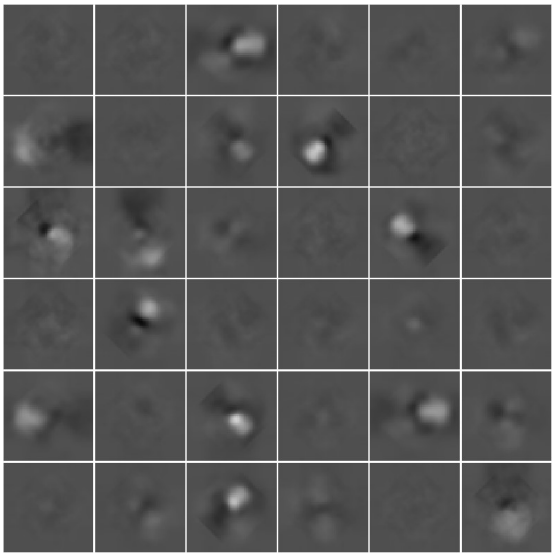


Figure 4.18: Visualization of 600 neurons from the first layer of the SAE. As expected, the plot shows detailed boundary features and orientation of the player.

stacked autoencoder classifier. Figure 4.19 shows the results of the estimated angle for a particular video. One may see that only reliable patches are used for the estimation of angles. From left to right, the estimated angles are 40.853°, 27.398°, 0.67059°, −34.702°, 0.000°5 respectively. We compute the angle of the (true) vertical axis of the video as the average of above angles, i.e. 6.3568°. The normalization is done by a rotation operation on the original video as illustrated in Figure 4.20. Rotating the video rectifies the camera rotation to recover the true vertical axis. This simplifies further automatic processing of the video, for example for person detection tasks, as well as providing a good view of the scene for human perception.

4.4 Chapter summary

Calibrating sports videos captured from a single camera is a challenging task. Previous works only consider calibrating a key frame by using clearly visible and

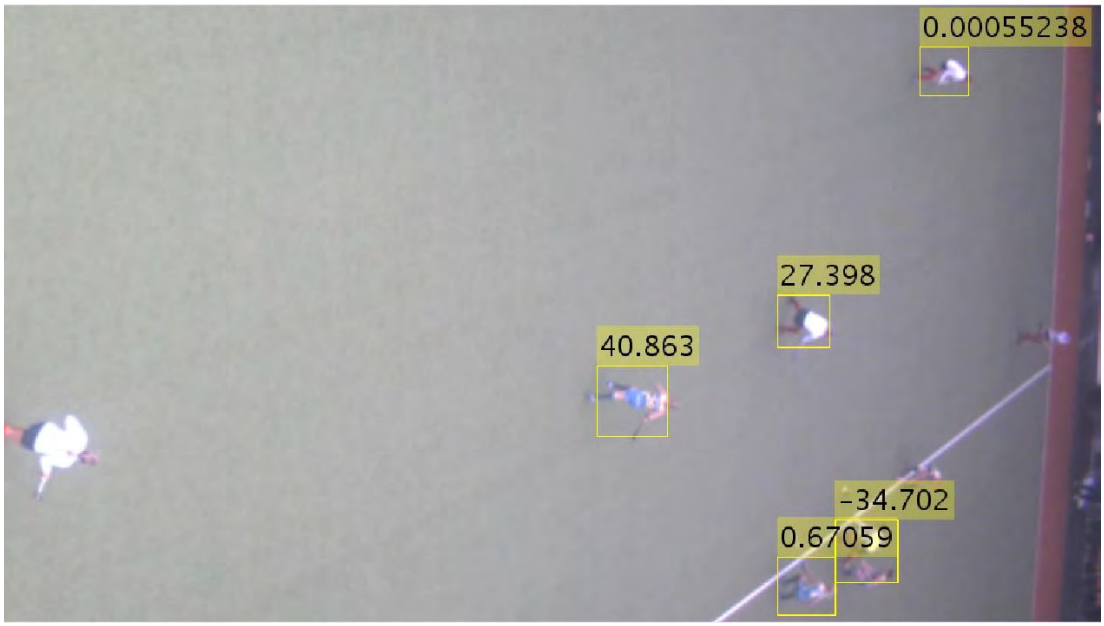


Figure 4.19: : Estimation of angles in a video via patches. From left to right, the estimated angles of the players are 40.853°, 27.398°, 0.67059°, −34.702°, 0.0005° respectively. The angles are computed in clockwise from the class 1 shown in Figure 4.13 to the class 8.

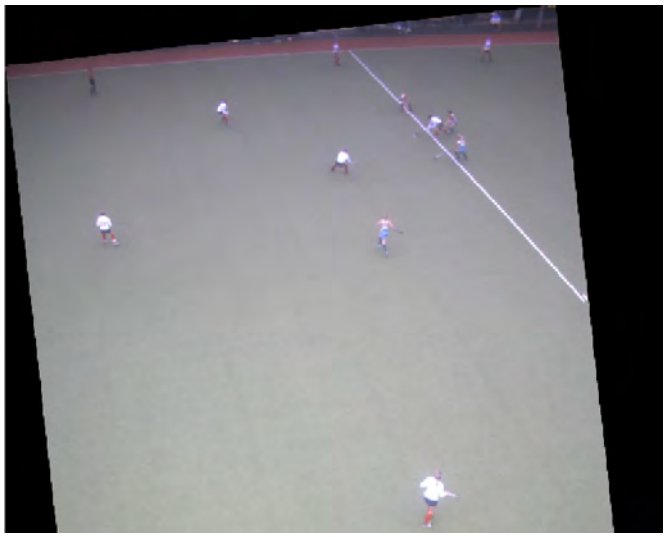


Figure 4.20: : A rotated video which has been normalized along the world vertical axis.

sufficient pitch lines, and then use the relationship between each frame to derive the homographies in remaining frames. These works may fail when a given video does not contain such a key frame. In this chapter, an innovative automatic camera calibration system for a single camera is proposed based on three components: a robust linear panorama generating module, a playing area estimation module, and a homography estimation module. This system calibrates each individual frame through calibrating the panorama generated from these frames first. It utilizes the holistic structure of the frames in a video rather than searching a key frame in isolation. Thus this system may still work well, even if a given video does not contain any a frame.

This chapter also has proposed the semi-automatic creation of a person orientation image dataset from sports video data, and proposes a method for estimating the true vertical axis of a given video to normalize the orientation of the video to enable the easy application of further analytics, and to provide improved video for human perception. Evaluation of our classifier on test data shows an accuracy of over 85%. The experiments conducted on hockey field video dataset show that the proposed system is able to estimate the true vertical axis of an input video accurately.

While the methods proposed in this chapter have demonstrated promise, there are still limitations. Specifically, the panorama system can be improved by considering the relationship between each video shot rather than each consecutive frame. This is because that when the camera stays in the same location a long time, the accumulated error in the panorama will increase significantly. Moreover, some prior knowledge such as camera motion (e.g. focal length, translation, panning, etc.) can be used to aid this process.

Chapter 5

Key Frame Generation for Camera Calibration in Broadcast Sports Videos

5.1 Introduction

Camera calibration for broadcast sports videos refers to the task of aligning video frames with an absolute coordinate system determined by a court template, and as such the objects of interest (e.g. ball, player, etc) in the playing area can be localized in real world coordinates. It is the most fundamental step for sports multimedia applications, such as player localization [63, 101, 143, 160], tactical analysis [87, 97, 159], virtual advertisement insertion [15, 89, 108], etc., all of which rely heavily on the knowledge of the real world coordinates of players.

Stemming from image registration [13, 162], the most common pipeline for camera calibration in broadcast sports video is to detect key frames within a video. For a

Figure 5.1: A typical frame in broadcast basketball video. Camera calibration is difficult due to the highly customized floor and fragmentary court structure, which does not contain sufficient standard land marks. In addition, large auditorium area and moving players introduce a large portion of dynamic pixels which adversely affect camera calibration.

key frame, a group of specific geometric features are used to register the frame with a standard court template. Then camera calibration for other frames is performed by using transformations between themselves and the key frames. While such geometry-based methods [34, 51, 63] work well when videos contains a sufficient number of high-quality key frames, calibrating cameras without such information is still challenging.

To avoid the instability of detecting and matching key frames, an alternative family of methods solves this task in a global manner. Specifically, [5, 7, 41, 88] attempt to minimize the pixel-wise discrepancy between a court template and all frames under the assumption that dynamic pixels (i.e. moving players) are sufficiently sparse in each frame. However, these pixel-based methods may fail when applied to a court-scale sport, such as volley ball, basketball, etc, where moving players

and spectators in auditorium areas are prominent and hence inevitability result in a large number of dynamic pixels.

The method presented in Chapter 4 addressed the lack of a key frame by generating a key frame for a given video. However, its performance may be significantly degraded when dealing with videos containing large numbers of dynamic pixels. Figure 5.1 further illustrates our motivation, which is driven by analyzing highly dynamic frames without distinctive geometric features. Frames such as that shown in Figure 5.1 are often captured when exciting events are occurring, and seldom contain the entire geometric structure of the court which is necessary for geometry-based camera calibration. Moreover, the highly customized floor, which makes detection of meaningful geometric features difficult, exacerbates this situation. Last but not least, moving players, an engaged audience, and relative motion between cameras brings a significant number of dynamic pixels, which adversely affects the performance of the key frame generation algorithm presented in Chapter 4. In addition, the method of Chapter 4 does not use the properties of broadcast cameras, which are typically a pan-tilt-zoom (PTZ) camera fixed in a specific location of a sporting stadium. Specifically, PTZ camera motion only contains rotations and intrinsic parameters changes. The degrees of freedom for the camera is five, including three for the rotation and two for focal length in the x and y axis respectively. Therefore, estimating nine-DOF homographies between frames is unnecessary, and estimated homographies which have nine-DOF may break the underlying geometry relationship that results from using PTZ cameras.

To address the aforementioned issues, the work presented in this chapter aims to combine the benefits of geometry-based and pixel-based methods simultaneously, and seeks to create a single camera calibration algorithm that does not rely on key frame detection by efficiently utilizing all frames in a global manner. In contrast to existing methods, our method reformulates the difficult key frame detection

problem as the process of generating a panoramic view of the court from all frames in a global manner. The generated court panorama, which serves as 'key frame', is then used to calibrate all other frames in the video. To improve the quality of generated key frames, we present a pipeline for camera calibration which is able to decompose estimated homographies using broadcasting sports PTZ cameras.

5.1.1 Related Work

A review of all relevant camera calibration literature is beyond the scope of this chapter (please refer to [54] for more details). Here, we discuss the most relevant existing approaches for calibrating broadcast sports video.

A significant number of camera calibration methods [34, 51, 61, 63, 63, 88] have been proposed using the prior knowledge of geometric layout of the court where sporting events are held. All of these methods are reliant on the same assumption, that sufficient geometric primitives appear clearly on the playing area, such as circles and lines. Farin *et al.* [34] present a sports video frame registration algorithm which utilizes the order of the lines in the court model. It is able to align a frame to a specified court model as long as the frame contains a sufficient number of straight lines. Han *et al.* [51] extend the homography obtained from [33, 34] to a simplified camera matrix by assuming that the camera roll-angle is zero. Similarly, Hsu *et al.* [61] presents a method which estimates the homography between a frame and the standard volleyball court model using two parallel lines on the net. Liu *et al.* [88] also report a novel camera calibration algorithm for sports video using straight lines. Hu *et al.* [63] first classifies basketball scenes into a right-side court view, a left-side court view, or a middle court view. Different camera calibration strategies are proposed for different kinds of court views based on [51]. This method works well when a frame can be successfully

classified according to the specific marks on the court. However it is hard to generalize to other sports and the large amount of prior knowledge required makes it cumbersome. As we demonstrate, the aforementioned methods are prone to failure when clearly visible geometric primitives are insufficient due to player occlusion, blur, illuminations, etc.

An alternative type of approach to bypass the search for key frames is to find redundant pixel-level correspondences between frames and the standard court template. For example, Lu *et al.* [93] detect the boundaries of the basketball court using a canny detector and extract all the pixels of the boundary lines as dense correspondences. Subsequently, frames are calibrated by applying iterated closest point (ICP) [153] which computes the transformation between the court template and dense correspondences. Similarly, [87, 146] extract dense correspondences from the middle part of the court and then warp frames to the template. A few recent methods [7, 41] exploit the use of every pixel in one frame to estimate camera parameters. The camera calibration problem is reformulated as a sparse optimization task where outlier pixels (e.g. moving players and spectators in auditorium areas) are assumed to be sparse in each frame. Note that assuming outlier pixels between adjacent frames are sparse could be invalid for many broadcast videos, as frames can be severely blurred when the camera undergoes quick pan-tilt-zoom motions. Moreover, continuous changes in camera position and highly dynamic spectator areas may degrade the performance.

5.1.2 Proposed Approach

Several pioneering approaches [140, 152] have explored the task of camera calibration without the need for key frame detection. The method proposed here superficially resembles [140], which also generates a panorama to assist camera

calibration. There are two important differences between [140] and our algorithm:

1. We estimate all camera intrinsic and extrinsic parameters simultaneously under the consideration of the properties of broadcast cameras instead of only using the homography, which breaks the underlying geometric constrains.

2. A novel camera parameter optimization function is designed for broadcast sports video, which enables generation of a high-quality panorama.

Compared to the method presented in Chapter 4, the approach proposed in this chapter has a better camera calibration accuracy and is able to generate a high-quality panorama of the whole playing area. In addition, our method is able to process broadcast videos and estimate all camera parameters of every single frame, owing to the novel camera calibration algorithm.

Figure 5.2 outlines the pipeline of our method. It consists of three components:

1. Camera parameter initialization. Camera intrinsic, extrinsic, and distortion parameters of every single frame are initially estimated using the novel proposed algorithm which considers the properties of broadcast sports cameras. In addition, a novel broadcast video pre-processing step is proposed to segment videos and provide player, scoreboard, and playing area masks for further use.

2. Key frame generation. A panoramic view of the playing area, which is regarded as a generated key frame, is reconstructed by optimizing all cameras parameters jointly. Moreover, the objective function used for optimization is designed to be suitable for high-quality panorama generation from broadcast video.

3. Camera calibration. Camera calibration on every single frame is performed using the transformation between itself and the aligned panorama, which is registered to a standard court template.

The main contribution of this work is three-fold:

1. This chapter presents a unified framework for camera calibration in a broadcast sports video. Instead of directly searching for key frames for registration, we generate them from highly dynamic frames using the proposed sports panorama technique.

2. We introduce a novel camera calibration method for sports video frames captured by PTZ cameras. Camera intrinsic, extrinsic, and distortion parameters can be estimated and jointly optimized by solving the proposed optimization function, which is designed under the constraints of sports scenes.

3. A novel image saliency algorithm for sports video frames is introduced by taking into account sports video properties. Our qualitative results show that the proposed method significantly outperforms previous state-of-the-art methods in terms for sports scenes.

The remainder of chapter is organized as follows: Section 5.2 describes the video preprocessing module used for video segmentation and mask generation. Afterwards, Section 5.3 introduces the PTZ camera model which is used to parameterize broadcast sports video. In Section 5.4, a coarse-to-fine camera parameter optimization pipeline is proposed for each frame. Section 5.5 generates key frames using the camera parameters of all the frames and registers all the frames to the court template. Qualitative and quantitative experiments on

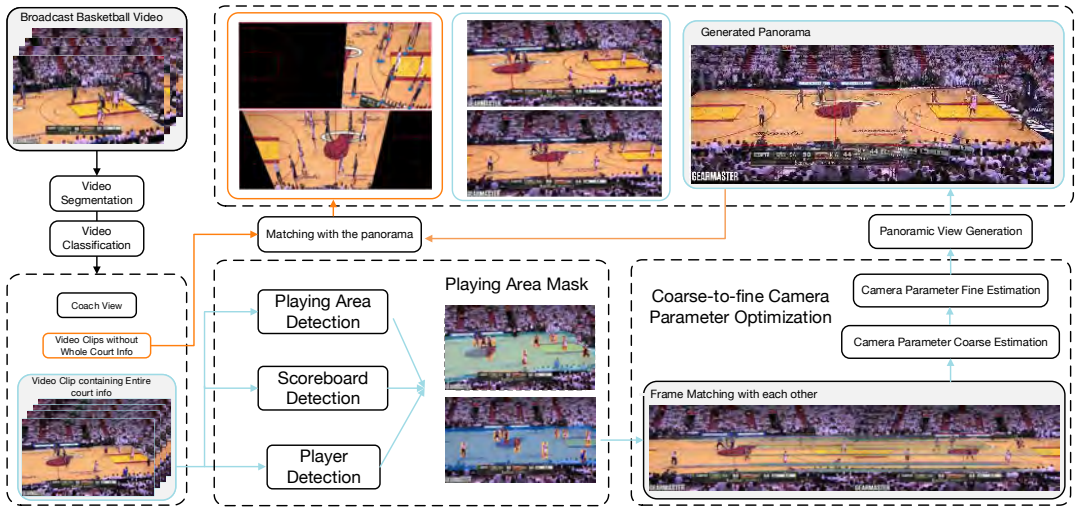


Figure 5.2: The pipeline of the proposed method. First, a complete broadcast video is segmented into short video clips utilizing with the proposed video segmentation approach. Then irrelevant video clips which are less informative to viewers (e.g. close-up, commercials) are filtered out according to the dominant color ratio and positions of lines. In the key frame generation module, the initial homographies between each frame pair are computed using detected correspondences and then those homographies are used to further compute the camera extrinsic parameters and camera intrinsic parameters. Finally, every frame is registered to the standard basketball template using a direct linear transformation (DLT).

broadcast basketball videos are shown in Section 5.6. We conclude the chapter in Section 5.7.

5.2 Video Preprocessing

A typical broadcast video may last longer than a hour. To make camera calibration easier, we first segment it into consistent short video clips using the proposed video shot boundary detector. Then the segmented video clips are classified into two categories [29], i.e., normal spectator-view shots, and close-up shots (see Figure 5.3 for an example), according to the dominant color ratio and positions of lines as used by [63]. Irrelevant shots (e.g. close-ups) that are less informative for

(a) Close-up (b) Normal spectator-view

Figure 5.3: Two kinds of segmented video. (a) A close-up view frame. (b) Two normal spectator-view frames. Since close-up frames are less informative in terms of sports frame registration, we filter them out directly.

player locations are filtered out. To generate the panoramic view of the court, our method requires that the camera pans from one side to another such that the whole court is captured. While this situation happens frequently as possession changes from side to side and each instigates an attack, a few video clips still do not contain such whole court information. Since all video clips share the same panoramic view, the video clips that do not contain the whole court can be calibrated using the panoramas generated by other video clips as per the strategy demonstrated in [140]. In the following subsections, we discuss the preprocessing in more details.

5.2.1 Video Shot Segmentation

The standard approaches [29, 93] for video segmentation are to train a video frame classifier using user-collected data to locate the boundary of video shots, i.e., a shot boundary is labeled when its frame category is different from its precedent frame. While these methods work well, it is often inconvenient for a user to annotate training data which is highly specific to the input video. To address this issue, we propose a simple video shot segmentor based on the observation that a peak of the histogram difference appears at the boundary of two video shots.

The shot boundary score S_{boundary} of the tth frame $\mathcal{F}_t$ can be computed using:

$$S_{\text{boundary}}(\mathcal{F}_t) = \left\| \begin{bmatrix} H(\mathcal{F}_{t-2}) & H(\mathcal{F}_{t-1}) & H(\mathcal{F}_t) \end{bmatrix} L \right\|_1, \tag{5.1}$$

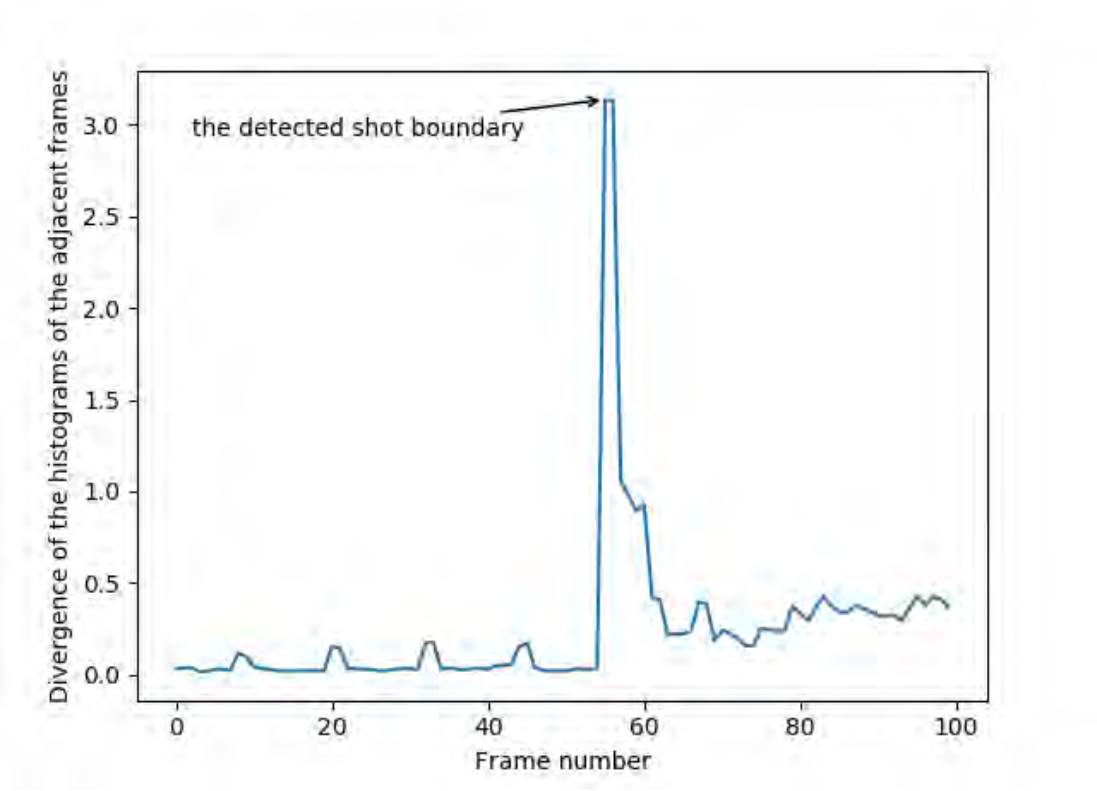


Figure 5.4: An illustration of the video shot segmentation output. The plot shows the shot boundary result over a portion of video. We can see that a peak appears at the boundary of two video shots.

where L is the one-dimensional discrete Laplace filter $\begin{bmatrix} 1 & -2 & 1 \end{bmatrix}^\top$, and $H(\cdot)$ is the histogram function. Specifically, we first compute the one-dimension histogram of each frame, i.e., $H(\cdot)$ outputs a $b \times 1$ vector where b is the number of bins in the histogram. Then we use the discrete Laplace operator to compute the divergence of the histograms of adjacent frames to locate shot boundaries. Figure 5.4 shows a plot of the output of Equation 5.1 over a portion of video, and a peak at the boundary of two individual video shots is clearly visible.

5.2.2 Playing Area Mask Generation

To reduce the difficulty of subsequent analysis, the playing area in each frame is located using a generated mask, where the scoreboard, players, and the auditorium area are removed.

Since the position of the scoreboard area in each frame is consistent, we can locate it once for all frames. Firstly, we randomly choose 20 frames from the input video and then perform pyramid meanshift filtering on every frame to suppress noise and small edges. The filtered frames are then used to calculate gradients using the Sobel operator. Finally, the scoreboard location is obtained by performing the *And* operation on all Sobel-operated frames.

Regarding the playing area, the basketball court mask is obtained using dominant color segmentation in the LUV color space. To improve the quality of the playing area mask, morphological operations including region-growing and small object elimination are employed. Moreover, we also train a player detector using Faster-RCNN [106] based on the InceptionV2-ResNet [128] architecture to filter out players, which introduce lots of dynamic pixels and may be harmful to camera calibration. More details on player detector can be found in Section 5.6.2. Figure 5.5 shows final masks for several examples.

5.3 Camera Model for Broadcast Sports Videos

Previous works [63, 93, 140, 152], which only use a homography are prone to errors due to the fact that an unconstrained single homography computed by correspondence pairs from any two frames cannot precisely model the relationship of the two. More specifically, a homography breaks the implicit geometric con-

Figure 5.5: Four examples of playing area mask generation. The areas of the scoreboard, auditorium, and players are filtered out. The playing areas are shaded using four different colors.

straints of two frames. Broadcast videos are typically recorded by pan-tilt-zoom (PTZ) cameras, which can only pan, tilt, and zoom around a fixed camera center and hence the relationship between two frames is implicitly encoded by geometric constrains (e.g. the pose matrix can only contain rotation and not translation, all frames share the same camera center, etc.). Furthermore, when the lens has a non-negligible distortion, only making use of a single homography may result in high calibration error [4]. To tackle this problem, we use a pinhole camera model [54] without translation parameters to model every single frame in the video. All camera parameters are changing while a PTZ camera is being operated. In the following, we first introduce a distortion-free camera model and then extend it to deal with lens distortion.

A point in the real world coordinate system and its corresponding projective image point in the frame $\mathcal{F}_t$ can be related by the following camera model:

$$\mathbf{p}_t \sim \mathbf{K}_t \mathbf{R}_t \mathbf{p}', \tag{5.2}$$

$$\mathbf{K}_t = \begin{bmatrix} f_{x,t} & & o_{x,t} \\ & f_{y,t} & o_{y,t} \\ & & 1 \end{bmatrix}, \qquad \mathbf{R}_t = \begin{bmatrix} r_t^{00} & r_t^{01} & r_t^{02} \\ r_t^{10} & r_t^{11} & r_t^{12} \\ r_t^{20} & r_t^{21} & r_t^{22} \end{bmatrix}, \tag{5.3}$$

where $\mathbf{p}' = [x_{\mathbf{p}'}, y_{\mathbf{p}'}, z_{\mathbf{p}'}]^\top$ is a point on the standard pitch plane represented by the real-world coordinate system, and $\mathbf{p}_t = [x_{\bar{\mathbf{p}}_t}, y_{\bar{\mathbf{p}}_t}, 1]^\top$ is its corresponding projection in the homogeneous frame coordinate system of $\mathcal{F}_t$. $\mathbf{K}_t$ is the 3×3 camera intrinsic matrix in which $f_{x,t}$, $f_{y,t}$, and $o_{x,t}$, $o_{y,t}$ represent the focal length and the optical center of the x and y axis respectively. $\mathbf{R}_t$ is the 3×3 rotation matrix between the world coordinate system and the camera coordinate system. The 3×3 $\mathbf{R}_t$ has only three degrees of freedom (DoF) as it can be represented as a 3×1 Euler–Rodrigues vector [54].

