Towards Automatic Broadcast of Team Sports

by

Jianhui Chen

B.Eng. Zhejiang University, 2004

M.Eng. Zhejiang University, 2011

MASc. York University, 2013

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Computer Science)

The University of British Columbia

(Vancouver)

August 2018

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Towards Automatic Broadcast of Team Sports

submitted by **Jianhui Chen** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Computer Science**.

Examining Committee:

Jame J. Little, Computer Science
Supervisor

Mark Schmidt, Computer Science
Supervisory Committee Member

Peter Carr, Argo AI
Supervisory Committee Member

Lenoid Sigal, Computer Science
University Examiner

Sidney Fels, Electrical and Computer Engineering
University Examiner

Michael S. Brown, York University
External Examiner

Abstract

Sports is the social glue of society as it allows people to interact with each other and appreciate games irrespective of their social status, age and ethnicity. Automatic sports broadcasting produces stream videos from vision sensors without human intervention. The goal is to predict where cameras should look and which camera should be "on air". The technique can benefit millions of people as most viewers participant in sports by watching TV or Internet broadcasting. The target team sports include basketball, soccer and ice hockey in which team members quickly move their positions in the game, excluding sports like baseball and cricket in which team members have relatively stable positions.

Automatic sports broadcasting covers areas of statistics, commentary, camera control and so on. We provide solutions for automatically setting camera parameters such as camera orientation angles and locations using computer vision. We restrict our attention to static pan-tilt-zoom (PTZ) cameras for television or live Internet broadcasting.

We propose three essential components of autonomous broadcasting: camera calibration, planning and selection. By learning from human demonstrations, our work can predict camera angles for single camera systems and camera viewpoints for multi-camera systems. We obtain human demonstrations from existing videos that are generated by professional camera operators. These videos contain camera angles and camera IDs if there are multiple cameras.

Because camera angles are not directly available, we first propose two novel camera calibration methods. We evaluate and compare our methods with previous algorithms. Our methods are more accurate and faster than previous algorithms.

With labeled data from human operators, we develop two methods for smooth camera planning which predict camera pan angles. The first method directly incorporates temporal consistency into a data-driven predictor. The second method optimizes the camera trajectory in overlapped temporal windows. We show they outperform previous methods in the literature.

We also propose two methods for selecting a broadcast camera view from multiple candidate camera views. The first method uses deep features for camera se-

lection. The second method augments the training data with Internet videos. We demonstrate comparable results with selections from human operators in soccer games.

Lay Summary

Billions of people enjoy team sports such as basketball and soccer through television or Internet broadcasting, which are usually controlled by human operators. This thesis designs an automatic system for team sports broadcasting. In the system, robotic cameras work as if they are controlled by human operators. Mimicking human operators is challenging because of the complex decision-making process and the limited training data. To overcome these challenges, we first develop computer vision techniques to collect labeled data from raw videos effectively. We also develop machine learning methods to build robust systems from a small number of training data. Using these novel techniques, we demonstrate state-of-the-art performance on three core components for automatic broadcasting: camera calibration, camera planning and camera selection.

Preface

This thesis presents original and independent research conducted under the guidance of my supervisor, Professor James J. Little. Parts of this thesis have been published as follows.

Chapter 2 describes two camera calibration methods that use camera location priors. The two-point method was published at IEEE Winter Conference on Applications of Computer Vision (WACV), 2018 [CZL18], co-authored by Fangrui Zhu and James J. Little. For this WACV paper, Fangrui Zhu helped with implementation and ran the experiments. I contributed to identifying the problem, designing the algorithm and experiments. The synthetic edge image method is based on yet-unpublished work. I conceived the initial idea, developed the algorithm and executed the experiments.

Chapter 3 describes two camera planning methods that learn from human demonstration. This work was based on three publications. A preliminary version was published at IEEE Winter Conference on Applications of Computer Vision (WACV) 2015 [CC15] co-authored by Peter Carr. In this WACV paper, Peter Carr contributed the initial idea and I contributed the implementation.

This preliminary work was later developed into two advanced methods. The first one was presented orally at IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [CLC+16], co-authored by Hoang M. Le, Peter Carr, Yisong Yue and James J. Little. In this CVPR paper, Hoang M. Le, Peter Carr and Yisong Yue contributed to problem formulation, algorithm design and experiments. I contributed to identifying the problem, algorithm design and most of the experiments. The iterative training procedure in the paper was developed by Hoang M. Le, thus is not included in the thesis. The second one was published in the journal Computer Vision and Image Understanding (CVIU) [CL17]. I conceived the initial idea, developed the algorithm, and designed and executed the experiments.

Chapter 4 describes two camera selection methods. The regularized visual importance ranking method was published at IEEE Winter Conference on Applications of Computer Vision (WACV), 2018 [CML18], co-authored by Lili Meng and James J. Little. Lili Meng contributed the implementation and the user study. I con-

tributed to problem formulation, algorithm design and experiments. The method of using Internet videos is based on yet-unpublished work with Keyu Lu, Sijia Tian and James J. Little. Keyu Lu contributed to player/ball detection. Sijia Tian contributed to experiments and writing. I contributed to problem formulation, algorithm design and experiments.

Dr. Peter Carr contributed to general guidance and writing in [CC15, CLC+16, CL17, CML18].

Professor James J. Little contributed to the planning and writing of most of the above publications and provided valuable overall guidance.

Table of Contents

List of Tables

List of Figures

Glossary

The table below provides a quick reference of the symbols used in this thesis.

Camera calibration symbols	
Symbol	Meaning
$\mathtt{P}$	Camera projection matrix
$\mathtt{K}$	Camera intrinsic matrix
$\mathtt{R}$	Camera rotation matrix
$\mathtt{C}$	Camera projection center, position in world coordinate
$\mathtt{I}$	Identity matrix
$\mathtt{H}$	Homography matrix
$\mathbf{x}$	2D image point
$\mathbf{X}$	3D world point
θ	pan angle of PTZ cameras
ϕ	tilt angle of PTZ cameras
f	focal length
Learning method symbols	
Symbol	Meaning
$\mathbf{x}$	Feature
y	Label
θ	Model parameter
Θ	Allowable parameter configurations
I	Information gain
S	A set of data
π	Policy
Π	All allowable policies

We define important terminologies to avoid confusion.

- Autonomous camera system (ACS): An autonomous camera system is a system that can detect and track objects in the scene by adjusting its location and direction.

- PTZ camera: A pan, tilt and zoom camera (PTZ camera) is a camera that is capable of remote directional and zoom control. Its location is fixed when it is used in sports broadcasting.
- Main PTZ camera: A main PTZ camera is a PTZ camera that is mostly used in broadcasting. Its location is often above and near the middle area of the playing ground.
- Broadcast video: A broadcast video is a video that is generated by human operators and delivered to subscribers and viewers. The video is captured from one or multiple PTZ cameras. When multiple cameras are used, human operators switch between cameras to create a composite broadcast video.
- CDF: Cumulative distribution function.
- CGAN: Conditional generative adversarial network.
- CRF: Conditional regression forest.
- DAgger: Dataset aggregation. It is an iterative algorithm that trains a deterministic policy that achieves good performance guarantees under its induced distribution of states.
- DLT: Direct linear transform is a homography estimation method that uses four point-to-point correspondences.
- FOV: Field of view.
- GAN: Generative adversarial network.
- HMM: Hidden Markov model.
- IoU: Intersection over Union.
- LSTM: Long short-term memory.
- KF: Kalman Filter.
- OHMM: Overlapped Hidden Markov model.
- PTZ: Pan, tilt and zoom.
- RANSAC: Random sample consensus.
- RMSE: Root mean square error.
- RNN: Recurrent neural network.
- RSF: Survival random forest.

- SA heatmap: Spatial-appearance heatmap.
- SG: Savitzky-Golay filter.
- SLAM: Simultaneous localization and mapping.

Acknowledgments

First of all, I would like to thank my three supervisory committee members, Jim Little (supervisor), Mark Schmidt and Peter Carr. Jim has always been supportive and encouraging, giving me excellent scholarly guidance and much freedom in research. Mark encouraged me to think about long-term impact problems. Peter guided me to the topic of my thesis and provided countless valuable feedback. They have also helped me greatly with the writing of this thesis.

I also would like to thank Prof. Michael Brown, the external examiner of the thesis, for his detailed and valuable report that helped me improve the final writing. I also thank Prof. Leonid Sigal and Prof. Sidney Fels for being my university examiners and carefully going through the writing. They gave feedback during the exam so that I had an opportunity to re-think my research from very different and important aspects.

I have been fortunate to work with outstanding colleagues in the laboratory of computational intelligence (LCI). Keyu Lu, Lili Meng, Fangrui Zhu and Sijia Tian directly contributed to this thesis. I also enjoyed working with Fred Tung, Lili Meng and Moumita Roy on various projects. Many thanks to Ankur Gupta, Alireza Shafaei, Julieta Martinez and Rayat Hossain for many interesting chats about our respective work.

I was lucky to spend four years with many friends in UBC. These include Victor Gan, Hao Ma, Xinxin Zhang, Lei Xiao, Yifan (Evan) Peng, Shuochen Su, Yufeng Zhu, Rui Ge, Shu Yang, Minchen Li, Wenqiang Dong, Zheng Dong, Zipeng Liu, Xing Zeng, Bo Zhao and many others. I would like to thank them for the fun we shared.

My research was partially funded by Natural Sciences and Engineering Research Council of Canada and a grant from Disney Research. I would like to acknowledge that my work was performed on the traditional, ancestral, and unceded territory of the Musqueam people.

Finally, I would like to thank my parents and my sister for their love and patience. I also would like to thank my family and many friends in China for the peaceful and enjoyable environment I can always find when I return home.

Chapter 1

Introduction

1.1 Motivation and task

An autonomous camera system (ACS) captures videos of particular scenes and sends these videos to viewers for different purposes such as surveillance and entertainment. For effective communication, the system requires high-level understanding from the raw videos. For example, an ACS perceives the environment by detecting objects, events and so on. Moreover, an ACS adjusts the camera direction and coordinates multiple cameras to accomplish the task. For example, a surveillance system continually monitors an intruder in a public place by scheduling multiple pan-tilt-zoom (PTZ) cameras.

Automatic cinematography is a core subarea of autonomous camera systems. It generates videos for stage performances, social events, sports and so on. It is an interdisciplinary research area that combines computer vision, machine learning and control. For example, an automatic video generation system like [GRG14] first tracks important actors and objects using computer vision techniques for stage performances. Then the system controls a virtual PTZ camera to generate sub-clips. Unlike other systems, automatic cinematography systems encode cinematographic practices in the system, resulting in aesthetically pleasing videos.

As a specific domain of automatic cinematography, automatic sports broadcasting can benefit millions of people. The sports market in North America was worth 69.3 billion dollars in 2017 and is expected to reach 78.5 billion dollars by 2021. As the biggest reason for market growth, media rights (game broadcasts and other sports media content) are projected to increase from 19.1 billion dollars in 2017 to 22.7 billion dollars in 2021 [Pri17]. Computational broadcasting is a promising way to offer consumers with various live game experiences and to decrease the cost of media production. It also provides multi-camera capture for amateur and

Figure 1.1: Pictures of a real broadcast system. Left: ESPN broadcating trucks; right: scenes inside a truck. Figure from Sports Video Group[1]. Photo credits: Joe Faraoni/ESPN Images; Brandon Costa/SVG.

semi-professional sports.

A professional sports broadcasting system is very complex. Figure 1.1 shows pictures of a real broadcast system used in the National Hockey League (NHL) by ESPN. Not only is the equipment expensive, but the system also demands a group of highly skilled individuals including the camera operator, director and commentator. For example, each camera should be controlled by a camera operator. If multiple cameras are used, the system requires a director to schedule all the cameras. The captured videos are post-processed and are broadcast with audio. Even for the minimum configuration, the system needs at least 3-5 well-trained operators.

The motivation of this thesis is to reduce the cost of human operators in group sports broadcasting. Big tournaments (such as the World Cup) can afford the high cost of hiring professional operators, while minor leagues usually are unable to afford the cost. In practice, minor leagues often use static cameras to record the game, in which players are small and details are lost. Thus, an affordable automatic broadcasting system is in high demand. Our techniques aim to provide informative and enjoyable videos for minor leagues and amateur users.

Let us imagine an ideal automatic soccer broadcasting system based on computer vision techniques. A couple of fixed location cameras look at the playing surface and detect where the players, referees and ball are. Using human pose estimation techniques, the system can also detect player actions and their intended movements. These cameras work like the eyes of human operators. At the same time, the detected information is processed by an intelligent agent (e.g., a computer). The process can use big data such as previous matches, player profiles and team information to make decisions. The decisions include where to put the broad-

[1] https://www.sportsvideo.org

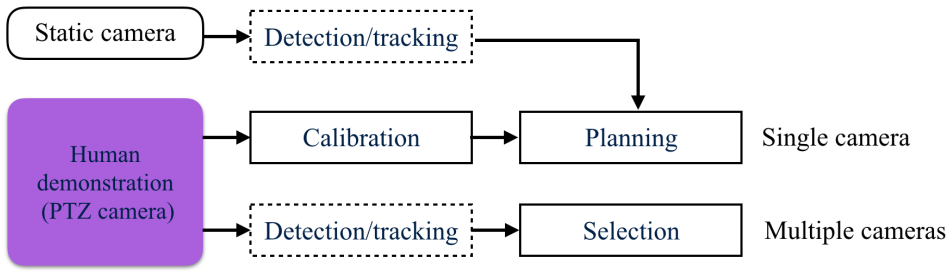


Figure 1.2: System overview. Solid rectangles: components of our system; Dashed rectangles: components on which our system relies but are not included in the thesis.

cast cameras, what are their poses and zoom factors and which camera should be "on air" at any time. In some circumstances, the PTZ cameras can be controlled by humans, or by computers in other scenarios.

The sports industry has recently deployed some automatic broadcasting systems. For example, the Spanish soccer league La Liga introduced an automated broadcasting system that uses up to eight high definition cameras in 2017-18 season [Adr18]. The system is semi-automated by one or two personnel on site: a camera operator filming handheld touchline clips and a vision mixer to switch the feeds. The system is being used by clubs as an inexpensive means to stream to official websites. This thesis provides techniques that can further reduce the number of human operators in such systems as well as improve their capabilities.

In practice, autonomous camera systems must solve three simultaneous problems [CC14]:

1. Planning: Where should cameras look?
2. Control: How should cameras move?
3. Selection: Which camera should be "on air"?

This work focuses on the first and third problems: camera planning and camera selection. Our fundamental idea is to replace human operators with computer programs and robotic PTZ cameras. The computer programs learn camera operations such as panning the camera and switching between cameras from human demonstrations. By doing so, the system aims to achieve competitive results with professional human operators but with much lower cost. Besides using PTZ cameras, other options include cropping sub-images from a wide-view static camera or switching between many static cameras. These methods are out of the scope of our interests.

	Camera planning		Camera selection	
Camera	Loc.	Pose	Loc. and pose	ID
Ideal	system	system (3DoF)	system	system
Ours	human	system (1DoF)	human	system

Table 1.1: Comparison of an ideal system and ours. In an ideal system, all the camera parameters are automatically controlled by the system itself. To simply the problem, our system is partially controlled by human operators or pre-fixed by human operators (e.g., there is only 1 degree of freedom (DoF) in camera planning).

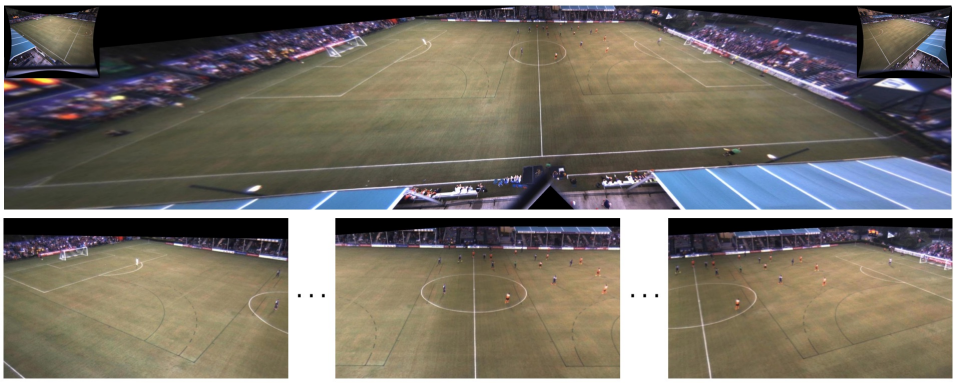


Figure 1.3: Camera planning task. Camera planning predicts proper camera angles for a main PTZ camera. Top: a panoramic view of the game; down: example images from three viewpoints.

Table 1.1 compares the ideal system with our system on the components that are controlled by human operators or by the system itself. As a research project, parts of our system are controlled by human operators so that we can simplify the problem and develop the remaining parts of the system. Building an automatic broadcast system requires lots of components and providing all the components is out of the scope of our work. The following components are not the contribution of this work: hardware design and video capture (e.g., video synchronization), player and ball detection/tracking and physical camera control.

Figure 1.2 shows the overview of our system. It has three components: camera calibration, camera planning and camera selection. Figure 1.3 shows the camera planning task which predicts proper camera angles for the main PTZ camera. Figure 1.4 shows the camera selection task which predicts a camera ID from multiple candidate cameras. This work also offers state-of-the-art solutions for sports cam-

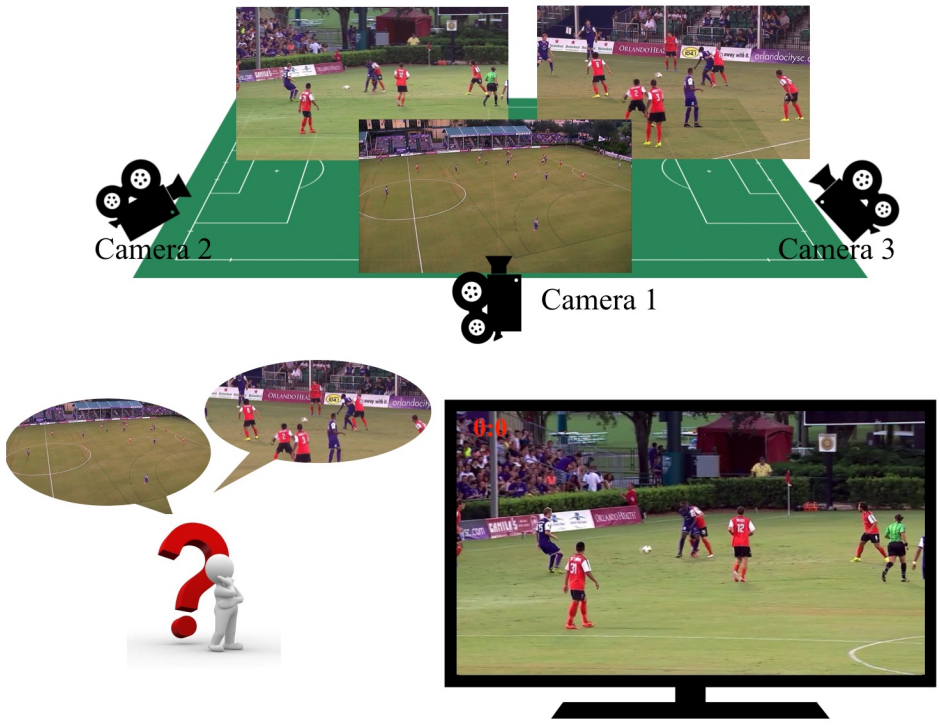


Figure 1.4: Camera selection task. Camera selection predicts a camera ID from multiple candidate cameras.

era calibration to obtain labels from human's demonstration. Figure 1.5 shows the camera calibration task which estimates a homography matrix from a geometric model to an image. The calibration task is integrated into the entire system by providing labels for the camera planning task.

The tasks are challenging from three perspectives. First, the broadcasting camera may focus on small areas and move quickly to capture interesting events. Thus, long-time camera calibration suffers from motion blur and a narrow field of view [Tho07]. Second, players act and interact with other players frequently in group sports. Consequently, how to effectively represent the multiple objects (e.g., players and ball) in an image is a hard problem in computer vision. Third, large amounts of training data are not directly available because researchers cannot access raw videos of all cameras that are used in broadcasting. The small training data degrades the performance of learning methods.

The tasks of camera planning and camera selection are closely related to sequential prediction. Sequential prediction operates on sequences, for example, pre-

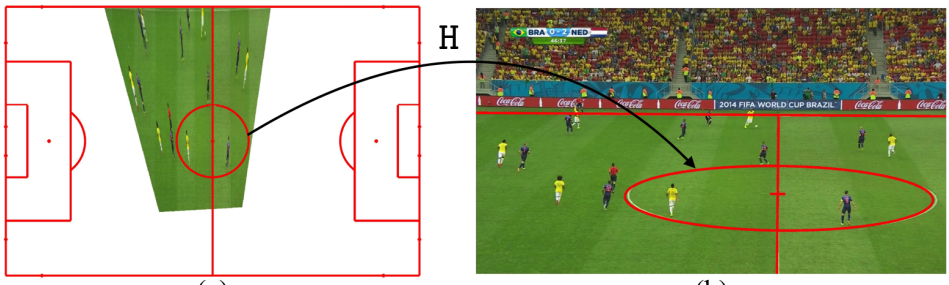


Figure 1.5: Camera calibration task. Camera calibration estimates a homography matrix H from a geometric model to an image.

dicting camera angles for a one-minute basketball game. The desired algorithms use temporal information as well as the appearance information in images. The predictions should follow the physical movement of the camera, for example, the camera trajectory should be smooth over time. Part of my work provides solutions to general sequential prediction problems such as how to predict the movements of a steering wheel for autonomous vehicles.

1.2 Applications of developed methods

The thesis presents a system for automatic sports broadcasting. The system is composed of a group of techniques that can be applied to a broad range of applications:

- **Two-point method:** This method can be applied to any PTZ camera with prior information of camera base.
- **Synthetic edge image calibration.** This method can be applied to many sports which have texture-less playing grounds such as soccer, volleyball, ice hockey, tennis and so on.
- **Recurrent decision trees:** This method can be applied to smooth temporal signal prediction in real time, for example, predicting the steering wheel angles in autonomous vehicles.
- **Overlapped hidden Markov model:** This method can be applied to smooth temporal signal prediction with some tolerance of delays.

1.3 Contributions

The work presented in the thesis contains several original contributions which can be summarized as follows:

- We developed two camera calibration methods. The first method only requires two point correspondences instead of four correspondences in previous work. The second method requires fewer human annotations by using synthetic edge images. The experiments demonstrated that our methods outperform existing methods on several public datasets.
- We developed two camera planning approaches. The first approach directly predicts smooth camera angles in real-time. The second method optimizes the camera trajectory in overlapped temporal windows. These two methods showed state-of-the-art performance on a basketball dataset and a soccer dataset, respectively.
- We designed two camera selection methods. The first method uses deep features for camera selection for the first time. The second method augments the training data with Internet videos. We demonstrated competitive results with the selection from human operators on a soccer dataset.

1.4 Outline

The remainder of this thesis is organized as follows: Chapter 2 is an overview of the background and related work; Chapter 3 addresses the problem of camera calibration for sports videos; Chapter 4 presents two camera planning approaches; Chapter 5 describes two camera selection methods. Finally, Chapter 6 summarizes the thesis and proposes several directions for future work.

Chapter 2

Background and Related Work

In Chapter 1, we show that an autonomous camera system is an important application of computer vision. Consequently, a large body of previous work has been devoted to this area and many methods have been published in the literature. This chapter presents background material and an overview of previous work. Previous work includes various autonomous camera systems, sequential supervised learning methods and sports camera calibration methods.

2.1 Preliminaries

In this thesis, we use camera models in Chapters 3, random forests in Chapters 3, 4 and 5, and deep neural networks in Chapters 3 and 5. To avoid repetition, we describe the fundamentals of these methods here.

2.1.1 Camera models

We use the pinhole model to describe the projection matrix of a camera

$$\mathtt{P} = \mathtt{KR}[\mathtt{I}|-\mathbf{C}], \tag{2.1}$$

where $\mathtt{K}$ is the intrinsic matrix, $\mathtt{R}$ is a rotation matrix from world to camera coordinates, and $\mathbf{C}$ is the camera's center of projection in the world coordinates. To simplify the problem, we assume square pixels and a principal point at the image center (u_c, v_c). We found this assumption holds well for broadcasting videos. As a

result, the focal length f is the only unknown variable in the intrinsic matrix:

$$\mathtt{K} = \begin{bmatrix} f, 0, u_c \\ 0, f, v_c \\ 0, 0, 1 \end{bmatrix}. \tag{2.2}$$

The 3×4 projection matrix $\mathtt{P}$ maps a homogeneous 3D point $\mathbf{X} = [x, y, z, 1]^\mathsf{T}$ to a homogeneous 2D image point $\mathbf{x} = [u, v, w]$ via

$$\mathbf{x} = \mathtt{P}\mathbf{X}. \tag{2.3}$$

The camera parameters can be estimated by minimizing the projection error using methods such as [Zha00] and [LMNF09].

When $z = 0$, the world coordinate is on the ground plane, and the ground to image homography $\mathtt{H}$ can be extracted from the projection matrix $\mathtt{P}$ (up to a scalar uncertainty):

$$\begin{aligned} \mathbf{x} &= \mathtt{P}[x, y, 0, 1]^\mathsf{T} \\ &= \mathtt{KR}[\mathtt{I} | -\mathbf{C}][x, y, 0, 1]^\mathsf{T} \\ &\cong \mathtt{H}\mathbf{x}' \end{aligned} \tag{2.4}$$

where

$$\mathtt{H} \cong \mathtt{KR} \begin{bmatrix} 1, 0, -\mathbf{C}_x \\ 0, 1, -\mathbf{C}_y \\ 0, 0, -\mathbf{C}_z \end{bmatrix}. \tag{2.5}$$

The homography can be estimated using at least 4 point-to-point correspondences [HZ03].

2.1.2 Random forests

A random forest is an ensemble learning method for classification, regression and other tasks [Bre01, CSK12]. In training, random forests construct a multitude of decision trees. In testing, random forests predict category labels (classification) or mean prediction (regression).

The essential idea in random forests is to average many noisy but approximately unbiased models, and hence reduce the variance [HTF09]. Random forests correct for the overfitting problem of decision trees by using bootstrap aggregating (bagging) and dimension sub-sampling. In training, bagging randomly chooses a part of the data to build each tree independently. The dimension sub-sampling op-

timizes the feature selection from a subset of feature dimensions. Random forests are very fast in testing.

Training and testing In our problem, we use random forests to predict a continuous scalar (camera angle). So we focus on random regression forests. A regression forest is an ensemble of T independently trained decision trees that minimize the regression error. Each decision tree is a binary-tree structured regressor consisting of split (or internal) nodes and prediction (or leaf) nodes. A set of example feature vectors $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ and their corresponding ground truth labels $\{y_1, y_2, \cdots, y_n\}$ are provided for training.

Following a top-down approach, the decision tree optimizes the parameters θ_i of the i^{th} node from the root node and recursively processes child nodes until leaf nodes are generated. The parameter θ_i is chosen from a set of randomly sampled candidates Θ_i.

At each internal node i, depending on the subset of the incoming training set S_i, random forests learn the function split that "best" splits S_i into S_i^L and S_i^R. S_i^L and S_i^R are sets of left and right sub-trees, respectively. This problem is formulated as the maximization of an objective function I at the i^{th} internal node

$$\theta_i = \arg\max_{\theta \in \Theta} I(S_i, \theta). \tag{2.6}$$

For scalar value regression, the mean squared error loss is widely used:

$$I(S_i, \theta) = \sum_{v \in S_i} (y - \bar{y}_i)^2 - \sum_{j \in L,R} \Big(\sum_{v \in S_i^j} (y - \bar{y}_i^j)^2 \Big), \tag{2.7}$$

where $\bar{y}_i$ indicates the mean value of y for all training samples reaching the i^{th} node. Node that the left and right subsets are implicitly conditioned on the parameter θ. In this loss function, the first term is a constant value for an internal node. The second term encourages smaller variances in child nodes. Node splitting is regularized by setting the minimum splitting size which is usually the same as the minimum leaf node size. The minimum leaf node size is a hyper-parameter that controls the complexity of a decision tree.

Training terminates when a node reaches a maximum depth D or contains too few examples. In tree t, one leaf node includes a set of samples and a prediction model is trained from these examples. For regression forests, the prediction model can be a constant value (e.g., mean value), a linear or a polynomial function that is fitted from the examples. The choice of the prediction model depends on the data. For example, if the y can be well approximated by piece-wise linear functions

(e.g., has small values of curvatures), we can choose the linear function as the prediction model. The constant-value model is also commonly used because it can approximate other models by constructing more and deeper trees. In this thesis, we use the constant-value model as the default prediction model.