The two most common types of lens distortion in PTZ cameras are radial and decentering distortion. Radial distortion is radially symmetric at the optical center of the lens and can be usually classified as either *barrel distortion* or *pincushion distortion* according to an inward or outward shift of the image point from the initial perspective projection. Decentering distortion is caused by the misalignment of the optical centers of the lens stack in the PTZ camera, and it has both a radial and a tangential component. By taking into account these two distortion types, the $\mathbf{p}_t$ and its distorted point $\bar{\mathbf{p}}_t$ can be related by:

$$\bar{\mathbf{p}}_t = \alpha_t(\mathbf{p}_t)\mathbf{p}_t + \beta_t(\mathbf{p}_t), \tag{5.4}$$

where $\alpha(\cdot)$ and $\beta(\cdot)$ are the radial and decentering distortion vectors respectively. They can be expressed as follows:

$$\alpha_t(\mathbf{p}_t) = 1 + k_{1,t} r_{\mathbf{p}_t}^2 + k_{2,t} r_{\mathbf{p}_t}^4, \tag{5.5}$$

and

$$\beta_t(\mathbf{p}_t) = \begin{bmatrix} 2j_{1,t}\frac{x_{\mathbf{p}_t}-o_{x,t}}{f_{x,t}}\frac{y_{\mathbf{p}_t}-o_{y,t}}{f_{y,t}} + j_{2,t}(r_{\mathbf{p}_t}^2 + 2\frac{x_{\mathbf{p}_t}-o_{x,t}}{f_{x,t}})^2 \\ j_{1,t}(r_{\mathbf{p}_t}^2 + 2\frac{y_{\mathbf{p}_t}-o_{y,t}}{f_{y,t}})^2 + 2j_{2,t}\frac{x_{\mathbf{p}_t}-o_{x,t}}{f_{x,t}}\frac{y_{\mathbf{p}_t}-o_{y,t}}{f_{y,t}} \end{bmatrix} \tag{5.6}$$

where $r^2_{\mathbf{p}_t} = (\frac{x_{\mathbf{p}_t} - o_{x,t}}{f_{x,t}})^2 + (\frac{y_{\mathbf{p}_t} - o_{y,t}}{f_{y,t}})^2$, k_1, k_2, and j_1, j_2 are two parameters for radial and tangential distortion respectively. To simplify the notation, we use:

$$\bar{\mathbf{p}}_t = \text{distort}(\mathbf{p}_t), \tag{5.7}$$

and

$$\mathbf{p}_t = \text{undistort}(\bar{\mathbf{p}}_t) \tag{5.8}$$

to represent the distort and undistort operation respectively.

5.4 A Coarse-to-fine Camera Parameter Optimization Method

5.4.1 Homography Estimation

Once the camera model for each frame in a video clip has been set up, we start to estimate all camera parameters of every frame in a coarse-to-fine way. In the coarse stage, we first compute homographies between each pair of frames using the correspondences, and decompose it into camera intrinsic, extrinsic, and distortion parameters using the proposed novel algorithms. In the fine stage, those coarse camera parameters are optimized using the proposed objective function which is designed according to the properties of the broadcast basketball video.

The homography between $\mathcal{F}_t$ and $\mathcal{F}_s$ can be obtained by relating the same point $\mathbf{p}'$ in Equation 5.2 by:

$$\mathbf{H}_{t,s}\mathbf{p}_t \sim \mathbf{p}_s, \tag{5.9}$$

which satisfies

$$\mathbf{H}_{t,s} = \mathbf{K}_s\mathbf{R}_{t,s}\mathbf{K}_t^{-1}, \tag{5.10}$$

$$\mathbf{R}_{t,s} = \mathbf{R}_s\mathbf{R}_t^{-1}. \tag{5.11}$$

$\mathbf{p}_t$ and $\mathbf{p}_s$ are a pair of correspondences falling on $\mathcal{F}_t$ and $\mathcal{F}_s$ respectively.

The most common way to estimate $\mathbf{H}_{t,s}$ is to first find sufficient correspondences (typically the number of the correspondences should be greater than 4) between two frames and then use RANSAC to compute the homography by filtering out false positives. In order to achieve this, finding good correspondences is crucially important. In our system, following the work in Chapter 4, speeded-up robust feature (SURF) [9] is employed to detect feature points and then the DAISY [131] descriptor is used to describe dense information around the detected points. The reasons for selecting SURF and DAISY have been presented in Chapter 4.

The obtained correspondence set is not perfect because it contains many false positives, which typically fall on moving objects (e.g. players, spectators in auditorium area, etc.) and the scoreboard. First, we filter out the false positives which fall outside the playing area using the generated mask outlined in Section 5.2.2. Subsequently, other kinds of outliers are removed by utilizing random sample consensus (RANSAC). The score function for RANSAC is computed iteratively as follows:

$$\min_{\mathbf{H}_{t,s}} \sum_{p}^{P_{t,s}} \|\mathbf{H}_{t,s}\bar{\mathbf{p}}_t^p - \bar{\mathbf{p}}_s^p\|^2, \tag{5.12}$$

where $P_{t,s}$ is the total number of correspondences between $\mathcal{F}_t$ and $\mathcal{F}_s$. Finally, the homography with the lowest score is selected as the best consensus. Figure 5.6 shows the detected correspondences in two frames.

Considering that what we are dealing with is a video sequence such that consecutive frames have a clear temporal order, a probability model [13] is not employed here to estimate the homography of one frame in relation to every other frame. A frame is only registered with its nearest 10 frames. In what follows, we describe the proposed pipeline to decompose this initially estimated homography to obtain the coarse camera intrinsic, extrinsic, and lens distortion parameters.

Figure 5.6: Detected correspondences and RANSAC matching examples. The first row shows two frames with the detected key points filtered by the generated masks. The second row shows the RANSAC matching result obtained from these two frames.

5.4.2 Homography to Camera Intrinsics

The camera intrinsic $\mathbf{K}_t$ can be further decomposed into the multiplication of two matrices $\mathbf{U}_t$ and $\mathbf{V}_t$ as follows:

$$\mathbf{K}_t = \mathbf{U}_t \mathbf{V}_t, \tag{5.13}$$

where

$$\mathbf{U}_t = \begin{bmatrix} 1 & & o_{x,t} \\ & 1 & o_{y,t} \\ & & 1 \end{bmatrix}, \qquad \mathbf{V}_t = \begin{bmatrix} f_{x,t} & & 0 \\ & f_{y,t} & 0 \\ & & 1 \end{bmatrix}. \tag{5.14}$$

Let $\hat{\mathbf{H}}_{t,s}$ represent the 3×3 decoupled homography which does not contain the optical center of $\mathcal{F}_t$ and $\mathcal{F}_s$, and as such it can represented by:

$$\hat{\mathbf{H}}_{t,s} = \mathbf{V}_s \mathbf{R}_{t,s} \mathbf{V}_t^{-1}. \tag{5.15}$$

By expanding $\hat{\mathbf{H}}_{t,s}$ and then utilizing the properties of the rotation matrix $\mathbf{R}_{t,s}$, the coarse focal length in $\mathcal{F}_t$ and $\mathcal{F}_s$ can be computed under the assumption that $f = f_{x,t} = f_{y,t} = f_{x,s} = f_{y,s}$ as follow [115].

$$f = \sqrt{f_1 f_2}, \tag{5.16}$$

where f_1 and f_2 are the two coarse estimates derived from $\hat{\mathbf{H}}_{t,s}$, and

$$f_1 = \sqrt{\frac{(\hat{\mathbf{H}}_{t,s}^{12})^2 - (\hat{\mathbf{H}}_{t,s}^{02})^2}{(\hat{\mathbf{H}}_{t,s}^{00})^2 + (\hat{\mathbf{H}}_{t,s}^{01})^2 - (\hat{\mathbf{H}}_{t,s}^{10})^2 - (\hat{\mathbf{H}}_{t,s}^{11})^2}}. \tag{5.17}$$

and

$$f_2 = \sqrt{-\frac{\hat{\mathbf{H}}_{t,s}^{02}\hat{\mathbf{H}}_{t,s}^{12}}{\hat{\mathbf{H}}_{00}\hat{\mathbf{H}}_{10} + \hat{\mathbf{H}}_{t,s}^{01}\hat{\mathbf{H}}_{t,s}^{11}}}. \tag{5.18}$$

By observing Equation 5.17 and 5.18, one can see that these two equations may suffer numeric instability when $\hat{\mathbf{H}}_{t,s}$ approximates the identity matrix. More specifically, the denominators of Equation 5.17 and 5.18 approach 0 and as such the computed f is a very large number. This situation frequently happens when applying this algorithm to the compute homography between adjacent frames. This is because the relative motion between two adjacent frames is small. Small motions mean that $\mathbf{H}_{t,s}$ is very close to a 3×3 identity matrix and the focal lengths of two frames are almost same. Therefore $\hat{\mathbf{H}}_{t,s}$ is also close to the identity matrix. Moreover, the misalignment of the detected correspondences in correspondences acquisition exacerbates this phenomenon. Another draw back is that [115] calculates focal lengths under the strong assumption that $f = f_{x,t} = f_{y,t} = f_{x,s} = f_{y,s}$, which does not hold in the real world.

To address the aforementioned issues, we propose a novel focal length estimation algorithm. This challenging problem is solved in an optimization manner and as such it is able to work even if $\mathbf{H}$ is an identity matrix.

From 5.10, we can obtain:

$$\mathbf{R}_{t,s} = \mathbf{K}_s^{-1}\mathbf{H}_{t,s}\mathbf{K}_t, \tag{5.19}$$

Since the computed $\mathbf{H}_{t,s}$ is in practice up to an arbitrary scale e, we can further rewrite Equation 5.19 to:

$$\mathbf{R}_{t,s} = e * \mathbf{K}_s^{-1}\mathbf{H}_{t,s}\mathbf{K}_t. \tag{5.20}$$

Moreover, considering $\mathbf{R}_{t,s}$ is a rotation matrix and hence $\mathbf{R}_{t,s}\mathbf{R}_{t,s}^{\top} = \mathbf{I}$, where $\mathbf{I}$ is a 3×3 identity matrix; by utilizing this property, an equation related to the camera intrinsic matrix can be established as follows:

$$e^2 * \mathbf{K}_s^{-1}\mathbf{H}_{t,s}\mathbf{K}_t\mathbf{K}_t^{\top}\mathbf{H}_{t,s}^{\top}\mathbf{K}_s^{-\top} = \mathbf{I}. \tag{5.21}$$

Equation 5.21 has 9 degrees of freedom, i.e., $f_{x,t}$, $f_{y,t}$, $o_{x,t}$, $o_{y,t}$, $f_{x,s}$, $f_{y,s}$, $o_{x,s}$, $o_{y,s}$, and e are unknowns. Considering one correspondence pair contributes two equations, five correspondence pairs can solve this deterministically. Despite this, we reformulate Equation 5.21 as an optimization problem which can use all detected correspondences to improve accuracy. The objective function is:

$$\arg\min_{\mathbf{K}_*,e} \left\| e^2 * \mathbf{K}_s^{-1}\mathbf{H}_{t,s}\mathbf{K}_t\mathbf{K}_t^{\top}\mathbf{H}_{t,s}^{\top}\mathbf{K}_s^{-\top} - \mathbf{I} \right\|_F^2, \tag{5.22}$$

where $*$ stands for one of s and t, and F is the Frobenius norm. In practice, the optical center is initialized as the center of image. We set the focal length to 1200 in the first optimization step.

5.4.3 Homography to Camera Extrinsics

Once the camera intrinsics have been coarsely computed, the extrinsics, i.e. the rotation matrix, of each frame can be obtained from Equation 5.19.

Due to the observation errors of the detected correspondences for each frame, the computed $\mathbf{R}_{t,s}$ is just a coarse estimate and as such it may not meet the requirements of a rotation matrix. In other words, its columns (or rows) are not orthogonal and the norm of them does not equal to 1. To refine the $\mathbf{R}_{t,s}$ further,

we approximate it with the most similar rotation matrix by solving the following optimization equation:

$$\arg\min_{\mathbf{R}} \|\mathbf{R} - \mathbf{R}_{t,s}\|_2^2 \quad \text{s.t.} \quad \mathbf{R}^\top \mathbf{R} = \mathbf{I}, \quad |\mathbf{R}| = +1, \tag{5.23}$$

where $\mathbf{I}$ is the identity matrix which has the same size as $\mathbf{R}_t$, and $|\cdot|$ is the determinant function. The solution of Equation 5.23 is constrained by two properties, i.e. it must form a rotation matrix and have a positive determinant.

5.4.4 Homography to distortion

As mentioned in Section 5.3, we use four parameters $k_{1,t}$, $k_{2,t}$, $j_{1,t}$, $j_{2,t}$ to represent the lens distortion model. In practice, lens distortion mainly comes from the radial distortion parameters, i.e., $k_{1,t}$, $k_{2,t}$, which are much more significant than others. To simplify the coarse computation of lens distortion [53], we let $j_{1,t}$, $j_{2,t}$ equal 0 and $k_{2,t} = -\frac{1}{2}k_{1,t}$, i.e., we only estimate $k_{1,t}$.

Therefore, the simplified lens distortion model can be represented as:

$$\bar{\mathbf{p}}_t = (1 + k_{1,t} r_{\mathbf{p}_t}^2)\mathbf{p}_t, \tag{5.24}$$

where $\bar{\mathbf{p}}_t$ is the undistorted coordinate of $\mathbf{p}_t$. In this simplified model, we assume that the lens distortion parameters in two consecutive frames are the same (while this is not strictly correct, the change between two consecutive frames is typically very small) and as such $k_{1,t} = k_{1,s}$. From Equation 5.9, we can find that:

$$[\mathbf{p}_s]_\times \mathbf{H}_{t,s} \mathbf{p}_t = 0. \tag{5.25}$$

By substituting Equation 5.24 into 5.25 and collecting terms, the relationship between lens distortion and distorted correspondences $\bar{\mathbf{p}}_t$ and $\bar{\mathbf{p}}_s$ can be related by:

$$(\mathbf{D}_1 + k_{1,t}\mathbf{D}_2 + k_{1,t}\mathbf{D}_3)\mathbf{h}_{t,s} = 0, \tag{5.26}$$

where $\mathbf{h}_{t,s}$ is the vectorized form of $\mathbf{H}_{t,s}$, and design matrices are of the form,

$$\mathbf{D}_1=\begin{bmatrix} 0 & 0 & 0 & -x_{\bar{\mathbf{p}}_t} & -y_{\bar{\mathbf{p}}_t} & -1 & x_{\bar{\mathbf{p}}_t}y_{\bar{\mathbf{p}}_s} & y_{\bar{\mathbf{p}}_t}y_{\bar{\mathbf{p}}_s} & y_{\bar{\mathbf{p}}_s} \\ x_{\bar{\mathbf{p}}_t} & y_{\bar{\mathbf{p}}_t} & 1 & 0 & 0 & 0 & -x_{\bar{\mathbf{p}}_t}x_{\bar{\mathbf{p}}_s} & -x_{\bar{\mathbf{p}}_s}y_{\bar{\mathbf{p}}_t} & -x_{\bar{\mathbf{p}}_s} \end{bmatrix} \tag{5.27}$$

$$\mathbf{D}_2=\begin{bmatrix} 0 & 0 & 0 & -r^2_{\mathbf{p}_s}x_{\mathbf{p}_t} & -r^2_{\mathbf{p}_s}y_{\mathbf{p}_t} & -r^2_{\mathbf{p}_t}-r^2_{\mathbf{p}_s} & 0 & 0 & 0 \\ r^2_{\mathbf{p}_s}x_{\mathbf{p}_t} & r^2_{\mathbf{p}_s}y_{\mathbf{p}_t} & r^2_{\mathbf{p}_t}+r^2_{\mathbf{p}_s} & 0 & 0 & 0 & 0 & 0 & -r^2_{\mathbf{p}_t}x_{\mathbf{p}_s} \end{bmatrix}, \tag{5.28}$$

$$\mathbf{D}_3=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -r^2_{\mathbf{p}_t}r^2_{\mathbf{p}_s} & 0 & 0 & 0 \\ 0 & 0 & r^2_{\mathbf{p}_t}r^2_{\mathbf{p}_s} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.29}$$

Let us assume that

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_1 \\ \mathbf{I} \end{bmatrix},\ \mathbf{B} = \begin{bmatrix} -\mathbf{D}_2 & -\mathbf{D}_3 \\ \mathbf{I} & \end{bmatrix},\ \tilde{\mathbf{h}}_{t,s} = \begin{bmatrix} \mathbf{h}_{t,s} \\ k_{1,t}\mathbf{h}_{t,s} \end{bmatrix}, \tag{5.30}$$

so that the solution of $k_{1,t}$ is found by iteratively solving $\tilde{\mathbf{h}}_{t,s}$ from

$$(\mathbf{A} - k_{1,t}\mathbf{B})\tilde{\mathbf{h}}_{t,s} = 0, \tag{5.31}$$

and updating $k_{1,t}$ by finding the smallest magnitude root of the scalar quadratic equation [79, 122]:

$$\tilde{\mathbf{h}}^\top_{t,s}(\mathbf{B}^\top + k_{1,t}\mathbf{A}^\top)(\mathbf{A} - k_{1,t}\mathbf{B})\tilde{\mathbf{h}}_{t,s} = 0. \tag{5.32}$$

After $k_{1,t}$ is obtained from the above process, we use it to initialize the distortion parameters of the frames.

5.4.5 Camera Parameter Optimization

The registration of the video clip using the initially estimated camera parameters is prone to error due to misalignment and inaccurate correspondence detection. Thus once the camera intrinsics and extrinsics of all frames have been initially

estimated, we optimize all the camera parameters jointly to further refine them by minimizing the sum of squared residuals as follows:

$$J(\mathbf{K},\mathbf{R})=\frac{1}{N_{\text{corres}}}\sum_{t}\sum_{p}W_t^{(p)}(\mathrm{Q}_{\mathrm{l}}^{(p)}+\mathrm{Q}_{\mathrm{c}}^{(p)}+\mathrm{Q}_{\mathrm{p}}^{(p)}), \tag{5.33}$$

where N_{corres} is the total number of correspondences in the video clip, and $W_t^{(p)}$ is the weight of $\mathbf{p}_t^{(p)}$ that denotes the pth correspondence on $\mathcal{F}_t$. Each residual term Q_* (where $*$ is one of l, c, p) is designed according to the camera properties in a sports frame. In the following we discuss the residual terms and weight term in more detail.

Location Adjustment Term

The observable correspondences explicitly provide strong clues on the relationships between two frames. However, the locations of these correspondences are prone to errors due to challenges existing in broadcast sports video. The challenges mainly come from two factors: camera flashes and motion. The camera flashes cause a large unpredictable increase in image intensities and camera motion blurs frames. Thus, we should provide the detected correspondences with flexibility, and allow them adjust to their correct location by themselves. In particular, considering the detected correspondences are all on the distorted frames, we minimize their distance using the lens distorted coordinates. To simultaneously optimize the camera parameters and adjust the locations of correspondences, the location adjustment term Q_{l} is used as follows:

$$\mathrm{Q}_{\mathrm{l}}=\left\|\text{distort}(\mathbf{K}_s\mathbf{R}_{t,s}\mathbf{K}_t^{-1}\hat{\mathbf{p}}_t^{(p)})-\bar{\mathbf{p}}_s^{(p)}\right\|_2^2 \tag{5.34}$$

where $\hat{\mathbf{p}}_t^{(p)}$ is the parameter used to adjust $\mathbf{p}_t^{(p)}$, which is the undistorted representation of $\bar{\mathbf{p}}_t^{(p)}$. $\bar{\mathbf{p}}_s^{(p)}$ is the distorted coordinate of $\mathbf{p}_s^{(p)}$. In practice, $\hat{\mathbf{p}}_t^{(p)}$ is initialized to the same value as $\bar{\mathbf{p}}_t^{(p)}$.

Location Constraint Term

While the *location adjustment term* gives each correspondence the ability to adjust their location by themselves, the process of optimizing all camera parameters still may violate the real geometry constraints. This is because the distance from the adjusted correspondences to the projected correspondence may be 0, and as such $\mathbf{H}_{t,s}$ would be an identity matrix. Thus we introduce the *location constraint term* $\mathrm{Q_c}$ to guide the process of adjusting the location of $\hat{\mathbf{p}}_t^{(p)}$ to avoid the position of the adjusted correspondence being far away from the original position. The location constraint term $\mathrm{Q_c}$ is defined as follows:

$$\mathrm{Q_c} = \left\| \mathrm{distort}(\mathbf{K}_t \mathbf{R}_t \mathbf{K}_s^{-1} \mathrm{undistort}(\bar{\mathbf{p}}_s^{(p)})) - \bar{\mathbf{p}}_t^{(p)} \right\|_2^2. \tag{5.35}$$

Local Patch Term

To further improve the accuracy of computing relationships between two frames, we introduce the local patch term $\mathrm{Q_p}$ to minimize the difference between surrounding pixels of $\bar{\mathbf{p}}_s^{(p)}$ and that of $\mathbf{H}_{t,s}\bar{\mathbf{p}}_t^{(p)}$. The local patch term $\mathrm{Q_p}$ can be defined as:

$$\mathrm{Q_p} = \left\| \mathrm{patch}(\bar{\mathbf{p}}_s^{(p)}) - \mathrm{patch}(\mathrm{distort}(\mathbf{H}_{t,s}\hat{\mathbf{p}}_t^{(p)})) \right\|_2^2, \tag{5.36}$$

where $\mathrm{patch}(\cdot)$ is a function which extracts the patch around the selected pixel. In our experiments, we set the patch size to 39×39. In addition, we remove the mean of the extracted patches and normalize the intensity of each color channel to $[-1, 1]$. This is because even at the same point, the pixel intensities may vary too much due to camera flashes. Figure 5.7 shows a scenario where the exact same patch has a large difference in pixel intensity due to camera flashes.

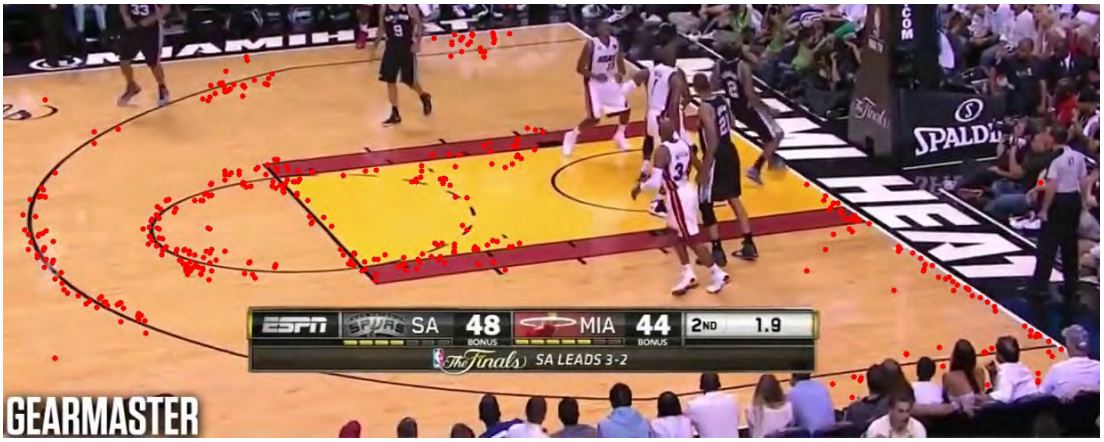
Figure 5.7: Illustration of the impact camera flashes which affect the patch matching.

Weight Term

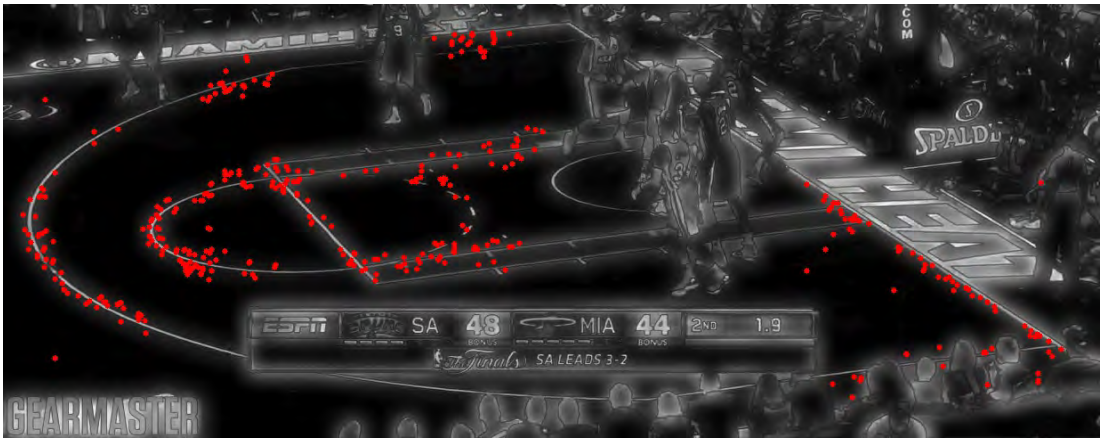
Figure 5.8 illustrates the motivation of the proposed weight term, which assigns a weight to each correspondence according to their intensity on the computed saliency map.

Inspired by [36, 103, 148], we propose a novel method to compute the saliency map for sports frames. The proposed saliency algorithm consists of three steps. Firstly, the input frame is over segmented into basic elements which abstract relevant structure, using [35]. Each obtained region is assigned a scale according to its size and shape [148]. Secondly, two different saliency cues are employed to find salient pixels in terms of color, size and position. Thirdly, the final saliency map is generated by adding maps obtained from these two saliency cues. The process of computing the two saliency cues is as follows:

Local Contrast: Based on a biological study of the human vision system, pixels which stand out from the background are generally more eye-catching [19]. The pitch line markings in sports visual content are printed using a color distinct from the color of the pitch to make them easier to recognize. Thus we model the local contrast saliency cue for the ath pixel in the frame using a weighted sum of color



(a) The original frame



(b) The saliency map generated by the proposed method.

Figure 5.8: Illustration of the motivation of the proposed weight term. Figure 5.8a is the original frame and Figure 5.8b is the saliency map computed by the proposed method. The *red points* represent the detected correspondences appearing on the playing surface. One may find that these correspondences mostly fall on or adjacent to the line markings of the playing surface, with only a few falling in areas of low texture. While the positions of the low texture correspondences are often not as accurate as those on or near lines, it is still invariably helpful to use them in camera parameter optimization thanks to the location adjustment term. To make use of all detected correspondences and avoid adverse affects from the latter kind of correspondences, we assign each correspondence a weight according to their saliency, i.e., the more salient the area where the correspondence is, the greater weight it obtains.

difference from surrounding pixels:

$$\mathrm{L}_a = \frac{1}{B}\sum_{b=1}^{B} \mathrm{K}(\mathrm{c}_b, \mathrm{c}_a, \sigma)\, \|\mathrm{c}_b - \mathrm{c}_a\|_2^2, \tag{5.37}$$

where c_a and c_b are the color of the ath and bth pixels respectively, and B is the number of neighbor pixels extracted from the patch whose center is at the ath pixel. $\mathrm{K}(c_a, c_b, \sigma)$ is the radial basis function (RBF) kernel weighting the distance between the ath and bth pixels. The argument σ constrains the influence of the neighboring pixels according its distance from the center. With this kernel, the closer the pixels are, the greater impact they have. In our experiments, σ is set to 5 to ensure salient pixels stand out from the local environment.

Global Rarity: Global rarity is generally defined as the uniqueness of the ath pixel given its color compared with other pixels in an image. Considering that what we are dealing with is a broadcast basketball video, the field of view always focuses on the playing area which is typically a single colour (e.g. yellow for the floor of a basketball court). Thus, to utilize this property, we compare each pixel to the dominant color in the frame as follows:

$$\mathrm{G}_a = \frac{1}{N}\|\mathrm{c}_a - \mathrm{c}_d\|_2^2 \sum_{b=1}^{B} \|\mathrm{c}_b - \mathrm{c}_d\|_2^2, \tag{5.38}$$

where B is the number of surrounding pixels and c_d is the centroid of the dominant color in the frame. In our implementation, we first get the maximum bin in the 16×16 2D LUV histogram and then c_d is set to the centroid color of this bin. The global rarity cue is also normalized to the range $[0, 1]$ for further use.

A widely used fusion technique is to formulate saliency cues as a linear summation [36]. The local contrast cue models the edges that are well defined from surrounding pixels. When applying this cue to a compact region filled with a uniform color, the result is a hollow region. Fortunately, global rarity overcomes this disadvantage. Thus we sum them together to get the final saliency of the ath pixel:

$$\mathrm{F}_a = \mathrm{L}_a + \mathrm{G}_a. \tag{5.39}$$

In practice, we notice that the time complexity of Equation 5.37 and 5.38 is very high and generally requires $O(N^2)$. When the radius of surrounding area increases, the time complexity may result in the curse of dimensionality. In Appendix A, we show that both of these cues can be computed in a linear time, $O(N)$, by transferring summation of surrounding pixels into an image filtering operation with the radial basis kernel.

5.4.6 Optimization

Once all the camera parameters are initialized by the aforementioned steps, these parameters are optimized jointly using the Trust Region Reflective (TRF) algorithm [54], which is particularly suitable for large sparse problems.

After the optimization of the camera parameters has converged, we can get the refined camera intrinsic, extrinsic, lens distortion parameters and the position-adjusted correspondences of each frame. In what follows, we illustrate the way to utilize the refined parameters to generate the panorama of the video clip, and then show how to register each frame on a standard basketball court template.

5.5 Camera Calibration Based on Generated Key Frames

The objective of frame registration is to register each frame to the standard basketball template $\mathcal{T}$, to enable the coordinate system transformation between the real world and frame coordinates. To achieve this goal, we first generate the panorama $\mathcal{P}$ of all frames in the video clip using the camera parameters optimized from the aforementioned process. Then the generated panorama $\mathcal{P}$ is registered

Figure 5.9: The panorama of a basketball broadcast video generated by our algorithm.

to $\mathcal{T}$ using the computed homography $\mathbf{H}$. In the last step, every single frame can be calibrated through the relationship between $\mathcal{P}$ and $\mathcal{T}$.

Figure 5.9 displays the panorama $\mathcal{P}$ which is generated from a set of frames using a linear blending technique. As one can see, the panorama covers the full view of the basketball court and the pitch markings are clear to see.

To register $\mathcal{P}$ to $\mathcal{T}$, appropriate point correspondences between $\mathcal{P}$ and $\mathcal{T}$ should be detected. We adopted the methods used in [63] to detect the top boundary line and the straight lines in the free-throw area. Once the positions of the court lines have been determined, the intersections of these lines are computed and as such they can be used as the correspondences to match the template $\mathcal{T}$ and the panorama $\mathcal{P}$.

Then the homography $\mathbf{H}$ which satisfies

$$\mathcal{T} = \mathbf{H}\mathcal{P}, \tag{5.40}$$

can be computed using a direct linear transform as follows:

$$\begin{bmatrix} \mathbf{0}^\top & -\mathbf{p}_{\mathcal{T}}^p & y_{\mathbf{p}_{\mathcal{P}}^p}\mathbf{p}_{\mathcal{T}}^p \\ \mathbf{p}_{\mathcal{T}}^p & \mathbf{0}^\top & -x_{\mathbf{p}_{\mathcal{P}}^p}\mathbf{p}_{\mathcal{T}}^p \end{bmatrix} \mathbf{h} = 0. \quad p = 1, \dots 4 \ , \tag{5.41}$$

Table 5.1: The difference between our method and previous state-of-the-art works.

Method	Distortion?	Parameters?	Need Key frame?	Pano.?
Ours	✓	✓	✗	✓
Hu *et al.* [63]	✗	✗	✓	✗
Okuma *et al.* [101]	✗	✗	✓	✗
Wen *et al.*[140]	✗	✗	✗	✓

where $\{\mathbf{p}_{\mathcal{P}}^{p}, \mathbf{p}_{\mathcal{T}}^{p}\}$ is the pth correspondence pair of $\mathcal{T}$ and $\mathcal{P}$, and $\mathbf{h} \in \mathbb{R}^{9\times 1}$ is the vectorized homography $\mathbf{H}$. Afterwards, by substituting the homography between $\mathcal{P}$ and $\mathcal{F}$ into 5.40, the transformation between the frame $\mathcal{F}_t$ and the template $\mathcal{T}$ can be computed easily using:

$$\mathbf{H}_{t,\mathcal{T}} = \mathbf{H}\mathbf{H}_{t,\mathcal{P}}. \tag{5.42}$$

5.6 Experiments

5.6.1 Baselines

To evaluate the performance of the proposed system fairly and thoroughly, we use the state-of-the-art methods [63, 101, 140] as baselines. Table 5.1 compares the differences between these methods. From the first column in Table 5.1, one can see that [63, 101, 140] do not have a proper way to deal with lens distortion, which often appears when the camera is low quality. Considering the second row, our method is able to estimate all camera parameters completely while the other three only consider the homography, which may break the underlying relationship among frames. The third column shows that our method and [140] do not rely on key frame searching due to a key frame generation strategy. In addition, ours and [140] are the only two methods which are able to generate a panoramic view of the court.

5.6.2 Implementation Details

To make fair comparison, we implemented the baselines [63, 101, 140] by eliminating the camera distortion parameters and replacing the camera intrinsic, extrinsic parameters with the homography using Equation 5.10.

We implement the proposed system and run the code on a desktop with a Core i7 3.6 GHZ CPU and 16GB RAM. Standard 720p broadcast basketball videos are used in our experiments. To speed up the calibration process of each frame, we first down sample each frame by a $2\times$ factor (i.e., the resolution is 640×360) and then generate the panorama using frames with an interval skip, which is typically set to 60 frames. The calibration of the skipped frames are obtained by the transformation between itself and nearest calibrated frame, which is used to generate the panoramic view of the court.

The predominant approaches for detection in sport scenes is to utilize low-level features to localize the player. [26, 63, 95] and extract player regions by subtracting the dominant color (i.e., color of pitch surface) from the sports image. [93, 140] trained a deformable part model (DPM) detector, which consists of 6 parts and 3 aspect ratios. While such low-level features have some success in the sports domain, their performance may be adversely affected by changes in illumination, camera motion blur, etc.

To alleviate the downside of these methods, we opt to train a Faster-RCNN model [107] that reliably detects players. Regarding the backbone of our network, Inception-ResNetV2 [127] is employed in this work for its state-of-the-art performance. Regarding the parameters of this player detector, we strictly follow the experimental settings of in [127].

One difficulty when training the network for player detection is that there is no

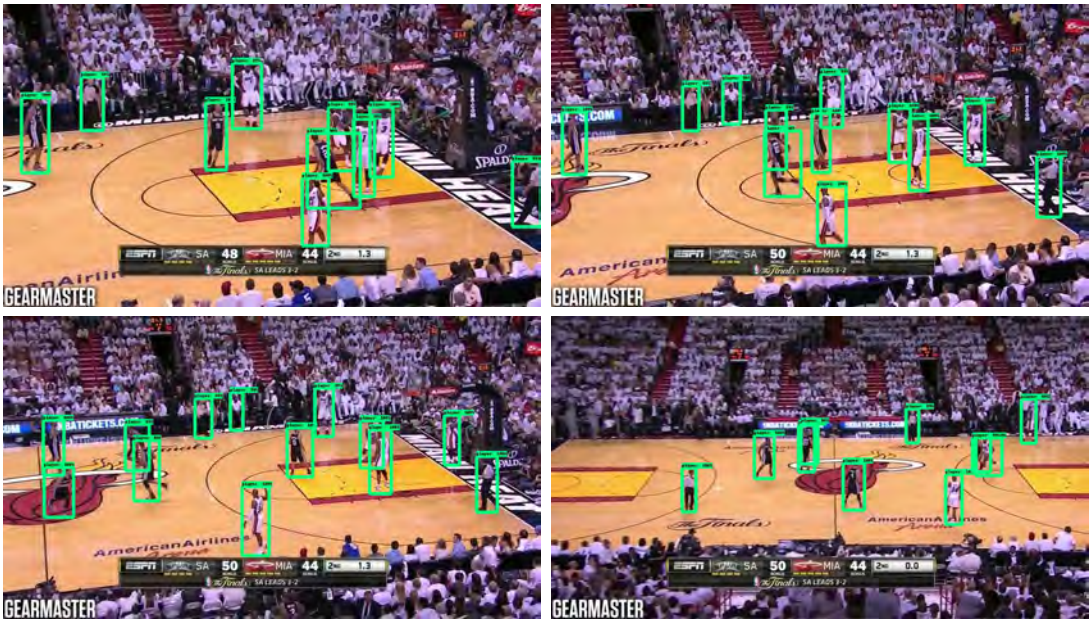


Figure 5.10: Player detection results obtained from the player detector in our work.

publicly available large-scale player dataset. To address this issue, we build a lightweight player dataset and then use transfer learning to reuse a model learned on a much larger dataset. Specifically, we manually labeled players in 100 frames, which are randomly drawn from the video. Then the network is trained using the weights pretrained from ImageNet. Figure 5.10 shows the detection results. One can easily find that almost all players are detected correctly with high confidences.

5.6.3 Evaluation

Figure 5.11: Comparison between the results obtained from our algorithm (left image) and [140] (right image) which only considers the homography. Two regions (i.e., *red region* and *green region*) are shown in detail to make the comparison clear. In these two regions, we can find that our algorithm outperforms others regarding the details of the panoramas (e.g. horizontal lines, backboard, etc.)

Figure 5.12: Calibration results of a video clip which contains the whole basketball court. The first column is the panoramic view of the court generated by our method, and the standard template (marked by red) has already been warped successfully to the panorama. The other three columns show the calibration results of the single frame which generates this panorama.

Figure 5.13: Calibration results for a video clip which does not contain information for the whole court. The first row shows bounding boxes detected by the player detection system in each frame. In the second row, the original frames are warped onto the standard court template using the homography between the panorama and frames themselves. The blue points in the second row marks the positions of players in the template.

We compare the panorama generation result with [140] since only ours and [140] are able to generate the panoramic view of the court. Figure 5.11 shows qualitative results. The left image is generated by our algorithm, which considers camera intrinsics, extrinsics, and lens distortion simultaneously and the right image is obtained from [140] which only considers homographies between each frame. To make the comparison clear, we zoomed in on two regions (i.e., *red region* and *green region*) in each image. In the left image, we can find that the basketball's backboard is well stitched in the *green region* and the horizontal court lines are correct in the *red region*. Regarding the right image, one may see that the horizontal line in the red region appears to be split due to the simple homography lacking the ability to model the relationship between frames. We also find that backboard in the *green region* does not stitch well due to the same reason.

Figure 5.12 shows the calibration results of representative frames in a video clip which contain the whole basketball court. The first column is the calibrated panorama. The other three are the representative frames in this video clip, which is calibrated from the generated panorama.

Regarding video clips that do not contain information for the whole court, we strictly follow the strategy proposed in [140], i.e., frames in such clips are registered to the court panorama generated by other video clips. Figure 5.13 presents camera calibration and player localization results of frames which fall in video clips that do not contain the entire court. The first row shows bounding boxes detected by the player detection system in each frame. The second row warps the original frames onto a standard basketball court template using the homography between the panorama and frames themselves. The process of obtaining the homographies is the same as in the aforementioned steps. One can clearly see that all frames are calibrated correctly. Player localization in the broadcast video is important because it provides vital information on player and team strategies. Based on the

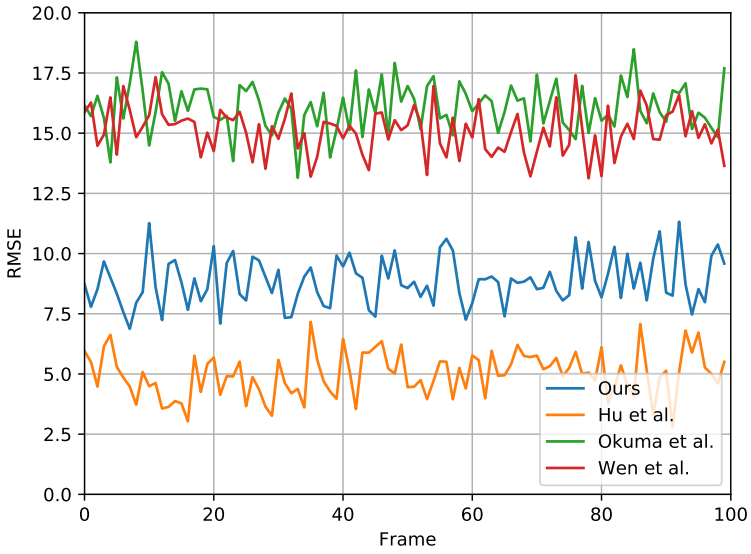


Figure 5.14: Root mean square error (RMSE) evaluation for each method over a test video.

proposed camera calibration system, we locate the players in the standard court template by averaging the bottom edge of the detected player bounding boxes. The player locations are marked using blue points in the standard template.

We also compare our method to [63, 101, 140] quantitatively using the same metric, i.e., Root Mean Square Error (RMSE), used in [63]. The RMSE is calculated from the calibrated positions and the ground truth positions, which are labeled using LabelMe[1]. Figure 5.14 shows RMSE for each frame in a continuous video clip. One can find that our method surpasses [101, 140] by a large margin. [63] has a lower RMSE than us due to its fine-grained registration strategy, which classifies court frames into three categories and then registers them with three independent templates. It has a high computational complexity and cannot work when frames cannot detect enough line markings due to the highly customized floor pattern.

A frame is regarded as well-calibrated when its intersection of Union (IOU) with the standard court template is larger than 0.8. Therefore the success rate of the

[1]https://github.com/wkentaro/labelme

Table 5.2: Comparison to previous methods considering IOU and camera calibration success rate.

Method	IOU	Success Rate
Ours	0.89	100
Okuma *et al.* [101]	0.75	82.3
Wen *et al.*[140]	0.82	100
Hu *et al.* [63]	0.92	82.3

camera calibration is defined by:

$$\text{Success Rate} = \frac{\text{Number of well calibrated frames}}{\text{Number of all the frames}} \tag{5.43}$$

Table 5.2 summarizes the results for these two metrics. We can see our method reaches 100 percent success for camera calibration due to the key frame generation strategy. [101] and [63] have the same low success rate because the highly customized court floor is difficult to detect and register. Our method outperforms [140] by a large margin because of the proposed camera calibration pipeline. While [63] obtains the highest performance in terms of IOU due to the complex registration strategy as mentioned in Section 5.1.1, it cannot work when a video clip does not contain a key frame.

We further investigate the impact of parameter settings in our system. Figure 5.15 examines the impact of the proposed *local patch term*. We can see that the free throw area in the left image contains errors because of split court lines. The right image is generated using the *local patch term* and as such the court lines in the free throw area stitched well. In addition, Figure 5.16 studies the influence of the patch size setting in the *local patch term*. We can see that the convergence speed of the optimization reaches a maximum when the patch size is set to 39, and also obtains the minimum mean squared error (MSE). The convergence speed increases as the patch size increases. This is because small patches extracted from the correspondences cannot convey useful information to the camera parameter optimizer. Also we note that the optimization speed and the MSE appear to be

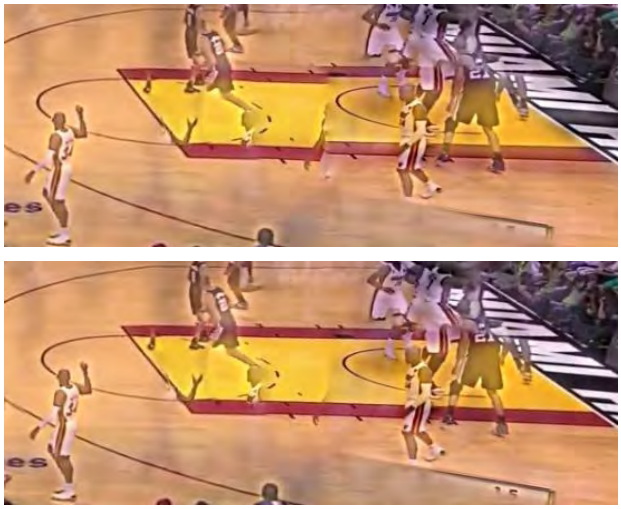


Figure 5.15: The impact of the *local patch term* on panorama generation. The top image shows the optimized result which does not contain the *local patch term*. We find that the straight court lines in the free throw area appear to be split, i.e., these lines are misaligned. The bottom image shows the result using the whole form of the proposed algorithm where the lines are aligned correctly.

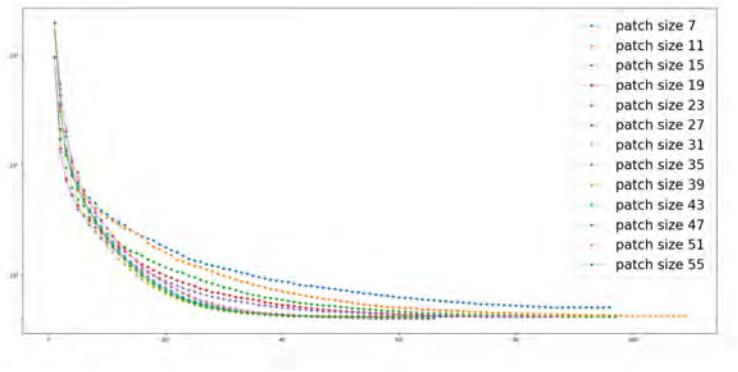


Figure 5.16: We compare the convergence speed and MSE for different patch sizes. The y axis is scaled to log space for the discrimination of subtle changes in the MSE. We find that the performance of the optimization process increases with an increases in patch size until the patch size reaches 39. When the patch size is 39, the speed of the convergence is the fastest and the error is at a minimum.

degraded when the patch size exceeds 47. The reason is that a big patch is likely to capture dynamic pixels which belong to players, and these player pixels destroy the relationship between the patches extracted from the correspondences.

The performance of the proposed saliency algorithm is also compared to existing state-of-the-art algorithms [3, 45, 60, 139, 148] qualitatively. Considering that there is no public basketball frame saliency dataset we provide a visual comparison of our proposed saliency detection with several state-of-art-methods as shown in Figure 5.17. One can see that our proposed method outperforms other kinds of saliency algorithms in the context of basketball frames. This is because most existing saliency detection methods are proposed to emphasize the dominant objects or regions in an image while basketball frames have their own properties. The proposed approach retains the edges of salient objects, including court lines and patterns, and eliminates the influence of the court floor in the saliency detection. The retained salient objects provide the key information of a basketball match such as where the player is, where the court lines are, etc. Moreover, the saliency map generated from our method provides substantial clues for the weight of each correspondence.

(a) SRC (b) CA [45] (c) HS [148] (d) AC [3] (e) SR [60] (f) GS [139] (g) Ours

Figure 5.17: Visual quality comparison of the-state-of-the-art saliency detections methods compared to our method (Ours). As you can see, Ours consistently outperforms context-aware (CA) saliency [45], hierarchical saliency (HS) [148], salient region detection (AC) [3], spectral residual saliency (SR) [60], and geodesic saliency (GS) [139]. Ours retains the edges of the salient objects, including court lines and court patterns, and eliminates unimportant regions of the court floor in the saliency detection.

The proposed calibration technique has been applied to player localization. Coaches, players, and spectators are expected to benefit from our technique because the system can localize players' positions, enabling rich analytics. Moreover, our camera calibration can invariantly benefit tactical analysis, sports panorama generation, player pose estimation, etc. Therefore, we intend to explore applications for these techniques in the future.

5.7 Chapter summary

In this chapter, an innovative camera calibration system for broadcast basketball video that does not require key frames has been presented. The major contribution of this framework is that this system calibrates each individual frame through calibrating the panorama generated from these frames. It utilizes the holistic structure of the frames in a video rather than searching for a key frame in isolation. Apart from this contribution, we propose a robust focal length estimation algorithm which can overcome the numerical instability present in the video registration task. Additionally, a novel camera optimization algorithm containing well-designed terms for the broadcast basketball task is proposed and experiments show that the camera model and optimization process outperform previous approaches which only consider the homography. Finally, an innovative saliency detection algorithm is proposed for sports videos. The newly proposed saliency detection algorithm can provide substantial clues for the importance of each point in a basketball frame.

The approach presented here builds directly on the method presented in Chapter 4. Compared to the method in Chapter 4, the proposed method in this chapter improves the quality of generated key frame significantly by decomposing estimated homographies using the properties of PTZ cameras. Furthermore, a new focal

length estimation algorithm and frame saliency generation algorithm are proposed for broadcast videos.

However, while the proposed approach performs well, there are a number of limitations. When point correspondence detection across adjacent frames fails, the homography estimation algorithm fails. Therefore, developing a homography estimation algorithm which is not reliant on local point correspondence detection and instead matches images globally is urgently needed. In Chapter 6, we introduce a deep learning based homography estimation algorithm, which takes input as a pair of images directly and outputs the homography. Therefore, it obviates the need for correspondence detection methods which can be problematic.

Chapter 6

Homography Estimation Using Perspective Fields

6.1 Introduction

Planar homography estimation, which is a fundamental task in computer vision, aims to provide a non-singular linear relationship between points in two images. As introduced in Chapters 3, 4, and 5, homographies have many practical applications, such as image rectification [21, 99, 161], image registration [46, 94], and camera calibration [50, 120].

The framework used in Chapters 4 and 5 is based on using a hand-crafted key point detector, followed by correspondence matching and the use of a direct linear transform (DLT) to determine the transformation between two images. While such frameworks have been successful to some extent, their performance is largely dependent on the accuracy of the key point detector and the correspondence matching. When the correspondences are misaligned, this framework may fail.

This situation often happens when hand-crafted features perform poorly in the presence of large variations in perspective, illumination, etc.

To alleviate the dependence on key-point detection and matching performance, estimating homographies in an end-to-end fashion using convolutional neural networks (CNNs) has been widely studied in the last two years. Most existing works [28, 67, 100] aim to regress the 4-point homography directly, i.e., the offsets of the four corners are embedded into a vector and then predicted by a CNN through a dense layer preceded by a fully-connected layer. Unfortunately, the strategy of formulating this task has three drawbacks:

1. The vector representation of the four corners computed via a dense layer breaks their spatial structure, and hence results in a loss of spatial information.

2. Four points are the minimum to estimate the homography. Perturbation of any one predicted corner may significantly affect the accuracy of the estimate. In other words, a 4-point homography is not robust.

3. Fully-connected layers are frequently used in these architectures, which increases the computation requirement and decreases the representation power of the network. For example, the fully-connected layers in networks such as VGGNet [116], AlexNet [75], etc., occupy nearly 20% (i.e., less representation capacity) for the weights but require 80% of the computation.

Aiming to address the aforementioned issues, we propose a conceptually simple, reliable, and general framework for homography estimation. The proposed parameterization uses the bijective pixel-to-pixel linear mapping described by a homography using the perspective field, which encodes every pixels' offsets while retaining their spatial information. Subsequently, this new formulation is

naturally learned by the perspective field network (PFNet), which contains a novel deconvolution block designed in a residual fashion.

The contribution in this chapter is two-fold:

1. We propose a new parameterization for homography estimation which can be easily applied to any existing FCN (Fully Convolutional Network).
2. We introduce the PFNet for homography estimation, endowed with the proposed novel deconvolution blocks, that allows for upsampling and predicting homographies efficiently. Thanks to the proposed fully convolutional architecture, the network has fewer hyper-parameters while greatly boosting the performance compared with existing state-of-the-art methods.

6.2 Related Works

Existing approaches for homography estimation can be divided into two groups: feature-based and end-to-end deep-learning-based approaches.