During testing, a testing sample traverses the tree t from the root node to a leaf node. The leaf outputs an estimated value using the mean value of the training examples stored in the leaf. The mean value of predictions from all trees is the final output.

In random forests, the prediction ability depends on the model parameters. The most influential parameters are the maximum allowed tree depth D, the amount of randomness (i.e., the number of random samplings for a decision boundary), the forest size T, the training objective and the choice of features. Generally, the maximum depth, the amount of randomness and the forest size increase when the number of training samples increases. The model parameters can be experimentally determined by cross-validation.

The backtracking regression forest is a useful variance of regression forests [MCT$^+$17]. In training, it stores mean values of samples $(\bar{\mathbf{x}}_l, \bar{y}_l)$ in leaf nodes. In testing, it outputs the feature distance $\|\bar{\mathbf{x}}_l - \mathbf{x}\|_2^2$ as well as $(\bar{\mathbf{x}}_l)$. The feature distance can be used to remove outliers.

Besides classification and regression, random forests can replace missing data with proper values in learning. For example, [LW$^+$02, Bre03] introduced the proximity approach which uses a proximity matrix to measure the similarity of data that occur within the same terminal node. The method first roughly imputes the missing values: missing values for continuous variables are replaced with the median of nonmissing values, or data are imputed using majority vote for categorical variables. Then, the method trains a random forest and re-imputes data using the proximity matrix. For continuous variables, the imputed data are a proximity weighted average of the nonmissing data. For categorical variables, the one with the largest proximity is used. The method repeats the second step for a few iterations to achieve a stable solution. One disadvantage of the proximity approach is that the trained forest cannot be used to predict on test data with missing values.

To improve the proximity approach, [IKBL08] proposed the random survival forest (RSF) method which dynamically imputes missing values when growing a tree. In data splitting, the method draws a random value for the missing value from the empirical distribution of nonmissing values. After splitting, the missing values are reset to be missing. In a terminal node, missing values are imputed: for continuous variables, the final imputed value is the average value of nonmissing values; for a category variable, the majority vote is used. The method is referred as *random survival forest* because it was first used in survival analysis which analyzes the expected duration of time until one or more events happen such as a death in

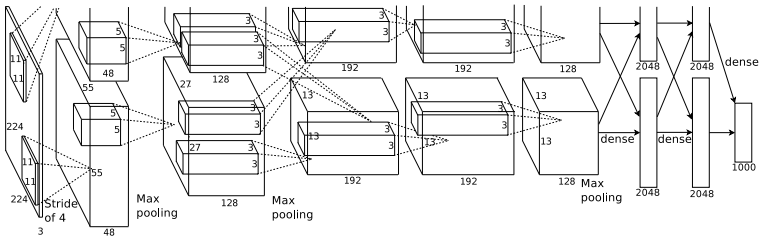


Figure 2.1: AlexNet. Figure reproduced from [KSH12], ©Neural Information Processing Systems Foundation, Inc.

biological organisms [IKBL08]. An alternative explanation of the name of random survival forests is from the data splitting process. Because the missing value is randomly imputed during data splitting, the missing data can be correctly imputed or incorrectly imputed with some probability. The probability accumulates along the path from a root node to a terminal node. When the missing data is assigned to a "correct" terminal node, the missing data *survives*, and vice versa.

2.1.3 Deep neural networks

Deep neural networks (DNNs) is important in many computer vision tasks such as object recognition, object detection and segmentation. The power of DNNs is that features are directly learned from raw images so that DNNs usually have higher performance given sufficient training data [GBC16]. DNN research includes network architecture [HZRS16], training techniques (optimization) [BCR$^+$17] and new applications.

The feed-forward convolutional neural network is one of the most common structures in computer vision. Figure 2.1 shows the structure of AlexNet [KSH12] which won the ImageNet challenge in 2012 [RDS$^+$15]. The input of the net is a raw RGB image and the output is the category of the object in the image. The network includes four standard layers of modern neural networks. The convolutional layer [LBBH98] extracts visual features in a local patch by applying a bank of learnable filters to the input. The pooling layer combines the outputs in a layer to its next layer. It decreases the dimension of the feature map. The fully-connected layer connects every neuron in one layer to every neuron in another layer. The loss layer computes the loss of the network output. For example, classification networks the softmax loss. The AlexNet uses the Rectified Linear Units (ReLUs) non-linearity which is $f(x) = max(0,x)$. It also uses dropout regularization which randomly sets hidden unites to zeros. The weights of the network are learned by error back propagation [Kel60, RHW86].

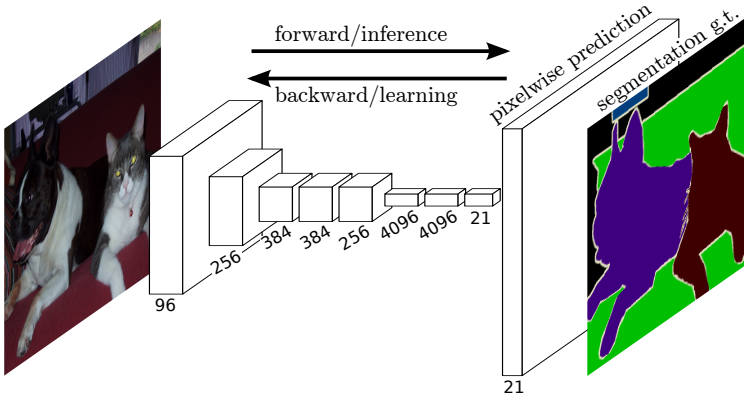


Figure 2.2: Fully convolutional networks. Figure reproduced from [LSD15], ©IEEE.

Besides these four layers, the deconvolution layer is also widely used in convolutional networks [LSD15] for spatially dense prediction tasks such as image segmentation, depth estimation and image generation. The network (see Figure 2.2) takes input of arbitrary size and produces correspondingly-sized output.

Modern deep learning frameworks like Caffe [JSD+14], TensorFlow [ABC+16] and PyTorch [PGC+17] provide basic components for network architecture and optimization methods. For example, they supply convolution, fully-connected, pooling, deconvolution layers and so on. They also provide stochastic gradient descent (SGD) [RM51], adaptive moment estimation (Adam) [KB14] and other optimizers. Some frameworks such as PyTorch also implement automatic differentiation methods so that users do not have to calculate the partial differentiation for new layers. These frameworks significantly boost the broad use of DNNs.

Among many deep networks, we are interested in feature extraction networks and generative adversarial networks (GANs).

Feature extraction. Feed-forward networks can be used to extract general purpose features for vision tasks [SEZ+13]. The easiest way is using the activation of some layers from a pre-trained network as the feature. When the data is associated with labels, the pre-trained network can be fine-tuned to improve the performance.

In some applications, input images only have neighborhood relationships but do not have meaningful labels. For example, a set of face images have similar/dissimilar labels for each pairs of images but do not have IDs for each image. We call similar/dissimilar labels as weak labels. When only weak labels are available, siamese networks have been used to learn deep features from paired images

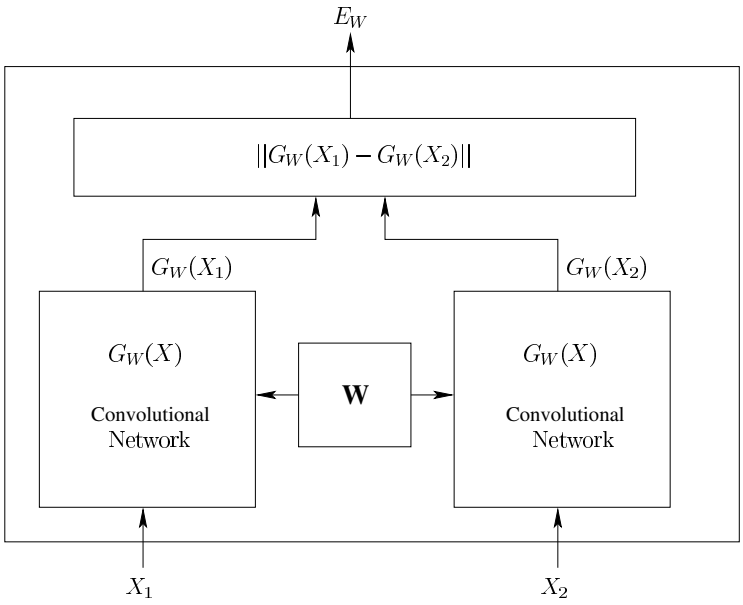


Figure 2.3: Siamese architecture. Figure reproduced from [CHL05], ©IEEE.

[CHL05]. Figure 2.3 shows the siamese architecture in which two branches share weights **W**. In training, the contrastive loss is minimized:

$$\mathscr{L}(\mathbf{w}, x_1, x_2, y) = yD_{\mathbf{w}}(x_1, x_2) + (1-y)\max(0, m - D_{\mathbf{w}}(x_1, x_2)), \tag{2.8}$$

where $D_{\mathbf{w}}(x_1, x_2) = \|G_{\mathbf{w}}(x_1) - G_{\mathbf{w}}(x_2)\|_2^2$, $G_{\mathbf{w}}(x)$ is the feature, and y is the label of positive/negative examples. When the paired example is similar ($y = 1$), it minimizes the distance between generated features. When the paired example is dissimilar ($y = 0$), the loss maximizes the distance according to the margin m. This method has been extended to triplet networks which have three weight-shared branches for inputs x_1, x_2 and x_3. The triplet network only requires relative similarities such as $D_{\mathbf{w}}(x_1, x_2) < D_{\mathbf{w}}(x_1, x_3)$. Thus, it can be applied to more applications. For example, [WL15] learned deep features from a triplet network to estimate 3D poses for small indoor objects.

Generative adversarial networks (GANs). GANs are generative models that learn a mapping from a random noise vector z to output an image y, $G : z \to y$ [GPAM$^+$14]. The generator G is trained to produce output that cannot be distinguished from

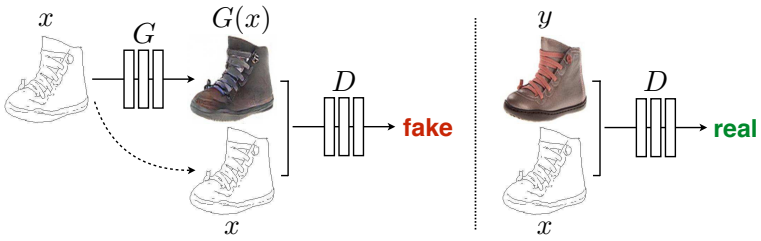


Figure 2.4: Conditional GAN example. A conditional GAN is trained to map edges to photo. The discriminator D, learns to classify between fake (synthesized by the generator) and real edge, photo tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and the discriminator observe the input edge image. Figure reproduced from [IZZE17], ©IEEE.

"real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator's "fakes". The objective of a GAN can be expressed as

$$\mathscr{L}_{GAN}(G,D) = \mathbb{E}_y[log(D(y))] + \mathbb{E}_z(log(1 - D(G(z)))), \tag{2.9}$$

where G tries to minimize the objective against an adversary D that tries to maximize it. In vision applications, the generative network is usually fully convolutional networks and its variations [LSD15, RFB15] so that the input and output are images.

Conditional GAN is an important extension of GAN. Conditional GANs learn a mapping from an observed image x and a random noise vector z, to y, $G : \{x,z\} \to y$. The generator G is trained to produce outputs that cannot be distinguished from "real" images by an adversarially trained discriminator D, which is trained to do as well as possible at detecting the generator's "fakes". The objection of a conditional GAN can be expressed as

$$\mathscr{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[log(D(x,y))] + \mathbb{E}_{x,z}(log(1 - D(x,G(x,z)))). \tag{2.10}$$

Previous approaches have found it beneficial to mix the GAN objective with traditional losses such as L2 or L1 distance [IZZE17]. For example, the task is to generate a photo from an edge image like the one shown in Figure 2.4. We can minimize the L1 distance between the generated photo and a real photo in training:

$$\mathscr{L}_1(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1]. \tag{2.11}$$

The discriminator's job remains unchanged, but the generator is tasked to not only fool the discriminator but also to be similar to the ground truth. Thus, the objective is

$$G^* = \arg\min_G \max_D \mathscr{L}_{cGAN}(G,D) + \lambda \mathscr{L}_1(G), \tag{2.12}$$

where λ is the weight of the L1 loss.

2.2 Autonomous camera systems

The autonomous camera systems described in the literature can be categorized in many ways: systems that are applied in different areas such as surveillance, education and sports; systems that work on the virtual or real world; systems that focus on different components such as planning, control and selection. There is some overlap between these categories and variants of the same method can be part of multiple categories.

In this section, we present some of the best-known systems in the literature. We group these systems into three categories: camera planning, camera control and camera selection. Camera planning determines the camera pose for a single camera. Camera control determines how a single camera moves to the desired configuration. Camera selection determines which camera should be selected to capture the scene in multi-camera systems. When one system fills into more than one categories (for example a system has both camera planning and camera control), we describe the most significant part of the system.

Camera planning. Camera planning addresses the problem of "where should cameras look?", and in real environments is solved by analyzing sensor data. Most real cameras are stationary robotic pan-tilt-zoom cameras, and so the planning algorithm must output the desired pan angle, tilt angle, and zoom factor. Virtual cameras[1], on the other hand, are not constrained to stationary positions and are instead often parametrized by a subregion of a real video frame to resample.

As one of the earliest autonomous camera systems, *intelligent studio* (see Figure 2.5) [PP95] was demonstrated for a scripted cooking show. The system used the contextual information of the script to select the necessary computer vision algorithms to search for the expected events unfolding in the scene. Human-defined rules were then used to generate the appropriate virtual camera subregion based on the position of the computer vision object detectors.

When video cameras became more affordable, autonomous video production

[1] Here we refer in particular to cameras that resample images from real-world videos, excluding virtual cameras in computer graphics.

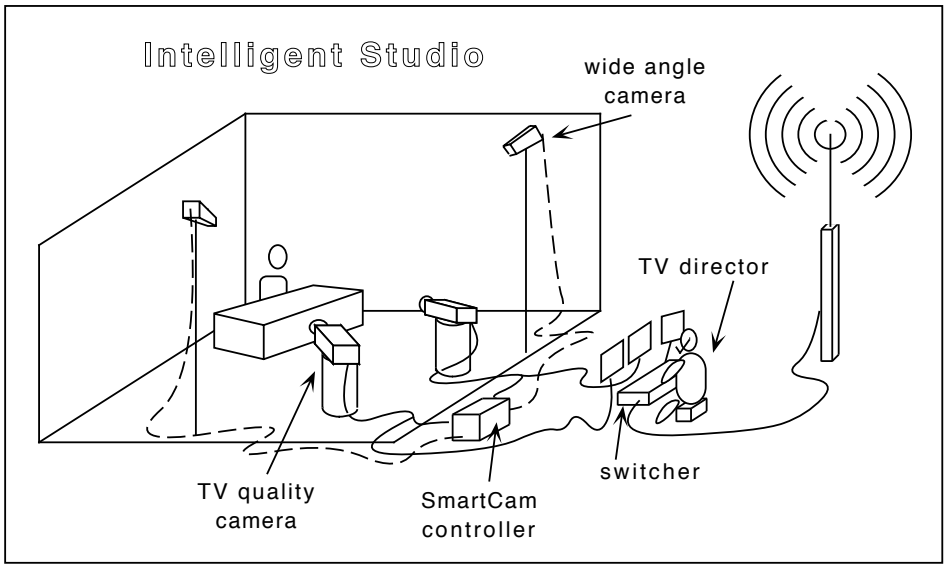


Figure 2.5: The concept of an intelligent studio, which uses wide-angle imagery to understand the events happening in the studio, and TV quality cameras to generate the images to be aired. Figure reproduced from [PP95], ©AAAI Press.

systems were proposed to record lectures [YF05, MAP+10, PMKS10]. The majority of systems used a fixed camera to detect and track the lecturer and used the raw tracking data to plan where the broadcast camera should look. [YF05] employed a virtual camera to generate the final output by cropping the appropriate subregion of the fixed camera. [MAP+10] tracked the position of the lecturer in a fixed camera, and use a bimodal planning algorithm that could switch between preset camera configurations, and a dynamic one which followed the lecturer.

More recently, virtual camera planning has been applied to more complex scenarios such as team sports. [AKK06] tracked the locations of soccer players and the ball using a fixed camera. Additionally, they defined rule-based classifiers to recognize game situations (such as penalty kicks and free kicks) based on the movement of the ball over a temporal window. These events were used to infer the zoom setting of the virtual camera, such as a wide shot for a free kick. Figure 2.6 shows the clipping examples of their method. A hybrid system that uses both static and PTZ cameras was used to predict the camera pan angle for basketball broadcasting [CMM13]. The system tracked basketball players using fixed cameras and determined the pan angle for a robotic camera by computing the centroid of the players'

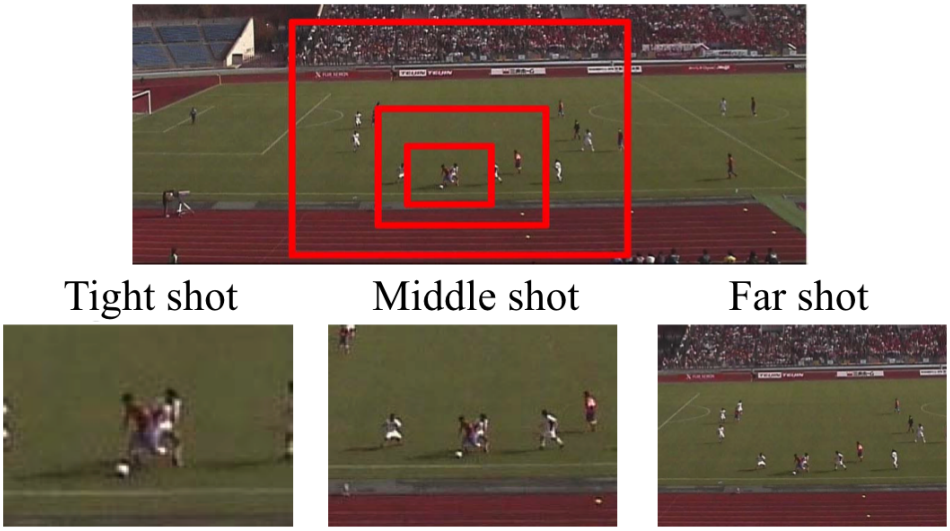


Figure 2.6: Virtual panning and zooming. Virtual panning and zooming technique clips subregions from hi-resolution images and control the region size and position. Figure reproduced from [AKK06], ©IEEE.

locations.

Most planning algorithms are based on tracked salient objects, such as faces in lectures, and player positions in team sports. For example, [CDV10] tracked basketball players and the ball using fixed cameras. The subregion for synthesizing a virtual camera was determined by a user-defined weighted sum of attentional interests (such as a following a 'star' player). Occasionally, additional features such as audience gaze direction [DO04] and visual saliency [PMKS10] have been incorporated as well. [CRS+15] extensively analyzed spectator behavior in ice hockey games by introducing the S-HOCK dataset which has annotations of people detection, head detection, head pose, posture, action, supported team and so on. If camera planning were performed in these games, it certainly could take advantage of these annotations. In almost all cases, the planning algorithm follows the tracked object, often in conjunction with a set of hand-crafted heuristics [Lin13].

Alternatively, learning based methods, such as SVM [PMKS10], neural networks [OIF09] and k-nearest neighbors [DDG07], have been investigated to produce more complex camera planning without explicitly modeling the underlying process. For example, [PMKS10] trained an SVM classifier within a lecture environment using features such as estimated center-of-attention and distance between the center-of-attention and the center of the nearest chalkboard to predict when a camera should pan.

Drone cinematography can capture videos from viewpoints that are unreachable by hand-held cameras. For example, [NMD+17] proposed a method for automated aerial videography in the dynamic and cluttered environment. The method takes high-level plans as input, alongside interactively defined aesthetic framing objective and jointly solves for 3D quadcopter motion plans. They evaluated the method with a number of challenging shots that involve multiple drones and actors.

When cameras capture extremely wide angle videos such as 360° panoramic videos, the video cannot be displayed naturally on a rectangular screen. Camera planning can generate normal field of view (NFOV) videos that include most important content and are pleasant to watch. For example, [SJG16] proposed the Pano2Vid system which automatically generates natural-looking normal field-of-view video from panoramic views. They extended the PanoVid by allowing the system to control its field of view dynamically and proposed a method to encourage various outputs [SG17]. [HLL+17] have proposed "deep 360°" for piloting through 360° panoramic sports videos. Their method learns an online policy to focus on a foreground object (e.g., a skateboarder) while minimizing both view angle loss from human annotated ground truth and smoothness loss between consecutive frames.

Camera control. Camera control moves a camera from its current parameter state to the desired configuration generated by the planning algorithm. In most applications, this task is a regulating process: the camera must move smoothly between fixation points to output aesthetic videos, but also move fast enough to follow its planned state space sequence. Camera planning and camera control are exchangeable in some literature, especially for virtual cameras because virtual cameras can transit to the desired configuration without any physical movements.

An early method for camera control is from NHK (Japan broadcasting corporation) [KYA+97]. It analyzed how cameramen operated their cameras in cooking shows and sports programs to figure out the exact characteristics of smooth camera motion — i.e. such as determining the speed at which panning is no longer aesthetic. The results illustrated an asymmetry in panning speed limits: acceleration can be higher when easing into a motion versus deceleration when easing out. In a subsequent study [KKK00], they found these characteristics only worked well in simple scenes: smoothly following a target with erratic movements was substantially more difficult.

Because virtual cameras resample recorded video, control algorithms for virtual cameras can be devised in an offline fashion i.e., determining a smooth approximation to the planned signal can use information about both previous and future planned states. When [YF05] used a virtual camera to follow a lecturer, the control

algorithm estimated and smoothed the trajectory of the tracked subregion using temporal differencing and bilateral filtering respectively. When the camera was undergoing a panning motion, they applied the learned parameters of [KYA+97] to regulate the position of the subregion. [CDV10] used a Gaussian Markov random field (MRF) to generate a smooth state-space trajectory of each virtual camera. [NO04] used a probabilistic roadmap to generate an initial estimate of linear segments which link the current camera state to the desired future camera state. The path was refined by fitting circular arcs between the segments and computing a smooth velocity plan which depended on path curvature limits.

The task of moving a physical camera to keep an object of interest within the field of view is referred as *visual servoing* in the robotics literature. [SO02] employed a proportion-only feedback control algorithm to adjust the pan-tilt angle of a camera mounted on the end of a human operated boom to keep a target object in the center of the camera image. [FAH05] used a proportion-only controller to position the centroid of detected image features near the center of the images of a stereo camera pair. [GHD09] used a task-priority kinematic controller to keep a set of interest points within the camera field of view. They showed how the mean and variance of the point cloud are independent objectives: pan-tilt values are modified to keep the mean near the center of the image, and zoom is regulated to keep the standard deviation within the image boundary. [CMM13] used a proportion-only controller to drive a robotic camera, but included delayed feedback from a virtual camera which resampled the raw video to generate a more stable output.

Camera selection. Camera selection determines which camera should be "on air" in multiple camera systems. The majority of existing camera systems use a set of human-defined rules based on low-level tracking data to compute the shot quality of each vantage point. For example, [DGSG04] used a network of fixed cameras to observe a subject moving through an office environment. A set of rules based on low-level tracking data were derived from cinematography conventions, such as using a 'long' shot to follow a moving target, and switching to a 'medium' shot when the subject comes to rest. A viewpoint score was computed for each camera at each frame, and a 'resistance' factor was used to ensure a cut to any new vantage point only happened when there was a significant change in viewpoint score from the current "on air" camera. [ZRCH08] presented the *iCam*2 system for lecture broadcasting. The system can switch between speaker, audience and questioner views using heuristic rules. For example, if the speaker has been on the air for a while, the system should switch to the audience view with a certain probability.

Sports camera selection is more complex than lecture camera selection as more people quickly move in the scenes. As a result, most previous work is on off-line

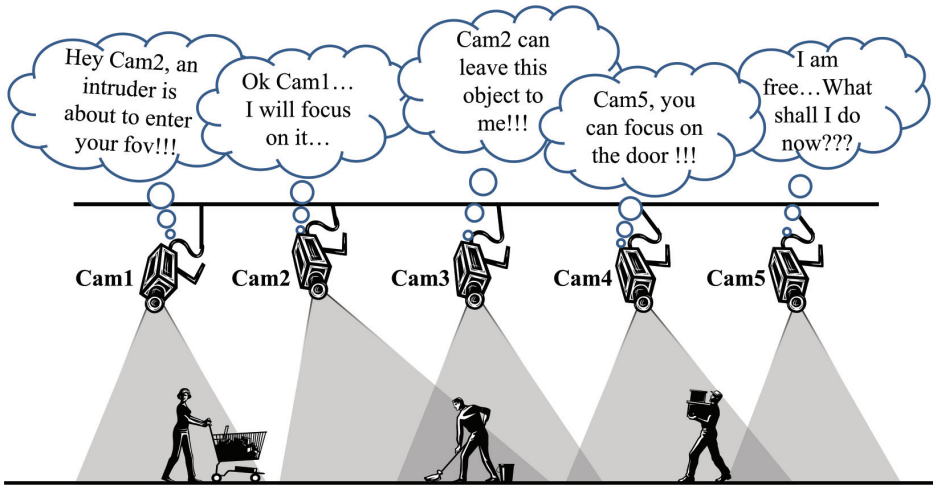


Figure 2.7: An example of surveillance system. Figure reproduced from [NAK15], ©ACM.

or static camera applications. For example, [WMHM14] proposed a soccer sequence recommendation system which carries out viewpoint evaluation and transition processes. Alternatively, [WXC+08] used a hidden Markov model (HMM) based on camera motion features to choose the best virtual camera to show in a soccer video production system. [CDV10] also used an HMM to select the best virtual camera for basketball production, but incorporated the size and visibility of user-selected 'salient objects' in the decision-making process as well. [DC11] proposed a multi-camera scheduling method based on view quality: the number of objects weighted by size and location, using a partially observable Markov decision process (POMDP) to minimize the number of inter-camera switches.

Instead of using predefined rules, methods that learn from human demonstrations have been proposed recently. For example, [CWH+13] proposed the automated director assistance (ADA) method by training a random forest classifier on field hockey tracking data to recommend the best view to broadcast directors. They used low-level features such as ball visibility, player locations and robotic camera movements. Previous broadcast data were used to learn a particular director's camera selection style.

Broadcast quality/preference measurement. Subjective experiments measure the quality by asking users (participants) to rate or compare the generated videos [WM08]. In the experiments, a number of participants are asked to watch a set

of video clips and rate their quality. For example, [GEL+15] introduced a pairwise comparison test to contrast the different camera movements that generate videos from panoramic videos for soccer games. Although subjective experiments are invaluable for quality measurement, they require a large number of viewers and usually cost more than other measurements.

An alternative way is measuring the similarity between videos generated by human operators and those produced by algorithms. The more similar, the better the algorithm is. For example, [FHS+15] use the F-measure to evaluate the accuracy of camera selection for soccer games, in which the ground truth selection is from the live TV broadcast of the same game. [SG17] proposed a "HumanEdit" metric that computes the overlap of camera field-of-view in each frame for generating normal field-of-view videos from 360° panoramic videos.

Instead of using professional videos as a reference, some recent methods focus on personalized broadcasting. For example, [WHE+18] learned the relationship between the viewer's personal viewpoint-selection tendency and the ball/player trajectory for soccer games. They evaluated three machine learning methods on a multi-static-camera dataset that includes 10 participants' selections. The experiments showed that the Gaussian mixture model (GMM) based method outperformed other methods.

Viewer behavior has been used to determine viewer preference for TV programs. For example, [TCNS15] developed a system for automatically estimating a viewer's rating of TV programs from his/her behavior in a home environment. They found that there is a strong correlation between the rating of viewers and the gaze (looking at the screen) ratios. The gaze ratio is the percentage of time that the viewer attends to the screen. With this assumption, the system first identifies whether the viewer is gazing at the screen or not by estimating head pose using a depth sensor. Then, the system fits a linear regression to predict the rating from the gaze ratio. In the experiment, the TV programs include 15 short clips that cover various topics. They evaluated the system from 30 participants individually. The experimental results showed that the system can robustly estimate a view's rating of TV programs.

Autonomous camera systems in other areas. In computer graphics, camera control encompasses viewpoint computation, motion planning and editing [CON08]. The nature of the camera control depends on these requirements which range from interactive approaches to fully automated control. Interactive methods aim to provide an intuitive interface for artists. For example, [LC15] introduced the *Toric space* for efficient virtual camera control. Their novel representation reduces the number of degrees of freedom (DoF) of camera control from 7 to 4 for problems

that involve at least two targets (e.g., two actors in a room) on the screen. Moreover, their method offers intuitive screen-space manipulation of virtual properties (such as size, vantage angle and location of targets). These techniques enable the rapid prototyping of complex camera sequences. Fully automated control uses some metrics to measure the "goodness" of views. For example, [VFSH01] defined "viewpoint entropy" based on information theory. Viewpoint entropy measures the amount of information contained in a scene that is conveyed by a given viewpoint of the scene. This method has been demonstrated on small indoor scenes (such as an office desk).