Feature-based homography estimation

Feature-based approaches have been explored in Chapters 4 and 5 of this thesis, and while good performance has been achieved in these works, the proposed approach has been carefully constructed to exploit the characteristics of the scene. As such, transferring these methods to a new domain is non trivial, and would require new domain knowledge to be incorporated and the method to be modified. For further information, we refer readers to Chapters 4 and 5, and [98] for a discussion of the properties of existing local invariant features.

Homography prediction with CNNs

Due to the remarkable advances of deep learning in the field of computer vision, research into homography estimation has been driven towards the use of CNNs. Detone *et al.* [28] was the first to use a CNN for regressing four corners offset from a pair of image in a VGG-like network with 14 layers. Their technique was shown to outperform traditional feature-based methods, such as SITF, ORB, SURF, etc., when sufficient features could not be detected. Inspired by this work, [67] proposed a twin CNN module, which processes two images in parallel in the first 4 convolutional layers and concatenates feature maps to generate estimates using another 4 convolutional layers and 2 fully-connected layers. Subsequently, they arrange the proposed module in a stacked manner and trained them in a hierarchical fashion to reduce estimation error bounds. While this iterative architecture significantly outperforms [28], it increases both the model and computational complexity significantly.

Alternatively, to improve the quality of the predicted homography, [100] develops a hybrid approach that combines the strengths of both a deep learning and a conventional feature-based method. Specifically, it relies on features to compute the homography estimates using a 4-point parameterization in the first half of the network and then optimizes the pixel-wise photometric loss of the two images in the second half.

These three approaches have two key commonalities: (1) they design a dedicated network which lacks flexibility and so cannot benefit from existing the state-of-the-art CNN architectures, and (2) the 4-point parameterization does not model the bijective pixel-wise linear mapping and breaks the spatial information among four corners' offsets.

Unlike these existing methods, we formulate this task as predicting a perspective

field, and then adapt the proposed fully convolutional network (FCN) to learn it simply and efficiently from a pair of images. Moreover, this new parameterization does not rely on any specific network and as such it has great generalization. It can be easily learned by any existing FCN to estimate homographies.

6.3 Homography Parameterization

The standard way to parameterize a homography is to find a 3×3 non-singular matrix which can establish a bijective projection between every pixel in two images. Suppose that we have a point $\mathbf{p} = (x_{\mathbf{p}}, y_{\mathbf{p}})$ on image I_{A} and its corresponding point on image I_{B} is denoted as $\mathbf{q} = (x_{\mathbf{q}}, y_{\mathbf{q}})$. Thus, the homography $\mathbf{H}$ that maps $\mathbf{p}$ to $\mathbf{q}$ can be expressed as,

$$\begin{bmatrix} x_{\mathbf{q}} \\ y_{\mathbf{q}} \\ 1 \end{bmatrix} \sim \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} & \mathbf{H}_{13} \\ \mathbf{H}_{21} & \mathbf{H}_{22} & \mathbf{H}_{23} \\ \mathbf{H}_{31} & \mathbf{H}_{32} & \mathbf{H}_{33} \end{bmatrix} \begin{bmatrix} x_{\mathbf{p}} \\ x_{\mathbf{p}} \\ 1 \end{bmatrix}. \tag{6.1}$$

Since the transformation is defined up to a scaling factor, it can be normalized by scaling $\mathbf{H}_{33} = 1$, and as such $\mathbf{H}$ can be parameterized by 8 values.

While this straightforward parameterization has been widely used in computer vision, it may be problematic when we employ it as the learning objective of a CNN. This is because $\mathbf{H}$ implicitly encodes the rotation, translation, focal length, skew, and optical center parameters; and the value range of each parameter has high variance. Typically, the magnitude of translation, focal length, and optical center are much greater than that of rotation, as such it is very hard to balance (normalize) all parameters' contribution to the loss function.

To address this issue, we parameterize $\mathbf{H}$ between I_{A} and I_{B} using a PF, F_{AB}. F_{AB} has two channels F_{AB_x}, F_{AB_y}, where each channel represents the pixel offset

caused by the homography in the x and y directions. Let W and H represent the width and height of I_{A} and I_{B} respectively. The displacement of $\mathbf{p}$ in terms of the x-axis is denoted $\Delta x_{\mathbf{p}} = x_{\mathbf{p}} - x_{\mathbf{q}}$. Similarly, $\Delta y_{\mathbf{p}} = y_{\mathbf{p}} - y_{\mathbf{q}}$ is the displacement of $y_{\mathbf{p}}$. Thus F_{AB_x}, F_{AB_y} can be expressed as follows,

$$F_{\mathrm{AB}_x} = \begin{bmatrix} \Delta x_{\mathbf{p}_{11}} & \Delta x_{\mathbf{p}_{12}} & \cdots & \Delta x_{\mathbf{p}_{1W}} \\ \Delta x_{\mathbf{p}_{21}} & \Delta x_{\mathbf{p}_{22}} & \cdots & \Delta x_{\mathbf{p}_{2W}} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta x_{\mathbf{p}_{H1}} & \Delta x_{\mathbf{p}_{H2}} & \cdots & \Delta x_{\mathbf{p}_{HW}} \end{bmatrix}, \tag{6.2}$$

and

$$F_{\mathrm{AB}_y} = \begin{bmatrix} \Delta y_{\mathbf{p}_{11}} & \Delta y_{\mathbf{p}_{12}} & \cdots & \Delta y_{\mathbf{p}_{1W}} \\ \Delta y_{\mathbf{p}_{21}} & \Delta y_{\mathbf{p}_{22}} & \cdots & \Delta y_{\mathbf{p}_{2W}} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta y_{\mathbf{p}_{H1}} & \Delta y_{\mathbf{p}_{H2}} & \cdots & \Delta y_{\mathbf{p}_{HW}} \end{bmatrix}. \tag{6.3}$$

To convert the PF back to the homography, we first simply restore the correspondences between two images by,

$$\{\mathbf{p}_i, \mathbf{q}_i\}_{i=1}^{HW} = \{\mathbf{p}_i, \mathbf{p}_i - \Delta\mathbf{p}_i\}_{i=1}^{HW}, \tag{6.4}$$

where i is the index of the correspondence, and $\Delta\mathbf{p}_i = (\Delta x_{\mathbf{p}_i}, \Delta y_{\mathbf{p}_i})$. Once the correspondences have been established by Equation 6.4, RANSAC is applied to further filter outliers, and then we use a DLT to solve the homography from a set of equations,

$$\begin{bmatrix} \mathbf{0}^\top & -\mathbf{q}_i^\top & y_{\mathbf{p}_i}\mathbf{q}_i^\top \\ \mathbf{q}_i^\top & \mathbf{0}^\top & -x_{\mathbf{p}_i}\mathbf{q}_i^\top \end{bmatrix} \mathbf{h} = 0. \quad i = 1, \ldots, HW \quad , \tag{6.5}$$

where $\mathbf{h} \in \mathbb{R}^{9\times 1}$ is the vectorized homography $\mathbf{H}$. Figure 6.1 visualizes the PF generated by a pair of images.

Figure 6.1: Illustration of the PF between a pair of images. From left to right: the original image, the warped image (using a random homography), the PF. The PFs are plotted on a green gird at a 10 pixel interval. The red quiver on the pixel represents the direction and length of the pixel offset. By combining this PF with the warped image, we can easily recover the original image and obtain the homography.

6.4 Methodology

In Section 6.4.1, we describe the architecture of the proposed fully convolutional residual network in detail. Then in Section 6.4.2 and 6.4.3, the strategies of building the loss function and optimization are introduced respectively. Figure 6.2 visualizes the proposed architecture.

6.4.1 FCN Architecture

Our objective is to design an end-to-end architecture that outputs the PF between a pair of images. Thus choosing an appropriate CNN backbone is crucially important. In this work, ResNet-50 [57] is employed as our FCN backbone for its state-of-the-art performance in many computer vision tasks. In addition, ResNet-50 does not have high computational complexity when compared to many deep networks such as ResNet-101, DenseNet, VGG, etc., while having an acceptable level of performance.

The proposed fully convolutional residual network consists of two parts: the

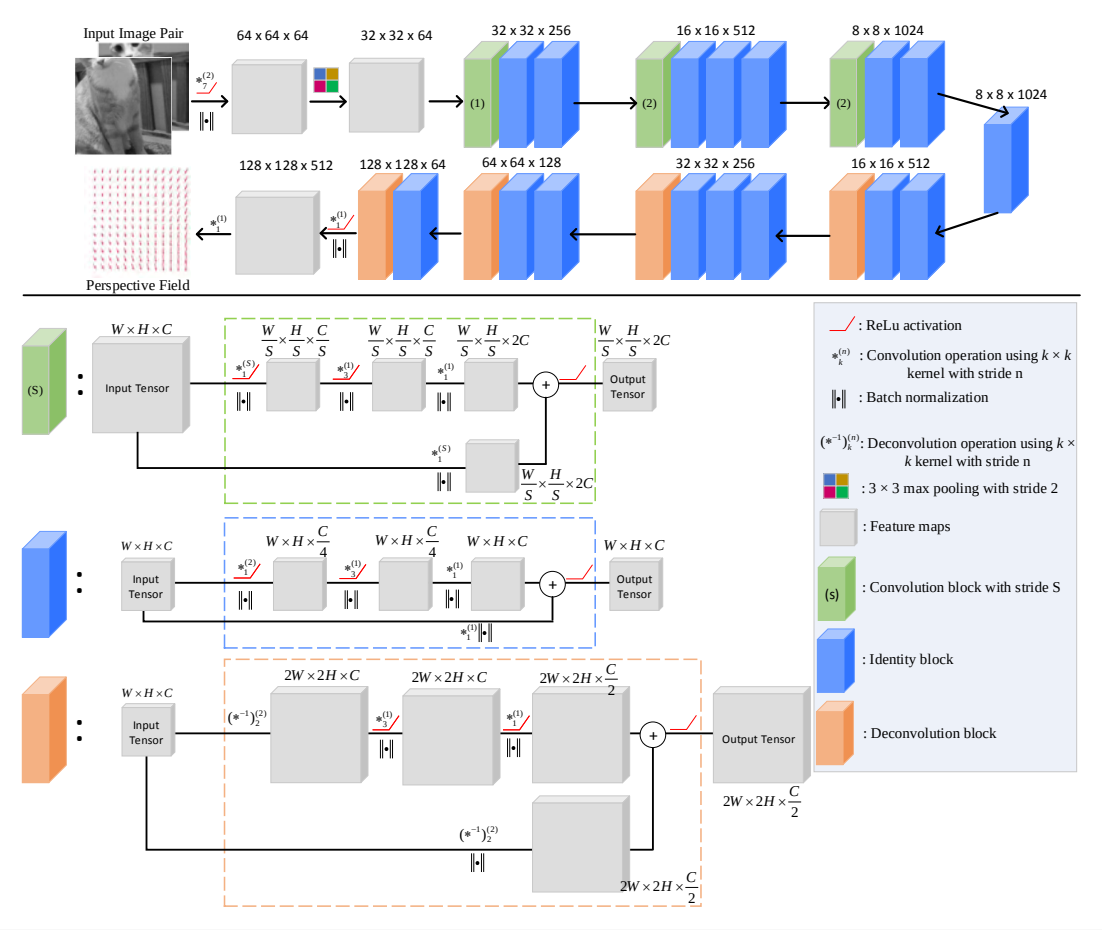


Figure 6.2: The architecture of the proposed network. This network consists of two parts: an encoder (the top row) and a decoder (the bottom row). In the encoder, the network takes as input a pair of images and encodes its information into a set of feature maps through a series of convolutions. In the decoder, the PF information embedded in the feature maps is recovered progressively. In the second last layer, we use 512 filters to expand the width to enhance the representation ability of the network. The last layer outputs F_{AB_x} and F_{AB_y} respectively.

encoder and the decoder. In the encoder, we strictly use the first 4 stages of ResNet-50, where each stage is constructed by a specific number of residual blocks [57] (including identity blocks and convolution blocks), and we remove the 5th stage and following fully-connected layer to enable the connection between the encoder and decoder. The encoder takes as input a $W \times H \times 2$ gray image pair and gives rise to a tensor of features of shape $W_m \times H_m \times C_m$ for stage $m = \{3, 4\}$, where $W_m = \frac{W}{2^m}$, $H_m = \frac{H}{2^m}$, and $C_m = 2^{6+m}$. The output of the encoder captures

the spatial correlation of the image pair at a set of uniformly sampled pixel locations, and forms the input to the decoder.

The decoder aims to learn how to generate the PF given the spatial correlations learned by the encoder. To achieve this goal, it progressively upsamples the feature maps through a series of identity blocks and the proposed deconvolution block, to ensure the output has the same size as the input image. The decoder is symmetrical to the encoder. It also consists of 4 stages, where each stage (denoted as $n = \{1, \ldots, 4\}$) has the same number of residual blocks but replaces the convolution block with the proposed deconvolution block. The output tensor generated from stage n is of shape $W_n \times H_n \times C_n$, where $W_n = \frac{W}{2^{4-n}}$, $H_n = \frac{H}{2^{4-n}}$, and $C_n = 2^{10-n}$. The deconvolution block embedded in each stage provides the ability to propagate low-resolution feature maps to high-resolution feature maps. More details are introduced in the following discussion.

Once the encoder and decoder have been set up, we convolve the last layer of the decoder with a $1 \times 1 \times 512$, and a $1 \times 1 \times 2$ filter successively to enable the prediction of the PF, as such the output size is $W \times H \times 2$. It is worth noting that our model does not have any long skip connections between m and the corresponding n, which are heavily used in segmentation tasks [91]. Skip connections aim to propagate pixel-wise information from low stages to high stages to assist segmentation. However, homographies have significantly distorted pixel-to-pixel spatial relationships between two images and as such we omit long skip connections from our architecture. Moreover, we do not use drop out to further regularize our architecture because we want to enable fair comparison with other baselines.

Deconvolution Block

The structure of the deconvolution block is depicted in the bottom half of Figure 6.2. The deconvolution block is the last residual-like block in each decoder stage. It receives an input tensor, and increases the spatial resolution using a deconvolution operation. The design of the deconvolution block is inspired by the philosophy of ResNet and as such we follow two simple rules:

1. the input tensor is added directly to the end of the block by being only convolved once to enable the residual operation, and
2. the block doubles the feature map size while halving the number of the filters.

More specifically, we first adopt two 2×2 deconvolutional kernels with stride 2 to convolve the input tensor to obtain two data streams. In the first data stream, the number of filters is the same as that of the input tensor. The second data stream halves the number of filters of the input tensor to enable a direct add operation between the output of the first data stream and the second.

6.4.2 Loss Function

A standard loss for dense prediction problems is the l_2 loss, which minimizes the Euclidean distance between predictions and the ground truth. However, this commonly used loss is not suitable for PFNet because the l_2 loss places emphasis on outliers, i.e., the optimization objective focuses on minor outliers rather than major inliers in the PF. As mentioned in Section 6.3, we further use RANSAC to filter out outliers predicted by PFNet, this operation means that PFNet is robust to any outliers. Thus what really determines the performance of PFNet is the

inliners in the estimated PF. To make use of this property, we choose a smooth-l_1 loss, which is robust to outliers, to optimize this network as follows,

$$\text{loss} = \begin{cases} 0.5 \times \left(F^{\text{pred}}_{AB_*} - F_{AB_*}\right)^2, & \text{if } \left|F^{\text{GT}}_{AB_*} - F_{AB_*}\right| \leq 1 \\ \left|F^{\text{pred}}_{AB_*} - F_{AB_*}\right| - 0.5, & \text{otherwise} \end{cases} \tag{6.6}$$

where $*$ denotes x or y, and superscript *pred* represents the predictions in terms of F_{AB_*}. More details and a comparison of the smooth-l_1 and l_2 loss on PFNet are provided in Section 6.6.1.

6.4.3 Training and optimization

We have implemented our system using Keras [20] and TensorFlow [1]. The full network is trained in an end-to-end fashion. We train this network from scratch, where all weights are initialized using the method of [44]. We choose NAdam [126] as our optimizer because it generally acts well for many datasets and has a favorable convergence speed. The parameters used for NAdam are set to the defaults, with momentum 0.9, $\beta_1 = 0.9$, and $\beta_2 = 0.99$. We use a learning rate of 1×10^{-3} in the first 40 epochs, and divide it by 10 after each 40 epochs. Each epoch contains 1000 steps in which the batch size is set to 64. The model has finished when training reaches 80 epochs. One batch takes approximately 2s on a NVIDIA Tesla M40 GPU and it takes about 44 hours of training for the model to converge.

6.5 Experiments

In this section, we introduce the dataset used in our experiments and the strategy for generating samples (see Section 6.5.1). The baseline models including CNN-

based and standard feature-based methods are outlined in Section 6.5.2. Section 6.5.3 evaluates the proposed network thoroughly with respect to CNN-based approaches to demonstrates the capacity of the proposed architecture and the contribution of the novel deconvolution block. Furthermore, a robustness analysis of our network and traditional feature-based methods, such as SIFT, and SURF, is examined in an extreme image contamination environment (see Section 6.5.4).

6.5.1 Synthetic Dataset Generation

The availability of large scale datasets is crucially important for the performance of CNNs. Unfortunately, there is no public dataset for this task and as such previous works use synthetic datasets to train and evaluate the approaches. By strictly following previous works' dataset generation rules [28], we trained our network using the Microsoft Common Objects in Context (MSCOCO) 2014 dataset [85], which contains 82,081 training images and 40,137 testing images. All images in the dataset have been transformed to gray scale and resized to 320×240 prior to further processing.

In the training phase, we generate the pair of input and output tensors by applying a random homography to an image. More specifically, we first randomly generate a 128×128 square of I_{A} with a random left top corner $\mathbf{p}_{\mathrm{ltc}} = (x_{\mathbf{p}_{\mathrm{ltc}}}, y_{\mathbf{p}_{\mathrm{ltc}}})$. Secondly, we perturb each corner of this square (four corners in total) using a random value in the range of $(-\rho, \rho)$. Then the ground truth homography can be obtained from the original corners and the perturbed corners. Please note that we set ρ to 32 and limit $32 < x < 224$ and $32 < y < 80$ to ensure that the four perturbed corners do not fall outside the image. Therefore, the warped image I_{B} can be computed using the randomly generated homography. To create the input for the proposed network, we crop two 128×128 patches from I_{A} and I_{B} at $\mathbf{p}_{\mathrm{ltc}}$ and then stack

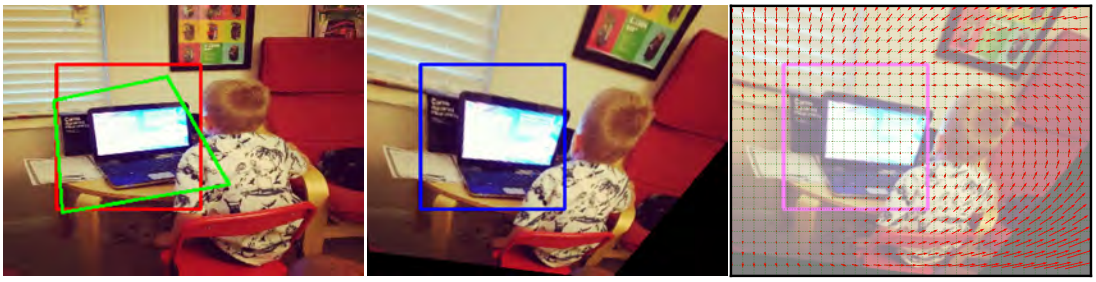


Figure 6.3: Visualization of training dataset generation. **Left**: we randomly generate the red square in the original image (I_{A}) and then perturb it's four corners in $(-\rho, \rho)$ to get the green quadrilateral. The ground truth $\mathbf{H}$ can be computed from the red and green box using 4-point DLT. **Middle**: The warped image (I_{B}) obtained by applying the inverse of $\mathbf{H}$ on I_{A}. The blue square has the exact same position as that of the red box. **Right**: the F_{AB} generated from I_{A} and I_{B}. The magenta box has the same position as the blue box. Therefore, we stack the gray-scaled red patch and blue patch to form the input tensor with shape $128 \times 128 \times 2$. To obtain the $128 \times 128 \times 2$ output tensor, we stack F_{AB_x} and F_{AB_y} together in the magenta square.

them together to form the input tensor whose shape is $128 \times 128 \times 2$. With respect to the output, we crop two patches of the same size at $\mathbf{p}_{\mathrm{ltc}}$ from F_{AB_x} and F_{AB_y}, such that the shape of the output tensor is $128 \times 128 \times 2$. Figure 6.3 visualizes our dataset generation strategy.

6.5.2 Baseline Models

To thoroughly evaluate the performance of the proposed network on this task, we select previous state-of-the-art CNN-based works [28, 67, 77] and the feature-based methods SIFT and SURF as the baseline models for comparison.

HomographyNet [28] uses a VGG-like network for this task which heavily utilizes fully-connected layers. HCN-x [67] stacks the proposed twin convolutional regression networks in a hierarchical way to estimate the homography between a pair of images, where x stands for the number of stacked modules. HomographyNet and

HCN-x both make use of the traditional homography parameterization. While FCRN [77] is proposed for depth estimation, which is another dense prediction task, its architecture is similar to ours. Both PFNet and FCRN are built upon ResNet-50. The difference between FCRN and PFNet is that FCRN uses the upsampling block [77] to do deconvolution and does not use any identity block in the deconvolution stage. To demonstrate the effect of the identity block in the architecture, we add identity blocks to FCRN in the same way as ours and refer this method to as FCRN-IB. Furthermore, the performance of the proposed deconvolution block is validated by comparing PFNet with FCRN-IB. To maximize the performance of PFNet, we train PFNet again using 300 epochs, where the learning rate is divided by 10 after each 100 epochs, and other parameters remain the same. We denote this method as PFNet-300. Besides applying the PF to FCRN and FCRN-IB, to testify the generalization of this parameterization, we also apply the PF on the state-of-the-art FCN-DenseNet [65, 155, 158], which has been successfully applied to many dense prediction problems.

Regarding SIFT and SURF, we directly employ them using their OpenCV implementations with default parameter settings to detect key points and extract local descriptors on the input patch. Subsequently, correspondences are established using a standard matching and RANSAC scheme to estimate the homography.

6.5.3 Evaluation

We evaluate the performance of our approach and CNN-based methods on test data generated using the strategy of [28, 67]. Specifically, we generate test samples from 5,000 randomly selected images from the test dataset of MS COCO 2014 by following the data generation process described in Section 6.5.1. To evaluate the baseline models fairly, we employ the mean average corner error (MACE) as

the metric, which is widely used in previous research [28, 67]. MACE averages the Euclidean distance between the ground truth and the estimated positions of the four patch corners. Table 6.1 reports the results of each method in terms of MACE and also reports the network size.

Focusing on the first two columns of Table 6.1, we find that PFNet consistently outperforms baseline models with respect to MACE. Specifically, compared with the previous state-of-the-art, HCN-4, we have reduced MACE from 3.91 pixels to 0.92. The performance of HomographyNet is the worst among learning-based methods because of the shallow architecture and 4-point parametrization. It is worth noting that FCRN performs worse than FCRN-IB, which adds the identity blocks in the decoder. The hypothesis is that the extra layers brought by identity block increases the representative capacity of the decoder in the network, and as such the decoder can recover the PF distribution from the output of the encoder. In particular, we obtain performance improvements when we replace the upsampling block [77] with the proposed deconvolution block in FCRN-IB to form PFNet, which suggests that the deconvolution block can avoid losing information while upsampling the input feature maps by a 2X ratio when compared to the upsampling block. This is because the input tensors in the upsampling block are convolved with an extra convolutional layer before it propagates information to the end of the block. The extra convolutional layer closes the shortcut between the input and output tensor, and as such it obstructs the information flow used to learn the residual function. The lower validation loss and MACE of PFNet in Figure 6.4 also demonstrates aforementioned analysis. For more details and comparisons among these three networks, readers are referred to the Section 6.6.2. All the four networks, i.e., FCN-DenseNet, FRCN, FRCN-IB, PFNet, have comparable performance with previous works and as such we claim the new homography parametrization for learning homographies has a good generalization ability.

Method	MACE	Computation Complexity	
		Architecture Depth	Parameters
HomographyNet [28]	9.20	14	34M
HCN-1 [67]	13.04	14	17M
HCN-2 [67]	6.39	28	34M
HCN-3 [67]	4.46	42	51M
HCN-4 [67]	3.91	56	68M
FCN-DenseNet [65]	4.20	121	20M
FRCN [77]	1.86	72	28M
FRCN-IB	1.72	96	31M
PFNet	**1.63**	96	31M
PFNet-300	**0.92**	96	31M

Table 6.1: Mean average corner error (MACE) comparison between our method and various baselines.

Observing the last two columns of Table 6.1, one may see that HomographyNet and HCN-x have a significantly larger hyper parameter space. This is because when they formulate homography estimation as needing to estimate 8 parameters, they have to use at least two fully-connected layers to regress the homography. Therefore, the architectures have a huge number of parameters when compared to FCNs with the same depth. Please note that HCN-x progressively improves the performance by stacking the modules one by one. As a consequence of this hierarchical approach, a network with only 56 layers depth contains 68M parameters. However, the depth of a deep learning architecture is crucially important for computer vision tasks as demonstrated in many studies [57].

Qualitative Results

We show example predictions from PFNet in Figure 6.5. The model generates the PF from given image pairs and the predicted homography is recovered from the

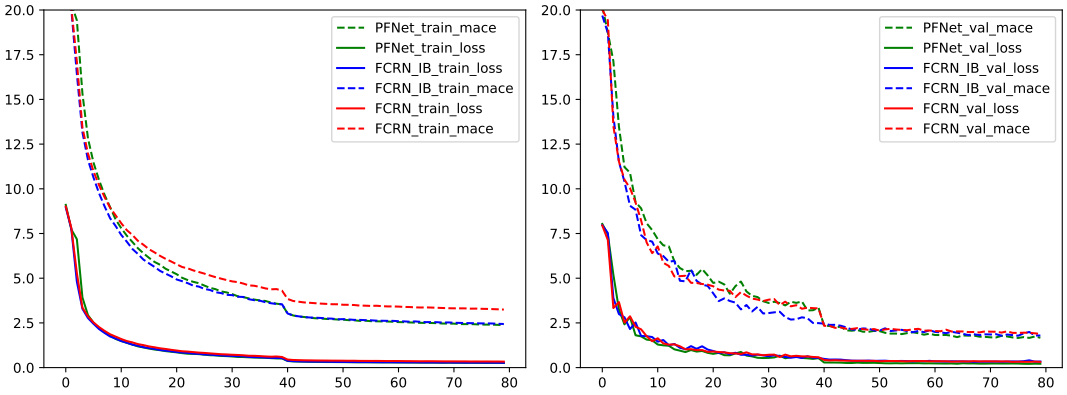


Figure 6.4: The MACE and the loss value in the training and validation phases for FCRN, FCRN-IB, and PFNet.

generated PF. Please note that the predicted PF is very similar to the ground truth. The same is true for the warped images. Additional qualitative experiments can be found in Section 6.6.3.

6.5.4 Evaluation on Noisy Data

In this section, we compare the proposed method with two feature-based methods, SIFT and SURF, in an extremely noise-contaminated environment to test the robustness of the algorithms. To simulate the noise contamination, we first create a image contaminator list which contains four contaminators: Gaussian blur (GB), dropout (DO), salt and pepper noise (SP), and Gaussian noise (GN); where their parameters are set in a range of (0, 4), (0, 0.4), (0, 0.4), and (0, 150) respectively according to the ImgAug toolbox [1].

To enable estimation of the PF in noise-contaminated images, we retrained our network by creating the dataset following the steps described in Section 6.5.1 but randomly select 0 to 4 contaminators to contaminate I_{A} and I_{B} separately. To analyze the robustness of the algorithms thoroughly, we split the testing

[1]ImgAug toolbox: https://github.com/aleju/imgaug

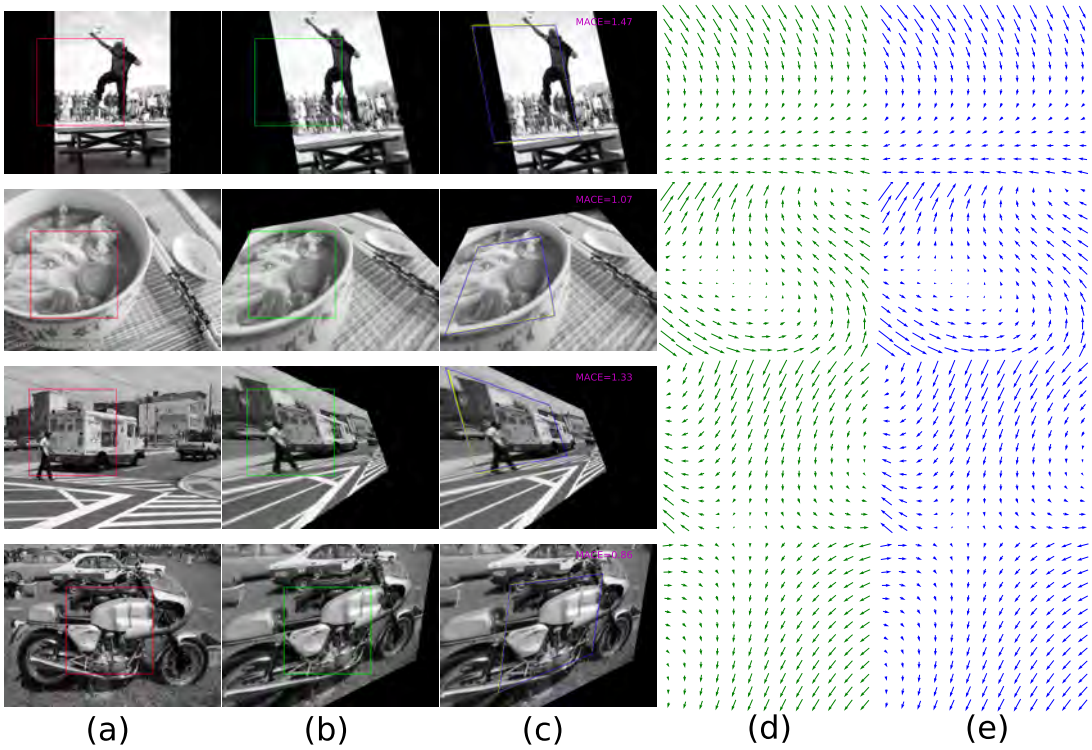


Figure 6.5: Example PF generated by PFNet and the predicted homography. (a) The original 320×240 gray image. The red box is used to generate input. (b) The warped image transformed by the ground truth homography. A green square is placed in the same position as that of red box to form the input image patch pairs. (c) The warped image transformed by the predicted homography. MACE error is annotated in the top right corner of the image. The blue quadrilateral represents the ground truth homography, and yellow is the predicted homography. (d) The ground truth PF. (e) The predicted PF.

experiments into 7 groups: 1. **N** (normal): the dataset is uncontaminated. 2-5. **GB** (Gaussian blur), **DO** (dropout), **SP** (salt and pepper noise), **GN** (Gaussian noise): only use one selected contaminator. 6. **M** (mix): mix 2 to 4 contaminators in a random order. 7. **HA** (heavy augmentation): mix 2 to 4 contaminators in a random order where each contaminator uses the maximum parameter. Testing samples in which MACE is less than 38.40 ($30\% \times 128$) are marked as successfully estimated samples. Using this, a success rate is employed to evaluate the robustness of each algorithm.

Table 6.2 reports the performance of the algorithms. We see that our method significantly outperforms feature-based methods in all cases. A 100% success rate in all cases means that our method has a high robustness when confronted with a noisy dataset. The significant drop in performance of SIFT and SURF when evaluating **DO** and **SP** contaminated images suggests that feature-based methods do not work when a large proportion of pixels are missing. It is worth noting that our method works acceptably for extremely contaminated image pairs, i.e., **HA**. However, SIFT and SURF do not work at all in these conditions. Figure 6.6 shows example images and computed homographies for the **HA** case.

Method	MACE							Success rate (%)						
	N	GB	DO	SP	GN	M	HA	N	GB	DO	SP	GN	M	HA
PFNet	**2.87**	**2.81**	**3.31**	**3.78**	**2.75**	**4.53**	**9.51**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
SIFT	25.35	25.54	25.80	26.18	25.28	25.78	fail	42.45	19.44	7.98	5.21	42.01	1.4	0
SURF	25.43	25.43	26.19	25.86	25.33	27.21	fail	40.13	21.22	4.49	2.5	38.67	0.4	0

Table 6.2: Comparison between the proposed network and the traditional feature-based methods during a robustness analysis.

6.6 Ablation Study

6.6.1 l_2 Loss V.S. Smooth l_1 Loss

We argue that the l_2 loss is not suitable for PFNet because the l_2 loss places emphasis on outliers. This property is contrary to the goal of PFNet, which aims to regress dominant inliers. A small quantity of outliers does not adversely affect the accuracy of a homography computed using PFNet as we use RANSAC to further process predictions. To verify this hypothesis, we train PFNet using a smooth-l_1 and l_2 loss respectively. Both of the training phases contains 300 epochs and the learning rate is divided by 10 after each 100 epochs. A sufficiently large number of epochs can help us examine loss the trend during training.

Figure 6.7 shows the MACE trend for training and validation for the two losses. We find that the smooth-l_1 consistently converges faster than the l_2 in both training and validation phases. The smooth-l_1 trained PFNet needs far fewer epochs to reach the same accuracy as the l_2 trained PFNet.

6.6.2 Comparison between FRCN, FRCN-IB, and PFNet

To thoroughly verify our hypothesis on the difference between these networks, we provide an extra set of experiments to compare the performance of FRCN, FRCN-IB, and PFNet. Regarding the experimental setting, we use 300 epochs for training and divide the learning rate by 10 after each 100 epochs. The loss function in this case is set to the l_2 loss to supplement the smooth-l_1 loss evaluation presented earlier. All other parameters remain the same. The 300 epoch training ensures that training for all networks has converged to best identify the most effectively network for homography estimation.

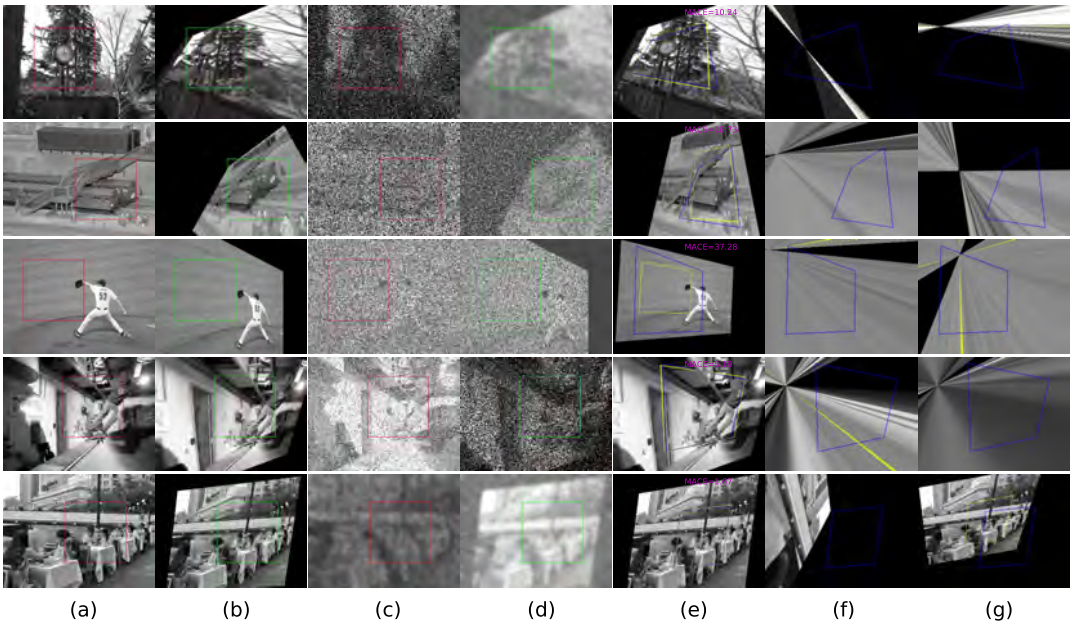


Figure 6.6: Visualization of estimating homographies in heavily contaminated image pairs. (a): Input image I_{A}. The red square shows the patch we extracted for training (b): The warped image I_{B}. The green square has the same location as the red one. (c): We randomly select two contaminators from the list and use the highest value in the value range to corrupt I_{A}. (d) I_{B} is contaminated in the same way as I_{A} but separately. (e)-(g): The results obtained from our method, SIFT, and SURF respectively. To improve visualization, we draw the ground truth homography using a blue square. The yellow squares are the predicted homographies.

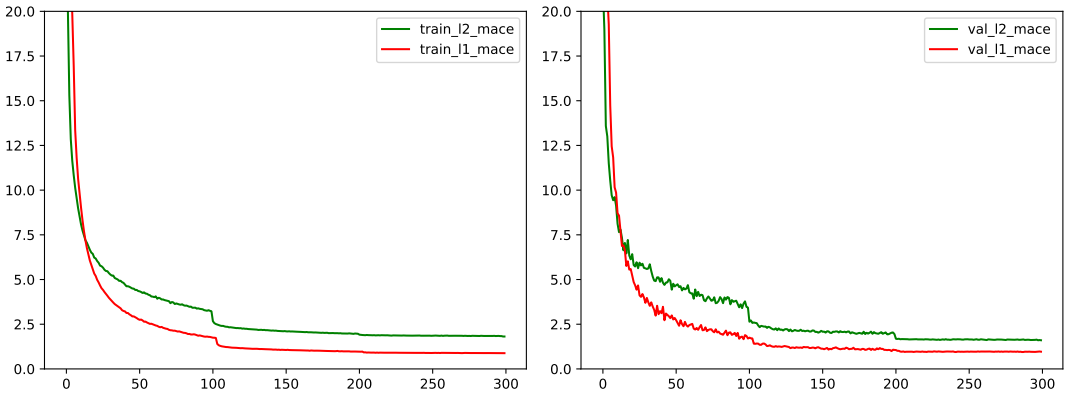


Figure 6.7: The training and validation MACE of PFNet regarding smooth-l_1 and l_2 loss.

Method	MACE
FRCN	3.21
FRCN-IB	2.39
PFNet	**1.46**

Table 6.3: Mean average corner error (MACE) comparison between FRCN, FRCN-IB, and PFNet using l_2 loss for training.

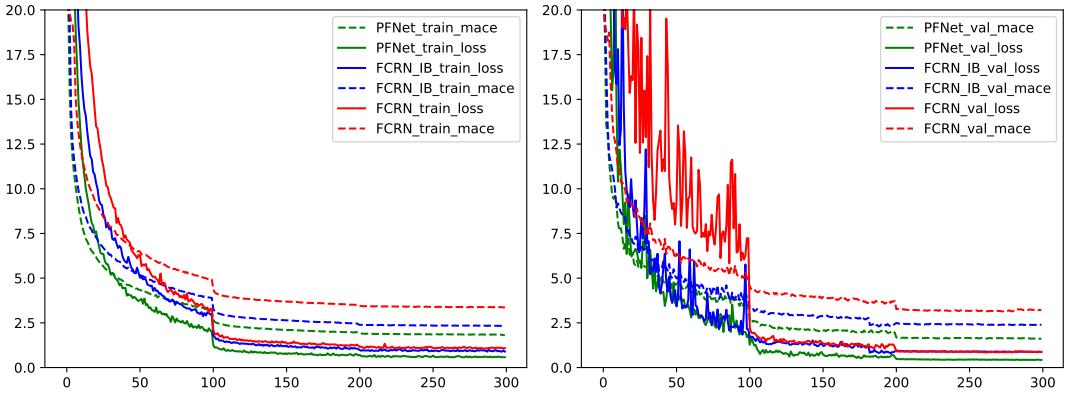


Figure 6.8: The MACE and the loss value in the training and validation phases for FCRN, FCRN-IB, and PFNet using an l_2 loss.

Table 6.3 reports the results of these three networks. We find that PFNet greatly outperforms the other two networks. The greater the number of training epochs, the more noticeable performance gap among these three networks.

Observing Figure 6.8, we can see that the FCRN validation loss oscillates significantly during the first 100 epochs, suggesting that the network does not generalise as well to unseen data as the proposed approach. With respect to training and validation metrics, PFNet consistently outperforms the other two networks.

6.6.3 Additional qualitative results

We provide additional qualitative results with a focus on examining the applicability of the PFNet to real world scenes. Considering there are no publicly available real world homography datasets, we use photos captured from an IPhone 6S as the experimental materials. To generate I_{A}, we capture a random scene on a university campus. Regarding the generation of I_{B}, we capture the same scene but randomly perturb the pose of the camera to artificially generate a perspective transform. All images are then resized to 320×240.

Figure 6.9 shows the qualitative results of the in-the-wild homography estimation. One can clearly see that the warped image generated by the predicted homography (see the last column of Figure 6.9) is almost the same as I_{B} (the second column of Figure 6.9). The high level of performance provides further evidence that PFNet, which is trained on a large-scale image dataset, has good applicability to real world scenes. More examples can be found in Figures 6.10 and 6.11.

6.7 Chapter summary

In this chapter, we introduced a new parameterization for homography estimation, i.e., the perspective field, which models the nature of the homography - pixel-to-pixel bijection. Unlike previous CNN approaches that require fully-connected layers for regressing, the PF can be easily learned by FCN architectures to significantly reduce computational complexity. To exploit this parameterization, we developed the PFNet architecture, which can naturally learn the PF and enables end-to-end training. The proposed deconvolution block follows the residual function fashion, and as such input information can be passed to the output directly by a

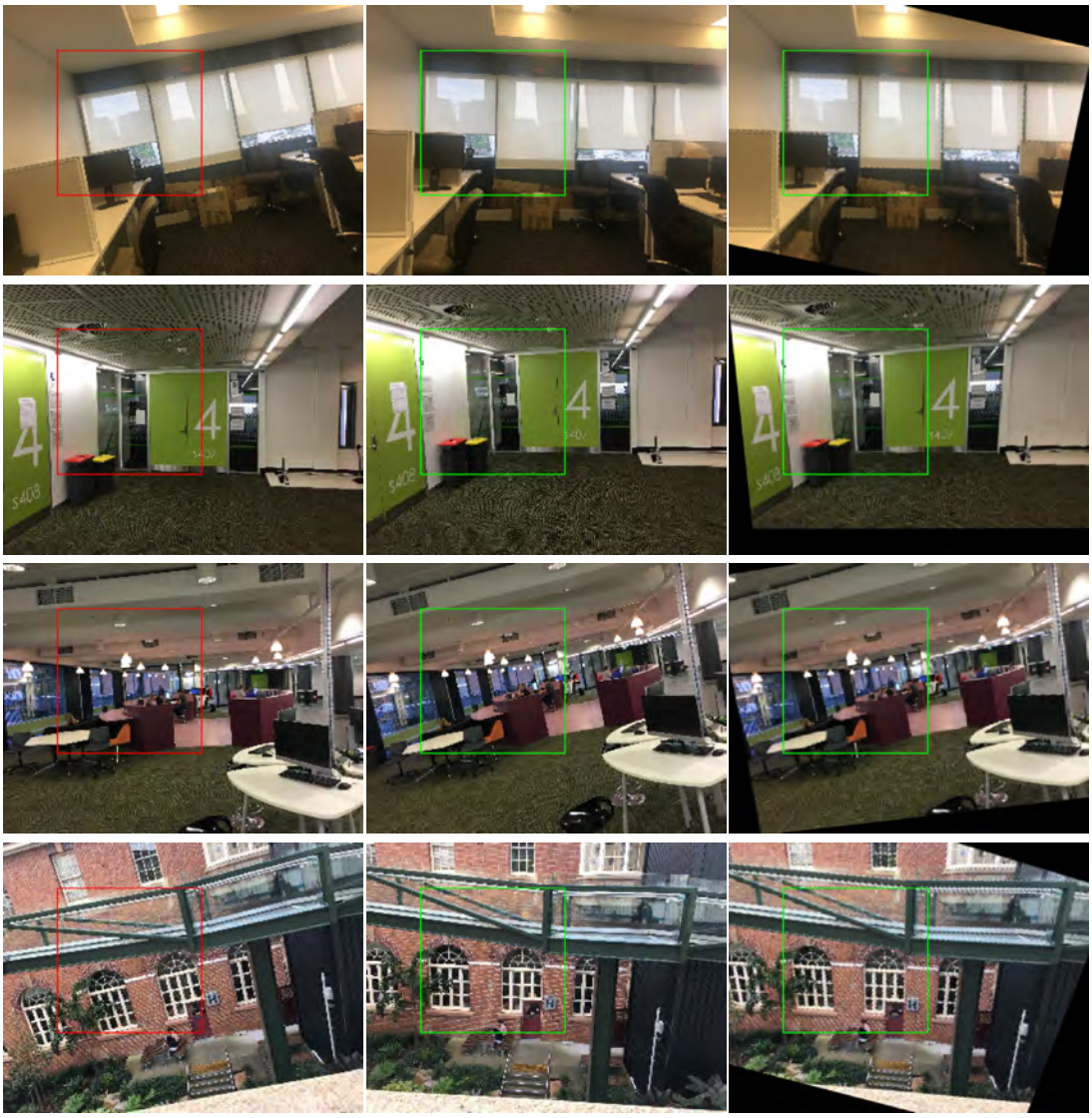


Figure 6.9: Demonstration of the applicability of the PFNet on real-world scenes. From left to right: I_{A}, I_{B} (captured from the same scene using a randomly perturbed camera pose), the image generated by warping I_{A} using the predicted homography. Every square has the same place in the image. The squares in the first two columns of the figure are used to generated the input tensor for PFNet. The image content in the green square in the last column is used to qualitatively compare with that shown in the second column. We clearly see that the image content contained in the squares of the second and third column are almost the same. These results demonstrate that our method has a good applicability to real-world scenes.

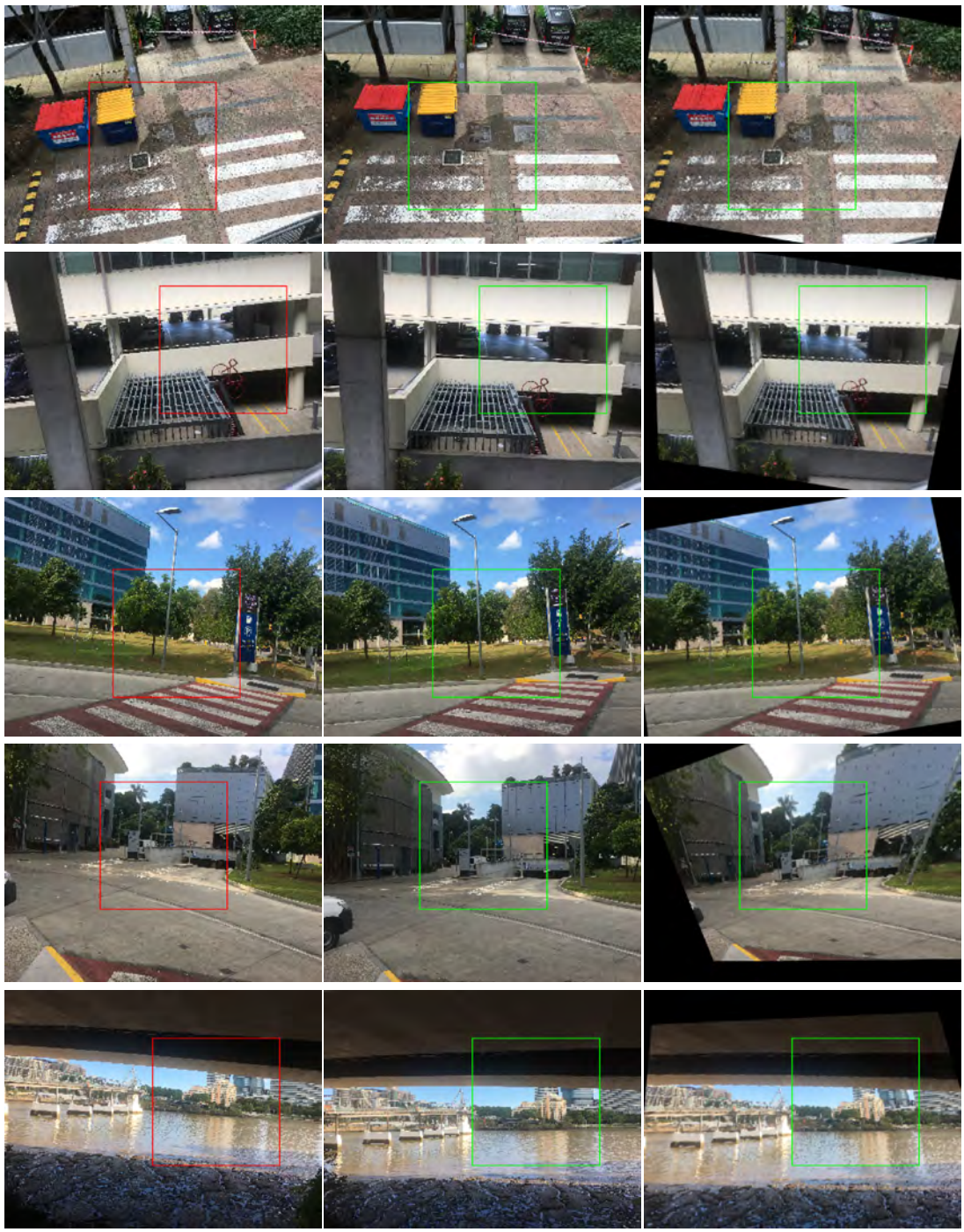


Figure 6.10: Extra examples of predictions by PFNet for real-world scenes. Column and red/green bounding box descriptions are as per Figure 6.9.

Figure 6.11: Extra examples of predictions by PFNet for real-world scenes. Column and red/green bounding box descriptions are as per Figure 6.9.

shortcut connection for upsampling. Our experiments demonstrate the capacity and robustness of our model with respect to baselines and previous state-of-the art works; and the qualitative experiments also show visually pleasing results.

Comparing this approach to the geometric methods used in Chapters 4 and 5, we can see from the evaluation with noisy data that the proposed approach is much more robust to challenging scene conditions. However it is important to note that the proposed approach makes no allowance for active pixels and potentially erroneous matches that are a result of these, as the method of Chapter 5 does.

Up to this point, projective geometry based and deep learning based homography estimation methods have been proposed in Chapters 4, 5, and 6 respectively. In the next chapter, we take a further step and combine the benefits of both deep learning and projective geometry in a single framework for assisting vehicle recognition.

Chapter 7

Geometry-constrained Car Recognition Using a 3D Perspective Network

7.1 Introduction

In Chapter 6 an approach to estimating a homography in a geometrically interpretable way was presented. While this achieved state of the art performance, it is only suitable for extracting geometric information for a pair of images as a whole. A growing body of work [74, 86] has sought to exploit 3D representations to aid object (or fine-grained object) recognition, however these methods are often reliant on 3D CAD models, or formulations that don't the extraction of geometrically correct representations [118]. In this chapter we seek to address this by proposing a fine grained vehicle classification approach that uses an accurately estimated 3D bounding box to extract a more discriminative feature representation.