In surveillance, autonomous camera systems effectively coordinate multiple cameras [NAK15]. For example, [QT07] proposed a camera scheduling method for collaborative tracking from multiple PTZ cameras in a virtual train station. Figure 2.7 illustrates a complex coordination among multiple cameras.

2.3 Sequential supervised learning

Sequential supervised learning [Die02] is broadly applied in many domains, including natural language processing tasks such as part-of-speech tagging [Col02] and computational biology tasks such as protein alignment [YJEP08].

Unlike standard supervised learning, sequential supervised learning operates on sequences: each training example is a sequence of features $\mathbf{x}_i = \langle x_{i,1}, \ldots, x_{i,T_i} \rangle$ and a corresponding sequence of labels $\mathbf{y}_i = \langle y_{i,1}, \ldots, y_{i,T_i} \rangle$. The learned predictor outputs a sequence of labels for an input sequence of features. The naive approach treats each time instant independently, which generally does not work well as it does not use any temporal information. The learning methods include sliding window methods, recurrent sliding windows, structured SVM [TJHA05], hidden Markov models, conditional random fields [LMP01] and recurrent neural networks [MJ99].

Some sequential learning methods assume access to all of $\mathbf{x}$ before predicting $\mathbf{y}$. For example, the hidden Markov model (HMM) is a probabilistic model that represents a joint distribution $P(\mathbf{x}, \mathbf{z})$ in which $\mathbf{x}$ is a sequence of observations and $\mathbf{z}$ are the hidden variables [Bis06]. The HMM has two assumptions. First, the HMM assumes that z_{t-1} and z_{t+1} are independent when z_t is given. Second, the HMM usually assumes that the conditional distribution $P(z_{t+1}|z_t)$ does not change over time (i.e., homogeneous transition model). With these two assumptions, an HMM can be trained by the forward-backward algorithm. The HMM is widely used in speech recognition [Jel97], natural language modeling [MS99] and biological sequence analysis [Yoo09]. The maximum entropy Markov model (MEMM) [MFP00] tried to overcome the limitations of the HMM. An MEMM learns the conditional distribution $P(z_t|z_{t-1}, x)$ so that the observation is not limited to the

current observation.

We are interested in time-series prediction, for example, predicting a sequence of camera angles when the input is a sequence of player locations. Time-series prediction has two properties. First, time-series prediction only has a prefix of the sequence up to the current time t. Second, in training, we have the true predicted values up to time $t-1$, whereas in testing we are not given any true values and the future prediction may depend on previous predictions which can be close to or diverge from the true values.

Recurrent neural networks (RNNs) are one of the most popular methods for time-sequence prediction. A recurrent neural network connects units from a directed graph along a sequence. RNNs use internal state (memory) to process sequences of inputs so that they easily deal with varying-length inputs. The long short-term memory (LSTM) [HS97] is a popular variation of RNN as it can deal with the exploding and vanishing gradient problem in traditional RNNs. Researchers can add high-level constraints on the RNN outputs. For example, [HL17] imposed a temporal smoothness constraint on the task of predicting 3D human poses from 2D human poses.

A recent trend of sequential learning is convolution across times because CNNs can take full advantage of GPU parallelism. For example, [GAG+17] designed a fully CNN method for machine translation (e.g., English-French). The method outperforms deep LSTM based methods on two very large datasets at an order of magnitude faster speed. The method first encodes the input sentence using a hierarchical CNN. Then, the method uses gated linear units and a separate attention module in each decoder layer to generate the translation.

While sliding window methods can predict temporal sequences by stacking features $\mathbf{x}_{1,\cdots,t}$ until the current time t, the prediction performance degrades for two reasons. First, including previous features with a fixed length is ineffective. The window size is probably either too small to capture long-distance interactions or too large to be efficient. Second, conventional sliding window methods do not explicitly model the dependence of the current prediction $\hat{y}_t$ on previous predictions $\langle \hat{y}_1, \cdots, \hat{y}_{t-1} \rangle$. As a result, the algorithm is not optimal when the current prediction has high correlations with previous predictions. A plausible solution is using ground truth $\langle y_1, \cdots, y_{t-1} \rangle$ in training and using predictions in testing, but this solution causes a mismatch between the training and testing distributions.

Alternatively, the predictor should be trained from "real" previous predictions, which leads to a "chicken and egg" problem of how to define a training set that depends on the predictions of the model to be trained. [DILM09] proposed the SEARN method that integrates search and learning to solve structured prediction problems. They iteratively train sequential classifiers on a weighted combination of ground truth data and previous predicted data. In practice, the weights are stochas-

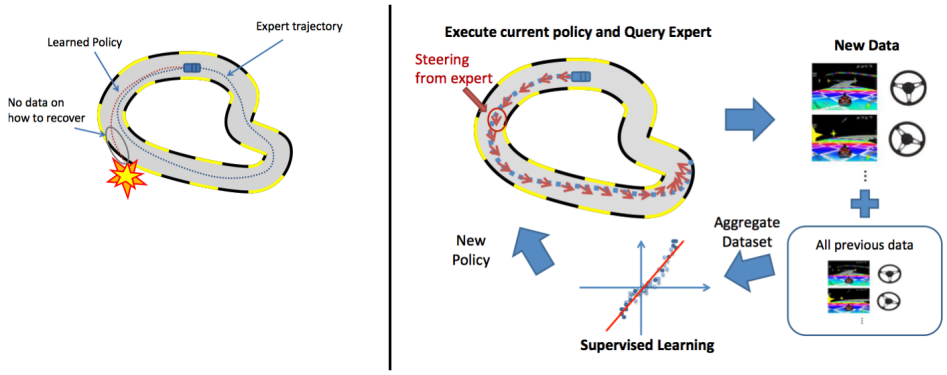


Figure 2.8: DAgger procedure in a driving scenario. Left: mismatch between the distribution of training and test inputs. Right: The dataset aggregation (DAgger) algorithm mitigates the mismatch problem by aggregating training data from many previous iterations. Figure reproduced from [Ros13], ©Stéphane Ross.

tically sampled. Each policy has different probabilities (weights) to be chosen to generate training examples in the current iteration. In training, the weights gradually shift from the ground truth data to the predicted data so that the final model is trained on the data that the predictor will actually expect to see in testing. The SEARN method is essentially a policy aggregation/interpolation method and may require a large number of iterations for the "gradual" shift. [RGB11] proposed the dataset aggregation (DAgger) method to train a stationary deterministic policy. Figure 2.8 shows one of the application scenarios of DAgger. In training, expert demonstrations of good behavior are used to learn a controller for autonomous cars. The prediction will be arbitrarily bad because the learner's predictions affect future input (e.g., observations/states) in testing. For example, the controller may drive the car out of the road and it cannot recover from this failure because there are no or very few similar examples in the training data. To overcome this problem, DAgger first collects data that are iteratively generated by previous controllers. Then, it trains the next controller under the aggregate of all collected data. The authors demonstrated the good performance of DAgger on a 3D car racing game and the *Super Mario Bros* game.

2.4 Sports camera calibration

Researchers typically assume the playing surface is flat so that camera calibration is equivalent to estimating the homography from the ground to the image. Previous

work [GLW11, PZVP11] first manually annotates several reference images. Then, they calibrate other images by finding correspondences between the reference images and the testing image.

Points and lines are two dominant features to find correspondences. Point features, such as KLT [TK91] and SIFT [Low04] have been used in most existing calibration systems [OLL04, PZVP11].The normalized direct linear transformation (DLT) algorithm [HZ03] provides a closed-form solution when four pairs of point-to-point correspondences are found.

However, point features are not descriptive in sports such as soccer and US football because the playing fields are almost texture-less. To improve performance, [HF07] proposed a method that maintains a core set of feature correspondences from both global distinctive and local distinctive features. Because a line covers a large area in an image, line features are more robust under motion blur. For example, [OLL04] computed the final homography matrix using edge pixels in a line model for rapid moving hockey games. [Tho07] proposed a modified form of the Hough transform to track pitch markings and achieved real-time responses.

Combining point features with other features produces more accurate estimation. For example, [DW08] extended the DLT algorithm to include point and line correspondences. Their method outperformed the point-only method on an ice hockey video registration task. Ellipse features were used together with point/-line features in calibration. For example, [GLW11] used point, line and ellipse features for rectification of broadcast ice hockey videos. The CalibMe method [CL17] tracked point, line and ellipse features for soccer video calibration.

When point, line and ellipse features are unable to provide sufficient visual information, other features have been used to calibrate the camera. For example, [GZA12] proposed a parameter-free registration method by formulating the problem as image alignment. [CSM12b] employed a 'long range' gradient method to align the edge image with the geometric model for field hockey games.

Fully-automatic methods estimate a homography matrix by associating data without any human interaction. For example, [WCW+16] first reconstructed a panoramic court image from a basketball video. Then, they warped the panoramic court to the court template. Their method is more robust than individual-frame approaches as they use court lines and corners from multiple frames.

Deep learning based methods use semantic information to estimate the homography matrix. For example, [HFU17] formulated the problem as a branch and bound inference in a Markov random field where an energy function is defined in terms of semantic cues (e.g. field surface, lines and circles) obtained from a deep semantic segmentation network. Their method achieves moderate accuracy in soccer and ice hockey games. The accuracy is measured by intersection over union (IoU) between ground truth and predicted playing field areas using a single image.

Recently, [SBGJ18] formulated camera calibration as a nearest neighbor search problem over a synthetically generated dictionary of edge images. Their method first warps the training image to the field template in which the training image becomes a quadrilateral. Then, the method simulates pan, tilt and zoom by modifying the quadrilateral to synthesize edge images. After that, it uses the HOG feature to describe edge images and builds a database of feature-pose pairs. In testing, the homography is retrieved from the database using the testing image feature.

In search based methods, the quality of the edge image is very crucial. The edge image is the projection of the template in the image. Edge image detection aims to distinguish field markings (e.g., line and arcs) pixels from other pixels (e.g., on players or non-field backgrounds). For example, [HPV04] use a cross-correlation score between the candidate pixels and a color-based kernel to find edge pixels. [CL17] use ellipse and line segment detector (ELSD) [PGvG12] to find out soccer field markings. Recently, [SBGJ18] use a conditional generative adversarial network (CGAN) [IZZE17] to directly generate the edge images from RGB images.

In the nearest neighbor search, edge images are represented by feature vectors for efficiency. Edge image representations achieve effective data association between field templates (e.g., in top view) and their images from various camera poses. For example, [Tho07] proposed a spatialized Hough transform method in which the genuine Hough peaks come from samples in field areas instead of samples from non-field areas. [SBGJ18] used the histogram of gradients (HOG) [DT05] to represent edge images for soccer games. In an early version of [SBGJ18], deep feature representation has been reported with good results on synthetic images but the performance degrades on real images. They concluded that the networks are susceptible to the noisy edge images in their experiments.

Chapter 3

Camera Calibration

3.1 Introduction

In this chapter, we address the problem of estimating camera parameters of sports broadcasting videos. Because the playing ground is a plane, the task is equivalent to estimating a homography matrix from a geometric model (e.g., soccer field model) to an image. The homography matrix is a 3×3 matrix whose number of degrees-of-freedom is 8.

The most used method for sports camera calibration is feature based. The feature-based approach first manually annotates several reference images by aligning points/lines in the image with these correspondences on the geometric model. This process requires at least 8 constraints which can be 4 point-to-point correspondences or 3 point-to-point plus 2 point-on-line correspondences and so forth. Then, it automatically calibrates other images by matching features between the testing image and the reference images. State-of-the-art algorithms assume that at least 4 points are needed to estimate the homography. We argue that the four-point assumption holds in general homography estimation but it is not necessary for a pan-tilt-zoom (PTZ) camera in sports broadcasting. A PTZ camera has a fixed location and base rotation so that the degrees of freedom of the projection matrix is 3 (2 for rotation and 1 for focal length). Thus, fewer than 4 points can fully calibrate the camera.

In this chapter, we present two novel calibration methods. The novelty is to use the prior knowledge of camera positions. The first method is a two-point method which assumes the camera position is exactly known from pre-processing. This method achieves higher accuracy than previous methods in calibrating narrow field-of-view cameras which may have fewer than four point correspondences. The second method uses a database of synthetic edge images to calibrate images from

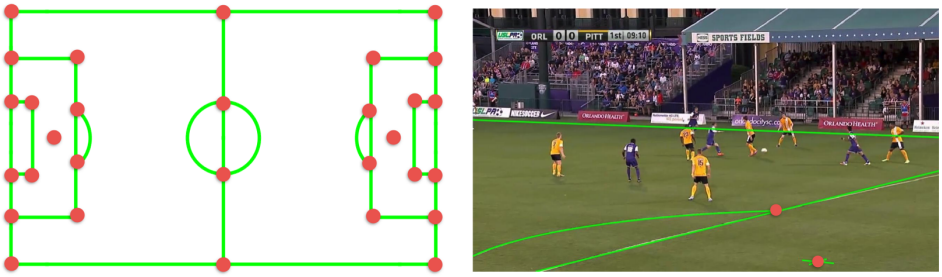


Figure 3.1: Two-point method example. (a) Soccer field model. The red points mark intersections of field curves and lines, and the penalty spots. These points are used in human annotation. (b) A narrow field of view image with overlaid markings (green lines). There are only two visible model points (red circles) in the image, making the four-point based annotation difficult. In this work, we propose a two-point algorithm to calibrate cameras of images like (b).

different stadiums. It relaxes the camera position constraint from exactly known to roughly known.

3.2 Two-point method

3.2.1 Method

We first propose a two-point method to calibrate a pan-tilt-zoom camera. In soccer broadcasting, the change in camera location and base rotation is very small. This phenomenon has been observed by Thomas [Tho07] and has been used to estimate PTZ configurations. Our work advances this research's direction to use the PTZ base information as a prior. Figure 3.1 shows a soccer field model and an example image calibrated by our method.

When the field-of-view (FOV) is narrow, calibrating new images using conventional reference image methods is time-consuming. Because each reference image only covers a small part of the environment (e.g., stadium buildings and commercial boards around the field), the new image query has to search through a large number of references to find the most similar one which still does not always provide sufficient matching points. To address this problem, we propose a *pan-tilt forest* algorithm to associate the data from many reference images effectively. In testing, the pan-tilt forest directly predicts the pan-tilt angles from local

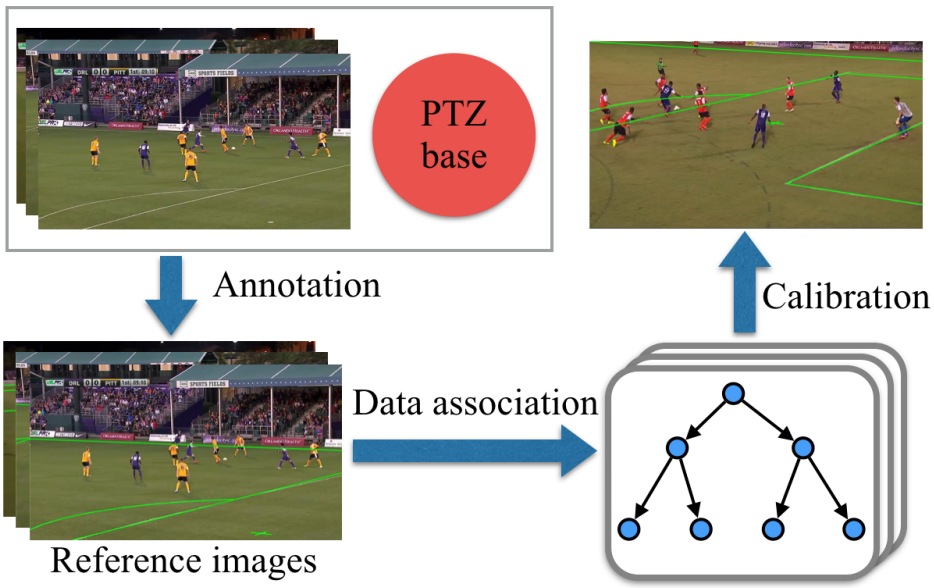


Figure 3.2: Pipeline of the two-point calibration system. Given the PTZ camera base information, we first manually annotate reference images using a two-point algorithm. We then use the pan-tilt forest method to encode the reference images. In testing, the predictions from the pan-tilt forest are used to calibrate a new image.

patch descriptors without image-to-image feature matching, greatly speeding up the algorithm.

Figure 3.2 shows the pipeline of our method. We first use a two-point algorithm to annotate reference images manually. Then, we use a pan-tilt forest algorithm to calibrate new images automatically.

Camera model. For the broadcasting camera, we assume that the center of rotation is the same as the center of projection and the lens distortion is negligible. We decompose the projection matrix (2.1) into:

$$\mathtt{P} = \underbrace{\mathtt{K}\mathtt{Q}_{\phi}\mathtt{Q}_{\theta}}_{\text{PTZ}}\underbrace{\mathtt{S}[\mathtt{I}|-\mathbf{C}]}_{\text{prior}}, \tag{3.1}$$

The combination of $\mathtt{Q}_{\phi}\mathtt{Q}_{\theta}\mathtt{S}$ describes rotations from world to camera coordinates. First, it rotates the camera to the PTZ camera base by $\mathtt{S}$ which is a general rotation

matrix with 3 degrees of freedom. Then the camera pans by $\mathtt{Q}_\theta$ and tilts by $\mathtt{Q}_\phi$. $\mathtt{K}$ is the intrinsic matrix.

$\mathtt{P}$ can be separated into two parts. The right part $\mathtt{S}[\mathtt{I}|-\mathbf{C}]$ is time invariant for a PTZ camera. This part is estimated from the training set and is used as a prior. The left part $\mathtt{K}\mathtt{Q}_\phi\mathtt{Q}_\theta$ describes the pan, tilt and zoom factor of the camera.

We estimate the prior part using a number of reference images that have sufficient point-to-point correspondences. First, each reference image (from the training set) is independently calibrated using point-to-point correspondences [Zha00]. Then, $\mathtt{S}$ and $\mathbf{C}$ are estimated by minimizing the reprojection errors in all the reference images. While calculating the estimates, all the reference images share the same $\mathtt{S}$ and $\mathbf{C}$ but have their own pan, tilt angles and focal length.

Because the camera only pans, tilts and zooms, the left part $\mathtt{K}\mathtt{Q}_\phi\mathtt{Q}_\theta$ projects a ray $\mathbf{r}=(\theta_\mathbf{p},\phi_\mathbf{p})$ to the image location $\mathbf{p}=(x,y)$:

$$\mathbf{p}=\underset{\{\theta,\phi,f\}}{proj}(\mathbf{r})=[f\tan(\theta_\mathbf{p}-\theta)+u_c, f\tan(\phi_\mathbf{p}-\phi)+v_c]^\mathsf{T}, \tag{3.2}$$

where we parameterize the ray by pan/tilt angles of its pixel location $\mathbf{p}$ and the left part $\mathtt{K}\mathtt{Q}_\phi\mathtt{Q}_\theta$ is represented by $\{\theta,\phi,f\}$. In particular, when $\mathbf{r}=(\theta,\phi)$, the ray passes through the image center (u_c,v_c). Note that the pan/tilt angles of a particular camera can be estimated by two pairs of $\{\mathbf{r},\mathbf{p}\}$ using (3.2).

Two-point algorithm. Given two 3D-2D point correspondences (e.g., from human annotation), our method first estimates the focal length from two points [Ged09]. Then it estimates the initial pan/tilt angles using one point [LZHT15]. Finally, it optimizes the PTZ parameters by minimizing the reprojection error using both points. The components of the method were introduced by [Ged09] and [LZHT15], separately. However, to the best of our knowledge, we are the first to integrate them and apply them to sports camera calibration.

We first estimate focal length from two point correspondences. Two 3D points $\mathbf{X}_1$ and $\mathbf{X}_2$ in the world coordinate are projected to image points $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively. $\mathbf{x}_1$ and $\mathbf{x}_2$ can be back-projected to two rays $\mathbf{d}_1$ and $\mathbf{d}_2$ in the camera coordinate. The cosine formula for the angle α between two rays is:

$$
\begin{aligned}
\cos\alpha &= \frac{\mathbf{d}_1^\top \mathbf{d}_2}{\sqrt{\mathbf{d}_1^\top \mathbf{d}_1}\sqrt{\mathbf{d}_2^\top \mathbf{d}_2}} \\
&= \frac{(\mathtt{K}^{-1}\mathbf{x}_1)^\top(\mathtt{K}^{-1}\mathbf{x}_2)}{\sqrt{(\mathtt{K}^{-1}\mathbf{x}_1)^\top(\mathtt{K}^{-1}\mathbf{x}_1)}\sqrt{(\mathtt{K}^{-1}\mathbf{x}_2)^\top(\mathtt{K}^{-1}\mathbf{x}_2)}} \\
&= \frac{\mathbf{x}_1^\top \omega \mathbf{x}_2}{\sqrt{\mathbf{x}_1^\top \omega \mathbf{x}_1}\sqrt{\mathbf{x}_2^\top \omega \mathbf{x}_2}},
\end{aligned}
\tag{3.3}
$$

where ω is the image of the absolute conic (IAC) and it is related to intrinsic matrix by $\omega = \mathtt{K}^{-\top}\mathtt{K}^{-1}$. The angle α is determined by three known points $\mathbf{C}$, $\mathbf{X}_1$ and $\mathbf{X}_2$. Thus the focal length f is the only unknown element in (3.3). It can be solved in the standard quadratic equation [Ged09] (details in Appendix A.1).

Then, we estimate pan, tilt from one point correspondence. With known focal length, a PTZ camera becomes a pan, tilt (PT) camera. [LZHT15] proposed a method for PT camera calibration using a single point. The method first builds a nonlinear PT camera model with respect to pan and tilt angles. Then a closed-form solution of pan and tilt can be derived by solving a quadratic equation of tangent of the pan angle. The solution for our camera model is in Appendix A.2.

With initial values of pan, tilt and focal length, our method further optimizes parameters by minimizing the reprojection error using Levenberg-Marquardt optimization. Figure 3.1(b) shows an example of the two-point method. There are fewer than 4 observable points (i.e. marking intersections and penalty marks) in the image. In that situation, the classic four-point method is not applicable and our two-point method is very useful.

Pan-tilt forests. We use a learning-based method to predict the ray $\mathbf{r}$ given its appearance in the image. We model this problem as a regression problem:

$$
\hat{\mathbf{r}}_\mathbf{p} = h(\mathbf{x}_\mathbf{p}), \tag{3.4}
$$

where $\mathbf{x}_\mathbf{p}$ is the feature (e.g., SIFT descriptor) that describes the image patch whose center is at $\mathbf{p}$. The label $\mathbf{r}_\mathbf{p}$ is given by:

$$
\mathbf{r}_\mathbf{p} = [\theta + \arctan\frac{x-u_c}{f}, \phi + \arctan\frac{y-v_c}{f}]^\top, \tag{3.5}
$$

where θ, ϕ are the pan and tilt angles of the camera. In training, $\{(\mathbf{x}_\mathbf{p}, \mathbf{r}_\mathbf{p})\}$ are paired training data. In testing, the ray $\hat{\mathbf{r}}_\mathbf{p}$ is predicted by the learned regressor $h(\cdot)$.

We use a random forest (Chapter 2.1.2) to predict the ray as it has achieved state-of-the-art in indoor and outdoor camera pose estimation [CGL+17, MCT+17] and denote it as *pan-tilt forest*. Please note that the pan-tilt predicts a ray that starts from the camera location. The ray may intersect the playing surface, resulting in a 2D point. However, a ray generally cannot be represented by a 2D point because the ray may hit the objects (e.g., buildings) above the surface.

We extract SIFT descriptors from sparse SIFT keypoints as features. We use standard regression forests for training. In a leaf node, we store mean values of samples that reaches that leaf node $(\bar{\mathbf{x}}_l, \bar{\mathbf{r}}_l)$. During testing, a sample travels the tree t from the root node to a leaf node. The leaf outputs the $\bar{\mathbf{r}}_l$ as the prediction. We also use the feature distance $\|\bar{\mathbf{x}}_l - \mathbf{x}\|_2^2$ (lower than a threshold) to remove outliers (e.g., features on players). As we will show in the experiment, this pre-processing is very effective in practice. We keep predictions from all trees and choose the one that minimizes the reprojection error in the camera pose optimization process.

Camera pose optimization. From the pan-tilt forest, we obtained a set of pixel location and predicted ray pairs $\{(\mathbf{p}, \hat{\mathbf{r}}_\mathbf{p})\}$. Camera pose optimization minimizes the reprojection error:

$$\{\theta, \phi, f\} = \arg\min_{\mathtt{P}} \sum_i \|\mathbf{p}_i - \mathtt{P}(\hat{\mathbf{r}}_{\mathbf{p}_i})\|^2, \tag{3.6}$$

where $\mathtt{P}$ is the projection matrix in (3.1).

Because the predictions from the pan-tilt forest may still have large errors, we use the RANSAC method [FB81] to remove outliers. In the internal loop of RANSAC, the minimal number of points required to fit the model is two. As RANSAC is a standard method to perform model estimation with the presence of outliers, the fewer iterations needed the better. Table 3.1 shows the number of iterations needed as a function of the minimum sets of points [Sca11]. The assumptions are: the probability of success is 99%, the percentage of outliers is 50%. Our two-point method requires much fewer iterations (16 vs. 71) than four-point based methods. That enables our method to find the optimal solution efficiently given the large ratio of outliers.

Min. set of points for model	Iteration #
1	7
2	16
4	71

Table 3.1: Number of RANSAC iterations.

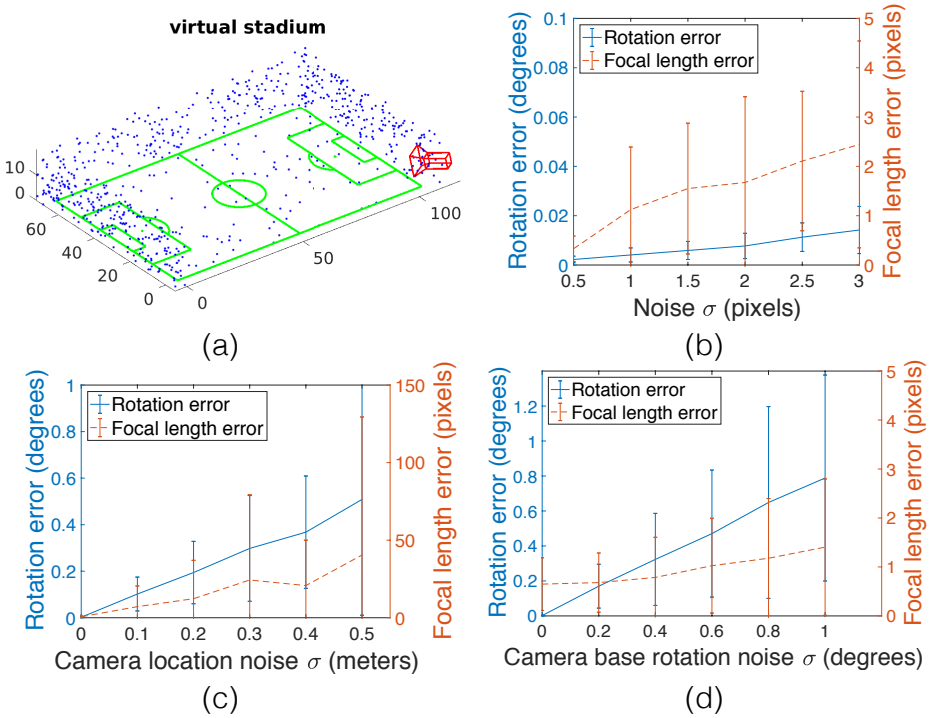


Figure 3.3: Synthetic experiments results. (a) An example of the synthetic stadium used in the experiment. (b-d) Mean errors as a function of the noise in feature location, camera base location and rotation, respectively. Error bars show standard deviations.

3.2.2 Experiments

Synthetic data

To estimate the accuracy of the method under typical feature location noise conditions, we conducted experiments with synthetic data. We set the camera base parameter as the one we estimated from a real dataset. That camera is located on the right corner of the playing field. Then, we randomly set pan, tilt and focal length values in the range of $[15°, 75°]$, $[-14°, -5°]$ and $[1500, 5000]$ pixels, respectively. Then, we randomly sample 200 rays in which about 90% of them are projected beyond the extent of the playing ground. The value of 90% is chosen experimentally to simulate keypoint distribution in real data in which most keypoints are on commercial boards and stadium structures. Finally, we add different levels of Gaussian noise to disturb the feature locations in the image and estimate the PTZ

configuration using our method. We simulate 100 cameras. For each camera, we repeated the experiment 100 times with different random values. The simulation process does not add outliers to the data.

Figure 3.3(b) shows the mean errors of rotation and focal length as a function of noise levels. Up to $\sigma = 3.0$, the mean rotation error is smaller than 0.02° and the mean focal length error is smaller than 2.5 pixels, demonstrating the stability of our method in terms of Gaussian noise in feature locations.

Because the estimation of camera base location and orientation may have errors in practice, we study the influence of estimation errors on our method. Figure 3.3(c,d) show the PTZ parameter errors as a function of camera base location and rotation uncertainties, respectively. Our method is robust to the uncertainties from camera location (see Figure 3.3(c)). One explanation is that the movement of the camera location is relatively small compared to the spatial extent of the stadium. On the other hand, the rotation error of our method is sensitive to camera base rotation uncertainties. It is not unexpected as the base rotation error can be directly propagated from the "prior" part to the "PTZ" part in (3.1). We report the absolute errors of the estimated focal length in Figure 3.3. The relative error of 100 pixels is about 3.2% of the ground truth.

Real data

Highlights dataset. This dataset was collected from the public soccer highlights dataset [LCLJH17] which was used to evaluate the accuracy of player detection algorithms. It contains four image sequences from two professional soccer games (two sequences for each game). In each game, the camera location is fixed. The image resolution is 1280×720 and the sample rate is about 6 FPS. The total frame number is 116.

The ground truth camera parameters are manually calibrated. We first calibrate the images that have at least four correspondences. Then we run a global optimization algorithm to estimate the PTZ camera base parameters. Then we calibrate the rest of the images using the two-point method by manual annotation. After that, we verify camera parameters by projecting the model to images. This process is repeated until there are satisfactory results.

World Cup dataset. The soccer dataset was collected by [HFU17] from World Cup 2014. The dataset has 10 games of 209 images for training, and 186 images from 10 other games for testing. The images consist of different views of the field with different grass textures, lighting patterns and heavy shadows. We select two games (BRA vs. MEX and BRA vs. NED) from the original World Cup dataset.

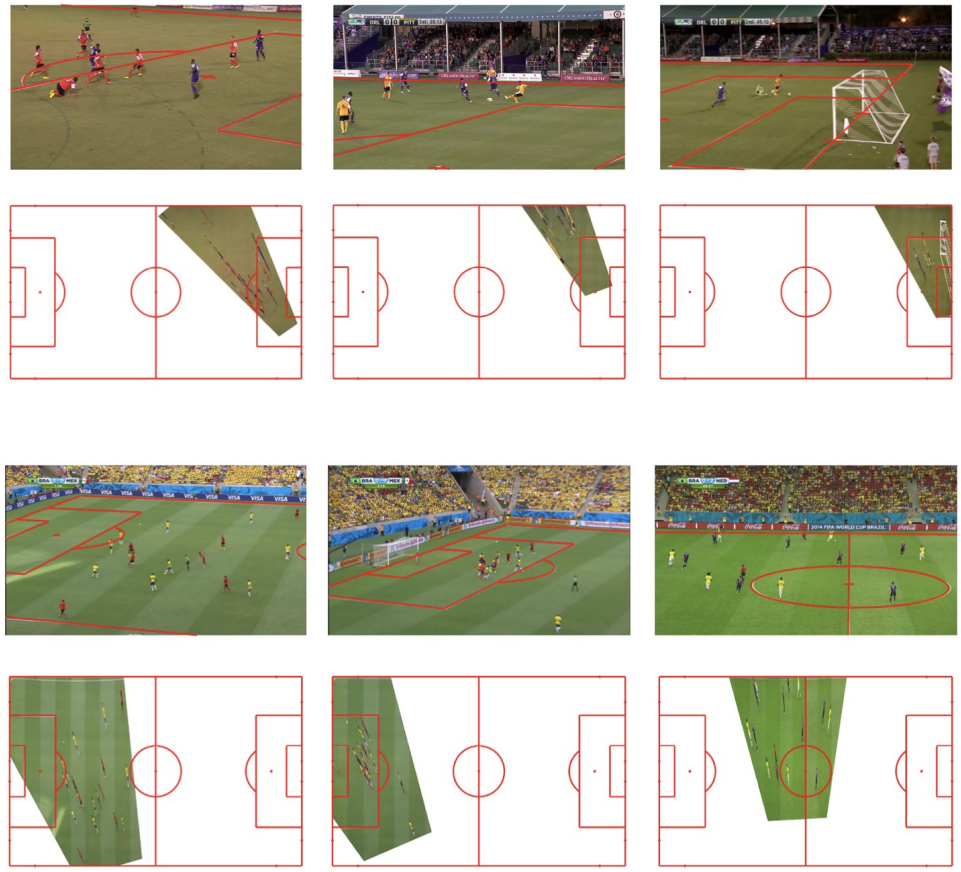


Figure 3.4: Examples of ground truth annotation. Top row: highlights dataset; bottom row: World Cup dataset.

The number of images is 42 and 33 for each game, respectively. In each game, images are randomly assigned to training or testing set with the ratio of 1:1. The ground truth camera pose is manual annotated. In these two games, we found that the assumption of fixed PTZ camera base holds in these two games.

Figure 3.4 shows examples of ground truth annotation by overlapping the markings on the image and warping the image to the model.

Error metric. We use the intersection over union (IoU) score [HFU17] to measure the calibration accuracy of different methods. The IoU is calculated by warping the projected model to the top view by the ground truth camera and the estimated camera (see the third row in Figure 3.6). We used the whole area of the model to

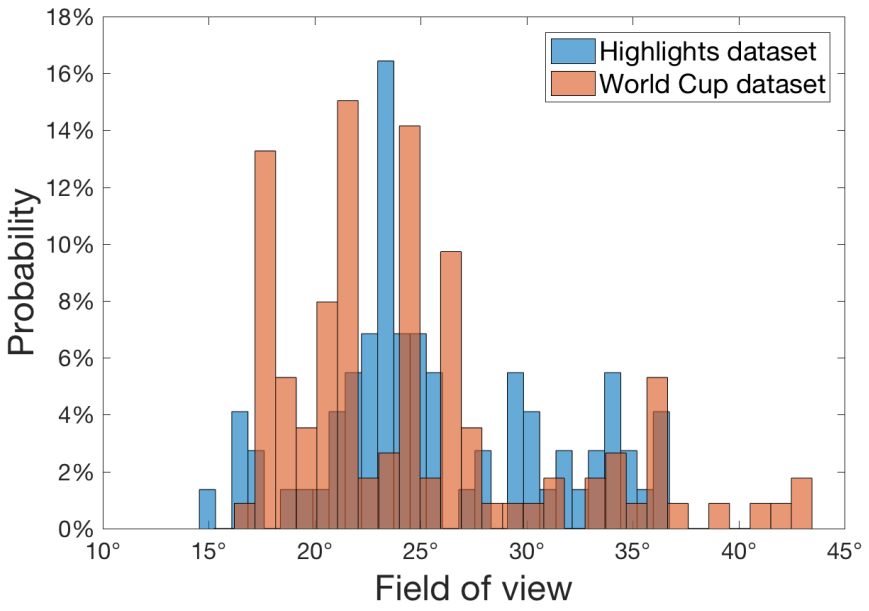


Figure 3.5: Distribution of field of view of the two datasets.

measure IoU values. For our method, we also report the rotation and focal length error distribution.

In the highlights dataset, we employ leave-one-sequence-out cross-validation (e.g., train on sequence 1, 2, 3 and test on sequence 4) in the experiment. In the World Cup dataset, half of the images were held out for training and the rest for testing in each game.

We compare our method with the CalibMe method [CL17]. CalibMe is a representative of reference-image-based methods and has achieved good performance on soccer videos. For each sequence, we manually select 3-5 reference images to cover the whole stadium to make sure the CalibMe method [CL17] works well. The number of the references is limited by the running time budget.

Main results. Figure 3.5 shows the distribution of field-of-view of two datasets. Both datasets have narrow FOV images (smaller than 25°) although the World Cup dataset has some wide FOV images.

In Table 3.2, we present mean IoU scores of different methods. The average IoU of our method is significantly higher (0.83 vs. 0.68) than the CalibMe method. Also, our method is faster (0.3 vs. 3.0 seconds) than CalibMe for two reasons.

Method	Range			CalibMe	Ours
	Pan (°)	Tilt (°)	Focal length (pix.)		
Seq1	[49,68]	[-13, -9]	[1586,4346]	0.75 ± 0.22	$\mathbf{0.88} \pm 0.06$
Seq2	[49,68]	[-11, -8]	[3330, 4947]	0.73 ± 0.27	$\mathbf{0.81} \pm 0.21$
Seq3	[17,67]	[-9,-5]	[1938, 4225]	0.61 ± 0.37	$\mathbf{0.69} \pm 0.36$
Seq4	[54, 69]	[-8, -7]	[2642, 4074]	0.62 ± 0.33	$\mathbf{0.94} \pm 0.04$
Avg	-	-	-	0.68 ± 0.30	$\mathbf{0.83} \pm 0.16$
Times/frame	-	-	-	3.0 s	0.3s

Table 3.2: Quantitative comparison. The table shows pan, tilt and focal length ranges in each sequence of the highlights dataset. The most right two columns are IoUs and corresponding standard deviations using different methods. The bottom line shows the average running time per frame for each method. The best performance is highlighted.

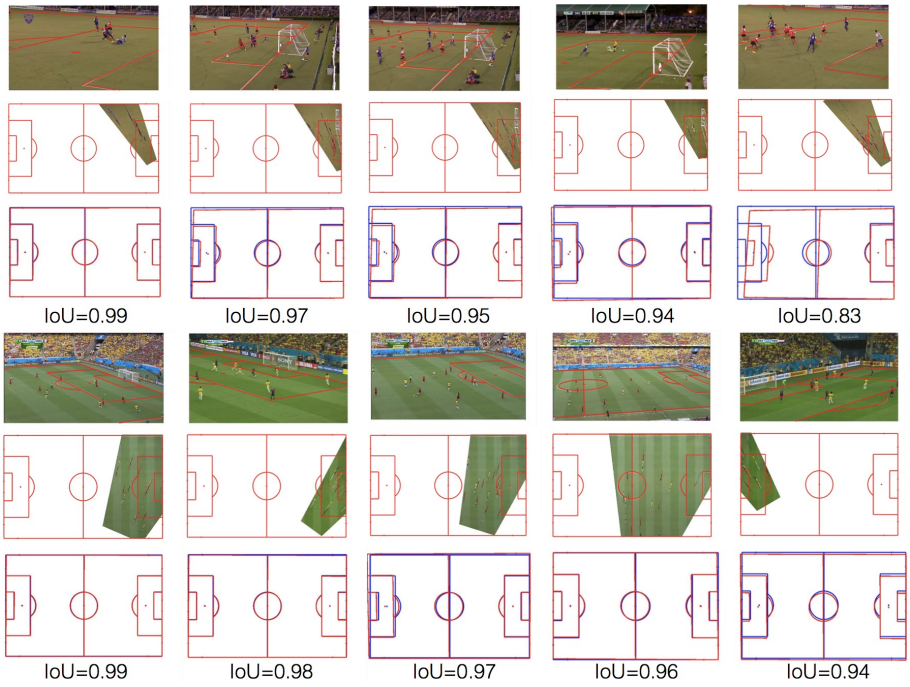


Figure 3.6: Qualitative results and corresponding IoU scores. Top row: results on highlights dataset; bottom row: results on World Cup dataset. In the top row, the most right column shows a typical challenging case for our method. The image has a very small texture-rich area, resulting in insufficient number of SIFT features for our method.

Games	CalibMe	Ours
BRA vs. MEX	0.84 ± 0.24	0.99 ± 0.01
BRA vs. NED	0.69 ± 0.37	0.98 ± 0.01

Table 3.3: World Cup dataset results. The values are mean IoU values and corresponding standard deviations.

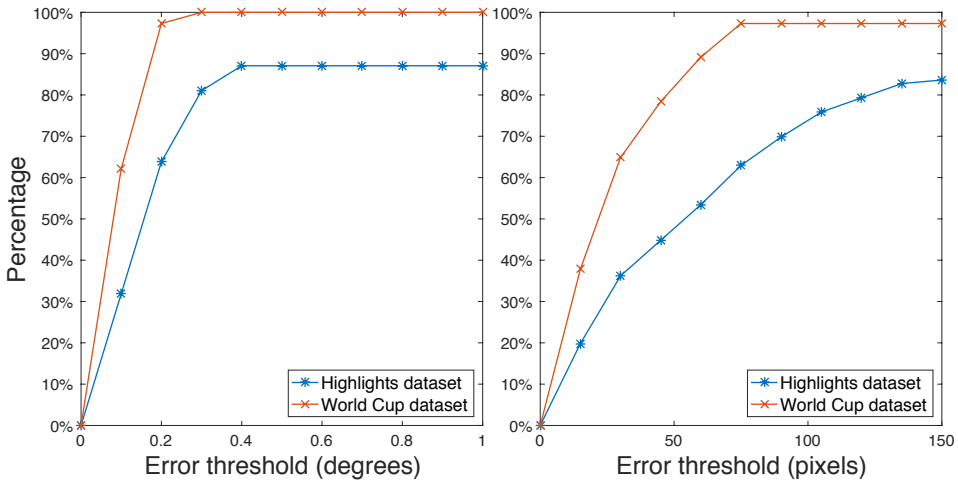


Figure 3.7: Cumulative distribution of the rotation and focal length estimation errors. Left: rotation errors; right: focal length errors. For the focal length, the relative error of 100 pixels is about 3.2% of the ground truth.

First, the running time of CalibMe increases linearly with the number of reference images. Because each narrow FOV image covers a small part of the stadium, CalibMe requires a large number of reference images to produce a good result. On the other hand, our pan-tilt forest groups nearby rays in the same leaf node, effectively encoding a large number of reference images. Second, our method requires fewer points (2 vs. 4) to estimate the initial camera parameters than CalibMe. Thus, it needs fewer iterations to find the optimal solution in RANSAC. Our method has a relatively low IoU score in the sequence 3. We found that the main reason is that the sequence 3 has a larger range of pan angles and some of them are not covered by the training data.

Table 3.3 shows the results on the World Cup dataset. Our method achieves very high IoU scores on this dataset because the images in this dataset cover larger areas than those in the highlights dataset.

	$\text{Inlier}_{0.5}$	Mean IoU
w/o threshold	0.09	0.53
w/ threshold	0.33	0.83
improvement	+0.24	+0.30

Table 3.4: Influence of using the feature distance threshold. $\text{Inlier}_{0.5}$ means the inlier percentage of pan-tilt predictions that has angular error less than 0.5 degrees.

Figure 3.7 shows the errors of our method measured by rotation error (degrees) and focal length error (pixels), separately. The rotation errors of more than about 85% of images are less than 1 degree. Our method has small focal length errors in both datasets, with the good performance being especially pronounced on the World Cup dataset.

Figure 3.6 shows qualitative results of our method. In most cases, our method can calibrate pan-tilt-zoom cameras from very challenging camera angles.

Further Analysis. Table 3.4 shows the influence of with and without the feature distance thresholding in pan/tilt angle prediction. Our method removes outliers using a simple feature distance threshold, resulting in a much higher (0.33 vs. 0.09) percentage of inliers (rotation error less than 0.5 degrees) in pan/tilt angle prediction. The accurate pan/tilt prediction provides a significantly higher (0.83 vs. 0.53) mean IoU score.

Figure 3.8 shows the IoU measurement as a function of the field of view values on both datasets of our method. When the FOV values are smaller than about 25°, our method has more incorrect estimates (red crosses) whose IoUs are smaller than a threshold (0.6). When the FOV values are larger than about 40°, our method has no incorrect estimates on these two datasets. This result suggests that the narrow field of view still causes incorrect estimates.

Implementation details. Our approach is implemented with C++ on an Intel 3.0 GHz CPU, 16GB memory Mac system. For the pan-tilt forest, the number of trees is 5 and the maximum depth is 20. In the test, the computation of SIFT feature costs about 0.2 seconds/frame. The pan/tilt prediction and camera pose optimization cost about 0.1 seconds/frame. At the current implementation, we do not optimize the speed which can be improved by using a GPU SIFT implementation and a small number of iterations in RANSAC . The code is available online[1].

[1] https://github.com/lood339/two_point_calib

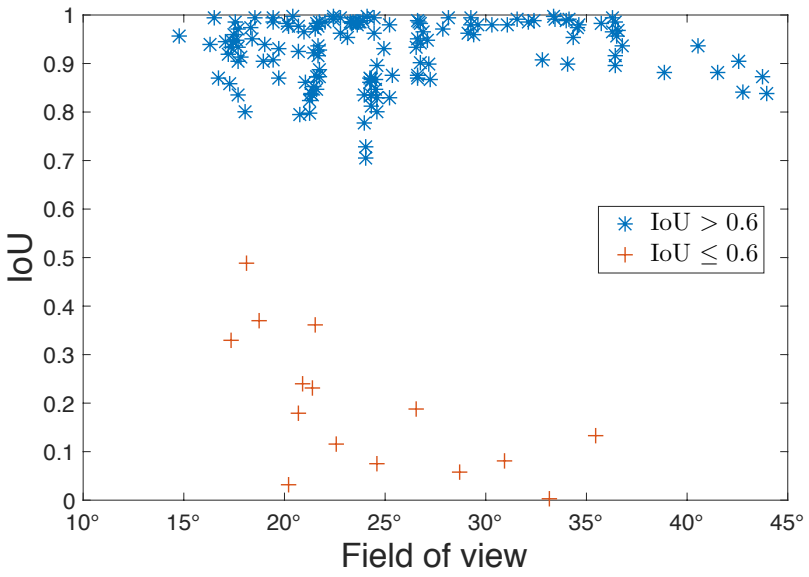


Figure 3.8: IoU as a function of the field of view values.

Limitations. Conversely, there are several limitations of this work. First, it assumes the camera base parameters are available and fixed. However, these parameters may change from one game to another. In that case, fully automatic methods [WCW+16, HFU17] can be used to estimate the camera base parameters before using our method. Second, our method uses a real-valued feature descriptor which is computationally expensive in running time. Third, our method assumes there is sufficient textured background in the image. This assumption may not hold when cameras zoom in to a small area of grass.

3.3 Synthetic edge image method

The two-point method relies on point features which are not invariant in different stadiums. To overcome this limitation, we develop a calibration method that uses synthetic edge images. The edge image is the projection of sports field templates.

The new method models the problem as the nearest neighbor search problem. The method first builds a camera pose database from synthetic edge images. Then, it queries the camera pose using the edge image of a testing image. Our method is inspired by [SBGJ18] which formulates camera calibration as a nearest neighbor search problem over a synthetically generated dictionary of edge images. However,

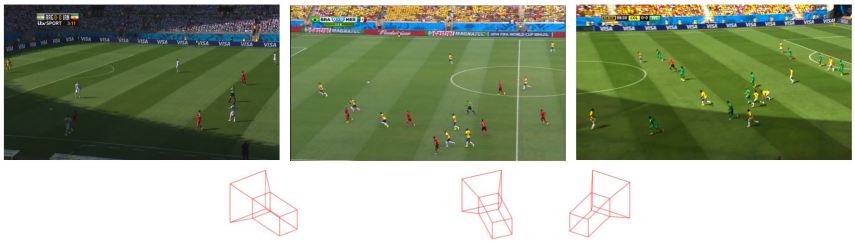


Figure 3.9: Image examples from the WorldCup dataset [HFU17]. Although these images are from different stadiums and have very different light conditions, all of them have a strong prior of the camera locations (roughly shown by red cameras).

our edge image generation process is more straightforward. Moreover, our method produces more compact features and achieves higher calibration accuracy.

We approach this problem from the prior knowledge of camera locations. Figure 3.9 shows image examples from the WorldCup dataset [HFU17]. When we look at these images, the first observation is that field markings (e.g., lines and circles) must be used in the calibration as they are the visible evidence of camera poses. The second observation is that the camera poses have some "default" settings. For example, the camera roughly locates near and above the middle lines. The camera has a large range of pans (from left to right) and a small range of tilts (from top to down). The second observation highly motivates our method.

When the rough location is available, the degrees of freedom (DoF) of the camera is 4 (3 for rotation and 1 for focal length). As a result, the edge image can be represented in a low dimensional space because it is fully determined by the camera pose and the template of the field. To achieve this goal, we propose to learn features of edge images via a siamese network [WL15]. The learned feature is much more compact and more effective than hand-craft features such as HOG or at least on par. With compact features, we can build a database of camera pose and edge image pairs for fast nearest neighbor search.

In testing, we first detect field markings. Then, we estimate the initial camera pose via nearest neighbor search. The initial camera pose is not perfect in general as the database has a limited number of camera poses. A straightforward solution is to refine the camera pose by warping the nearest neighbor edge image to the observed edge image via Lucas-Kanada (LK) algorithm [BM04]. However, directly applying LK warp does not work well because edge images have narrow ranges of gradients. To overcome this problem, we propose to use the distance transform [FH12] to efficiently approximate the long-range gradient so that LK warp con-

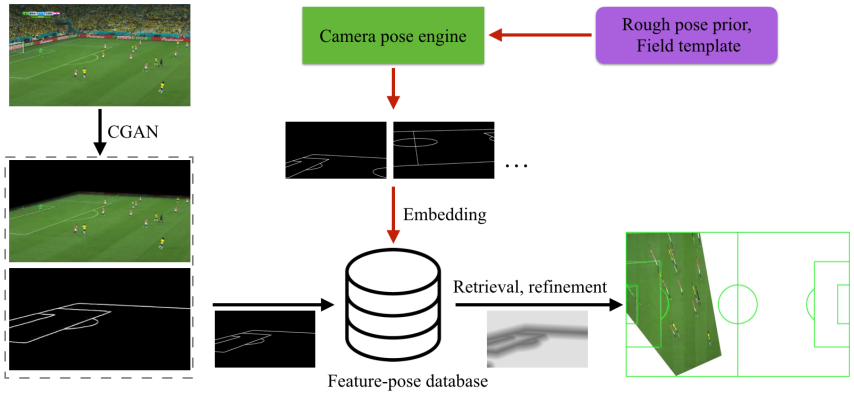


Figure 3.10: Synthetic edge image method pipeline. Our method has two phases. In off-line phase (red arrows), we use a camera pose engine to synthesize a set of edge images. The edge images are embedded into a low dimension space. In this way, we obtain a feature-pose database. In the test phase (black arrows), we use conditional generative adversarial networks (CGAN) to detect field markings in an image. The camera pose of the testing image is quickly retrieved in the database and is further refined using the distance image.

verges well.

Figure 5.8 shows the pipeline of our method. It first builds a feature-pose database. In the database, the camera pose is generated by a pose engine which takes camera priors and the field template as input. The feature is learned from a siamese network. In testing, the method segments field areas (e.g., grassland in soccer) from the image to robustly extract edge images. Then, a nearest neighbor camera pose is quickly retrieved from the database. Finally, the camera pose is refined by distance images.

3.3.1 Method

Camera model. We decompose the base rotation $\mathtt{S}$ in (3.1) to

$$\mathtt{S} = \mathtt{S}_{\theta'}\mathtt{S}_{\rho'}\mathtt{S}_{\phi'}, \tag{3.7}$$

where ϕ', ρ', and θ' are tilt, roll and pan angles of the camera base. In the two-point method, we use all angles of $\mathtt{S}$ together because it is the known input of that method. Here, we decompose $\mathtt{S}$ into three angles to simplify the camera generation process. When we put (3.7) into (3.1), $\mathtt{S}_{\theta'}$ can be eliminated because composition

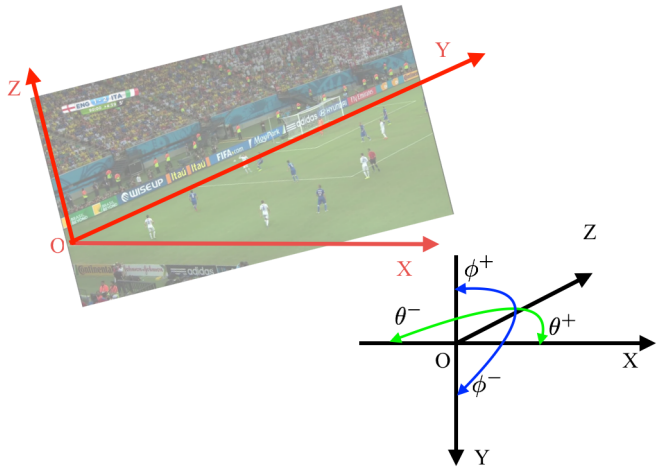


Figure 3.11: World (red) and camera (black) coordinate systems.

effect of $\mathtt{Q}_\theta\mathtt{S}_{\theta'}$ can be represented by $\mathtt{Q}_\theta$ with $\theta = \theta + \theta'$. Thus, the camera model becomes

$$\mathtt{P} = \underbrace{\mathtt{K}\mathtt{Q}_\phi\mathtt{Q}_\theta}_{\text{PTZ}}\underbrace{\mathtt{S}_{\rho'}\mathtt{S}_{\phi'}[\mathtt{I}|-\mathbf{C}]}_{\text{prior}}. \tag{3.8}$$

Without loss of generality and for easy explanation, we set the coordinate system as in Figure 3.11. The world origin is at the left bottom of the field template. When pan and tilt are zeros, the camera looks along the Y-axis of the world coordinate.

In (3.8), we want to minimize the number of free parameters to reduce the dimension of sample space. We set $\phi' = -90^\circ$ with which cameras are set up to be "level" because the effect of $\mathtt{S}_{\phi'}$ will be canceled by $\mathtt{Q}_\phi$. ρ' varies in a small range ($\pm 0.1^\circ$) because the camera base prevents the camera from rotating about its direction-of-view. As a result, the free parameters of the projection matrix become f, ϕ, θ and $\mathbf{C}$. For sports fields, $\mathbf{C}$ is further constrained in practice, for example, most cameras are above and along the center line for soccer games. With this model, we can generate an arbitrary number of camera poses and corresponding edge images.

Learning edge image feature. Given edge images and their poses, we learn compact features via a siamese network [HCL06]. The input of the network is edge image pairs. The label is similar or dissimilar. A pair of edge images is set as similar if their pan, tilt and focal length differences are within pre-defined thresholds,

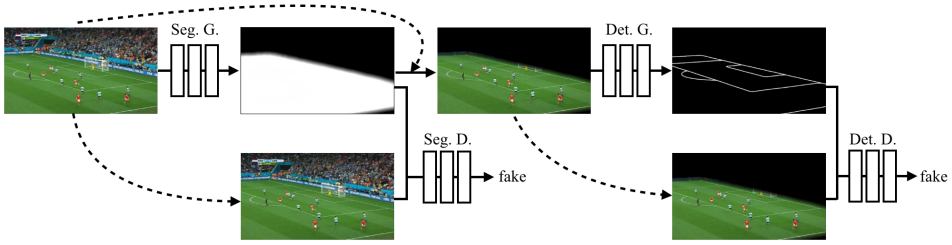


Figure 3.12: Two-GAN model. We use a two chained conditional GAN to detect field markings. "Seg." and "Det." are short for segmentation and detection, respectively. "G." and "D." are short for generative network and discriminative network, respectively.

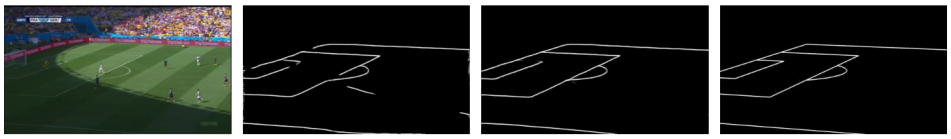
and vice versa.

We have tried many structures of CNNs for the siamese network. We found max-pooling and batch normalization has a slightly negative impact on the performance, which agrees with [DBKK16]. The network consists of 5 stride-2 convolutions (kernel size 7, 5, 3, 3, 3) followed by a 6×10 convolution and a $L2$ normalization layer. The learned feature dimension is 16.

Field marking detection. We use two conditional GANs (Chapter 2.1.3) to detect field markings from testing images. We first use a segmentation GAN to segment the playing surface from the whole image. Then, we use a detection GAN to detect field markings from the playing surface. The motivation of using two GANs is to avoid the negative influence of background objects (e.g., white lines on commercial boards) for the generative model of GANs.

In our problem, the GAN generates field markings from a soccer game image: "segmentation" and "detection" are named from the two sub-tasks in the GAN. The first task distinguishes grassland pixels from non-grassland pixels, which can also be described as "segmenting" the grassland from the image. The second task distinguishes field marking pixels from non-field-marking pixels in the segment result. The name of "detection" is from traditional detection tasks such as line detection and circle detection. In practice, the lines are "generated" by the GAN and the result is better than traditional detection methods. For example, it is hard for a traditional detection method to detect a line that is occluded by players but GANs can recover this line to some extent.