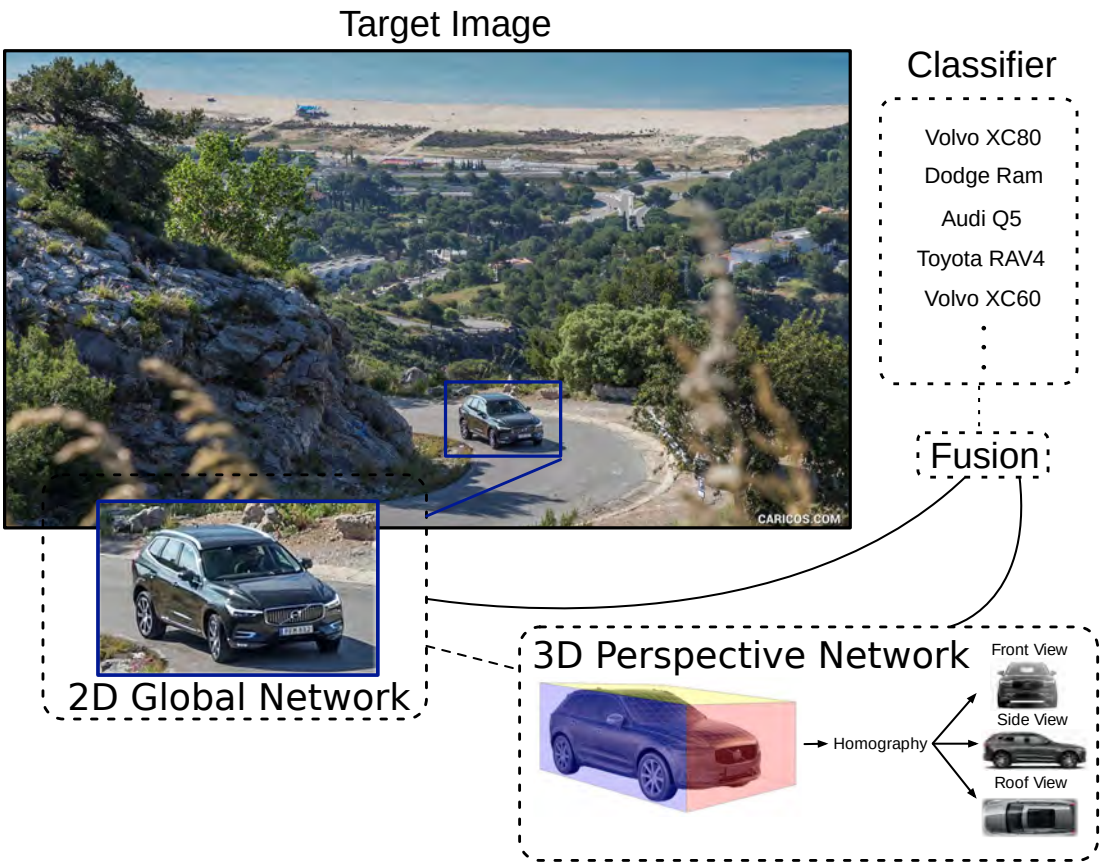


Figure 7.1: Illustration of 3D bounding box based representation for car recognition. Consider the image of the car shown above. Even though the vehicle is shown on a flat 2D image, our model can estimate and leverage knowledge from 2D appearance as well as the rigid 3D bounding box of the vehicle to produce a viewpoint-normalized representation, which is able to improve vehicle recognition performance.

Vehicle classification is an important task in intelligent transportation. A holy grail for traffic surveillance is the ability to automatically recognize and identify vehicles from visual information alone, which could enable automated car model analysis, which is helpful for innumerable purposes including regulation, description, and indexing vehicles.

One key idea shared by recent vehicle recognition algorithms is to use an ensemble of local features extracted from discriminative parts of the vehicle [56, 66, 72, 73, 102, 142, 156], which can be located using either part annotations

or attention mechanisms. These approaches, given part annotations, [56, 72] learn the corresponding part detectors and then assemble these to obtain a uniform representation of the vehicle, which is used for category classification. To overcome the need for part annotations, recent advances [66, 73, 102, 142, 156] make use of attention mechanisms to identify salient spatial regions automatically. Despite these part-aware methods successfully leveraging spatial information, they are still 'flat', i.e., built on independent and 2D views.

3D-aware methods have been shown to be promising alternatives to part-aware approaches. For instance, [74, 86] exploit the fact that aligning a 3D CAD model or shape to 2D images significantly eliminates the variation caused by viewpoint changes, which is shown to be the main obstacle for vehicle categorization. However, these methods have limited generality as they require 3D CAD models for vehicles.

To address these issues, we instead propose to directly use the 3D bounding box of the vehicle to normalize viewpoint variation. Our work is summarized in Figure 7.1. It combines the benefits of a 3D bounding box representation with a 2D appearance-based depiction of a vehicle. Our proposed method has three components: the Global Network (GN), the 3D Perspective Network (3DPN), and the Feature Fusion Network (FFN). GN detects and extracts relevant global appearance features of vehicles from input RGB images. 3DPN predicts the 3D bounding box under the geometric constraints of the vanishing points using the proposed vanishing point loss. With the assistance of the predicted 3D bounding box, the 3DPN further generates a viewpoint-aligned feature representation using the proposed RoI perspective pooling layer to extract 3D data in a geometrically correct manner. Finally, the features generated from the GN and the 3DPN are merged in the FFN and then used for vehicle recognition. Our contributions can be summarized as follows:

- A unified network architecture for vehicle recognition which takes full advantage of the 2D and 3D representations is presented. To our best knowledge, our method is the first work which extracts a meaningful feature representation from the 3D bounding box to enhance vehicle recognition.
- A 3D bounding box predictor, i.e. 3DPN, is proposed for vehicle recognition to use 3D information in a meaningful and correct manner, without the challenge of requiring a 3D CAD model.
- We propose a new parameterized CNN pooling layer termed RoI Perspective pooling, which grants the CNN the adaptive transformation modeling capability to normalize vehicle viewpoints in the feature space given the 3D bounding box.
- We introduce a geometrically interpretable loss (vanishing point loss) to elegantly enforce the consistency of the predicted 3D bounding box to improve regression accuracy.

We evaluate our proposed method on the vehicle classification and verification tasks in the BoxCars benchmark and show that without using 3D annotations, we achieve better results than the state-of-the-art methods that only use 2D information.

7.2 Related Work

We review the previous works on vehicle recognition, cuboid detection, and RoI pooling, which are related to our approach.

Vehicle classification: Since our model uses only a single image to recognize vehicles, methods which use extra information, such as 3D CAD models, are not reviewed. 2D vehicle recognition can be classified into two categories: part-annotation (PA) and attention-mechanism (AM) methods. While PA methods [56, 72, 118] are

able to achieve high performance by extracting local feature representation from detected vehicle parts, they are reliant on part annotations. The labor intensive annotation is usually not possible during inference when applying such methods to a real scene. [56] detects each discriminative part of a vehicle and then generates a uniform feature using the HOG descriptor. [72] trains a classification CNN by combining both local and global cues, which have been previously annotated. Similarly, [118] uses a pre-annotated 3D bounding box to generate a 2D "flat" representation. To alleviate the essential requirement of annotations, AM methods [37, 66, 137, 149] have been extensively researched in recent years. One of their common features is to locate discriminative parts of a vehicle automatically using attention mechanisms. [66] aims to determine an affine transformation to map a entire vehicle to its most discriminate viewpoint in a global way. [37, 137, 149] generate features locally without using a projective transformation, and then uses them for recognition.

In contrast to previous methods, we take a further step towards taking full advantage of both the 2D and 3D representation of a vehicle. Compared with PA and AM methods, our method is able to predict the 2D and 3D bounding box simultaneously. It can generate viewpoint normalization features using appropriate geometric constraints in a geometrically explainable way. To our best knowledge, our work is the first to use a 3D bounding box for feature representation, which enhances recognition performance. Moreover, compared to 3D-aware methods, our method is totally free from 3D CAD models, which are difficult to obtain in practice.

Cuboid Detection: It has been observed that vertex localization typically aids cuboid detection. [49, 58, 80, 141] localize vertices using corner detectors and then construct cuboids through the geometric relationships among all vertices. Following the success of these geometry-based methods, [30] regresses vertices of the cuboids through a Faster-RCNN-based model. Subsequently, vertex predictions are refined by utilizing vanishing points [54]. However, this refinement step is separate from

the network training stage, and the vanishing points computed from inaccurate predictions often lead to significant error. Unlike [30], we use the proposed vanishing point (VP) regularization to encode the VP constraint of the eight vertices during network training. It allows our model to avoid any post refinement to redress vertices.

RoI Pooling: RoI pooling layers, such as RoIAlign [55] and RoIWarp [25], have been extensively used in extracting fixed-size features from a rectangular RoI. However, applying them to a quadrilateral RoI is problematic. For example, extracting features from the 2D bounding box of a quadrilateral RoI results in unrelated regions being included in the representation due to the irregular shape. This situation often happens when projecting rectangular regions to the image plane using a perspective transformation. To address this issue, we propose RoI Perspective (RoIPers) pooling to efficiently and effectively extract fixed-size features from quadrilateral regions caused by perspective warping. One of the key contributions is that RoIPers learns to sample corresponding feature points from the source to target through a perspective transformation.

7.3 Methodology

Overview: Our goal is to design an architecture that jointly extracts features in terms of both the 2D and 3D representation for vehicle recognition. The proposed model (see Figure 7.2) is composed of three sub-modules:

1. *Global Network*, which is a learned 2D detection network, and aims to produce global feature maps representing 2D texture information, and localize regions of interest (RoIs) that contain vehicles.
2. *3D Perspective Network*, which contains two task-specific components, 3D bounding box regression and 3D perspective feature extraction, which is obtained using the RoIPerspective layer to normalize the vehicle viewpoint in

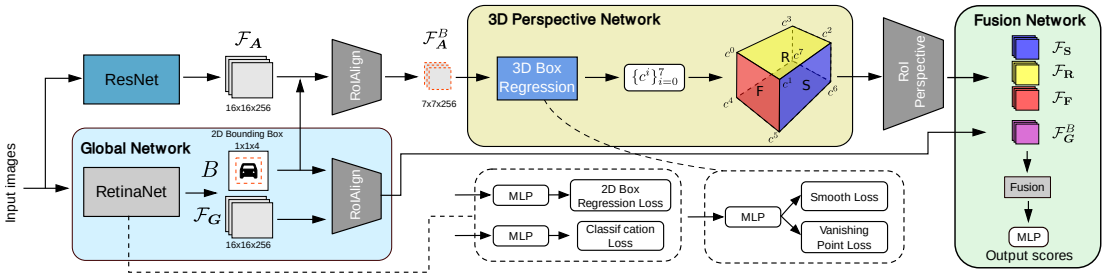


Figure 7.2: Overview of the proposed model. The model is composed of three main components: (A) Global Network (GN), which aims to localize the vehicle and extract its 2D features. (B) 3D Perspective Network (3DPN), which performs 3D bounding box regression by taking the anchor from the predicted 2D bounding box and generates normalized viewpoint features of the three main faces (front, roof, and side) of the vehicles. (C) Feature Fusion Network, which fuses the features from the **GN** and **3DPN** by applying multi-modal compact bilinear (MCB) [38] pooling. **F**, **R**, **S** in the predicted 3D bounding box represents the front/rear, roof, and side respectively.

feature space.

3. *Feature Fusion Network*, which combines 2D global and 3D perspective features using multimodal compact bilinear pooling (MCB) [38], and then outputs the category of the detected vehicle.

7.3.1 Global Network (GN)

The GN uses a variant of RetinaNet [84] to localize the vehicle using a 2D bounding box. RetinaNet is a dense object detector composed of a CNN-FPN [83] backbone, which aims to extract a convolutional feature map, $\mathcal{F}_{\boldsymbol{G}}$, over an entire input image. Two task-specific subnetworks are attached to the backbone network to perform object classification and 2D object box regression respectively. RetinaNet offers comparable performance to complex two-stage detectors [55, 106] while retaining the advantages of one-stage detectors such as inference speed and model optimization. These attributes are desirable when adopting it as one component in an end-to-end classification framework. To adapt the original RetinaNet as the

part of our network, we make the following modifications:

ROIAlign process: We add an ROIAlign layer [55] after the 2D box decoding process. The detected 2D bounding box of the vehicle is denoted $B = (B_x, B_y, B_w, B_h)$, where B_x, B_y are the left-top corner coordinates with respect to x and y axis. B_w, B_h represents the width and height of B respectively. The ROIAlign layer combined with the detected 2D bounding box coordinates is able to produce a fixed-sized global feature representation which comprises the vehicle, termed $\mathcal{F}_{\boldsymbol{G}}^{B}$. In particular, this modification ensures that errors in the extracted 2D coordinates can be back propagated through the GN when trained jointly with other network components.

7.3.2 3D Perspective Network (3DPN)

Figure 7.2 illustrates the architecture of the 3DPN. Its role is to provide geometrically-interpretable features by normalizing the vehicle viewpoint to account for perspective distortion. To achieve this, the 3DPN takes as input $\mathcal{F}_{\boldsymbol{A}}^{B}$, which is the feature map pooled from B at $\mathcal{F}_{\boldsymbol{A}}$ using RoIAlign. $\mathcal{F}_{\boldsymbol{A}}$ is the auxiliary feature map extracted from an off-the-shelf CNN. We then estimate the coordinates of eight vertices' of the 3D bounding box, $C : \{c^i\}_{i=0}^{7}$, using a 3D bounding box regression network. Subsequently, C is used to normalize the viewpoint of the vehicle in $\mathcal{F}_{\boldsymbol{A}}^{B}$ using our proposed **RoI Perspective** (**RoIPers**) layer. As a result, $\mathcal{F}_{\mathbf{R}}$, $\mathcal{F}_{\mathbf{F}}$, and $\mathcal{F}_{\mathbf{S}}$, representing perspective transformed feature maps from the quadrilaterals formed by the roof (**R**), front (**F**), and side (**S**) of the vehicle, are extracted. Below we describe the 3D bounding box regression network with the proposed vanishing point loss, and RoIPers respectively.

3D bounding box regression branch: Instead of using the absolute coordinates of the 3D box in the image space directly, we estimate them in an RoI relative coordinate system by leveraging the 2D bounding box as an anchor. For

each $\{c^i\}_{i=0}^7$ in the image coordinate system we first transform those points to the 2D-bounding-box relative coordinate system: $\hat{c}_x^i = (c_x^i - B_x - B_w/2)/B_w$, and $\hat{c}_y^i = (c_y^i - B_y - B_h/2)/B_h$, where $\{\hat{c}^i\}_{i=0}^7$ is the training target of this branch. The 3D bounding box regression network takes $\mathcal{F}_{\boldsymbol{A}}^B$ as the input feature map. Then it utilizes two convolution layers ($3 \times 3 \times 256$) and a multilayer perceptron (512×16) to regress all x and y coordinates of $\{\hat{c}^i\}_{i=0}^7$ (leaky ReLu are used as activations). The loss function used to train this sub-network is:

$$L_{\text{3Dbranch}} = L_{\text{smooth}l_1}(\hat{c}^*, \hat{c}) + L_{\text{vp}}, \tag{7.1}$$

where $\hat{c}^*$ is the ground-truth locations for $\hat{c}$, $L_{\text{smooth}l_1}$ is the standard smooth-l_1 loss and L_{vp} is the proposed vanishing point regularization loss to ensure that C satisfies perspective geometry (i.e., every parallel edge of C intersects at the same vanishing point).

Vanishing point regularization: A standard smooth-l_1 loss lacks the capacity to impose perspective geometry constraints on $\{c^i\}_{i=0}^7$, which constructs a projective cuboid in the image plane. We thus propose a 3D geometric vanishing point regularization loss, which forces $\{c^i\}_{i=0}^7$ to satisfy perspective geometry during regression, as such the predicted vertices don't require camera calibration data or post preprocessing for refinement [30].

In projective geometry, the two-dimensional perspective projections of mutually parallel lines in three-dimensional space appear to converge at the vanishing point. The required condition for convergence of three lines is that the determinant of the coefficient matrix is zero. The proposed vanishing point loss encodes this geometry constraint (as shown in Figure 7.3) by minimizing the determinants of all sets of three parallel edges of the vehicle. Formally, taking three parallel lines $\mathbf{l}_{c^0c^3}$, $\mathbf{l}_{c^1c^2}$, $\mathbf{l}_{c^5c^6}$ in $\mathbf{F}$ as examples (as shown in Figure 7.3), the vanishing point

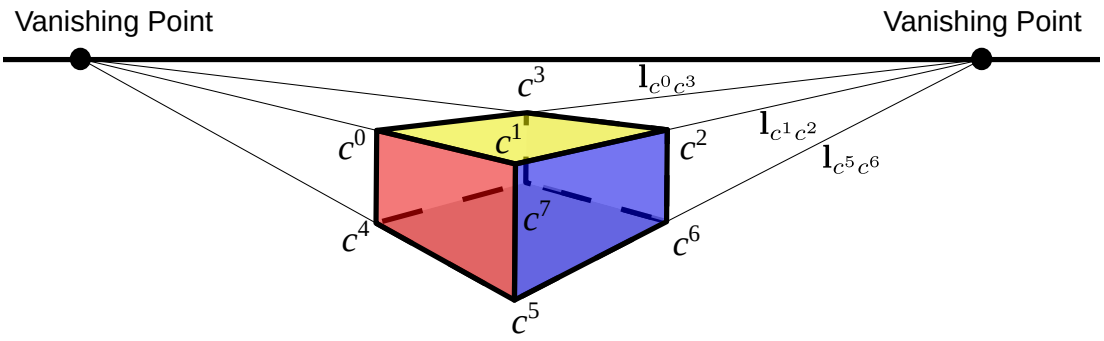


Figure 7.3: Illustration of the vanishing point. To simplify the visualization, only two vanishing points (in the **F** and **S** directions) are plotted. The lines $\mathbf{l}_{c^0c^3}$, $\mathbf{l}_{c^1c^2}$, and $\mathbf{l}_{c^5c^6}$ contribute to the first part of $L_{\text{vp}_\mathbf{F}}$.

loss and the coefficient matrix are expressed as:

$$L_{\text{vp}_{\mathbf{F}_1}} = (D_{\text{vp}_{\mathbf{F}_1}})^2, D_{\text{vp}_{\mathbf{F}_1}} = \begin{vmatrix} m_{c^0c^3} & n_{c^0c^3} & l_{c^0c^3} \\ m_{c^1c^2} & n_{c^1c^2} & l_{c^1c^2} \\ m_{c^5c^6} & n_{c^5c^6} & l_{c^5c^6} \end{vmatrix}, \tag{7.2}$$

where $m_{c^ic^j}x + n_{c^ic^j}y + l_{c^ic^j} = 0$ is the line equation of $\mathbf{l}_{c^ic^j}$, and D is the determinant of the matrix. $L_{\text{vp}_{\mathbf{F}_1}}$ is the first part of $L_{\text{vp}_\mathbf{F}}$ using the first three lines ($\mathbf{l}_{c^0c^3}$, $\mathbf{l}_{c^1c^2}$, and $\mathbf{l}_{c^5c^6}$; see Figure 7.3 for details.). Similarly, we build the second part, $L_{\text{vp}_{\mathbf{F}_2}}$, using the last three lines ($\mathbf{l}_{c^0c^3}$, $\mathbf{l}_{c^4c^7}$, and $\mathbf{l}_{c^5c^6}$) in the diagonal to form up the final vanishing point regularization $L_{\text{vp}_\mathbf{F}} = L_{\text{vp}_{\mathbf{F}_1}} + L_{\text{vp}_{\mathbf{F}_2}}$ for the **F** direction, and repeat for the R and S directions. Therefore, the vanishing point loss of the whole vehicle, $L_{\text{vp}} = L_{\text{vp}_\mathbf{R}} + L_{\text{vp}_\mathbf{S}} + L_{\text{vp}_\mathbf{F}}$.

RoIPerspective: We propose RoIPerspective (RoIPers) pooling to ensure that when extracting features from arbitrary quadrilateral images, we sample uniformly within the quadrilateral and do not sample from points outside of it. RoIPers consists of two steps: parameterized sampling grid generation, and differentiable feature sampling. Suppose that we have a source feature map $\mathcal{F}_{\text{source}}$, which is extracted from the input image using a standard CNN, and a corresponding

quadrilateral RoI, Q. The objective of RoIPers is to map the feature inside Q of $\mathcal{F}_{\text{source}}$ to a fixed-size target feature map, $\mathcal{F}_{\text{target}}$. We first use a four-correspondence DLT [54] to obtain the homography $\mathbf{H}$ between Q and $\mathcal{F}_{\text{target}}$:

$$\begin{bmatrix} q_x^i \\ q_y^i \\ 1 \end{bmatrix} \sim \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} & \mathbf{H}_{13} \\ \mathbf{H}_{21} & \mathbf{H}_{22} & \mathbf{H}_{23} \\ \mathbf{H}_{31} & \mathbf{H}_{32} & 1 \end{bmatrix} \begin{bmatrix} t_x^i \\ t_y^i \\ 1 \end{bmatrix}, \tag{7.3}$$

where $\{t^i\}_{i=1}^4$ and $\{q^i\}_{i=1}^4$ are the four corners of $\mathcal{F}_{\text{target}}$ and Q respectively. Thus given the coordinate of each pixel in $\mathcal{F}_{\text{target}}$, we can obtain their corresponding sampling points in $\mathcal{F}_{\text{source}}$ using $\mathbf{H}$. In the feature sampling step, the exact value of each sampling point at $\mathcal{F}_{\text{source}}$ can be computed easily using bilinear interpolation at four regularly sampled locations, i.e.,

$$\mathcal{F}_T^k = \sum_n^H \sum_m^W \mathcal{F}_S^{mn} \max(0, 1-|s_x^k - m|) \max(0, 1-|s_y^k - n|), \tag{7.4}$$

where k is the pixel index of $\mathcal{F}_{\text{target}}$, and s^k is the corresponding sampling point in $\mathcal{F}_{\text{source}}$. H and W are the height and width of $\mathcal{F}_{\text{target}}$ respectively. Using this approach, the feature inside Q is extracted as a fixed-size target feature map $\mathcal{F}_{\text{target}}$. Figure 7.4 visualizes the process of generating 3D features from $\mathcal{F}_{\mathbf{S}}$. $\mathcal{F}_{\mathbf{R}}$ and $\mathcal{F}_{\mathbf{F}}$ can be obtained in a similar manner.

7.3.3 Feature Fusion Network (FFN)

Figure 7.5 visualizes the architecture of the FFN, which is designed to merge feature maps extracted from the GN and 3DPN to recognize a given vehicle. Three 3D feature representations $\mathcal{F}_{\mathbf{S}}$, $\mathcal{F}_{\mathbf{R}}$, $\mathcal{F}_{\mathbf{F}}$ and one global feature $\mathcal{F}_G^B$ are processed through two identity blocks [57], followed by a global average pooling (GAP) layer, to generate refined feature vectors respectively. Please note that the three feature vectors from **F**, **R**, and **S** are concatenated together to form a

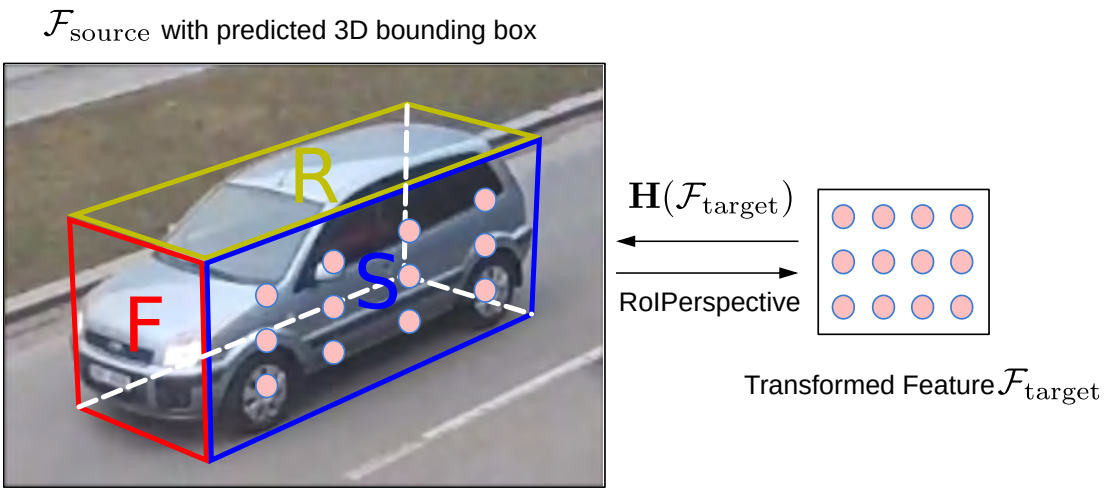


Figure 7.4: The process of extracting perspective corrected features from **S** using RoIPers $\mathcal{F}_{\text{source}}$ and $\mathcal{F}_{\text{target}}$ are the source and target feature maps respectively. To improve visualization, we overlay the input image with $\mathcal{F}_{\mathbf{S}}$ to show where the predicted 3D bounding box and sampling points (colored by pink) are. The sampling points in **S** are generated using **H** by projecting every pixel in $\mathcal{F}_{\text{target}}$ back to $\mathcal{F}_{\text{source}}$. Then the exact value of each point is computed via bilinear interpolation at four regularly sampled locations. Finally, these points are gathered to generate $\mathcal{F}_{\text{target}}$.

single perspective feature vector carrying discriminative perspective information representing different vehicle views. The final feature vector of dimension $16,000$ is obtained by applying multi-modal compact bilinear (MCB) [38] pooling on the global and perspective feature vector. The reason for using MCB is that it is normally used to facilitate the joint optimization of two networks generating features which lie on different manifolds. The two feature vectors are obtained from two different networks (GN vs. 3DPN), i.e., they lie on different manifolds. The final feature vector is passed through two fully-connected (fc) layers of size 2048 and the number of categories, respectively. Up to this point, our full model, which is composed of three network components, can be trained jointly with a single optimization process using the following multi-task loss function:

$$L = \lambda_1 L_{\text{2DGN}} + \lambda_2 L_{\text{3DBranch}} + \lambda_3 L_{\text{CrossEntropy}}, \tag{7.5}$$

where L_{2DGN} is the focal loss [84] used to train the GN, L_{3DBranch} is defined in

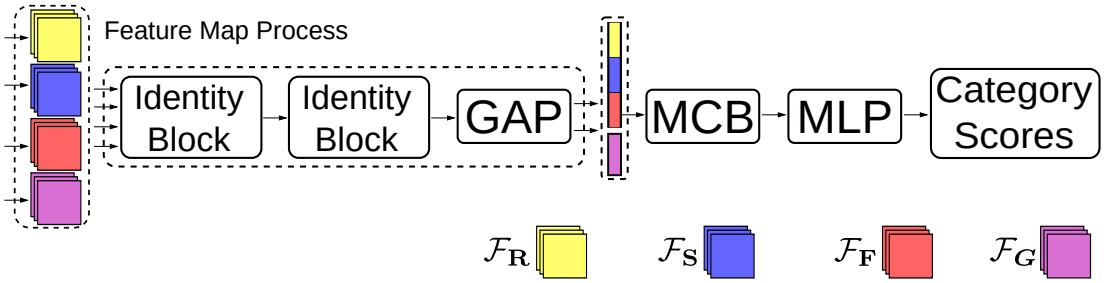


Figure 7.5: Feature Fusion Network (FFN) architecture. Four feature maps are processed by two identity blocks and global average pooling (GAP) to generate corresponding feature vectors. Then these feature vectors are merged using multi-modal compact bilinear (MCB) pooling, followed by two fully-connected layers, to output the final category scores.