Figure 3.12 shows the pipeline of our two-GAN method. The top row shows the pipeline of the generative networks. The first GAN (segmentation GAN) segments foreground (e.g., grassland in soccer games) areas from the input RGB image and



(a) (b) (c) (d)

Figure 3.13: An example of field marking detection. (a) input; (b) using one GAN [SBGJ18]; (c) ours; (d) ground truth.

outputs a mask image. We obtain a foreground image using the mask image and the input RGB image. The second GAN (detection GAN) detects field markings from the foreground image. Each GAN has a discriminator network. The segmentation discriminator predicts whether the mask image is real or fake. The detection discriminator predicts whether the edge image is real or fake.

We first train the segmentation GAN and detection GAN independently. Then, we train the whole network end to end. In training, we found the segment boundary has considerable influence on the detection. When a binary segmentation boundary is used, the detection GAN tends to memorize the boundary and that will cause artifacts in testing. We overcome this problem by using a soft (alpha-blending) boundary. The hard foreground is 1 and the hard background is 0, with linearly interpolated values between them. We randomly set the width of the alpha-blending band in the range of $[30,50]$ pixels to prevent the detection GAN from memorizing the bandwidth. Figure 3.13 shows an example of field marking detection, in which our method produces fewer artifacts than alternative methods.

Distance transform and camera pose optimization. Given the detected edge image, we first extract its feature using our siamese network. Then, we estimate its initial camera pose by retrieving the nearest neighbor in the database. The initial camera is not accurate because the database has a limited number of camera poses. To refine the camera pose, we first obtain two distance images from the edge images (one is from testing image and another is from the nearest neighbor camera pose) via distance transform. Then, we estimate the homography matrix from the nearest neighbor distance image to the testing distance image.

In a distance image, each pixel has a real value which is the shortest distance to edge pixels. We found a truncated distance transform works best. Let $\mathscr{G}$ be a regular grid and $f : \mathscr{G} \to \mathscr{R}$ a function on the grid. P is a set of points defined by a binary edge image. The truncated distance transformation of point sets is:

$$D_P(p) = \min_{q \in \mathscr{G}}(d(p,q), 1(q)), \tag{3.9}$$

(a) (b) (c)

Figure 3.14: Camera pose retrieval and refinement results. (a) retrieval; (b) retrieval + LK warp using edge images; (c) retrieval + LK warp using distance images (ours).

where $1(q)$ is an indicator function for membership in P:

$$1(q) = \begin{cases} 0 & \text{if} \quad q \in P, \\ \tau & \text{otherwise.} \end{cases} \tag{3.10}$$

τ is the truncation threshold. When $d(.)$ is measured in $L2$ norm, a linear-time algorithm is available [FH12].

With the distance image, we use the Lucas-Kanada algorithm [BM04] to estimate the homography matrix from the nearest neighbor image to the testing image. Figure 3.14 visualizes an example of initial and refined camera poses. In this example, the LK algorithm with distance images significantly improves the camera calibration accuracy. On the other hand, LK with edge images fails because the LK algorithm cannot converge with the narrow range of gradients.

3.3.2 Experiments

We evaluate our method on the World Cup dataset from [HFU17].

Error metric. We use the intersection over union (IoU) score to measure the calibration accuracy of different methods. The IoU is calculated by warping the projected model to the top view by the ground truth camera and the estimated camera. [HFU17] measures the IoU on the whole area of the model, while [SBGJ18] measures the IoU only on the area that is visible in the image. We denote these two metrics by IoU_{whole} and IoU_{part}, respectively. For a fair comparison, we report both metrics for our method.

Camera pose engine parameters. The camera pose engine parameters are estimated from the training data. We first estimate the camera location distribution $\mathcal{N}(\mu, \sigma^2)$ ($\sigma \approx \pm[2.2, 9.3, 2.9]^\mathsf{T}$ meters). Then, we randomly generate 100k

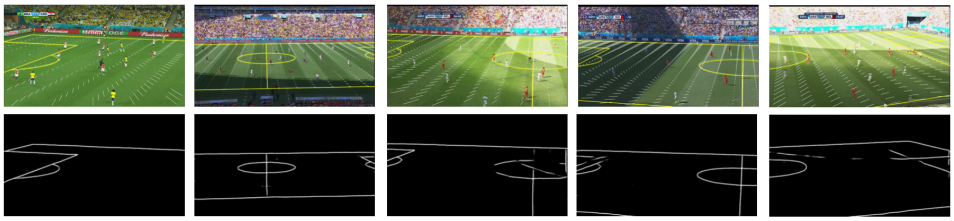


Figure 3.15: Qualitative results.

camera poses by following settings: (1) The camera centers are sampled from $\mathcal{N}(\mu, \sigma^2)$; (2) The pan, tilt and focal length values are in the range of $[-35°, 35°]$, $[-15°, -5°]$ and $[1000, 6000]$ pixels, respectively; (3) the tilt of camera base ϕ' is fixed ($-90°$) and the roll angle is a random value from $[-0.1°, 0.1°]$.

Method	IoU_{whole}		IoU_{part}	
	Mean	Median	Mean	Median
DSM [HFU17]	83	–	–	–
Dict. + HOG [SBGJ18]	–	–	91.4	92.7
Ours	**89.4**	**93.8**	**94.5**	**96.1**

Table 3.5: Comparison on the soccer dataset.

Table 3.5 shows the mean and median IoU measurement on the soccer dataset. Our method is significantly more accurate (89.4 vs. 83) than the DSM method [HFU17]. It is also more accurate (94.5 vs. 91.4) than [SBGJ18] and has a much simple and interpretable camera pose engine.

Table 3.6 shows the component analysis of our method. It shows that both the segmentation GAN and the LK warp contribute to the final result. On this dataset, it is interesting to see that the HOG feature is as good as our method in accuracy but much less compact (1,860 vs. 16). It shows that HOG is a strong baseline on this dataset.

Method	Mean IoU_{whole} (%)	Improvement
Ours	**94.5**	–
w/o segmentation	90.2	4.3
w/o LK warp	91.5	3.0
w/o segmentation and LK warp	88.9	5.6
deep feature $\rightarrow$ HOG	94.5	0.0

Table 3.6: Components analysis on the soccer dataset.

Qualitative results. Figure 3.15 shows the qualitative results of detected field markings and estimated camera poses. On the soccer dataset, the heavy shadow is the main reason for incorrect field marking detection. However, our method can estimate camera poses in these difficult situations.

Implementation details. Our approach is mainly implemented in Matlab on an Intel 3.0 GHz CPU, 16GB memory Mac system. In the current implementation, the speed is not optimized. In testing, the running time is about 0.5 seconds/frame.

Our two-GAN field marking detection implementation is based on the pix2pix network [IZZE17] (PyTorch version). Both generator and discriminator use modules of the form convolution-BatchNorm-Relu. The generator uses skip connections and follows the shape of a U-Net [RFB15]. In training, we set the weight of L1 loss $\lambda = 100$. We alternate between one gradient descent step on D, then one step on G. The losses of the segmentation GAN and the detection GAN have equal weights. We set the batch size to 1 because of limited GPU memory (12GB). The number of epochs is 200 and the learning rate linearly decays from 0.0002. We augment the training set by first resizing the image to 300×300 then randomly cropping it to 256×256. The two-GAN code is available online[2].

In our method, the parameters are not sensitive to the final result. For example, we found 30-50 pixels work well for the distance parameter τ in (3.10). For the siamese network, the input dimension is 180×320 and output dimension is 16. For the positive camera poses, the thresholds of pan, tilt and focal length differences are 1°, 0.5° and 30 pixels, respectively.

Limitations Currently, our method still requires a small number of human annotations. In the future, we would like to develop fully automatic camera calibration methods.

3.4 Summary

In this chapter, we have proposed two methods for sports camera calibration. Both methods take advantage of the camera location information to simplify the problem. The first method requires only two point correspondences, overcoming the challenge of the narrow field of view camera calibration. The second method builds a synthetic camera pose database for accurate camera calibration from edge images. Both methods have achieved the state-of-the-art accuracy on challenging soccer datasets.

[2] https://github.com/lood339/pytorch-two-pix2pix

Chapter 4

Camera Planning

4.1 Introduction

In this chapter, we are interested in the task of predicting camera angles of the main PTZ camera. The key idea is learning from human demonstrations in which training labels are from human operator's generated videos.

Figure 4.1 gives an overview of camera planning for basketball games. The input is player positions on the playing ground and the output is pan angles of the PTZ camera. We want the predicted camera views not only to capture interesting events of the game, but also to maintain a smooth trajectory over time.

In preliminary work [CC15], we modeled camera planning as a structured regression problem

$$\hat{y}_t = h(\mathbf{x}_t), \tag{4.1}$$

where $\hat{y}_t$ is the planned pan angle of the camera for a particular time, $h(\cdot)$ is the learned regressor, and $\mathbf{x}_t$ is a feature vector extracted from the current player tracking data. To learn the regressor, the work of an expert human camera operator is analyzed to generate exemplar pan angles $\{y_t\}$ for the observed tracking features $\{\mathbf{x}_t\}$. Using these paired data $\{(y_t, \mathbf{x}_t)\}$, we first use a random forests regressor to generate a suitable camera planning algorithm $h(\cdot)$. Then, we use a first-order Savitzky-Golay (SG) filter [SG64] to smooth the predicted pan angles.

Based on this method, we have developed two methods to predict smooth pan angles. The first method directly incorporates temporal consistency into a data-driven predictor. The input is a state which includes recent features and predictions. The temporal consistency is forced by an independently trained autoregressor. In training, the method iteratively updates the input (i.e., the state) in a decision tree framework so that we denote it recurrent decision tree (RDT). The second method

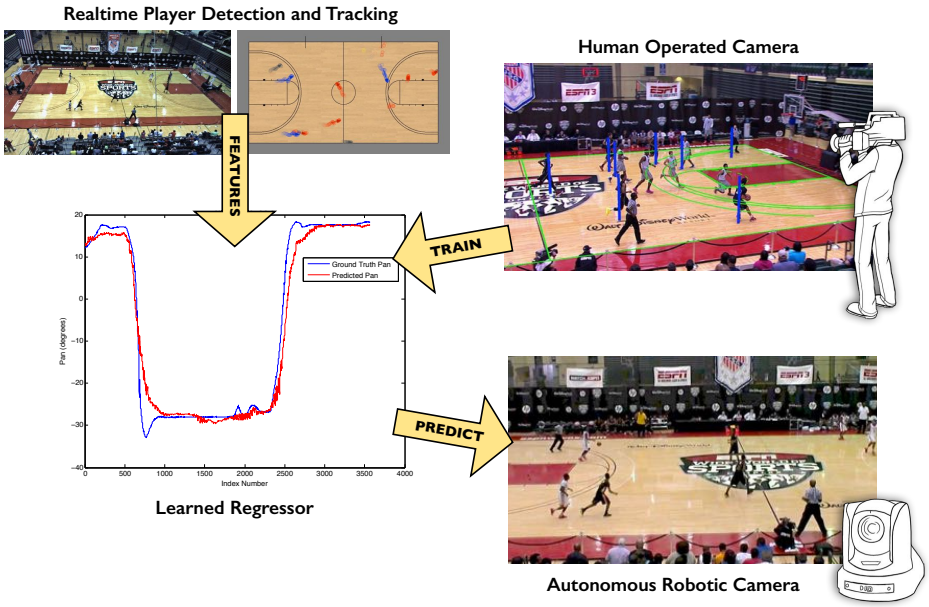


Figure 4.1: Camera planning. Using a stationary machine vision camera, we extract a fixed length feature vector from tracking data to describe the game state. Meanwhile, we estimate the pan-tilt-zoom configuration for each video frame of a human operated camera. Using these data, we train a regressor to model the relationship between camera pan angle and game state. At test time, we use the regressor to generate target pan angles for an autonomous robotic camera based on the input tracking data.

optimizes the camera trajectory in overlapped temporal windows. It estimates the most probable camera angles by repeatedly using the hidden Markov Model (HMM) framework. We denote it overlapped hidden Markov model (OHMM).

4.2 Recurrent decision trees

This method incorporates temporal consistency into a data-driven predictor given noisy detections. The input of the RDT method is a sequence of the player locations (current and previous) and previous predictions. The output of the RDT method is the predicted camera pan angle for the current time.

4.2.1 Method

Let Π denote a class of policies that our learner is considering. Given a stream of inputs $\mathbf{x} = \langle x_1, \ldots, x_T \rangle$, each $\pi \in \Pi$ generates a stream of outputs $\mathbf{y} = \langle y_1, \ldots, y_T \rangle$. Here, y_i is the absolute pan angle (not the change in pan).

We first extend the input from a single frame feature x_t to a feature *state* $s_t = \{x_t, \ldots x_{t-\tau}, y_{t-1}, \ldots, y_{t-\tau}\}$ which is composed of the recent inputs and predictions. The state makes predictions conditioned on its previous predictions (i.e., y_{t-1}), which allows it to learn temporal patterns within the data (in addition to any direct feature-based relationships). The output is a pan angle $y_t = \pi(s_t)$.

We instantiate the policy using decision trees (Chapter 2.1.2). A decision tree specifies a partitioning of the input space (i.e, the space of all possible states s). Let $D = \{(s_m, y_m^*)\}_{m=1}^M$ denote a training set of state/target pairs. Conventional regression tree learning aims to learn a partitioning of the data such that each leaf node, denoted by `node`, makes a constant prediction to minimize the squared loss:

$$\bar{y}_{\texttt{node}} = \arg\min_y \sum_{(s,y^*) \in D_{\texttt{node}}} (y^* - y)^2, \tag{4.2}$$

where $D_{\texttt{node}}$ denotes the training data from D that has been partitioned into the leaf `node`.

Using the state as input can improve temporal consistency (i.e., smoothness). However, that smoothness purely depends on data and will not be maintained when the input has abrupt changes, for example, the number of detected players suddenly drops. In practice, camera movement is constrained by inertia so that the camera generally has a smooth trajectory. Thus, we add a smoothness term in the decision tree.

$$\hat{y} \equiv \pi(s) \equiv \arg\min_y \ (y - \bar{y}_{\texttt{node}(s)})^2 + \lambda (y - f_\pi(s))^2, \tag{4.3}$$

where $\texttt{node}(s)$ denotes the node of the decision tree that s branches to, and $\lambda \geq 0$ trades off between $\hat{y}$ matching the target signal $\bar{y}_{\texttt{node}(s)}$ versus the smooth autoregressor $f_\pi(s)$.

We use a regularized linear autoregressor as $f_\pi(s)$. Let $f_\pi(y_{-1}, \ldots, y_{-\tau})$ denote an autoregressor of the temporal dynamics of π over the distribution of input sequences $d_{\mathbf{x}}$, while *ignoring* the exogenous inputs x. In other words, at time step t, f_π predicts the behavior $y_t \equiv \pi(s_t)$ given only $y_{t-1}, \ldots, y_{t-\tau}$.

In general, it can be difficult to exactly solve (4.3) due to the circular dependency between the distribution of states and the policy under consideration. We developed an iterative training procedure that alternates between estimating states over a fixed policy and optimizing the policy over fixed states [CLC+16].

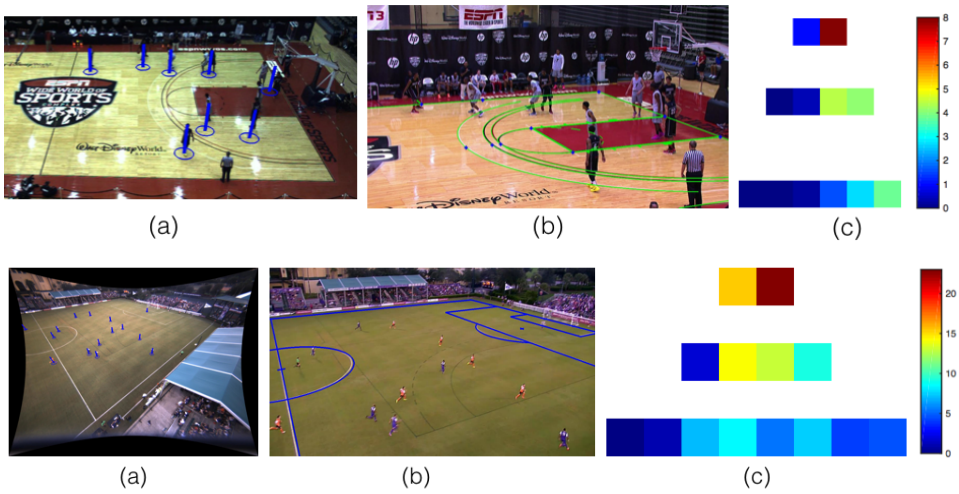


Figure 4.2: Features and labels. (a) player detections, (b) pan/tilt/zoom parameters, and (c) spherical quantization scheme for generating features. In the feature, the unit is the number of players.

4.2.2 Experiments

Basketball dataset. The basketball dataset is from a high school basketball game. The match was recorded by two cameras. The broadcast camera (1280×720 at 60 fps) is operated by a human expert. The observation camera (1936×1456 at 25 fps) remains stationary so that a computer can detect and track players automatically. We manually segmented the recorded video to remove all non-continuous periods of play (such as timeouts and free throws). It is about 32 minutes of data divided into roughly 50 segments (each about 40 seconds long), with two held out for validation and testing. The testing sequence is about 80 seconds long and has dynamic camera pan angles.

We detect/track players within the video of the stationary camera using the method from [CSM12a, LCCL13]. Unlike [CDDV11], we do not have an estimate of the players' identities. Player identities can potentially improve the performance of camera planning because the algorithm can use more specific information, for example, the trajectory of a famous player. However, player identification (or re-identification) requires much more computation and is not always available, for example, when players are very small in images. For convenience, we linearly interpolated the tracking data to 60 fps. We estimate the camera parameters (pan, tilt and focal length) from the broadcast camera using the method described in [CC15].

Soccer dataset. The soccer dataset is from a semi-professional soccer match. It was recorded using three cameras: two stationary cameras near the floodlights for player detection, and a robotic PTZ located at mid-field remotely operated by a human expert. The videos were synchronized at 60 fps. About 91 minutes was used for training, and two 2 minute sequences were held out for validation and testing.

The soccer players were detected/tracked from the two stationary cameras using a method similar to [CSM12a, LCCL13]. The camera parameters were estimated using the CalibMe system described in [CL17]. We also focus on predicting an appropriate pan angle on the soccer data. The ground truth pan angle is smoothed using a Savitzky-Golay (SG) filter [SG64].

Features and labels. The ground locations of players were determined from 3D geometric primitives which best justified the background subtraction results using the method of [CSM12a]. Each ground position was projected to a spherical coordinate system centered and aligned with the broadcast camera. Because the number of detections varied due to clutter and occlusions, a fixed length feature vector was constructed using spatial frequency counts. Because the goal is to predict a pan angle for the PTZ camera, there is an inherent non-linear spherical projection between the world coordinate system and the pan-tilt-zoom domain of the camera. Therefore, we generated a heat map $\mathbf{x}_t$ on the unit sphere of the PTZ camera by projecting the pan angles. As a result, the spherical map is generated for resolutions 1×2, 1×4, and 1×8, resulting in a 14 dimensional feature vector $\mathbf{x}_t$ [CC15]. We denote this feature by spherical pan map (SPM). Because the tilt and zoom of the camera do not vary significantly over the dataset, we use the pan angles of the broadcast camera as labels. Figure 4.2 shows examples of features and labels for the basketball and soccer datasets.

Evaluation. We test the RDT method on the basketball and soccer datasets. We also compare the RDT method with five baselines: Savitzky-Golay [CC15], Kalman Filter, Dual Kalman Filter, Conditional regression forests [DGFVG12] and filter forests [FKK$^+$14]. We tune the parameters of our method and the baselines using the validation sequence. For our method, an important parameter is the λ in (4.3). A small λ allows more accurate but jittery predictions, and a large λ leads to smoother but less accurate predictions. We found $\lambda = 300$ is a good trade-off between accuracy and smoothness.

Figure 4.3 shows the benchmark performance evaluated for both basketball and soccer. We evaluate using joint loss which is the sum of the squared angular difference and the squared velocity (weighed by 1 vs. 500). The precision and

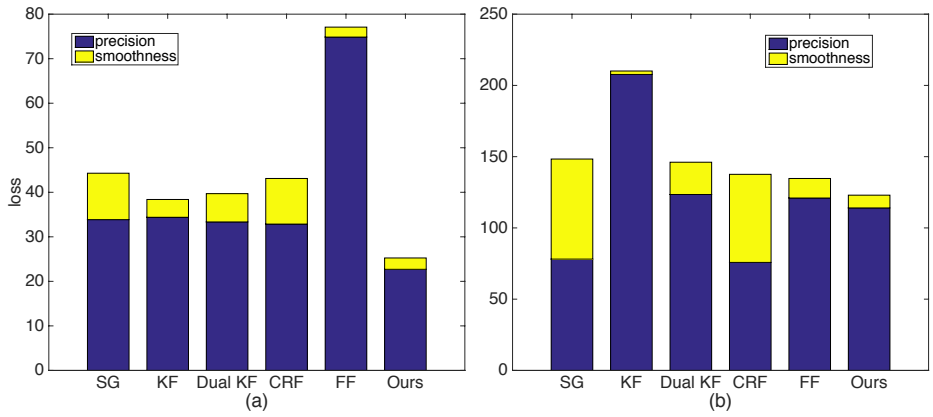


Figure 4.3: Prediction loss. (a) Basketball; (b) Soccer. The precision is measured by the squared angular prediction error. The smoothness is measured by the squared velocity (weighted by 500).

smoothness losses are plotted separately to illustrate their relative contributions to the joint loss. For both settings, we see that our approach achieves the best performance, with the performance gap being especially pronounced in basketball.

We also observe that our approach achieves very low smoothness loss, despite not utilizing a post-processing smoothing step. For instance, in the basketball setting, our approach achieves the smoothest performance. In fact, for the basketball setting, our approach dominates all baselines in both imitation loss and smoothness loss. The KF and FF baselines tend to achieve competitive smoothness loss, but can suffer substantial imitation loss.

We note that the soccer setting is significantly more challenging than basketball, and no method performs particularly well for soccer. One possible explanation is that soccer camera planning using only player detections is unreliable due to players not following the ball (unlike in basketball). A visual inspection of the generated videos also suggests that lack of ball tracking in the input signal $\mathbf{x}$ is a significant limitation in the soccer games.

Visual inspection. Figures 4.4 and 4.5 show a visualization of the predictions. From this visual inspection, we can verify qualitatively that our method achieves the best balance of precision and smoothness. Our predicted trajectory remains close to the human operator's trajectory and has less jitter than the other methods. Even with post-smoothing, SG and CRF exhibit significant jitter. KF struggles between jitter and overshooting when the noise is not Gaussian. Surprisingly, dual

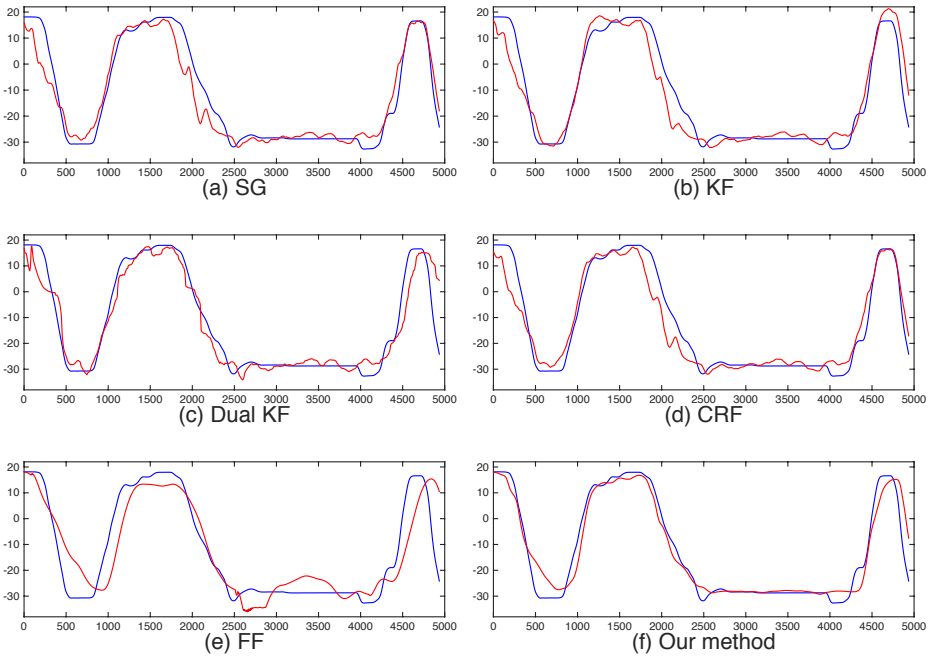


Figure 4.4: Comparison on basketball data. (a) Method of [CC15], (b) Kalman filter, (c) Dual Kalman filter , (d) Conditional regression forests [DGFVG12], (e) Filter forests [FKK+14], (f) Our method. The horizontal axis is the frame number and the vertical axis is the pan angle (measured in degrees). The blue line is from human operators and the red line is from predictions. Note our method does not need any post-processing.

KF performance is worse than KF, which is again presumably because the noise is not Gaussian, and errors in the process estimation propagate to the state estimation. FF is very competitive in the basketball dataset, but its performance suffers from large jitter in the more challenging soccer dataset. We also generate videos[1] by resampling the captured video from predicted camera angles.

User study. We also conducted a user preference study to complement our benchmark experiments. Videos were generated by warping the video captured by the human operator. Figure 4.6 shows our user study interface. We evaluated our approach against the five baseline algorithms for both basketball and soccer. Par-

[1] https://youtu.be/JW9ffEILQKE, left: our method, right: SG.

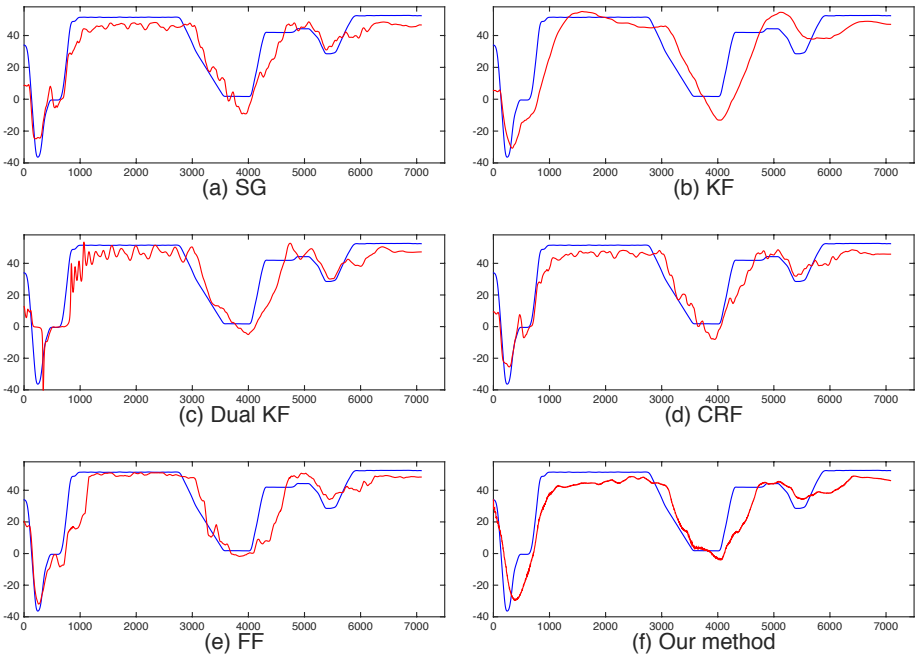


Figure 4.5: Comparison on soccer data. (a) Method of [CC15], (b) Kalman filter, (c) Dual Kalman filter , (d) Conditional regression forests [DGFVG12], (e) Filter forests [FKK+14], (f) Our method. The horizontal axis is the frame number and the vertical axis is the pan angle (measured in degrees). The blue line is from human operators and the red line is from predictions. Note our method does not need any post-processing.

ticipants are from Amazon Mechanical Turk. We do not ask if they are sports fans but we set a high threshold for their annotation performance. Each comparison has 25 trials and the total number of trials is 300. In each trial, participants were instructed to choose the video that was more appealing (our method was randomly assigned the left or right view).

Table 4.1 shows the results. For basketball, our method is significantly preferred over the other methods based on a two-tailed binomial test ($p < 0.001$). For soccer, none of the methods performed particularly well, making it challenging for users to judge which method generated better videos. Note that there is still a sizable preference gap compared to the human expert.

Figure 4.6: User study screenshot. Users were asked which video was more pleasant to watch, and to consider both composition and smoothness in their assessment.

Comparison	Basketball win / loss	Soccer win / loss
Ours vs. SG	22 / 3	14 / 11
Ours vs. KF	23 / 2	12 / 13
Ours vs. dual KF	25 / 0	24 / 1
Ours vs. CRF	24 / 1	12 / 13
Ours vs. FF	23 / 2	14 / 11
Ours vs. human	1 / 24	4 / 21

Table 4.1: User study results. For basketball, our method is significantly preferred over all baselines. For soccer, all methods performed poorly, and users did not have a strong preference. There is still a sizable preference gap compared to expert human.

4.3 Overlapped hidden Markov model

In the soccer game, there are many interesting areas at one time because the players are sparsely located on the soccer field. When there are multiple choices of camera angle, the cameraman chooses one of them according to the previous states of the camera and the prediction of the future evolution of the game. In the data, the same formation of players may have multiple possible suitable pan angles — i.e., $h(\mathbf{x}_t) \mapsto \{y_t, y_t', y_t'', \dots\}$. As a result, $h(\mathbf{x}_t)$ is not strictly a single-valued function but a multivalued function. The multivalued function raises ambiguities in the prediction, resulting in jittery camera trajectories.

To reduce jittery from the multivalued function, we propose a new method that

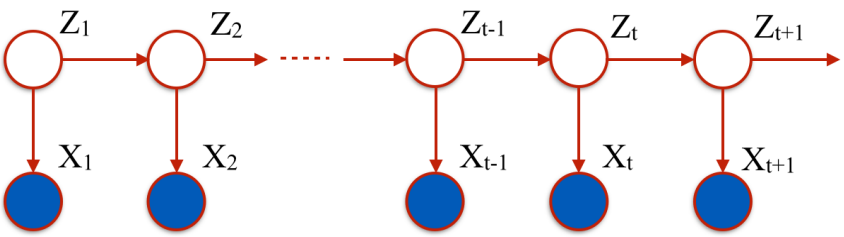


Figure 4.7: The graphical model of an HMM. We can represent sequential data using a Markov chain of latent variables, with each observation conditioned on the state of the corresponding latent variables.

optimizes camera trajectory in temporal windows. In this method, we assume that audiences will tolerate a short delay, which is already the case in practice. For instance, there is a seven-second delay in live broadcasting to prevent undesirable material from airing. We believe relaxing the real-time response requirement and constraining the smoothness in the temporal space is an intuitive and effective way to improve the smoothness of the camera trajectory. Guided by this intuition, we optimize predictions in a temporal window.

4.3.1 Method

Our method is based on the hidden Markov model (HMM) [Bis06]. Figure 4.7 shows the graphical model of an HMM in which each observation is conditioned on the discrete state of the corresponding latent variable and each latent variable is conditioned on its previous state. An HMM optimizes the parameter θ of the joint probability distribution over both hidden and observed variables:

$$p(\mathbf{X},\mathbf{Z}|\theta) = \underbrace{p(\mathbf{z}_1|\pi)}_{\text{initial}} \underbrace{[\prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1},A)]}_{\text{transition}} \underbrace{\prod p(\mathbf{x}_m|\mathbf{z}_m,\phi)}_{\text{emission}} \tag{4.4}$$

where $\mathbf{X} = \{x_1,\cdots,x_N\}$ are observations, $\mathbf{Z} = \{z_1,\cdots,z_N\}$ are hidden variables, and $\theta = \{\pi,\mathbf{A},\phi\}$ denotes the set of parameters governing the model. π, $\mathbf{A}$ and ϕ are the initial, transition and emission parameters, respectively. In our problem, player locations are the observations and camera pan angles are the hidden states.

Because previous emission probabilities vanish when the sequence becomes longer, domain knowledge is used to constrain the length of the sequence. For example, natural language processing (NLP) techniques maximize the joint probability distribution in each sentence. For soccer sequences, a trivial way is to partition the sequence into several short sequences (chunks), and apply the Viterbi algo-

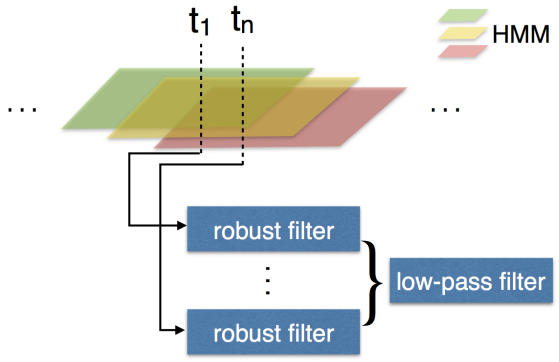


Figure 4.8: Pipeline of overlapped hidden Markov model. First, multiple HMMs are applied to local temporal windows independently. Then, multiple HMMs are overlapped to gather confidential predictions using robust filters. Finally, a low-pass filter further eliminates quantization artifacts.

rithm on each chunk. This simple partition method caused large errors as shown in [NVS06]. The real challenges are how to predict the correct chunk duration and ensure the planned trajectories are continuous across the boundaries between chunks. Hence we propose an overlapping window approach that separates the video into short sequences with overlaps. In this way, we first obtain the optimal path using the Viterbi algorithm on a chunk:

$$\mathbf{z}_t^* = \arg\max_{\mathbf{z}_t} p(\mathbf{x}_t|\mathbf{z}_t,\theta), s.t.\ t = \{t_s,\cdots,t_e\} \tag{4.5}$$

where $\{t_s,\cdots,t_e\}$ is a temporal window.

Then we apply a robust filter (truncated mean using data within 3 standard deviations) to $\mathbf{z}_t$ to get the optimal prediction $\mathbf{z}_t$ from multiple overlapped windows. We denote this method the overlapped hidden Markov model (OHMM) since the optimal value is obtained from a number of temporally overlapped windows. We further use a low-pass filter [SG64] to remove quantization artifacts. Figure 4.8 illustrates the OHMM method. To the best of our knowledge, we are the first ones to study OHMM and apply it to smooth noisy predictions.

In camera planning, the hidden Markov model (4.4) has to estimate initial, transition and emission probabilities from the training data. For transition probabilities, we use the normalized histogram of frame-to-frame velocity (see Figure 4.9 (a)). For the emission probability, we use a discriminative model (random forests). Also, we use the emission probability to approximate the initial probability.

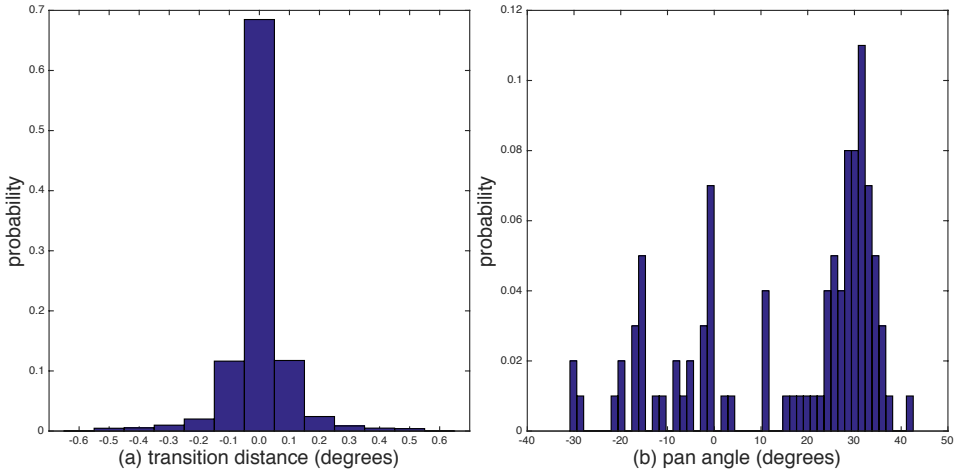


Figure 4.9: Hidden Markov model parameter examples. (a) Transition probability, (b) An example of emission probability.

The random forest is used to estimate the emission probability. It has multiple, typically up to hundreds, of decision trees. In our case, we first train a random regression forest to predict pan angles at current time [CC15]. Then, we estimate the emission probability by voting on a histogram using quantized predictions from every regression tree. Figure 4.9 (b) shows an example of the emission probability.

Complexity of OHMM. An HMM has three parameters: the number of hidden states N_h, N_{tr} the number of states that each state can transition to, the window size w. The complexity of an HMM is $O(N_h N_{tr} w)$. An OHMM has an extra parameter which is the overlap ratio r. For a sequence of length L with overlap ratio r, the complexity of OHMM is $O(\frac{N_h N_{tr} L}{1-r})$, where $r \in (0,1)$. Because the complexity increases very rapidly when r is close to 1, we deliberately analyze how to choose the overlap ratio and other parameters in Appendix B.1. A proper setting of parameters can achieve real-time response (with a constant delay).

4.3.2 Experiments

We test the OHMM method on the soccer dataset described in Section 4.2.2.

Evaluation criteria. We use the root mean square error (RMSE) for pan angles ($RMSE_p$) and velocities ($RMSE_v$) as the error measurements. For qualitative comparison, we plot the predicted camera angle trajectory along with the ground truth

trajectory. We also synthesize videos using predicted camera angles to inspect the camera planning result.

We implement six baseline methods for comparison.

Savitzky-Golay. [CC15] learns a predictor using a random forest trained using only current player locations. A Savitzky-Golay (SG) filter smooths the predictions but induces a delay. Our implementation of this method augments the current player locations with previous player locations. This modification makes the instantaneous predictions more reliable as the predictor has additional temporal information.

Kalman Filter. We replace the Savitzky-Golay filter with a Kalman filter employing a constant velocity process model. Parameters were determined using the validation data.

Conditional Regression Forests. Conditional regression forests (CRFs) split the training data into multiple subsets [DGFVG12]. We tested various splitting methods based on camera position and velocity, such as dividing the data into $4, 8$ and 16 subsets of pan angles. We also tried both disjoint sets and joint sets with different overlap ratios. We report the best result from 8 subsets with 50% overlap. The output of the CRF is further smoothed by an SG filter.

Recurrent Neural Network. Recurrent neural network (RNN) allows feedback connections so that the internal memory can process arbitrary sequences of inputs. In the experiment, we have tested many RNN architectures and found that the sequence to sequence learning [SVL14] performs the best. In this method, one training example is a sequence of features and labels. We use a single layer LSTM [HS97] to map the input sequences (features) to output sequences (labels). The number of hidden variables is 64 and the length of each sequence is fixed at 180 (3 seconds). To avoid overfitting, we randomly sample 20,000 sequences from the original sequence. In other words, the overlap ratio of the training sequence is about 10%. We denote this method by RNN.

Recurrent Decision Tree. Recurrent decision trees (Section 4.2) can smoothly predict camera angles without any post-processing. The method jointly optimizes the accuracy and smoothness using an iterative learning algorithm. We denote this method by RDT.

Method	$RMSE_p$	$RMSE_v$	$delay(sec)$
SG [CC15]	8.35	0.079	1.1
KF	14.27	0.080	**0.0**
CRF [DGFVG12]	**8.24**	0.079	1.1
RNN	9.00	0.082	1.5
RDT [CLC$^+$16]	10.70	0.128	**0.0**
OHMM (ours)	9.82	**0.067**	2.1
OHMM (with ball)	7.07	0.069	offline

Table 4.2: Prediction errors and delays. The best performance in each measurement is emphasized. The last row uses offline ball detection method, indicting the potential performance gain if the ball tracking is available.

OHMM (our method). In the OHMM method, the hyperparameters are window size w, overlap ratio r and the number of transitions N_{tr}. They are experimentally set as $N_{tr} = 5$, $w = 256$ and $r = 0.95$. A detailed analysis of these parameters is in Appendix B.1.

Quantitative comparison. Table 4.2 shows the $RMSE_p$ and $RMSE_v$ for the methods. The SG and CRF are the best in terms of correctness. It is not unexpected because both of them directly minimize the $RMSE_p$ in training. However, they sacrifice smoothness so that the trajectory is very jittery even with the low-pass filters (see Figure 4.11). Our method is moderate in $RMSE_p$ and achieves the best smoothness in $RMSE_v$. Our method has about 2 seconds delay which may cause unsatisfactory results such as missing quick player movements.

We also test the performance of our method when the ball position is available. The result shows that the ball position can significantly reduce the prediction error (from 9.82 to 7.07 in $RMSE_p$). It suggests that a real-time ball detection method is valuable for camera planning in soccer games.

To further analyze the performance of our method in different situations, we conducted 10-fold and half-match cross-validation using both of the player-location-only feature *spherical pan* and the player-and-ball-location feature. Figure 4.10 shows the mean absolute error and the standard deviation. We have the following observations. First, predictions with the ball are significantly better (about 24% improvement on average) than those without the ball. Second, our method has large variance even with the ball, indicating the training data is not large enough to cover all the cases that appear in the testing because of the diversity of the play. Third, the precision of half-match cross-validation is only moderately worse than the precision of 10-fold cross-validation. In the game, the host team played more

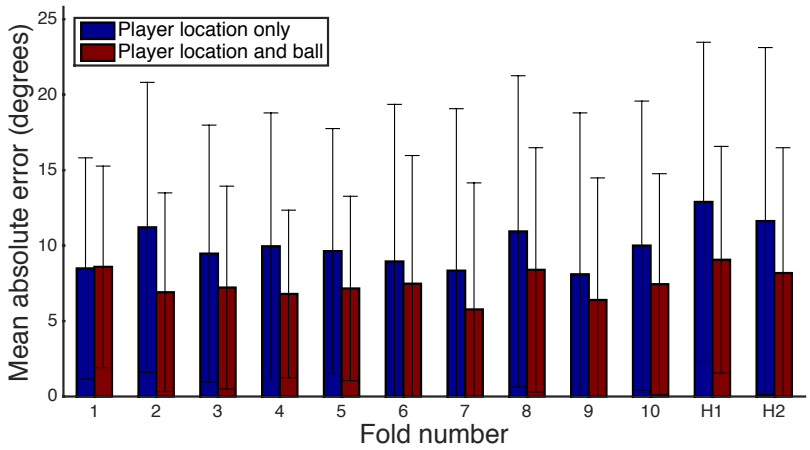


Figure 4.10: 10-fold and half-match cross-validation error and standard deviation. The first 10 groups are the 10-fold cross-validation results. The "H1" and "H2" are the half-match cross-validation results. When the ball tracking is available, the prediction errors (red bars) drop significantly.

aggressively than the guest team and led the game all the time. This result indicates that our method can be applied to multi-game predictions in which the games in the testing may be different from the training games in terms of team formation or tactics.

Qualitative comparison. Figure 4.11 shows the predictions with the ground truth. The predicted trajectory of our method is close to that of the human operator and has less jitter than the other methods. Even with post-smoothing, SG and CRF exhibit significant jitter. KF struggles between jitter and overshooting when noise is not Gaussian. RNN gives moderately good results in both correctness and smoothness. It has less jitter than KF and CRF as it predicts a sequence each time instead of a single frame. The low-frequency noise is eliminated in RDT. However, it still has considerable high-frequency noise. Our method has another advantage that it is easy to train and requires only about 10 percent of the training time compared with the RDT method. We also generate videos[2] by resampling the captured video from predicted camera angles.

[2] https://youtu.be/0Iff_QnqI0M, left: our method, right: human operator.

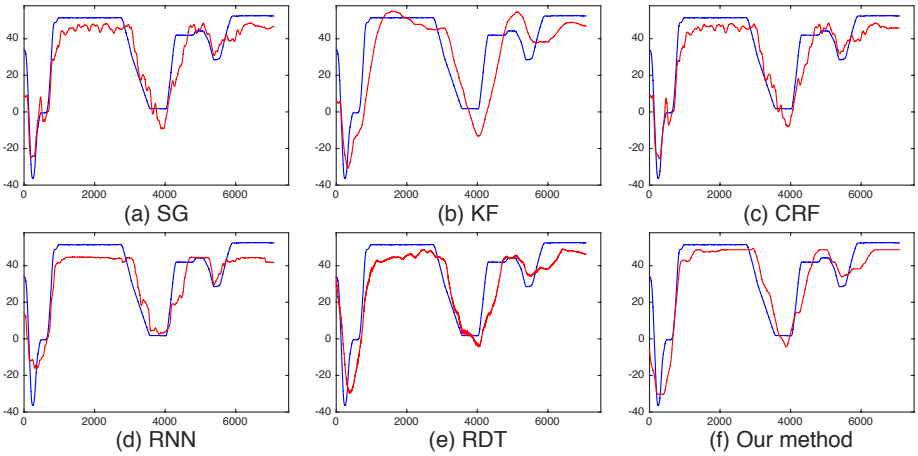


Figure 4.11: Comparison with other methods. (a) Method of [CC15], (b) Kalman Filter, (c) Conditional regression forests [DGFVG12], (d) RNN, (e) Recurrent decision tree, (f) Our method. The horizontal axis is the frame number and the vertical axis is the pan angle (measured in degrees). The blue line is from human operators and the red line is from predictions. Note our method balances correctness and smoothness better than other methods.

Limitations. The OHMM method is designed for smoothing scalar values (e.g., pan angles). It is computationally inefficient for structure predictions, for example, for pan, tilt angles and zoom factors predictions. In that case, we have to develop a 3D Viterbi decoding method. Furthermore, the distances between the pan, tilt and zoom factors are not Euclidean. The capability of extending OHMM to vector smoothness is left as future work.

The method can be improved in several ways. For example, the semi-Markov model [Mur02, Yu10] may be a better alternative to the Markov model in camera planning because each state (i.e., a particular camera angle) has a variable duration in the semi-Markov model and a number of observations being produced while in the state. In term of the description of observations, one can replace the player detection features with deep features from the raw images of the observation camera. Using deep features can simplify the pipeline by removing the player detection part. Also, one can extend the prediction to tilt angles and zoom factors. That will advance camera planning for real applications.

4.4 Summary

In this chapter, we present two methods to predict smooth camera pan angles from player locations. The recurrent decision tree method predicts pan angles in real time. It works well on the basketball dataset. The overlapped hidden Markov model method optimizes the camera trajectory in temporal windows to overcome the artifacts from the multivalued function on the soccer dataset. The latter method has a constant delay which is in the tolerance of broadcasting systems.

Chapter 5

Camera Selection

5.1 Introduction

In this chapter, we are interested in the task of selecting one camera from multiple cameras for broadcasting. The input is multiple synchronized streams of images from candidate cameras. The output is the selected camera ID.

In this chapter, we propose two novel methods for soccer camera selection. The first method is based on regularized visual importance ranking. Visual importance ranks the camera views using the frequency at which viewers select a camera view from all views. The method first predicts the visual importance of each camera view independently. Then, it regularizes the visual importance value to maintain a similar camera duration as human operators. This method is trained on a dataset with multiple camera streams plus the stream of broadcast images as selected by the human producer. The second method augments training data with Internet videos. The purpose of using Internet videos is to mitigate the data scarcity problem that data with all views are not available to researchers. Our method introduces the random survival forest (RSF) method (Chapter 2.1.2) to impute the missing views in the Internet videos so that they can be used as extra data for training. We evaluate these methods on a real soccer game with three candidate cameras. The video that is generated by our method has a comparable preference to the video that is generated by professional operators.

5.2 Regularized visual importance ranking

By analyzing existing systems and watching broadcast videos, we observed that a good camera selection algorithm maintains two important components. First, the selected camera views must provide valuable information about the game [CMM13].

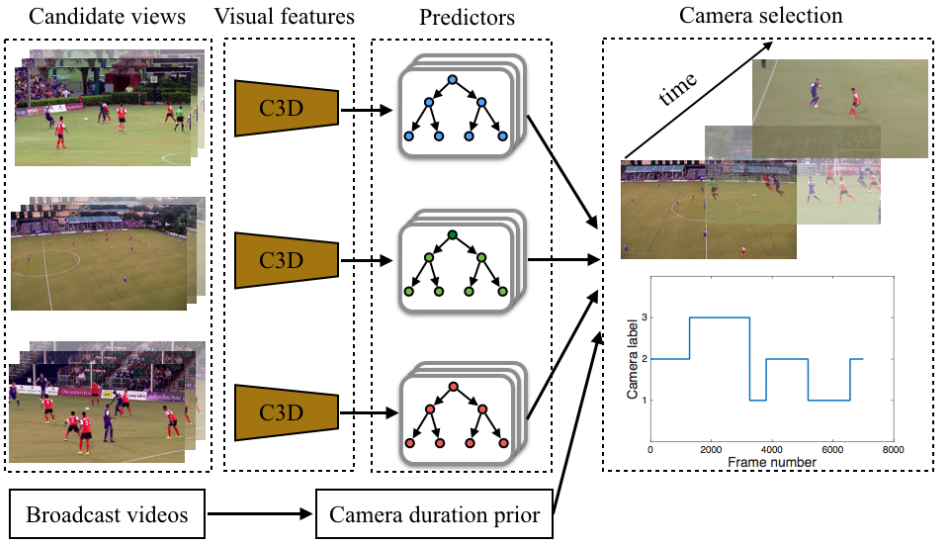


Figure 5.1: Regularized visual importance ranking pipeline. First, visual features are extracted from a shared convolutional neural network for multiple candidate cameras. Then, a group of independent smooth regressors predicts the visual importance of each camera view. Finally, the predictions are regularized by the camera duration prior that is estimated from existing broadcast videos, producing natural camera selection.

In our case, it should contain items of visual importance such as ball locations, interesting player actions, events (i.e. passes and goals) and so on. Second, the camera selection process should maintain a proper camera duration distribution [Coh13]. The camera duration is the length of a sequence without a camera transition. Durations that are too short distract the attention of viewers. On the other hand, durations that are too long produce monotonous footage or miss details of the game.

Our method learns both components from human demonstrations. Figure 5.1 shows the pipeline of our method. First, our method extracts visual features from a convolutional neural network, which simplifies feature extraction compared with previous hand-crafted methods. Then, a group of independent predictors evaluates the visual importance of each camera view based on the human demonstration. Meanwhile, we estimate a camera duration prior from the training data. Finally, our method greedily selects the best camera by combining the predicted visual importance and the camera duration prior, providing natural camera selection.

5.2.1 Method

Problem formulation. We have a sequence of T frames and a set of N cameras. $\mathbf{y} = \langle y_1, y_2, \cdots, y_T \rangle$ is a sequence of camera labels, where $y_t \in [1, N]$. Let $\mathbf{I}_n^t$ denote the image from camera n at time t. Then $\mathbf{I}_\mathbf{y}^T = \langle \mathbf{I}_{y_1}^1, \mathbf{I}_{y_2}^2, \cdots \mathbf{I}_{y_T}^T \rangle$ is an image sequence that corresponds to one possible solution $\mathbf{y}$. Let $\mathbf{Y}$ denotes the set of all possible solutions. The goal is to find the optimal selection $\mathbf{y}^*$:

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in \mathbf{Y}} (h(\mathbf{I}_\mathbf{y}^T) + \lambda g(\mathbf{I}_\mathbf{y}^T)) \tag{5.1}$$

where $h(\cdot)$ is an arbitrary non-linear function that predicts the visual importance of $\mathbf{I}_\mathbf{y}^T$. For example, it can be a neural network, a random forest or a combination of both. $g(\cdot)$ is a regularization term that prevents too short (brief glimpse) or too long (monotonous selection) camera selections. $\lambda \geq 0$ controls the trade-off between these two terms.

Simplification and interpretation. Since $\mathbf{Y}$ is growing exponentially with the length of the video and the number of cameras, optimally solving (5.1) quickly becomes intractable. Therefore, we simplify (5.1) based on three assumptions. First, the visual importance can be predicted independently in each camera. This assumption makes the prediction easier but may cause bias which will be addressed by the regularization term. Second, the visual importance can be predicted with a limited number of frames and previous predicted visual importance values (i.e., those of previous τ frames). Third, the regularization term $g(\cdot)$ only depends on the duration of the currently selected camera, which is also camera independent.

By these simplifications, we transform the original optimization problem to a ranking problem. We define $f_n(t) \in \mathbb{R}$ as the "score" of camera n at current time t. As a result, the optimization becomes:

$$y_t^* = \arg\max_n f_n(t), \tag{5.2}$$

where

$$f_n(t) = h(\mathbf{I}_n^{t,\tau}, h_n^{t-1,\tau-1}) + \lambda g(d_n), \tag{5.3}$$

with

$$g(d_n) = \begin{cases} reg(d_n), & \text{if camera } n \text{ is selected at time } t \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

where $\mathbf{I}_n^{t,\tau} = \langle \mathbf{I}_n^{t-\tau}, \cdots, \mathbf{I}_n^{t-1}, \mathbf{I}_n^t \rangle$ is a sequence of images with window size τ from camera n. $h_n^{t-1,\tau-1}$ are the previous $\tau - 1$ visual importance values. d_n is the camera

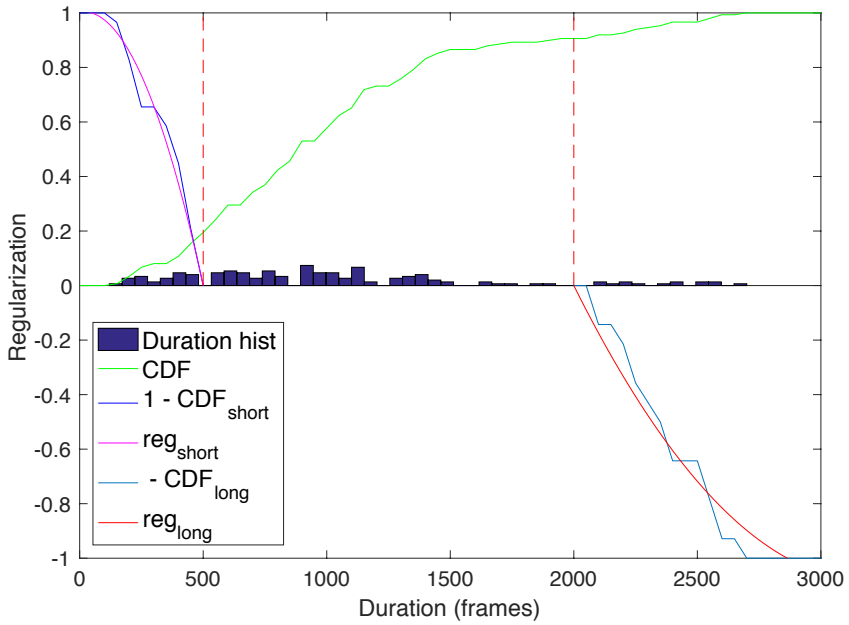


Figure 5.2: Regularization term when $(\varepsilon_1, \varepsilon_2) = (500, 2000)$. The blue bar chart shows the histogram of camera duration d from the training data. The green line shows the cumulative distribution function (CDF) over all durations. The regularization has three segments. Two of them (reg_{short} and reg_{long}) are derived from the CDFs in each duration intervals, respectively. The third segment is zero (when $d \in [\varepsilon_1, \varepsilon_2]$). Best viewed in color.

duration of camera n, and $reg(\cdot)$ is a regularization function that will be specified later.

This simplification decouples the correlation of visual importance prediction between cameras. Since $h(\cdot)$ is recurrently predicted, one concern is too frequent camera switching if the values of $h(\cdot)$ are jerky along time. To prevent this problem, we adopt the recurrent decision tree method [CLC+16] to smoothly predict the $h(\cdot)$ term, making the optimization temporally smooth.

The novelty of our method is on the feature representation part. Previous work use various types of handcrafted features [APS+14, DC11, CWH+13, WMHM14] based on the player detection data. To the best of our knowledge, our approach is the first to use deep features in camera selection, making the feature extraction more effective.

Data-driven Regularization. Our regularization is the reward of keeping the current selection. It encourages the selected camera duration to have a similar distribution to the human demonstration. Therefore, we use the cumulative distribution function (CDF) to represent the camera duration prior. The CDF is defined as:

$$F_D(d) = P(D \leq d) \tag{5.5}$$

where $P(D \leq d)$ is the probability that the random variable D (camera duration) takes on a value less than or equal to d.

The regularization term is defined as a piecewise function:

$$reg(d) = \begin{cases} 1 - F_D(d), & \text{if } 0 \leq d < \varepsilon_1 \\ 0, & \text{if } \varepsilon_1 \leq d \leq \varepsilon_2 \\ -F_D(d), & \text{if } d > \varepsilon_2 \end{cases} \tag{5.6}$$

where d is the duration of a selected camera. (ε_1, ε_2) is a camera duration range that has neither punishment nor awards (reg_{zero}), serving as a hyperparameter of the regularization. When $d < \varepsilon_1$, the $F_D(d)$ is a discrete function bounded in $[0,1]$ to the CDF. To make the whole regularization term continuous, we fit a quadratic function from discrete samples of $F_D(d)$ with the constraint $F_D(\varepsilon_1) = 1$ [NW06]. The same strategy is applied when $d > \varepsilon_2$. Figure 5.2 shows the regularization term when $(\varepsilon_1, \varepsilon_2) = (500, 2000)$. The estimated quadratic functions are close to the real distribution. When $d < \varepsilon_1$, the regularization is a positive value, which encourages keeping the current selection until $d = \varepsilon_1$. By contrast, it encourages switching to other cameras when $d > \varepsilon_2$.