Equation 7.1, and $L_{\text{CrossEntropy}}$ is the cross entropy loss to train the last softmax layer in the FFN.

7.4 Experiments

To the best of our knowledge, the BoxCars dataset [118] is the only dataset which provides both 3D and 2D bounding box annotations for vehicle recognition in the computer vision community. Therefore, we use it to evaluate our model. BoxCars contains 63,750 images, which are collected from 21,250 vehicles of 27 different makes. All images are taken from surveillance cameras. BoxCars consists of two challenging tasks: classification and verification. Regarding the classification task, the dataset is split into two subsets: *Medium* and *Hard*. The *Hard* protocol has 87 categories and contains 37,689 training images and 18,939 testing images. The *Medium* protocol is composed of 77 categories and has 40,152 and 19,590 images for training and testing respectively. The main difference between the *Medium* and *Hard* splits is that *Hard* considers make, model, submodel, and model year; while *Medium* does not differentiate model year.

With respect to the verification task, BoxCars has three well defined protocols that provide *Easy*, *Medium*, and *Hard* cases. The *Easy* protocol is composed of pairs of vehicle images recorded from the same unseen camera. Camera identities are no longer fixed in the *Medium* protocol. The *Hard* protocol not only draws vehicle pairs from different unseen cameras, but also takes into account vehicle model years.

7.4.1 Implementation Details

GN and 3DPN Architecture: The RetinaNet [84] backbone used in the GN is built on MobileNetV2 [110]. ResNet101 [57] with the first 4 stages is selected as the 3DPN architecture. We select different backbones for the 3DPN and GN because features produced from them lie on different manifolds. This can enhance the representation of the unified network.

Training and Optimization: We have implemented our model using Keras [20] and TensorFlow [1]. In this work, we adopt a pragmatic 2-step training approach to optimize the whole network. In the first step, we train the GN solely so that it can output the 2D bounding box correctly, which is important to train the 3DPN which takes the 2D bounding box as input. In the second step, we train all three network components, the GN, 3DPN, and FFN, together in an end-to-end manner. λ_1, λ_2, λ_3 are set to 1, 0.1, and 1 respectively. SGD is chosen as our optimizer and its momentum is set to 0.9. The initial learning rate is 0.02, and is divided by 10 after every 15 epochs. The batch size is set to 30. The model optimisation is ceases when training reaches 45 epochs. Each batch takes approximately 2s on a NVIDIA Tesla P100 GPU and in total the model takes about 12 hours to converge.

7.4.2 Vehicle Classification Results

Table 7.1: Overall classification accuracy on BoxCars dataset. *M* and *H* represent the *Medium* and *Hard* splits. Top-1 and -5 accuracy are denoted as T-1 and T-5.

Method	Input Size	Detection?	3D?	Attention?	*M* T-1	*M* T-5	*H* T-1	*H* T-5
RetinaNet [84]	256×256	✓	✗	✗	66.52	-	59.4	-
Faster-RCNN [106]	256×256	✓	✗	✗	67.23	-	62.73	-
Ours-*det*	256×256	✓	✓	✓	**78.45**	**93.39**	**75.18**	**91.53**
NTS [149]	224×224	✗	✗	✓	80.40	92.37	76.31	90.42
DFL [137]	224×224	✗	✗	✓	76.78	91.94	70.25	88.405
BoxCar [118]	224×224	✗	✗	✓	75.4	90.1	73.1	89
RACNN [37]	224×224	✗	✗	✓	72.21	88.47	67.5	86.83
STN [66]	224×224	✗	✗	✓	64.33	81.92	59.76	80.13
Ours-*cls*	224×224	✗	✓	✓	**81.27**	**93.82**	**77.08**	**91.97**

(a) The correctly predicted examples.

(b) The incorrectly predicted examples.

Figure 7.6: Qualitative results visualization of Ours-*det* on the BoxCars dataset. (a): examples in which the 2D and 3D bounding box are correctly predicted. (b): examples containing errors in prediction. The first row shows 2D and 3D bounding box estimations from our proposed method. The detected 2D bounding box is denoted using a magenta box. The 3D bounding box is colored to indicate quadrilateral regions: red is the front, blue is the side and yellow is the roof. The second row represents the corresponding sampling points computed by RoIPers (Sec. 7.3.2).

Baselines: Since our model recognize vehicles from a single image, only state-of-the-art methods [57, 75, 84, 106, 110, 118] which use a single image are selected to compare with our method. These methods are divided into two evaluation categories: (1) detection-like (*det*-like) networks [84, 106], in which localization and classification of the vehicle are performed simultaneously; and (2) classification-like (*cls*-like) networks [37, 66, 118, 137, 149] in which vehicles are cropped using the annotated bounding box before network training. Since our model is composed of a RetinaNet built by MobileNetV2, detection-like networks [84, 106] also built with MobileNetV2 are compared to ours to demonstrate that the performance gain is due to the use of the 3D feature representation. Regarding attention mechanism methods, [37, 66, 118, 137, 149] are used to demonstrate the efficiency of our method that uses both the 2D and 3D feature representation. We note that [118] superficially resembles our method. However, [118] requires 3D bounding box annotations to preprocess images before network training, and the vehicle in the image is still "flat", i.e., lacking of 3D meaning. Furthermore, [118] requires annotated 3D boxes at inference time, while the proposed approach operates over the raw image with no further input. With respect to classification-like networks, all images are resized to 224×224. Regarding detection-like networks, images are resized to the same scale of 256 pixels as in [83]. To make fair comparison, we use the official implementations of these methods without any parameter changes.

***det*-like network results**: The upper half of Table 7.1 shows the results of *det*-like networks. One can see that Ours-*det* surpasses all *det*-like baselines by a significant margin. Since RetinaNet [84] and Faster-RCNN [106] share the same backbone (MobileNetV2) with the GN in Ours-*det*, we confirm that the additional 3D feature representation significantly improves the performance obtained compared to using traditional 2D features. From Table 7.1, one can see that RetinaNet and Faster-RCNN do not have a top-5 accuracy recorded. This is because RetinaNet and Faster-RCNN output confidence scores of predicted boxes. After non-maximum suppression, the boxes with high confidence scores

are merged, and as such there is only a top-1 accuracy. Although non-maximum suppression is also performed in our method, we can still obtain top-5 accuracy due to the use of the softmax layer in the FFN.

***cls*-like network results**: To make a comparison between *cls*-like baselines and the proposed approach, we modify the 2D processing component of our model. Specifically, MobileNetV2-based RetinaNet is replaced with a vanilla MobileNetV2, in which the last global average pooling layer and following classification layer are removed. Therefore the output of this network is used as a global feature for the vehicle. The modified model for *cls*-like experiments is denoted Ours-*cls*.

The second half of Table 7.1 showcases overall classification accuracy (percent) for *cls*-like networks. We observe that Ours-*cls* consistently performs better than all baseline models with respect to classification accuracy in both the *Medium* and *Hard* splits. One can see that [37, 66] perform poorly among all *cls*-like methods, as [37, 66] only search for the most discriminative part of a vehicle. This strategy discards parts of the global feature, which captures important pose information and other subtle details. Moreover, an affine transformation used in [66] significantly increases the difficulty of vehicle viewpoint normalization. This is because an affine transformation of the 2D vehicle bounding box distorts the shape of the vehicle, and does not consider its 3D geometry. We next compare Ours-*cls* with previous state-of-the-art methods [118, 137, 149], which extract discriminative features without considering the 3D geometry. From the results, we conjecture that the combined 2D and 3D representation used in our method has better a capability for distinguishing vehicle details compared to other methods. Last but not least, Ours-*cls* is the only method which can extract the 3D structure of a vehicle. In contrast to [37, 66, 118, 137, 149], Ours-*cls* normalizes the viewpoint of a car in a geometrically correctly way, i.e., performing a perspective transformation on each side of the predicted 3D bounding box. This 3D-bounding-box based feature representation retains pose information as well as discriminative texture details

from each face, and as such the feature representation has good generality.

Qualitative results: Figure 7.6 visualizes qualitative results on BoxCars images. In Figure 7.6 (a), we see that Ours-*det* is able to determine the correct 2D location and 3D bounding box estimation of the vehicle. Figure 7.6 (b) shows some mis-estimated images. The first two columns show 3D bounding box regression performance, when the 2D bounidng box is incorrect, and the 3D estimation cannot recover from the earlier error. The last column shows a case where 2D location is predicted correctly and the 3D box estimator fails. We see that the 3D bounding box estimation tries to compensate for errors made by the 2D bounding box estimation. In addition, the sampling points computed by RoIPers for **F**, **R**, and **S** are also shown. One can see that the sampling points perfectly cover the three main sides of the vehicles, and therefore extract perspective invariant features.

7.4.3 Ablation experiments

An ablation study is conducted to analyze the effectiveness of the individual proposed components including RoiPers, the VP regularization loss and network design in the 3D bounding box regression component.

RoIPers vs. RoIAlign: In this experiment we compare the performance of the proposed RoIPers to the frequently used RoIAlign on the *Hard* and *Medium* splits. The RoIPers layers in Ours-*det* are replaced with RoI Align to evaluate the impact of RoIPers. The results are shown in Table 7.2. It can be observed that the network employing RoIPers instead of RoI Align achieves an approximately 6.1% and 2.1% improvement on *Medium* and *Hard* splits respectively. As discussed in Sec. 7.3, RoIPers can extract relevant features from arbitrary quadrilateral regions. In contrast, RoIAlign introduces unrelated information/noise using the bounding rectangles of quadrilateral RoIs, which may harm the process of extracting useful

Table 7.2: Classification accuracy results with different RoI layers in terms of the *Medium* and *Hard* splits on the BoxCars dataset.

RoI Pooling Type	*M.* T-1	*M.* T-5	*H.* T-1	*H.* T-5
RoI Align	73.97	91.23	73.65	91.13
RoI Pers.	**78.45**	**93.39**	**75.18**	**91.53**

Table 7.3: Evaluation of 3D bounding box localization and quality in terms of percentage of correct keypoints (PCK) and the proposed cuboid quality (CQ). e stands for the number of training epochs.

	e=5		e=10		e=15	
	PCK	CQ	PCK	CQ	PCK	CQ
Ours-*det*	**85.35**	1.48	**85.66**	**1.98**	**87.15**	**2.12**
DeepCuboid[30]	85.03	1.48	85.58	1.64	86.70	1.69

features from the unaligned vehicle images.

VP regularization: We evaluate the proposed VP regularization loss on 3D bounding box detection. The baseline we compare against is [30], where a Faster-RCNN-based model is implemented to detect cuboids. To make a fair comparison, we add VP regularization directly to [30], and as such the only difference between these two methods is the use of vanishing point regularization or not. Table 7.3 reports the results obtained in the *Hard* split in terms of two metrics, the percentage of correct points (PCK) [134, 150] and the proposed cube quality (CQ). A predicted vertex is considered to be correct if it lies within $0.1 \times (max(height, width))$ pixels of the ground truth annotation of the vertex. CQ is computed via $-\log L_{\text{VP}}$, where L_{VP} is the loss defined in Sec. 7.3.2. From the results, we can see that the 3D bounding box obtained from our method with VP regularization consistently outperforms that of [30] in terms of both metrics. In the early and mid training stages (the fifth and tenth epochs), our method is equal or slightly better than [30]. However, as the model is trained with more epochs, our method surpasses [30] by a large margin in terms of both the PCK and CQs; demonstrating that the proposed vanishing point regularization is of benefit to the 3D cuboid prediction in both quality and accuracy.

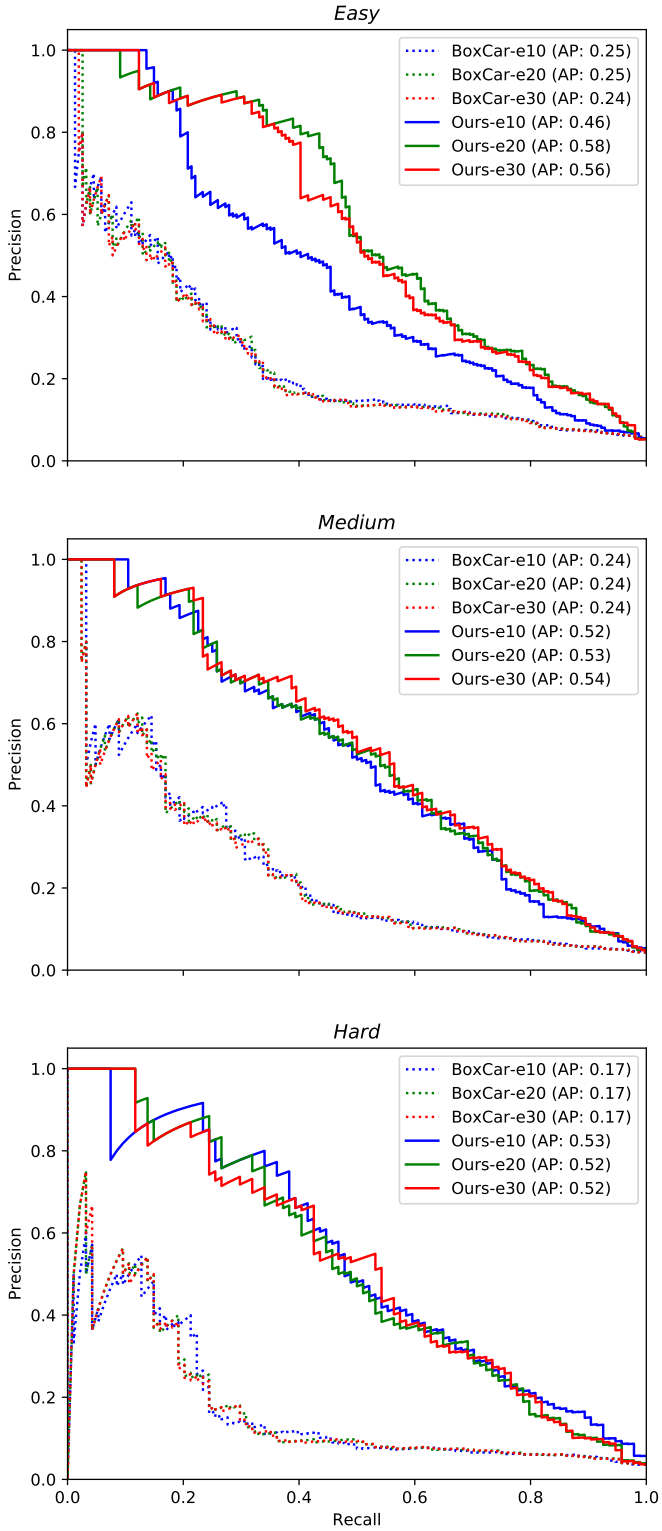


Figure 7.7: Precision-Recall (PR) curves of different models with different numbers training epoch (e denotes training epoch x). Three verification protocols *Easy*, *Medium*, and *Hard* are shown in the figure. Average Precision (AP) is given in the plot legends. Baseline results (shown as BoxCar-ex) are taken from [118].

Table 7.4: The converged smooth-l_1 loss obtained from different designs of the 3D bounding box regression branch. Fully convolutional networks (FCN) and multi-layer perceptrons (MLP) architectures are compared.

	3D Bounding Box Regression Branch	loss
FCN	conv: $256 \rightarrow 256 \rightarrow 512$	0.018
MLP	fc: $256 \rightarrow 256 \rightarrow 512$	0.016
MLP	fc: $256 \rightarrow 256 \rightarrow 512 \rightarrow 512$	0.015

3D bounding box regression branch: Here we explore various architectures of the sub-network in terms of two designs, i.e., fully convolution networks (FCN) and multi-layer perceptrons (MLP) in the 3DPN (Sec. 7.3.2). Table 7.4 reports the converged smooth-l_1 loss for test images obtained from different designs. We show that the FCNs achieve approximately the same smooth-l_1 loss as an MLP-based branch does. However, the number of parameters in the MLP network is 100 times more than the FCN. Considering computational complexity and model efficiency, we employ an FCN to perform the 3D bounding box regression.

7.4.4 Vehicle Verification

Vehicle verification is the problem of determining whether two gallery samples belong to the same category. It is an important and challenging task for intelligence transportation, especially when the system is working in new scenarios with unseen and misaligned categories.

To demonstrate the generality and robustness of the proposed method, we conduct experiments on the verification task of BoxCars. In this experiment, we follow the same method of [118] to perform verification, i.e., 3,000 image pairs are randomly selected to test the performance of various models in each case.

For all networks, we use the output of the second last layer (the layer preceding the last softmax classification layer) as the representation feature vector for the

given image. For each image pair, we use the cosine distance [130] to obtain the similarity of two gallery images, which is then used to compute precision, recall, and average precision.

The precision-recall (PR) curves presented in Figure 7.7 show that the proposed approach outperforms the baseline method [118] on all three dataset protocols. The performance gain of our method provides an absolute performance gain of 33% in Average Precision (AP) on *Easy*, and an even better 36% AP on the *Hard* split. It is worth noting that the size of feature vector of [118] is 4096 while ours is 2048, which indicates a better data distribution and faster speed for model inference.

7.5 Chapter Summary

In this chapter, we present a unified framework to perform vehicle recognition, which takes full advantage of both the 2D and 3D perspective representations. The proposed method achieves state-of-the-art results both in car classification and verification on the BoxCars dataset. Furthermore, we propose a vanishing point regularization for cuboid detection, which is intuitively appealing and geometrically explainable, and avoids the need for a post detection refinement processes, as used by existing methods. Last but not least, the proposed RoI Perspective is able to extract features from RoIs warped by perspective efficiently; and is able to extract features that are more effective than the state-of-the-art method [55].

Chapter 8

Conclusions and Future Works

Estimating homographies automatically and efficiently is helpful for many computer vision applications, which require the geometry information provided by homographies to improve their performance. Homography estimation is challenging in cases where a given image pair is taken by two widely placed cameras, or contains many dynamic pixels. The research presented in this thesis makes contributions in three primary research directions relating to homography estimation. The first is the further development and improvement of camera calibration methods for sports videos. The outcomes from this research direction are outlined in Section 8.1. The second is the perspective fields, which aims to provide a robust, geometrically interpretable, and low computational complexity parameterization for deep learning based homography estimation. The outcomes of this research direction are concluded in Section 8.2. The third is the exploration of the combination of deep learning and projective geometry. A vehicle recognition system has been proposed by combining both 2D appearance and 3D viewpoint aligned features. The viewpoint aligned features are generated using the proposed RoI Perspective layer, which is able to transform features in a quadrilateral into a fix-sized rectangle using the homography. Moreover, a vanishing point loss has

been proposed for accurate 3D bounding box prediction. The outcomes of this research direction are discussed in Section 8.3. Ideas for future research directions in homography estimation are discussed in Section 8.4.

8.1 Camera Calibration in Sports Scenes

Camera calibration for broadcast sports video has been a challenging task of a broad interest for many years, and which can enable advanced sports analytics by transforming player coordinates from a frame to a court template coordinate system. The standard pipeline for existing methods is to calibrate key frames, which are detected by searching for a group of specific geometric features on the playing area, and extrapolate the calibration results to the remaining frames. However, key frame detection is prone to failure when video clips do not contain a key frame. Chapters 4 and 5 present an innovative method to calibrate sports videos using key frames generated from the video itself, when no key frames are available in the video. Our proposed method consists of three steps:

1. We estimate camera intrinsic and extrinsic parameters of every frame using a novel camera parameter initialization algorithm while considering the properties of broadcast videos and cameras.

2. A panoramic view of the playing area, which can be regarded as a generated key frame, is reconstructed by optimizing all cameras parameters jointly. Moreover, the objective function used for optimization is designed to be suitable for high-quality panorama generation from broadcast video.

3. Camera calibration on every single frame is performed using the transformation between itself and the aligned panorama, which is registered to a standard court template.

Experiments have been conducted on challenging video clips that lack key frames and specific geometric features due to a highly-customized playing surface. Experimental results demonstrate that our method outperforms all previous state-of-the-art methods.

This research has been published in the following papers: **Rui Zeng**, Simon Denman, Ruan Lakemond, Sridha Sridharan, Clinton Fookes, Stuart Morgan, "Key Frames Generation for Camera Calibration in Broadcast Basketball Video," submitted to IEEE Transaction on Multimedia.

Rui Zeng, Ruan Lakemond, Simon Denman, Sridha Sridharan, Clinton Fookes, "Calibrating cameras in poor-conditioned pitch-based sports games," in *International Conference on Acoustic, Speech, and Signal Processing (ICASSP), 2018.*

Rui Zeng, Ruan Lakemond, Simon Denman, Sridha Sridharan, Clinton Fookes, Stuart Morgan, "Vertical axis detection for sport video analytics," in *Digital Image Computing: Techniques and Applications (DICTA), 2016.*

8.2 Homography Estimation using Perspective Fields

Planar homography estimation refers to the problem of computing a bijective linear mapping of pixels between two images. While this problem has been studied with convolutional neural networks (CNNs), existing methods simply regress the location of the four corners using a dense layer preceded by a fully-connected layer. This vector representation ignores the spatial structure of the corners since they have a clear spatial order. Moreover, four points are the minimum required to compute the homography, and so such an approach is susceptible to perturbation.

Chapter 6 proposes a conceptually simple, reliable, and general framework for homography estimation. In contrast to previous works, we formulate this problem as a perspective field (PF), which models the essence of the homography - pixel-to-pixel bijection. The PF is naturally learned by the proposed fully convolutional residual network, PFNet, to retain the spatial order of each pixel. Moreover, since the displacement of every pixel can be obtained from the PF, it enables robust homography estimation by utilizing dense correspondences. The experiments demonstrate that the proposed method outperforms traditional correspondence-based approaches and state-of-the-art CNN approaches in terms of accuracy while also having a smaller network size. In addition, the new parameterization of this task is general and can be implemented by any fully convolutional network (FCN) architecture.

This research has been published in the following paper: **Rui Zeng**, Simon Denman, Sridha Sridharan, Clinton Fookes, "Rethinking Planar Homography," in *Asian Conference on Computer Vision (ACCV), 2018.*

8.3 Geometry-constrained Car Recognition

Chapter 7 presents a novel learning framework for vehicle recognition from a single RGB image. Unlike existing methods using only attention mechanisms to locate 2D discriminative information, our unified framework learns a joint representation of the 2D global texture and 3D-bounding-box in a mutually correlated and reinforced manner. These two feature representations (2D and 3D) are combined by a novel fusion network, which predicts the vehicle's category.

The 2D global feature is extracted using an off-the-shelf detection network, where the estimated 2D bounding box assists in finding the region of interest (RoI). With

the assistance of the RoI, the 3D bounding box and its corresponding features are generated in a geometrically correct way using a novel 3D perspective Network (3DPN). The 3DPN consists of a convolutional neural network (CNN), a novel vanishing point loss, and novel RoI perspective layers. The 3DPN regresses the 3D bounding box under the guidance of the proposed vanishing point loss. The loss encodes projective geometry constraints implicitly during the training phase of the 3DPN, and hence that the regressed 3D bounding box conforms to the geometric vanishing point constraint: i.e., parallel lines in the real world converge at the same point in images. Thanks to the proposed RoI perspective layer, the estimated 3D bounding box is used to generate 3D perspective features from the feature map space. It is shown that the variation caused by viewpoint changes is corrected by this viewpoint-aligned feature, enhancing the feature representation.

Chapter 7 presents qualitative and quantitative results for the proposed method on the vehicle classification and verification tasks in the BoxCars dataset. The results demonstrate that, by learning how to extract features from the 3D bounding box, we can achieve comparable or superior performance to methods that only use 2D information.

This work has been published in the following paper: **Rui Zeng**, Zongyuan Ge, Simon Denman, Sridha Sridharan, Clinton Fookes, "Geometry-constrained Car Recognition Using a 3D Perspective Network," submitted to *International Conference on Computer Vision (ICCV), 2019.* https://arxiv.org/abs/1903.07916

8.4 Future Works

The key components used to calibrate cameras in environments like sports scenes, such as correspondence detection and homography computation, are reaching

maturity. It is expected that future research into homography estimation in sports scenes will only yield incremental improvements over existing methods. This thesis recognizes this limitation and has begun to explore techniques that go beyond traditional homography estimation methods. The presented camera calibration system decomposes obtained homographies into the multiplication of camera intrinsic and extrinsic matrices while considering the properties of sports videos. Therefore, the proposed method achieves a higher performance compared to existing methods which do not consider underlying geometry constraints arising from PTZ cameras used in sporting videos recording.

Possibilities for future research exist within the key modules of the proposed camera calibration system, including panorama generation and camera parameter optimization. The panorama generation of a whole playing area could include more advanced image stitching algorithms such as ghosting artefact elimination, debluring, etc. The camera optimization process could be improved by using the geometric constraint that all correspondences falling on the playing area are coplanar in the real world coordinate system. By taking this into account, errors arising from correspondence misalignment could be further reduced.

The proposed PFNet approach can also be further extended to sports or other environments. While it has been shown to be very accurate on both synthesized datasets and photos taken in both indoor and outdoor scenes, at present it has limitations which prevent its application to a video clip. The formulation of PFNet presented in this thesis is only able to estimate homographies between each frame pair in isolation. PFNet does not take the global context into account when dealing with a given video clip. Therefore, PFNet cannot correct poorly estimated homographies using pre-estimated homographies, and cannot incorporate prediction of camera motion to improve performance. Moreover, PFNet has a high computational complexity due to the fact that each frame is used twice when

predicting homographies within one video clip. For instance, when estimating homographies in a video clip, which has n frames, we need to generate n fame pairs. A feasible work around for addressing these two issues simultaneously may be the use of a long short-term memory network. It could predict the homography of the current frame using the implicit camera position knowledge obtained from previous frames. In this way, even if a single frame is very difficult to calibrate, its homography can still be computed according to the camera motion momentum stored in the memory units of LSTM. Another possible future research direction is to fuse the PFNet with object detection networks. Since the performance of the PFNet on sports videos is significantly degraded by moving players, the fusion of the PFNet and object detection networks may be able to provide a disentangled video frame representation, which could distill the homography representation by filtering out detected players.

As noted in Chapter 7, the use of 3D bounding box for vehicle recognition aims to generate viewpoint aligned features, which helps to eliminate viewpoint variations. However, manipulating projective geometry in the feature space directly still does not enable networks to fully comprehend the geometric structure of objects from 2D images. Therefore, investigating a new feature representation, which could disentangle the 3D model, appearance, and relative distance, is a key to improve vehicle (and more broadly object) recognition performance.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning.," in *OSDI*, vol. 16, pp. 265–283, 2016.

[2] A. E. Abdel-Hakim and A. A. Farag, "Csift: A sift descriptor with color invariant characteristics," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 1978–1983, IEEE, 2006.

[3] R. Achanta, F. Estrada, P. Wils, and S. Süsstrunk, "Salient region detection and segmentation," *Computer Vision Systems*, pp. 66–75, 2008.

[4] M. Ahmed and A. Farag, "Nonmetric calibration of camera lens distortion: differential methods and robust estimation," *IEEE Transactions on Image Processing*, vol. 14, no. 8, pp. 1215–1230, 2005.

[5] F. Angehrn, O. Wang, Y. Aksoy, M. Gross, and A. Smolić, "Mastercam fvv: Robust registration of multiview sports video to a static high-resolution master camera for free viewpoint video," in *IEEE International Conference on Image Processing*, pp. 3474–3478, Oct 2014.

[6] A. Antoniou, *Digital signal processing*. McGraw-Hill, 2016.

[7] I. Atmosukarto, B. Ghanem, S. Ahuja, K. Muthuswamy, and N. Ahuja, "Automatic recognition of offensive team formation in american football plays,"

in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 991–998, 2013.

[8] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," *arXiv preprint arXiv:1412.7755*, 2014.

[9] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European Conference on Computer Vision*, 2006.

[10] S. Berretti, A. Del Bimbo, P. Pala, B. B. Amor, and M. Daoudi, "A set of selected sift features for 3d facial expression recognition," in *International Conference on Pattern Recognition*, pp. 4125–4128, IEEE, 2010.

[11] G. Boutry, M. Elad, G. H. Golub, and P. Milanfar, "The generalized eigenvalue problem for nonsquare pencils using a minimal perturbation approach," *Journal on Matrix Analysis and Applications*, vol. 27, no. 2, pp. 582–601, 2005.

[12] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[13] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, 2007.

[14] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

[15] K. Calagari, M. Elgharib, S. Shirmohammadi, and M. Hefeeda, "Sports vr content generation from regular camera feeds," in *ACM Conference on Multimedia*, pp. 699–707, ACM, 2017.

[16] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[17] J. Chamorro-Martínez, P. Martínez-Jiménez, and J. M. Soto-Hidalgo, "A fuzzy approach for retrieving images in databases using dominant color and texture descriptors," in *FUZZ-IEEE*, 2010.

[18] J. Chen, T. N. Pappas, A. Mojsilovic, and B. E. Rogowitz, "Adaptive perceptual color-texture image segmentation," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1524–1536, 2005.

[19] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu, "Global contrast based salient region detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 569–582, 2015.

[20] F. Chollet *et al.*, "Keras," 2015.

[21] O. Chum and J. Matas, "Planar affine rectification from change of scale," in *Asian Conference on Computer Vision*, pp. 347–360, Springer, 2010.

[22] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *International Joint Conference on Artificial Intelligence*, 2011.

[23] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *arXiv preprint arXiv:1202.2745*, 2012.

[24] T. S. Cohen and M. Welling, "Transformation properties of learned visual representations," *arXiv preprint arXiv:1412.7659*, 2014.

[25] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3150–3158, 2016.

[26] B. Dang, A. Tran, T. Dinh, and T. Dinh, "A real time player tracking system for broadcast tennis video," in *Asian Conference on Intelligent Information and Database Systems*, pp. 105–113, Springer, 2010.

[27] S. Denman, T. Lamb, C. Fookes, V. Chandran, and S. Sridharan, "Multi-spectral fusion for surveillance systems," *Computers & Electrical Engineering*, vol. 36, no. 4, pp. 643–663, 2010.

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

[29] L.-Y. Duan, M. Xu, Q. Tian, C.-S. Xu, and J. S. Jin, "A unified framework for semantic shot classification in sports video," *IEEE Transactions on Multimedia*, vol. 7, pp. 1066–1083, Dec 2005.

[30] D. Dwibedi, T. Malisiewicz, V. Badrinarayanan, and A. Rabinovich, "Deep cuboid detection: Beyond 2d bounding boxes," *arXiv preprint arXiv:1611.10010*, 2016.

[31] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2154, 2014.

[32] F. Erlik Nowruzi, R. Laganiere, and N. Japkowicz, "Homography estimation from image pairs with hierarchical convolutional networks," in *IEEE International Conference on Computer Vision*, pp. 913–920, 2017.

[33] D. Farin, J. Han, and P. H. N. de With, "Fast camera calibration for the analysis of sport sequences," in *IEEE International Conference on Multimedia and Expo*, pp. 4 pp.–, July 2005.

[34] D. Farin, S. Krabbe, W. Effelsberg, *et al.*, "Robust camera calibration for sport videos using court models," in *Storage and Retrieval Methods and Applications for Multimedia*, vol. 5307, pp. 80–92, International Society for Optics and Photonics, 2003.

[35] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, pp. 167–181, Sep 2004.

[36] H. Fu, X. Cao, and Z. Tu, "Cluster-based co-saliency detection," *IEEE Transactions on Image Processing*, vol. 22, pp. 3766–3778, Oct 2013.

[37] J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4438–4446, 2017.

[38] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, "Multimodal compact bilinear pooling for visual question answering and visual grounding," *arXiv preprint arXiv:1606.01847*, 2016.

[39] C. Geng and X. Jiang, "Face recognition using sift features," in *International Conference on Image Processing*, pp. 3313–3316, IEEE, 2009.

[40] R. Gens and P. M. Domingos, "Deep symmetry networks," in *Advances in Neural Information Processing Systems*, pp. 2537–2545, 2014.

[41] B. Ghanem, T. Zhang, and N. Ahuja, "Robust video registration applied to field-sports video analysis," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2012.

[42] A. Gil, O. Reinoso, O. M. Mozos, C. Stachniss, and W. Burgard, "Improving data association in vision-based slam," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2076–2081, IEEE, 2006.

[43] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[44] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

[45] S. Goferman, L. Zelnik-Manor, and A. Tal, "Context-aware saliency detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 1915–1926, Oct 2012.

[46] M. Gong, S. Zhao, L. Jiao, D. Tian, and S. Wang, "A novel coarse-to-fine scheme for automatic image registration based on sift and mutual information," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, pp. 4328–4338, July 2014.

[47] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Pearson Press, 2007.

[48] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[49] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert, "From 3d scene geometry to human workspace," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1961–1968, IEEE, 2011.

[50] H. Ha, M. Perdoch, H. Alismail, I. S. Kweon, and Y. Sheikh, "Deltille grids for geometric camera calibration," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5344–5352, 2017.

[51] J. Han, D. Farin, and P. H. de With, "Broadcast court-net sports video analysis using fast 3-d camera modeling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1628–1638, 2008.

[52] C. G. Harris, M. Stephens, *et al.*, "A combined corner and edge detector.," in *Alvey vision conference*, vol. 15, pp. 10–5244, Citeseer, 1988.

[53] R. Hartley and S. B. Kang, "Parameter-free radial distortion correction with center of distortion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 8, pp. 1309–1321, 2007.

[54] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[55] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *IEEE International Conference on Computer Vision*, pp. 2980–2988, IEEE, 2017.

[56] H. He, Z. Shao, and J. Tan, "Recognition of car makes and models from a single traffic-camera image," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3182–3192, 2015.

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

[58] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering free space of indoor scenes from a single image," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2807–2814, IEEE, 2012.

[59] R. Hess and A. Fern, "Improved video registration using non-distinctive local image features," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[60] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2007.

[61] C. C. Hsu, H. T. Chen, C. L. Chou, and S. Y. Lee, "Spiking and blocking events detection and analysis in volleyball videos," in *IEEE International Conference on Multimedia and Expo*, pp. 19–24, July 2012.

[62] M.-C. Hu, M.-H. Chang, J.-L. Wu, and L. Chi, "Robust camera calibration and player tracking in broadcast basketball video," *IEEE Transactions on Multimedia*, vol. 13, no. 2, pp. 266–279, 2011.

[63] M. C. Hu, M. H. Chang, J. L. Wu, and L. Chi, "Robust camera calibration and player tracking in broadcast basketball video," *IEEE Transactions on Multimedia*, vol. 13, pp. 266–279, April 2011.

[64] X. Hu, Y. Tang, and Z. Zhang, "Video object matching based on sift algorithm," in *International Conference on Neural Networks and Signal Processing*, pp. 412–415, IEEE, 2008.

[65] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks.," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, p. 3, 2017.

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

[67] N. Japkowicz, F. E. Nowruzi, and R. Laganiere, "Homography estimation from image pairs with hierarchical convolutional networks," in *IEEE International Conference on Computer Vision Workshops*, pp. 904–911, Oct 2017.

[68] L. Juan and G. Oubong, "Surf applied in panorama image stitching," in *International Conference on Image Processing Theory, Tools and Applications*, pp. 495–499, IEEE, 2010.

[69] A. Kanazawa, A. Sharma, and D. Jacobs, "Locally scale-invariant convolutional neural networks," *arXiv preprint arXiv:1412.5104*, 2014.

[70] Y. Ke and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in *null*, pp. 506–513, IEEE, 2004.

[71] P. Kovesi, "Video surveillance: Legally blind?," in *Digital Image Computing: Techniques and Applications*, pp. 204–211, IEEE, 2009.

[72] J. Krause, T. Gebru, J. Deng, L. Li, and L. Fei-Fei, "Learning features and parts for fine-grained recognition," in *International Conference on Pattern Recognition*, pp. 26–33, Aug 2014.

[73] J. Krause, H. Jin, J. Yang, and L. Fei-Fei, "Fine-grained recognition without part annotations," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5546–5555, 2015.

[74] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013.

[75] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[76] I. Kviatkovsky, A. Adam, and E. Rivlin, "Color invariants for person reidentification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1622–1634, 2013.

[77] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *IEEE International Conference on 3D Vision*, pp. 239–248, IEEE, 2016.

[78] R. Lakemond, *Multiple camera management using wide baseline matching*. PhD thesis, Queensland University of Technology, 2010.

[79] R. Lakemond, C. Fookes, and S. Sridharan, "Practical improvements to simultaneous computation of multi-view geometry and radial lens distortion," in *The International Conference on Digital Image Computing: Techniques and Applications*, 2011.

[80] D. C. Lee, M. Hebert, and T. Kanade, "Geometric reasoning for single image structure recovery," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2136–2143, IEEE, 2009.

[81] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 991–999, 2015.

[82] Y. Li, Y. Wang, W. Huang, and Z. Zhang, "Automatic image stitching using sift," in *International Conference on Audio, Language and Image Processing*, pp. 568–571, IEEE, 2008.

[83] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection.," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, p. 4, 2017.

[84] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

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

[86] Y.-L. Lin, V. I. Morariu, W. Hsu, and L. S. Davis, "Jointly optimizing 3d model fitting and fine-grained classification," in *European Conference on Computer Vision*, pp. 466–480, Springer, 2014.

[87] A. Linnemann, S. Gerke, S. Kriener, and P. Ndjiki-Nya, "Temporally consistent soccer field registration," in *IEEE International Conference on Image Processing*, pp. 1316–1320, Sept 2013.

[88] S. Liu, J. Chen, C. H. Chang, and Y. Ai, "A new accurate and fast homography computation algorithm for sports and traffic video analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2017.

[89] H. Liu, S. Jiang, Q. Huang, and C. Xu, "A generic virtual content insertion system based on visual attention analysis," in *ACM International Conference on Multimedia*, pp. 379–388, ACM, 2008.

[90] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman, "Sift flow: Dense correspondence across different scenes," in *European Conference on Computer Vision*, pp. 28–42, Springer, 2008.

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

[92] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157, IEEE, 1999.

[93] W.-L. Lu, J.-A. Ting, J. J. Little, and K. P. Murphy, "Learning to track and identify players from broadcast sports videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1704–1716, 2013.

[94] J. Ma, H. Zhou, J. Zhao, Y. Gao, J. Jiang, and J. Tian, "Robust feature matching for remote sensing image registration via locally linear transforming," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, pp. 6469–6481, Dec 2015.

[95] M. Manafifard, H. Ebadi, and H. A. Moghaddam, "Multi-player detection in soccer broadcast videos using a blob-guided particle swarm optimization method," *Multimedia Tools and Applications*, vol. 76, no. 10, pp. 12251–12280, 2017.

[96] M. Meingast, S. Oh, and S. Sastry, "Automatic camera network localization using object image tracks," in *International Conference on Computer Vision*, 2007.

[97] C. Micheloni, B. Rinner, and G. L. Foresti, "Video analysis in pan-tilt-zoom camera networks," *IEEE Signal Processing Magazine*, vol. 27, pp. 78–90, Sept 2010.

[98] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[99] P. Monasse, J.-M. Morel, and Z. Tang, "Three-step image rectification," in *British Machine Vision Conference*, pp. 89–1, BMVA Press, 2010.

[100] T. Nguyen, S. W. Chen, S. S. Shivakumar, C. J. Taylor, and V. Kumar, "Unsupervised deep homography: A fast and robust homography estimation

model," *IEEE Robotics and Automation Letters*, vol. 3, pp. 2346–2353, July 2018.

[101] K. Okuma, J. J. Little, and D. G. Lowe, "Automatic rectification of long image sequences," in *Asian Conference on Computer Vision*, 2004.

[102] Y. Peng, X. He, and J. Zhao, "Object-part attention model for fine-grained image classification," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1487–1500, 2018.

[103] F. Perazzi, P. Krähenbühl, Y. Pritch, and A. Hornung, "Saliency filters: Contrast based filtering for salient region detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 733–740, June 2012.

[104] V. Pradeep, G. Medioni, and J. Weiland, "Visual loop closing using multi-resolution sift grids in metric-topological slam," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1438–1445, IEEE, 2009.

[105] J. Puwein, R. Ziegler, L. Ballan, and M. Pollefeys, "Ptz camera network calibration from moving people in sports broadcasts," in *IEEE Winter Conference on Applications of Computer Vision*, 2012.

[106] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, pp. 91–99, 2015.

[107] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[108] H. Sabirin, Q. Yao, K. Nonaka, H. Sankoh, and S. Naito, "Toward real-time delivery of immersive sports content," *IEEE MultiMedia*, vol. 25, pp. 61–70, Apr 2018.

[109] E. Salahat and M. Qasaimeh, "Recent advances in features extraction and description algorithms: A comprehensive survey," in *IEEE International Conference on Industrial Technology*, pp. 1059–1063, IEEE, 2017.

[110] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[111] D. Schleicher, L. M. Bergasa, R. Barea, E. López, M. Ocaña, and J. Nuevo, "Real-time wide-angle stereo visual slam on large environments using sift features correction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3878–3883, IEEE, 2007.

[112] J. Schmidhuber and R. Huber, "Learning to generate artificial fovea trajectories for target detection," *International Journal of Neural Systems*, vol. 2, no. 01n02, pp. 125–134, 1991.

[113] S. Se and P. Jasiobedzki, "Photo-realistic 3d model reconstruction," in *IEEE International Conference on Robotics and Automation*, pp. 3076–3082, IEEE, 2006.

[114] L. Sha, P. Lucey, S. Sridharan, S. Morgan, and D. Pease, "Understanding and analyzing a large collection of archived swimming videos," in *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[115] H.-Y. Shum and R. Szeliski, "Construction of panoramic image mosaics with global and local alignment," in *Panoramic Vision*, pp. 227–268, Springer, 2001.

[116] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[117] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "Feature tracking and matching in video using programmable graphics hardware," *Machine Vision and Applications*, vol. 22, no. 1, pp. 207–217, 2011.

[118] J. Sochor, A. Herout, and J. Havel, "Boxcars: 3d boxes as cnn input for improved fine-grained vehicle recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3006–3015, June 2016.

[119] K. Sohn and H. Lee, "Learning invariant representations with local transformations," *arXiv preprint arXiv:1206.6418*, 2012.

[120] A. N. Staranowicz, G. R. Brown, F. Morbidi, and G.-L. Mariottini, "Practical and accurate calibration of rgb-d cameras using spheres," *Computer Vision and Image Understanding*, vol. 137, pp. 102 – 114, 2015.

[121] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 246–252, IEEE, 1999.

[122] R. Steele and C. Jaynes, "Overconstrained linear estimation of radial distortion and multi-view geometry," in *European Conference on Computer Vision*, 2006.

[123] G. P. Stein, "Tracking from multiple view points: Self-calibration of space and time," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[124] A. Suga, K. Fukuda, T. Takiguchi, and Y. Ariki, "Object recognition and segmentation using sift and graph cuts," in *International Conference on Pattern Recognition*, pp. 1–4, IEEE, 2008.

[125] M. Suhling, M. Arigovindan, P. Hunziker, and M. Unser, "Multiresolution moment filters: Theory and applications," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 484–495, 2004.

[126] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on Machine Learning*, pp. 1139–1147, 2013.

[127] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.," in *AAAI Conference on Artificial Intelligence*, vol. 4, p. 12, 2017.

[128] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[129] R. Szeliski, "Image alignment and stitching: A tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 2, no. 1, pp. 1–104, 2006.

[130] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, 2014.

[131] E. Tola, V. Lepetit, and P. Fua, "A fast local descriptor for dense matching," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[132] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 815–830, 2010.

[133] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[134] S. Tulsiani and J. Malik, "Viewpoints and keypoints," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1510–1519, 2015.

[135] T. Tuytelaars, K. Mikolajczyk, *et al.*, "Local invariant feature detectors: a survey," *Foundations and trends® in computer graphics and vision*, vol. 3, no. 3, pp. 177–280, 2008.

[136] K. Van De Sande, T. Gevers, and C. Snoek, "Evaluating color descriptors for object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1582–1596, 2010.

[137] Y. Wang, V. I. Morariu, and L. S. Davis, "Learning a discriminative filter bank within a cnn for fine-grained recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4148–4157, 2018.

[138] P. Wang, D. Zhang, J. Wang, Z. Wu, X.-S. Hua, and S. Li, "Color filter for image search," in *ACM Multimedia*, 2012.

[139] Y. Wei, F. Wen, W. Zhu, and J. Sun, "Geodesic saliency using background priors," in *European Conference on Computer Vision*, pp. 29–42, Springer, 2012.

[140] P. C. Wen, W. C. Cheng, Y. S. Wang, H. K. Chu, N. C. Tang, and H. Y. M. Liao, "Court reconstruction for camera calibration in broadcast basketball videos," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, pp. 1517–1526, May 2016.

[141] M. Wilczkowiak, P. Sturm, and E. Boyer, "Using geometric constraints through parallelepipeds for calibration and 3d modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 2, pp. 194–207, 2005.

[142] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang, "The application of two-level attention models in deep convolutional neural network for fine-grained image classification," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 842–850, 2015.

[143] J. Xing, H. Ai, L. Liu, and S. Lao, "Multiple player tracking in sports video: A dual-mode two-way bayesian inference approach with progressive observation modeling," *IEEE Transactions on Image Processing*, vol. 20, pp. 1652–1667, June 2011.

[144] J. Xu, S. Denman, V. Reddy, C. Fookes, and S. Sridharan, "Real-time video event detection in crowded scenes using mpeg derived features: A multiple instance learning approach," *Pattern Recognition Letters*, vol. 44, pp. 113–125, 2014.

[145] J. Xu, S. Denman, S. Sridharan, and C. Fookes, "Activity modelling in crowded environments: A soft-decision approach," in *International Conference on Digital Image Computing: Techniques and Applications*, pp. 107–112, IEEE, 2011.

[146] X. Xu and H. Man, "Interpreting sports tactic based on latent context-free grammar," in *IEEE International Conference on Image Processing*, pp. 4072–4076, Sept 2015.

[147] J. Xu, L. Xiang, R. Hang, and J. Wu, "Stacked sparse autoencoder (ssae) based framework for nuclei patch classification on breast cancer histopathology," in *IEEE International Symposium on Biomedical Imaging*, pp. 999–1002, IEEE, 2014.

[148] Q. Yan, L. Xu, J. Shi, and J. Jia, "Hierarchical saliency detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1155–1162, June 2013.

[149] Z. Yang, T. Luo, D. Wang, Z. Hu, J. Gao, and L. Wang, "Learning to navigate for fine-grained classification," in *European Conference on Computer Vision*, pp. 420–435, 2018.

[150] Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2878–2890, 2013.

[151] G. Yu and J. Yuan, "Fast action proposals for human action detection and search," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1302–1311, 2015.

[152] R. Zeng, R. Lakemond, S. Denman, S. Sridharan, C. Fookes, and S. Morgan, "Calibrating cameras in poor-conditioned pitch-based sports games," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1902–1906, IEEE, 2018.

[153] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[154] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[155] H. Zhang and V. M. Patel, "Densely connected pyramid dehazing network," *arXiv preprint arXiv:1803.08396*, 2018.

[156] H. Zheng, J. Fu, T. Mei, and J. Luo, "Learning multi-attention convolutional neural network for fine-grained image recognition," in *International Conference on Computer Vision*, vol. 6, 2017.

[157] H. Zhou, Y. Yuan, and C. Shi, "Object tracking using sift features and mean shift," *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 345–352, 2009.

[158] Y. Zhu and S. Newsam, "Densenet for dense flow," *arXiv preprint arXiv:1707.06316*, 2017.

[159] G. Zhu, C. Xu, Q. Huang, Y. Rui, S. Jiang, W. Gao, and H. Yao, "Event tactic analysis based on broadcast sports video," *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp. 49–67, 2009.

[160] X. Zhuang, X. Cheng, S. Huang, M. Honda, and T. Ikenaga, "Motion vector and players' features based particle filter for volleyball players tracking in 3d space," in *Pacific Rim Conference on Multimedia*, pp. 393–401, Springer, 2015.

[161] B. Zitova and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[162] B. Zitová and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977 – 1000, 2003.

Appendix A

Algorithm Derivations

This appendix lists the fast implementation of frame saliency generation algorithm, which is presented in Chapter 5, Section 5.4.

A.1 Fast Implementation for Frame Saliency

Equation 5.37 can be further expanded as follows:

$$L_i = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{K}(c_j, c_i, \sigma)\, \|c_j - c_i\|_2^2 \tag{A.1}$$

$$= \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{K}(c_j, c_i, \sigma)\, \|c_j\|_2^2 + \frac{1}{N} c_i^2 \sum_{j=1}^{N} \boldsymbol{K}(c_j, c_i, \sigma) \tag{A.2}$$

$$- \frac{1}{N}\left\| 2c_i \circ \sum_{j=1}^{N} \boldsymbol{K}(c_j, c_i, \sigma) c_j \right\|_1, \tag{A.3}$$

where $\circ$ is the Hadamard product. From Equation A.3, we can find that all the three terms equate to performing 2D convolution operations on the ith pixel. Thus we can extend the local contrast cue of the ith pixel to the saliency map of

the whole frame $\mathcal{F}$ by applying a 2D convolution as follows:

$$\begin{aligned} L =& \frac{1}{N}\boldsymbol{K}(x,y,\sigma) * \mathcal{F}^2(x,y) \\ &+ \frac{1}{N}\mathcal{F}^2(x,y) \circ \boldsymbol{K}(x,y,\sigma) * \mathbf{1}(x,y) \\ &- \frac{2}{N}\mathcal{F}(x,y) \circ \boldsymbol{K}(x,y,\sigma) * \mathcal{F}(x,y), \end{aligned} \tag{A.4}$$

where $\mathcal{F}^2(x,y)$is the squared sum of all color channels of the frame $\mathcal{F}$, $*$ is the 2D convolution operator. $\mathbf{1}$ is a frame which has the same size as $\mathcal{F}$ and all pixels within are set to 1.

Similarly, Equation 5.38 can be expanded as follows:

$$\begin{aligned} G_i =& \frac{1}{N}\|c_i - c_d\|_2^2 \sum_{j=1}^{N} \|c_j - c_d\|_2^2 \\ =& \frac{1}{N}\|c_i - c_d\|_2^2 \left[\sum_{j=1}^{N} \|c_j\|_2^2 + \sum_{j=1}^{N} \|c_d\|_2^2 - 2 \left\| c_d \circ \sum_{j=1}^{N} c_j \right\|_1 \right]. \end{aligned} \tag{A.5}$$

Again, we extend the pixel-wise global rarity cue to the saliency map of the whole frame by evaluating the following equation:

$$\begin{aligned} G = & \frac{1}{N}(\mathcal{F}(x,y) - \mathcal{F}_{c_d}(x,y)) \circ \boldsymbol{N}(x,y) * \mathcal{F}^2(x,y) + \\ & \frac{1}{N}(\mathcal{F}(x,y) - \mathcal{F}_{c_d}(x,y)) \circ \mathcal{F}_{c_d}^2(x,y) \circ \boldsymbol{N}(x,y) * \mathbf{1}(x,y) \\ & -\frac{1}{N}(\mathcal{F}(x,y) - \mathcal{F}_{c_d}(x,y)) \circ 2\mathcal{F}(x,y) \circ \boldsymbol{N}(x,y) * \mathcal{F}(x,y), \end{aligned} \tag{A.6}$$

where $\boldsymbol{N}(x,y)$ is the mean filter.