It is worth noting that our regularizer is different from conventional regularizers such as the L2 norm. It has both positive and negative values since it measures the reward of keeping the current selection based on camera duration but not the camera duration itself. For example, a short camera duration should not be rewarded as it causes jump cuts, but extending the short sequence to have a longer duration should be rewarded.

5.2.2 Experiments

Dataset. We collected data from the game of Orlando City vs. Charlotte Eagles in 2014 season. Figure 5.3 shows the camera configuration and example images. One camera is located at the mid-field, overlooking the game. The other two cameras are located behind the left and right goals respectively, slightly close to the field corners. The middle camera gives an overview of the game. The left and right

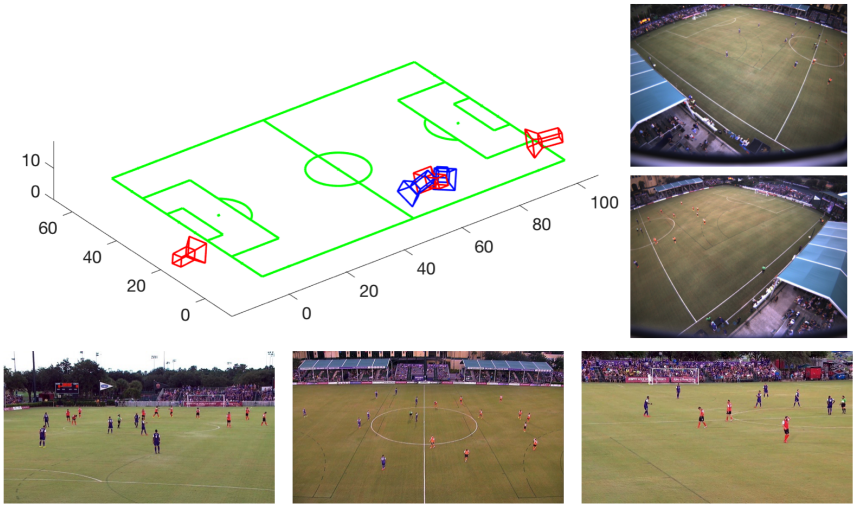


Figure 5.3: Camera settings of the main game and image examples. Blue: static cameras; red: PTZ cameras.

cameras provide detailed views of the game. A broadcast video was composited from the three PTZ cameras by a professional operator. The videos were synchronized at 60 fps with resolution 1280×720. Replays were manually removed. The length of the game is about 94 minutes. There are also two static cameras in the system but our method only uses information from the three PTZ cameras to select the broadcast camera. Static cameras are only used in one of the baselines.

As a proof-of-concept, we use the data from the first half game for evaluation. It has about 42 minutes of data for training and a 5-min sequence for testing. The testing sequence displays a range of camera switches.

We have two sources of labels. The first is one group of labels from a real broadcast video (professional selection). The second is five groups of labels that are collected from trained amateur users. We build a web application to collect data from amateur users. On the top of the web page, three windows show synchronized videos of the soccer game at real-time speed. The users are asked to switch between these three views by pressing keyboard buttons. The images of the selected stream are shown in another window (under the candidate views). Before the data collection, the users practice the web application until they feel confident in their selections. The original labels are binary values of whether a frame is selected or not. In our method, the training labels are real values that are weighted from the two sources of labels in each frame. We put a higher weight (5 vs. 1) on professional labels. The test labels are from professional selections only.

Conv1	Conv2	Conv3-5	fc6	fc7	fc8	softmax
64	128	256	2048	1024	2	–

Table 5.1: Visual importance network architecture. The network has 5 convolution followed by 5 max-pooling separately, and 3 fully connected layers. All convolution kernels are $3 \times 3 \times 3$ with stride 1 in both spatial and temporal dimensions. The numbers of filters are denoted in the second row. All pooling kernels are $2 \times 2 \times 2$, except for *pool*1 which is $1 \times 2 \times 2$.

Implementation. We use the C3D network [TBF+15] as a basis to learn the representation of an image sequence. The network architecture is shown in Table 5.1. The network has 5 convolution layers, 3 fully connected layers and a *softmax* layer.

We initialize the network from a pre-trained C3D network using the UCF101 dataset [SZS12]. Then, we fine-tune the network using broadcast training examples to predict if a sequence is selected for broadcasting (binary classification). We rescale the input image to 200×112, mostly keeping the original aspect ratio (16×9). As a result, the input dimension is $3 \times 16 \times 112 \times 200$. In training, we set the mini-batch size as 20 and use an initial learning rate of 0.003. The training is stopped after 5 epochs and takes about 2-3 hours. The fc6 layer is used as the visual feature.

Because the dimension of the fc6 layer is too high to be suitable for RDT learning, we compress the visual feature using PCA (about 78% variance explained), resulting in a 32-dimension feature. The motivation for using PCA is that decision tree optimization is more efficient in an orthogonal feature space. Using PCA also effectively compresses the feature compared with original features (64 times smaller). An alternative is to decrease the dimension of the fully-connected layers. Because feature extraction is just a part of our method which contains other approximations, we believe using PCA is a good trade-off between effectiveness and compactness in our problem. We denote this feature by C3D feature. Given the visual feature, we use the RDT method to predict smooth visual importance in each camera independently.

Besides our method, we also implement two alternative methods for comparison. **Automated Director Assistant (ADA)** [CWH+13]: ADA learns a random forest classifier using player location distribution and game flow at the current time as the feature. Our implementation adds temporal information by concatenating the features in a 2-second sliding window. **LSTM**: This method models the camera selection as a sequential multivariate classification problem. It also uses the C3D feature and learns the prediction using the long short-term memory (LSTM)

Method	Accuracy				Switch #
	Left	Middle	Right	All	
ADA	59.2	77.3	2.3	60.7	151
LSTM	48.8	75.9	**42.3**	60.5	46
RDT	30.3	91.0	0.0	54.6	**14**
ADA + CDF	**62.0**	81.3	8.3	64.4	23
LSTM + CDF	43.0	83.3	23.8	59.2	28
Ours (RDT+CDF)	49.1	**92.1**	22.3	**65.6**	17
Prof. operator	–	–	–	–	12

Table 5.2: Camera selection accuracy and switch numbers. ADA: the method of [CWH+13], LSTM: the method of [HS97]. CDF is the cumulative distribution function. The best performance in each category is highlighted.

prediction [HS97]. We use a single layer LSTM to map the input sequences (features) to output sequences (labels). The number of hidden variables is 256. The length of each sequence is fixed at 180 (3 seconds). For a fair comparison, we also apply our regularization method to ADA and LSTM. We first smooth the selection probability from ADA and LSTM. Then, we combine the probability and the CDF method to select the camera.

Results. Table 5.2 shows the performance of the methods evaluated by the accuracy and the number of camera switches compared with the human operator's result. Our method and the two alternatives have achieved reasonably good prediction accuracy. When there is no CDF regularization, the switch number of ADA and LSTM is very large, which will cause unpleasant viewing experience. RDT tends to select the middle camera. When we apply the CDF regularization to all three methods, the switch numbers of ADA and LSTM drop significantly. Meanwhile, the overall prediction accuracy of ADA and our method increases most probably because the CDF mitigates the monotonous selections, for example, the prediction accuracy of the right camera is significantly increased (0.0 vs. 22.3). Our method achieves the highest accuracy that is slightly higher than ADA. Meanwhile, our method has a reasonable number of camera switches compared with the human operator's (17 vs. 12). ADA has achieved the second best overall accuracy. However, it has very low accuracy in the right camera. In the experiment, we found LSTM is very strong. It has a quite high accuracy on each camera. However, its switch number is relatively large.

In (5.1), the weight λ controls the importance of the camera duration prior. Figure 5.4 shows the camera selection accuracy with different regularization weights

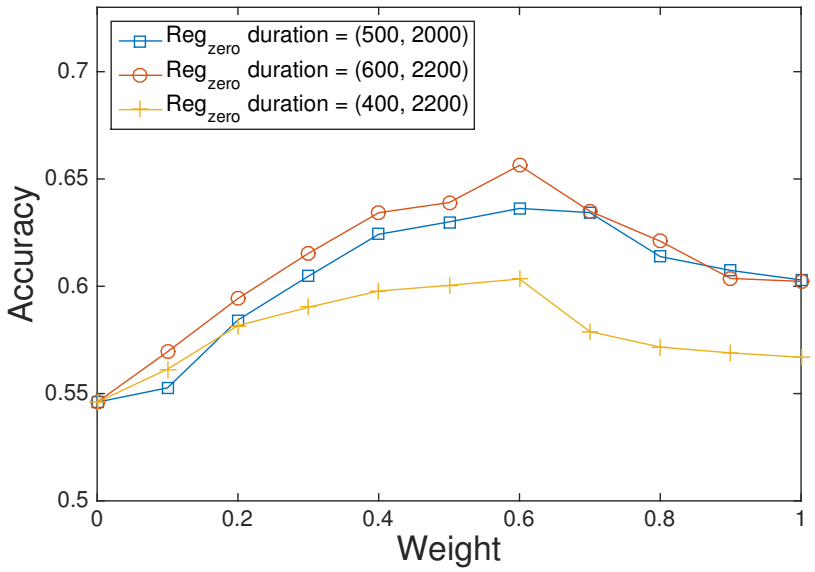


Figure 5.4: Accuracy with varying regularization weights. For three groups of camera duration hyperparameters $(\varepsilon_1, \varepsilon_2)$, the weight setting $\lambda = 0.6$ constantly provides the best performance in each hyperparameter setting, justifying the setting of 0.6. When a camera duration is smaller than ε_1, the regularization term is positive. When a camera duration is larger than ε_2, the regularization term is negative.

and different camera duration hyperparameters. The weight setting $\lambda = 0.6$ constantly produces the best performance, demonstrating the robustness of our weight setting. Figure 5.4 also shows that the change in other parameters such as $(\varepsilon_1, \varepsilon_2)$ does not affect the optimal setting for λ.

Visual inspection. Figure 5.5 shows a visualization of the predictions. From this visual inspection, we can verify qualitatively that our method achieves a good balance of prediction accuracy and camera duration distribution. Our predictions mostly overlap with the human operator's choice and have fewer jump cuts (narrow spikes in the figure). Without the regularization, ADA and LSTM have obvious jump cuts which will severely deteriorate the broadcast quality. We also generate videos[1] using predicted camera IDs.

Figure 5.6 shows qualitative results of visual importance prediction. The line

[1] https://youtu.be/xzJ2syXLIAA, left: our method, right: human operator.

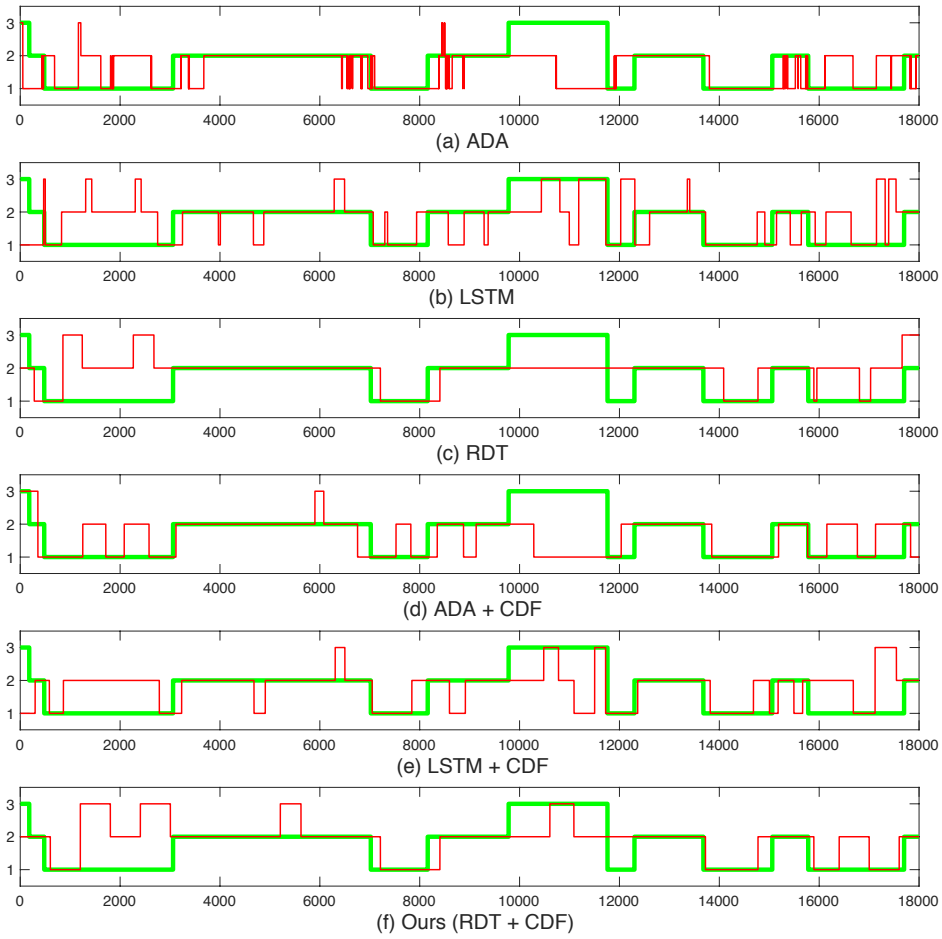


Figure 5.5: Comparison with other methods. (a) ADA [CWH+13], (b) LSTM [HS97], (c) RDT [CLC+16], (d) ADA + CDF, (e) LSTM + CDF, (f) Our method. The horizontal axis is frame numbers and vertical axis is camera labels. The green line is the ground truth and the red line is the predictions. Our method remains close to human operator's selection and has fewer jump cuts. Best viewed in color.

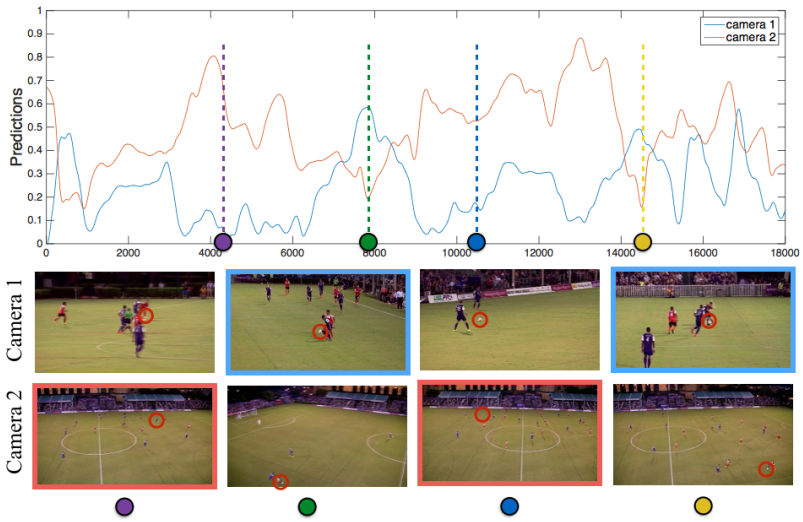


Figure 5.6: Visual importance prediction. The top figure shows that our approach smoothly predicts the visual importance of two cameras. The bottom two rows show four columns of sampled images whose time are denoted by four colors. In each column, the image with a color bounding box has higher visual importance. Our predictions are precise. For instance, camera 1 has a good view of the ball position and active players in "time green" (second column). At the same time, camera 2 has a poor view of the ball position, corresponding to the lower visual importance value on the top figure. Best viewed in color.

chart shows that our approach smoothly predicts the visual importance of camera views. The bottom two rows show the selected/un-selected camera views from four time instances. The selected frames are informative to viewers about what happens in the game such as interesting actions, ball positions and team formations.

User preference study. We also conducted a user preference study to complement our quantitative experiments. We evaluated our approach against the two alternatives. Three side-by-side video clips were synthesized using predicted camera labels. For ADA and LSTM, we do not apply the CDF regularization. Each video clip is about 1 minute in length. The number of trials is 225. Each clip was evaluated by 25 different CrowdFlower [Cro17] workers who have a diverse distribution of ages, genders and geographic locations. We did not ask if participants are soccer fans or not. In each trial, participants were instructed to choose the video that has

Comparison	Clip 1 win / loss	Clip 2 win / loss	Clip 3 win / loss	All win/loss
Ours vs. ADA	18/7	22/3	14/11	54/21
Ours vs. LSTM	15/10	16/9	19/6	50/25
Ours vs. Prof.	12/13	9/16	14/11	35/40

Table 5.3: User study results. Our method is preferred over all alternatives. Comparing with professional operator's, our method has comparable preference (35 vs. 40).

more natural/pleasant camera views (our method was randomly shown on the left or right side).

Table 5.3 shows the user study results. Our method is preferred over all alternatives with safe margins. Moreover, our method has a comparable preference from users when it is compared with the professional operator's selection, demonstrating the practicality of our method.

5.3 Data augmentation using Internet videos

For sports camera selection, a significant limitation of existing work is that most of them use a single game (main game) in the experiment [CWH+13, CML18] because researchers cannot access the data (including candidate videos and broadcasting videos) that are owned by broadcasting companies. On the other hand, broadcast videos are widely available on the Internet (e.g., Youtube). These games (auxiliary games) provide a large number of positive examples. Using these Internet videos can scale up the training data with negligible cost.

In practice, arbitrarily choosing auxiliary games does not necessarily improve the performance of camera selection, when main games are from minor leagues while auxiliary games are from premier leagues. So, the main game and the auxiliary games should be similar in terms of camera locations and the action of players. Although a universal camera selection model should be the final goal, models for specific teams have their own value. For example, minor-league teams can reduce the cost of live broadcasting for host games. Targeting at these applications, we constrain that main games and auxiliary games are from the same stadium and similar camera setups across games.

The primary challenge of using auxiliary games is the missing views in the video composition. Omitting non-broadcast views is the default setting for TV channels and live streams on the Internet. As a result, the amount of complete and incomplete data is highly unbalanced. To overcome this challenge, we introduce

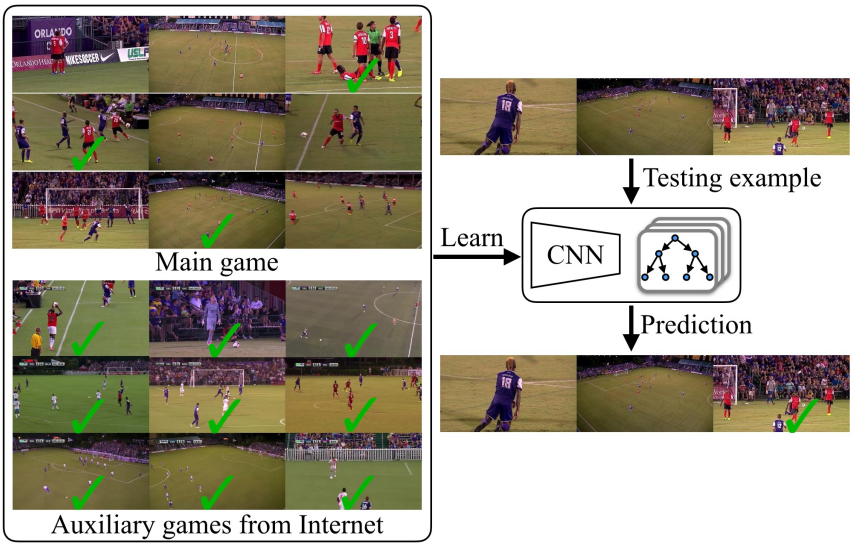


Figure 5.7: Learning camera viewpoint selection from Internet videos. The goal of our work is to select one camera from multiple candidate cameras for sports broadcast. The traditional way is trained from a dataset that has all candidate views such as the main game shown in the figure. However, it is costly to obtain this kind of data as it requires high effort from human experts. Our method uses existing Internet videos as auxiliary data to train a model with state-of-the-art prediction accuracy. Best viewed in color.

the random survival forest (RSF) method [IKBL08] from statistic learning to impute the missing data. To the best of our knowledge, we are the first to use Internet videos and RSF in camera selection.

The second challenge is from the potentially negative impact of background information in auxiliary games. The background information includes different lighting, fan celebration and stadium decoration. In practice, camera operators are trained to follow players and keep balls visible at particular locations [Owe15]. Based on this observation, we propose a spatial-appearance heatmap to represent locations and appearances of foreground objects jointly.

Our main contributions are: (1) Using Internet data and random survival forests to address the data scarcity problem in camera selection for soccer games. (2) Proposing a spatial-appearance heatmap to represent foreground objects.

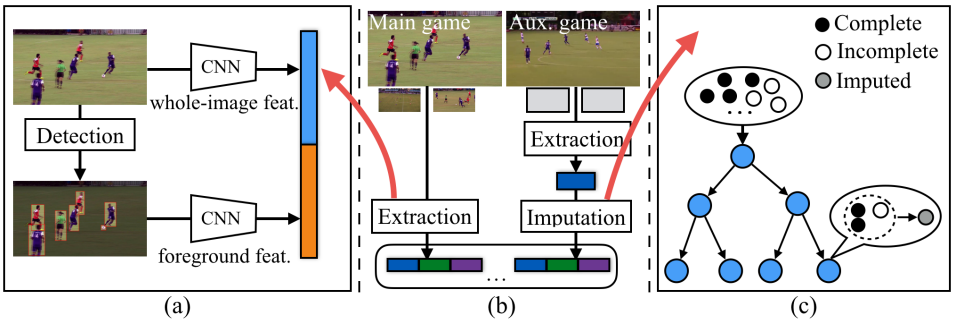


Figure 5.8: Main components of our method. (a) Feature extraction. Two CNNs are used to extract whole-image and foreground features. (b) Training process. We first extract features from both main game and auxiliary game frames. The feature of auxiliary games is imputed for the missing data. Both data are then used to train the final model. (c) Data imputation. Best viewed in color.

5.3.1 Method

Problem setup. We have two sources of data. One is from complete games which have videos and selections. Another is from auxiliary games which only have one broadcasting video and selections. We model the problem as a classification task given hybrid data $D = \{D_{com}, D_{incom}\}$ in which D_{com} is the *complete* data and D_{incom} is the *incomplete* data. Let $D_{com} = \{X_{com}, Y\}$ where X_{com} is the feature representation of all candidate views and $Y \in \{1,2,3\}$ is the corresponding label. X can be an arbitrary feature representation for an image. Let $D_{incom} = \{\{X_{obs}, X_{mis}\}, Y\}$ where X_{obs} is the *observed* data and X_{mis} is the *missing* data (e.g., unrecorded views). Our goal is to learn a classifier from the whole data to predict the best viewpoint from multiple candidate viewpoints (e.g., an unseen X_{com}):

$$y_t = f(\mathbf{x}_t). \tag{5.7}$$

We do instantaneous single frame prediction and $\mathbf{x}_t$ is a feature representation from all camera views. During training, $\mathbf{x}_t$ is either a raw feature extracted from the main game, or a raw plus imputed feature from an auxiliary game. We only test on the main game.

Our primary novelty is to use auxiliary data from the Internet. On the other hand, this choice creates considerable challenges because of the missing data.

Assumptions and interpretation. Our method has three assumptions. First, the imputed feature $X_{inputed} = \{X_{obs}, \hat{X}_{mis}\}$ ($\hat{X}$ means the inferred values) and $X_{inputed}$ has a similar distribution to X_{com}. This assumption is reasonable since both types of games are collected from the same stadium with the same host team. Also, we expect the broadcast crew to have consistent operation across games to some extent. Second, images from different viewpoints are correlated at a particular time instance. Camera operators (from different viewpoints) cooperate to tell the story of the game. For example, the same group of players is captured from different viewpoints (i.e., joint attention). In this case, the observed data X_{obs} has a strong indication of the missing data X_{mis}. A similar assumption was successfully used to complete household objects from depth sensors [FMAJB16]. Third, our method models the viewpoint prediction problem as a single frame prediction problem without using temporal information. Single-frame prediction is the focus of our work. We will briefly show the adaptation of our method to a temporal model in the experiment.

Random survival forest. With these assumptions, we randomly draw imputed data from the joint posterior distribution of the missing data given the observed data [VVA15].

$$X_{mis} \sim p(X_{mis}|X_{obs}, Y) \tag{5.8}$$

with

$$p(X_{mis}|X_{obs}, Y) = \int p(X_{mis}|X_{obs}, \theta) p(\theta|X_{obs}, Y) \mathrm{d}\theta, \tag{5.9}$$

where θ is the model which is decision trees in our method and Y is the label. Please note this process is in the training phase so that Y is available. However, it is often difficult to draw from this predictive distribution due to the requirement of integrating over all θ. Here we introduce random survival forests to simultaneously estimate θ and draw imputed values.

A random survival forest (RSF) is an ensemble of random survival trees, which was originally designed to identify a complex relationship between long-term survival and attributes of persons (e.g., body mass, kidney function and smoking). Each decision tree recursively splits training data into sub-trees until the stopping criterion is satisfied. The statistics (e.g., mean values of labels for regression) of training examples in the leaf nodes are used as the prediction. A survival tree imputes missing data as below.

1. In internal nodes, only observed data is used to optimize tree parameters such as the decision boundary by minimizing the cross-entropy loss. This step estimates the model θ from the distribution $p(\theta|X_{obs}, Y)$ in (5.9).

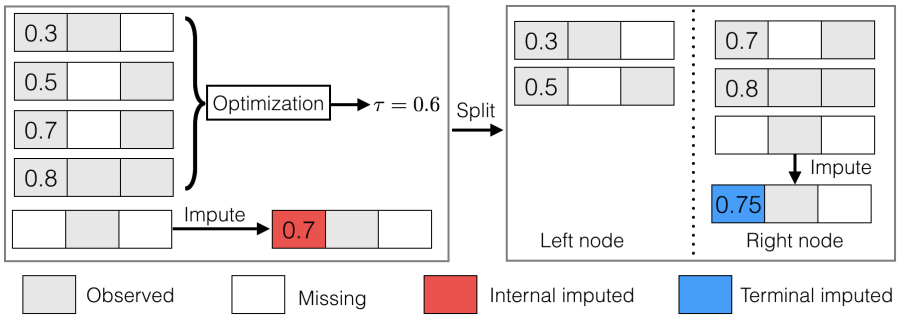


Figure 5.9: A two-level random survival tree. Each row represents a three-dimensional feature. The first dimension of the fifth feature is imputed. τ is the decision boundary. Labels are omitted for clarity. Best viewed in color.

2. To assign an example with missing data to the left or right sub-trees, the missing value is "imputed" by drawing a random value from a uniform distribution $U(x|a,b)$ where (a,b) are the lower/upper bounds of X_{obs} of the target dimension. This step draws samples from $p(X_{mis}|X_{obs},\theta)$ in (5.9).

3. After the node splitting, imputed data are reset to missing and the process is repeated until terminal nodes are reached.

4. Missing data in terminal nodes are then imputed using non-missing terminal node data from all the trees. For categorical variables, a majority vote is used; a mean value is used for continuous variables.

Figure 5.9 shows a data imputation example in a two-level random survival tree. Specific details of RSF can be found in [IKBL08, TI17]. With the RSF method, we impute the missing data $\{\{X_{obs},X_{mis}\},Y\}$ with substituted values to obtain the new data $\{X_{imputed},Y\}$. To the best of our knowledge, we are the first to introduce RSF from statistic learning to solve vision problems. Besides, we will experimentally show that it outperforms other alternatives in our problem.

Foreground feature. We develop a novel spatial-appearance (SA) heatmap to represent foreground objects. First, we quantized the image space into a 16×9 grid. Then, we represent the location of each player using five points (four corners and one center point) of its bounding box. Each point contributes "heat" to its located and neighboring cells. In the conventional heatmap, the "heat" is pre-defined values such as the number of players [CLC+16]. In our heatmap, the "heat" is the object appearance feature that is learned from the data.

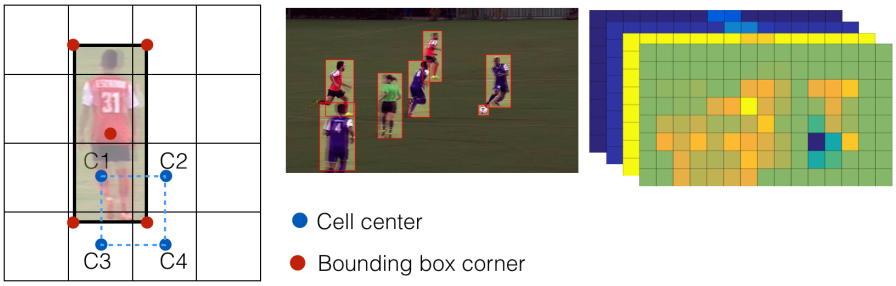


Figure 5.10: Spatial-appearance heatmap. Left: one player on a 4×4 grid; right: an example of detected objects and corresponding heatmap.

Figure 5.10 (left) illustrates how we compute the SA heatmap on a 4×4 grid. The bottom right corner of the bounding box contributes weighted "heats" to four cells C_1, C_2, C_3 and C_4. The weight is the inverse bilinear interpolation coefficient of the corner with respect to the cell centers. We use the heatmap as input to train a binary classification CNN and its second-last fully connected layer is used as the foreground feature. The classification CNN has 3 convolution layers and a *softmax* layer.

Appearance feature learning. Given the detected bounding boxes of the objects, we use a siamese network (Chapter 2.1.3) to learn object appearance features. The idea is to use the tracking information between frames as supervision to train the siamese network which uses paired examples in training and is used to extract features from image patches in testing. To train the network, we obtain positive (similar) examples from tracked players [LTLM13] in consecutive frames (e.g., from frame 1 to frame 2). The underlying assumption is that the tracked players in consecutive frames have a similar appearance, pose and team membership. Any player not part of a track is likely to be dissimilar.

5.3.2 Experiments

USL broadcast dataset. We collect a dataset from United Soccer League (USL) in the 2014 season. The dataset has one main game and twelve auxiliary games. The main game has been described in Section 5.2.2 when we introduce the visual importance ranking method. Our system only uses information from the three PTZ cameras to select the broadcast camera. Static cameras are only used in one of the baselines.

Dataset	Game	Length (min.)	# Game	# Camera	Camera	Ground truth
APIDIS [DC11]	basketball	15	1	5	static	non-prof.
ADS [CWH+13]	field hockey	70	1	3	PTZ	prof.
OMB [FCL+13]	basketball	~ 16	1	2	PTZ	non-prof.
VSR [WMHM14]	soccer	9	1	20	static	non-prof.
CDF [CML18]	soccer	47	1	3	PTZ	hybrid
Ours	soccer	94 + 108	1+12	3	PTZ	prof.

Table 5.4: Dataset comparison. In our dataset, the 108 minutes data (column four) is sparsely sampled from total $1,080$ minutes data.

Twelve auxiliary games are collected from Youtube. These games are hosted in the same stadium as the main game. They are typically 1.5 hours long. Unlike the main game, each auxiliary game only has a composited broadcast video (640×360). Figure 5.7 (bottom left) shows image examples from the auxiliary games.

In the main game, we manually annotate the ball locations on the static cameras, and two detailed view PTZ cameras at 1 fps. In the auxiliary games, we manually check the classified camera IDs around detected camera transition points. In all games, we detected bounding boxes of players and balls. Table 5.4 compares our dataset with previous camera view selection datasets. To the best of our knowledge, ours is the first dataset with dense annotations for such long dynamic multi-camera image sequences.

Implementation. We pre-process Internet videos for training labels. Given a raw video, we first detect shot boundaries using [ETM03]. We call the consecutive frames at the shot boundary *boundary frames* for simplicity. Given boundary frames, we train a CNN to classify their camera IDs into four categories (i.e., left, middle, right and other-view). The other-view images are commercials, replay logos or frames that are captured from other viewpoints. To train the camera-ID CNN, we first randomly sample 500 training frames from each PTZ video of the main game. For the other-view, we sample the same number of images from a non-sports video. Then, we apply the trained model to classify boundary frames. The classification result is manually checked and refined. The refined boundary frames are used to re-train the CNN. This process is repeated for each video. After five games, the prediction accuracy is about 85%. We found this performance is sufficient to lighten the workload of human annotation. Initialized by the CNN then manually corrected, we collect $1,634$ pairs of boundary frames from twelve videos.

Each frame is represented by two types of features: the whole-image feature and the foreground feature. The whole-image feature (16 dimensions) is from a binary classification CNN to classify if an image is selected or not by human op-

erators. The foreground feature (16 dimensions) is described in Section 5.3.1. In training, it is important to balance the number of positive and negative examples. For the main game, we choose the positive candidate view and one of the negative camera views at sampled times. For the auxiliary games, we randomly sample negative examples from the main game.

The learning data are from the main game and the auxiliary games. For the main game, we uniformly sampled $6,000$ examples from the video at 1 fps. For the auxiliary game data, we randomly sampled $4,000$ frames around camera shot boundaries (within 2 seconds) and imputed the missing values using the SRF method. The imputed data are verified by a model trained from the complete examples (about $2,100$ data passed verification).

We use the random forest method to fuse features from all candidate cameras since it is relatively easy to train. The dimension of the feature is 96 ($16 \times 3 \times 2$ for two types of features from three candidate cameras). The parameters of the random forest are: tree number 20, maximum depth 20 and minimum leaf node number 5.

We evaluate our method on the main game using 3-fold leave-one-sequence-out cross-validation. We use the camera selection from human operators as the ground truth and report the classification accuracy.

Results. For comparison, we also implement six baselines. **Baseline 1**: constantly select one camera that has the highest prior in the training data. This baseline always selects the middle camera. **Baseline 2**: select the camera that is closest to the human-annotated grounded truth location of the ball. **Baseline 3**: predict the camera using the team occupancy map introduced in [BLC$^+$13]. The team occupancy map describes the player distribution on the playing ground using tracked players from the static cameras. **Automated director assistant (ADA)** [CWH$^+$13]: it learns a random forest classifier using player distribution and game flow at the current time. Our implementation augments temporal information by concatenating the features in a 2-second sliding window, making the predictions more reliable. **C3D** [TBF$^+$15]: it is a deep CNN modified from the original C3D network. First, images from three cameras pass through the original C3D network, separately. Then their fc6 activations are concatenated and fed into a fully connected network ($1024 \times 32 \times 3$) with *Softmax* loss. **RDT+CDF** [CML18]: it uses the recurrent decision tree (RDT) method to predict visual importance and a cumulative distribution function (CDF) regularization to predict the final camera selections in a sequence. Because [CML18] requires real-valued visual importance as labels in training, we only compare with it on the dataset from [CML18].

In our method, the auxiliary data are used in two steps. First, they are used to train the feature extraction networks as extra positive examples. Second, their

Feature	Main				Main + aux.			
	L	M	R	All	L	M	R	All
whole-image	53.4	74.4	57.5	63.2	62.8	77.8	61.4	68.5
foreground	45.9	84.1	39.5	59.7	53.1	**86.2**	41.3	63.2
both	58.3	78.0	58.7	66.5	**70.0**	85.2	**68.9**	**75.9**

Table 5.5: Camera selection accuracy. "Main" and "Main+aux." mean the training data is from the main game only and is with auxiliary videos, respectively. L, M and R represent the camera on the left, middle, and right side, respectively. The highest accuracy is highlighted in bold.

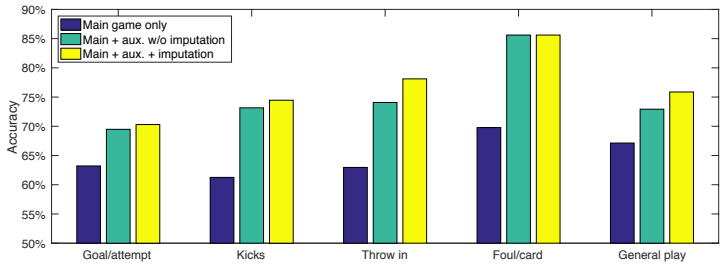


Figure 5.11: Prediction accuracy with and without auxiliary games (grouped by events).

imputed examples are used as extra training examples in the final model. Table 5.5 shows the main results of our method. First, auxiliary data provides significant performance improvement (about 9.4%). The improvement is from two stages: feature extraction and data imputation. Figure 5.11 shows details of the improvement by separating these two stages and grouping the frames into different events. Overall, the main improvement is from the feature extraction stage (about 6.6%). It implies that extra positive examples are very helpful in feature extraction. Data imputation provides an extra 2.8% improvement, which is significant in "throw in" and "general play". This result justifies our motivation of using auxiliary data. Second, the foreground feature improves performance, especially when the auxiliary games are used. The main reason might be the foreground feature excludes the negative impact of backgrounds (e.g., different fan groups, weather and light conditions) of auxiliary games.

Table 5.6 shows the comparison with the baselines. Our method outperforms all the methods by large margins. Baselines 1-3 do not work well on this dataset mainly because they omit the image content from the PTZ cameras. This result suggests that heuristic techniques using ball and player locations are not enough for

Method	Accuracy (%)	Improvement
Constant selection	40.9	35.0
Closest to ball (GT.)	37.6	38.3
Team occupancy map [BLC$^+$13]	49.8	26.1
ADA [CWH$^+$13]	54.1	21.8
C3D [TBF$^+$15]	64.3	11.6
Ours	**75.9**	–
w/o auxiliary data	66.5	9.4
w/o whole-image feature	63.2	12.7
w/o foreground feature	68.5	7.4

Table 5.6: Comparison with baselines and components analysis.

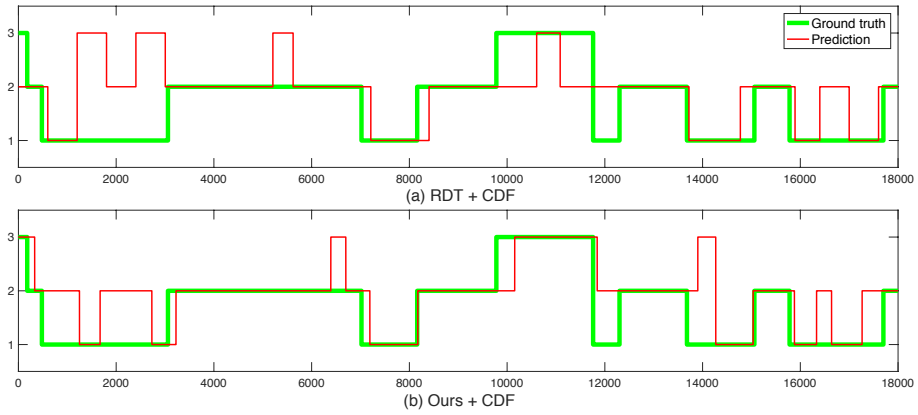


Figure 5.12: Prediction on a 5-minute testing sequence. The horizontal axis is frame numbers and vertical axis is camera labels. (a) Result of [CML18]. (b) Ours. Best viewed in color.

dynamic PTZ camera selection. ADA has substantial challenges on this dataset. It is partially because of hand-crafted features (such as player flow) are quite noisy from fast-moving PTZ cameras. C3D works reasonably well as it learns both appearance and motion features end-to-end. Its performance is slightly better than our whole-image-feature model. However, our full model is significantly more accurate (11.6 %) than C3D. It is worth noting that training C3D with auxiliary data is very difficult because the input of C3D is consecutive frames of all the views.

Combine with temporal models. To test the capability of our method with temporal models, we conducted experiments on the dataset from [CML18]. This dataset

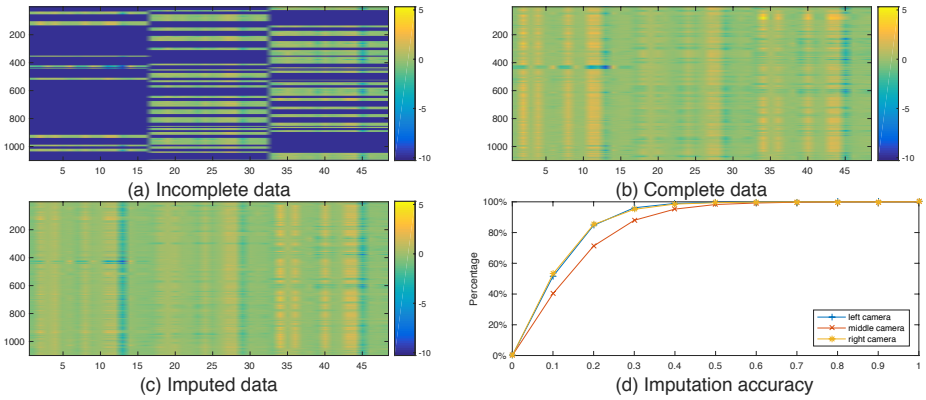


Figure 5.13: Imputation result on the main game data. (a),(b),(c) Color-coded visualization of imputed feature sequence. For the purpose of visualization, the incomplete data in (a) is set to a constant value. The x-axis is the feature dimension. The y-axis is the frame index. Colors visualize the feature values. Blue color blocks indicate the missing data. (d) The imputation accuracy as a function of the error thresholds. Best viewed in color.

has 42 minutes (60 fps) data for training and a 5-min sequence for testing. In the experiment, we feed the selection probability to the cumulative distribution function (CDF) method from [CML18]. The CDF method prevents too short (brief glimpse) and too long (monotonous selection) camera selections. The experiment shows our method is more accurate than [CML18] (70% vs 66%). Figure 5.12 shows a visualization of the prediction.

Data imputation accuracy. We analyze the accuracy of our imputation method using the main game data because the missing data in the auxiliary videos have no ground truth. We use the last $1,100$ frames as testing data by masking the features from the un-selected cameras as missing data. A random survival forest model is trained from the rest of the data. The error is measured by absolute errors normalized by the range of the feature data in each dimension. This error metric is a good indication of the performance of imputed data in the final model. The final model (i.e., a random forest) uses the sign of the difference between the feature value and the decision boundary to guide the prediction. Figure 5.13(a)(b)(c) visualizes the incomplete data, complete data and imputed data, respectively. The imputed data is visually similar to the ground truth. Figure 5.13 (d) shows the imputation accuracy as a function of the error thresholds. When the error threshold is 0.2, about 80% of

Method	Accuracy (%)	Improvement
RSF	**75.9**	–
NN	72.2	3.7
OptSpace [KMO10]	68.6	7.3
Autoencoder	73.9	2.0

Table 5.7: Comparison of RSF with alternatives. RSF is our method. NN is a nearest neighbor method.

	location	appearance	Accuracy (%)	Improvement
SA heatmap	✓	✓	**59.7**	–
Avg pool.		✓	41.8	17.9
Max pool.		✓	42.4	17.3
Heatmap in [CC15]	✓		48.4	11.3

Table 5.8: Comparison of SA heatmap with alternatives. The SA heatmap is our method. Average and max pooling methods pool features from multiple players.

the data are correctly predicted. Although the accuracy is tested on the main game, it suggests a reasonably good prediction on the auxiliary games.

To evaluate the performance of RSF on the real data, we also imputed the missing values using nearest neighbor (NN), OptSpace [KMO10] and a neural autoencoder. Table 5.7 shows that RSF outperforms all of them with a safe margin.

Foreground feature aggregation. We conducted the experiments on the main game to compare the performance of the SA heatmap with other alternatives. All the methods use the same appearance feature from the siamese network as input. Table 5.8 shows that the SA heatmap outperforms other alternatives mainly because it encodes both location and appearance information.

Qualitative results. Figure 5.14 shows predicted image sequences with the ground truth and contributing sequences. The contributing sequence is from the most dominant contributing examples in the leaf nodes for each prediction. Figure 5.14(b) (last column) shows an example of incorrect predictions. The ground truth camera is kept as the middle camera. Our prediction switches to the right camera. By inspecting the video, we found the human operator's selection has better temporal consistency while ours tends to provide more information in single frames. This

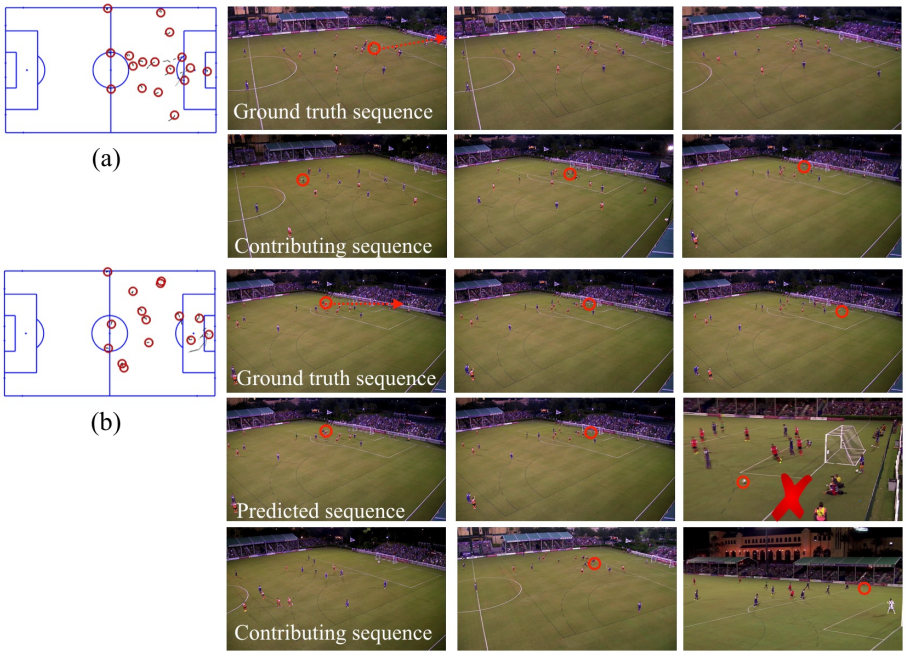


Figure 5.14: Qualitative results. The ground truth row shows the ground truth image sequences (about 3 seconds). The predicted sequence row shows the predictions from our method (omitted if all predictions are correct such as in (a)). The contributing sequence row shows the most dominant training example in the leaf node. In each sequence, player trajectories are visualized on the playing field template. The ball locations and their trajectories (dashed lines) are overlaid on the original images. The red cross mark indicates incorrect predictions. Best viewed in color.

link [2] shows a generated video using our method.

Limitations. In real applications, more than three candidate cameras are used. However, we found most of the shots are from the three cameras that cover the left goal area, the middle filed and the right goal area. We also qualitatively verified that the camera setting in the proposed dataset is representative for soccer games from [WMHM14] and [HFU17]. It indicates that our method can apply to many real situations, especially in small-budget broadcasting.

Although we collected the largest-ever training data from the Internet, the test-

[2] https://youtu.be/2qm285vU_8c, left: our method, right: human operator.

ing data is from one game. We mitigate this limitation by using dense testing (3-fold cross-validation).

If the testing game is from different stadiums, the trained model cannot be directly applied to the testing game because deep features encode team color and stadium. In that case, fine tuning the feature extraction part is necessary.

5.4 Summary

In this chapter, we proposed two methods for sports camera selection. The first method uses a cumulative distribution function to regularize the camera duration. The second method uses Internet videos to augment the training data. The methods are straightforward and both produce reasonably good results.

Our two methods have different design principles. The first method predicts the visual importance of each camera view independently so that the system has the flexibility to add/remove cameras. For example, when we add a new camera to a three-camera system, we do not have to retrain the models of the existing cameras. However, the design of the first method has two disadvantages. First, the method requires real-value labels for each camera because the first method is a regression method. Real-value labels are more expensive than binary labels. Second, the number of models increases linearly with the number of cameras, so that it is not friendly to users when the number of cameras is large. For example, training twelve models for a twelve-camera system. In contrast, the second method trains a single model that sees all the views and only requires binary labels. However, the second method has to retrain the model when the system adds/removes cameras.

At the current stage, we do not investigate other typical behavior of human directors. One is event anticipation. For example, camera directors often anticipate the event of the game so that they can send instructions to camera operators in advance. Once the event indeed happens, the camera director can quickly switch to the anticipated camera view. If the system wants to mimic this ability of the human director, it will require long-term event forecasting and a planning-selection coordinating method.

Chapter 6

Conclusions

This thesis presents three essential components for automatic broadcast of team sports. The key idea is learning from human demonstrations.

In Chapter 3, we present two camera calibration methods that are used to estimate human demonstration labels from broadcasting videos. The two-point method overcomes the problem of insufficient point correspondences in narrow field of view cameras. The edge image based method can automatically estimate camera poses with a database of synthetic images. We compare and evaluate these methods on public datasets and find that they are more accurate and faster than state-of-the-art methods. When images are captured from the same stadium, the first method is preferred because it provides higher accuracy. When images are captured from unknown stadiums, the second method is preferred because it has fewer assumptions for the camera location and does not rely on the point features.

In Chapter 4, we present two camera planning methods that are used to predict smooth camera angles. The recurrent decision trees method predicts smooth camera angles in real time. The overlapped hidden Markov model method handles the jitters produced by multiple possible choices of camera angles in camera planning. Experiments demonstrate that our methods significantly outperform other methods in the literature.

Broadcasting from multiple cameras is visually more appealing than single camera broadcasting. In Chapter 5, we present two camera selection methods that are used to select an "on air" camera from multiple candidate cameras. The first method regularizes the camera duration of the selected camera to avoid sudden jumps and monotonous selections. The second method augments the training data with Internet videos to alleviate the lack of multi-camera examples. Our methods achieve higher accuracy than other methods and the generated video is competitive with the result of a human operator.

6.1 Future directions

We conclude this thesis by proposing several directions for future exploration:

- **Simultaneous localization and mapping for sports cameras.** The camera calibration methods presented in this thesis assume the environment or map (e.g., landmarks in sports stadiums) does not change over time. Although we enhance the content of the map by using a random forest, dynamically adding new landmarks to the map can improve the performance. Simultaneous localization and mapping (SLAM) is a standard technique in robotic navigation. Researchers have adapted SLAM to PTZ cameras. For example, [LMPDB16] proposed a PTZ SLAM system for object tracking in a parking lot. In that system, the camera state and scene landmarks are simultaneously updated. A future direction is to develop a PTZ SLAM system for sports applications such as virtual reality and augmented reality [RKSCS18].

- **Camera planning/selection with player identities.** In this thesis, we do not use player identities in the camera planning and camera selection because player identities are not available. When player identities are available, we can use more expressive features to describe particular players. For example, we can use a relative occupancy map [LCCL13] to describe a star player in basketball games. By doing so, we expect a better performance.

- **Player pose for camera planning.** The presented method uses player locations and appearances (i.e., deep feature from player patches) as features for camera planning. Players have more detailed features such as body pose [WRKS16, NYD16]. Body pose describes the motion of players and provides cues to their intentions. For example, when a ball runs towards a player, the player can approach the running ball or wait for the ball to run to him/her. A comprehensive camera planning system should recognize these minor differences and give appropriate responses.

- **Detailed-view camera planning.** In Chapter 4, we have shown camera planning for overview cameras. We have not explored camera planning for detailed-view cameras or "ISO" cameras because they cover specific (isolated) anticipated actions points. These cameras are good at capturing the zoom-in view of individual players, for example, a player who is taking a penalty kick. In addition to panning, these cameras employ zoom settings over a large range of scales. A future direction is the study of zoom factors in detailed-view cameras, which can also be applied to non-sports applications such as TV talk shows and stage performances.

- **Joint optimization of planning and selection.** In the thesis, planning and

selection are two independent tasks that do not cooperate with each other. In practice, they are jointly conducted in broadcasting. For example, the director gives an instruction that "camera 2, please focus and follow the ball". When camera 2 follows the ball, the director will switch to that camera. One of the future direction is to mimic a similar operation that optimizes planning and selection jointly.

- **Use high-level semantic structures of games.** In the thesis, our data is from a single game in both basketball and soccer, preventing us from learning general models that are suitable for many games. When more data are available, the high-level semantic structures such as team tactics will play an important role. For example, when the two wings of a 4-4-2 team have many goals or goal attempts in a soccer game, the broadcast director often switches to detailed-view cameras. In the future, we can categorize games by semantic structures such as counterattack, "total football" and "long ball". Then, we train multiple models on different categories of the games. In testing, the broadcasting system can choose a suitable model for a particular game.

- **Automatic commentator.** This thesis focused on generating broadcasting videos, while audio comments are critical for audience experience. There is much room for improvement in existing automatic commentator systems [TNN$^+$98, VAHR98] which use ruled-based methods. On the other hand, computer vision techniques can generate diverse sentences and paragraphs directly from images/videos [AAL$^+$15, SLHS17, XSJL17]. Thus, we believe an automatic commentator is practically useful and technically feasible.

- **Learning from synthetic games.** Our work uses data from real games to train planning and selection algorithms. Real data are expensive and are not always available. Using synthetic data is one of the ways to eliminate this limitation [SLS16, RVRK16]. For example, [SLS16] demonstrated that synthetically generated RGB images can improve the performance of image segmentation and depth estimation. If synthetic games are used in training automatic camera systems, the challenge will become how to make sure the synthetic game has enough resemblance to the real games. At the same time, the behavior of synthetic game directors may be generated from some rules. How to integrate these rules into the current system is also an interesting direction.

Bibliography

[AAL+15] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. VQA: Visual question answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. → pages 94

[ABC+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: A system for large-scale machine learning. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, volume 16, pages 265–283, 2016. → pages 13

[Adr18] Pennington Adrian. Tesla for sports broadcasting: how automated live production is gaining ground. https://www.svgeurope.org/blog/headlines/tesla-for-sports-broadcasting-how-automated-live-production-is-gaining-ground/, 2018. → pages 3

[AKK06] Yasuo Ariki, Shintaro Kubota, and Masahito Kumano. Automatic production system of soccer sports video by digital camera work based on situation recognition. In *IEEE International Symposium on Multimedia (ISM)*, 2006. → pages 17, 18

[APS+14] Ido Arev, Hyun Soo Park, Yaser Sheikh, Jessica Hodgins, and Ariel Shamir. Automatic editing of footage from multiple social cameras. *ACM Transactions on Graphics (TOG)*, 33(4):81, 2014. → pages 70

[BCR+17] Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. In *International Conference on Learning Representations (ICLR)*, 2017. → pages 12

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. → pages 23, 59

[BLC+13] Alina Bialkowski, Patrick Lucey, Peter Carr, Simon Denman, Iain Matthews, and Sridha Sridharan. Recognising team activities from noisy data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. → pages 85, 87

[BM04] Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision (IJCV)*, 56(3):221–255, 2004. → pages 42, 47

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. → pages 9

[Bre03] Leo Breiman. Manual–setting up, using, and understanding random forests v4. https://www.stat.berkeley.edu/~breiman/Using_random_forests_v4.0.pdf, 2003. → pages 11

[CC14] Jianhui Chen and Peter Carr. Autonomous camera systems: A survey. In *Workshops at the AAAI Conference on Artificial Intelligence*, 2014. → pages 3

[CC15] Jianhui Chen and Peter Carr. Mimicking human camera operators. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2015. → pages vi, vii, 50, 53, 54, 56, 57, 61, 62, 63, 65, 89

[CDDV11] Fan Chen, Damien Delannay, and Christophe De Vleeschouwer. An autonomous framework to produce and distribute personalized team-sport video summaries: A basketball case study. *IEEE Transactions on Multimedia (TMM)*, 13(6):1381–1394, 2011. → pages 53

[CDV10] Fan Chen and Christophe De Vleeschouwer. Personalized production of basketball videos from multi-sensored data under limited display resolution. *Computer Vision and Image Understanding (CVIU)*, 114(6):667–680, 2010. → pages 18, 20, 21

[CGL+17] Tommaso Cavallari, Stuart Golodetz, Nicholas A. Lord, Julien Valentin, Luigi Di Stefano, and Philip H. S. Torr. On-the-fly adaptation of regression forests for online camera relocalisation. In

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. → pages 33

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. → pages 14

[CL17] Jianhui Chen and James J Little. Where should cameras look at soccer games: Improving smoothness using the overlapped hidden Markov model. *Computer Vision and Image Understanding (CVIU)*, 159:59–73, 2017. → pages vi, vii, 26, 27, 37, 54

[CLC+16] Jianhui Chen, Hoang M. Le, Peter Carr, Yisong Yue, and James J. Little. Learning online smooth predictors for realtime camera planning using recurrent decision trees. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. → pages vi, vii, 52, 63, 70, 76, 82

[CML18] Jianhui Chen, Lili Meng, and James J Little. Camera selection for broadcasting soccer games. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018. → pages vi, vii, 78, 84, 85, 87, 88

[CMM13] Peter Carr, Michael Mistry, and Iain Matthews. Hybrid robotic/virtual pan-tilt-zoom cameras for autonomous event recording. In *ACM International Conference on Multimedia (ACMMM)*, 2013. → pages 17, 20, 67

[Coh13] Neil Cohn. Visual narrative structure. *Cognitive Science*, 37(3):413–452, 2013. → pages 68

[Col02] Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2002. → pages 23

[CON08] Marc Christie, Patrick Olivier, and Jean-Marie Normand. Camera control in computer graphics. In *Computer Graphics Forum*, volume 27, pages 2197–2218, 2008. → pages 22

[Cro17] CrowdFlower. CrowdFlower. https://www.crowdflower.com/, 2017. → pages 77

[CRS+15] Davide Conigliaro, Paolo Rota, Francesco Setti, Chiara Bassetti, Nicola Conci, Nicu Sebe, and Marco Cristani. The S-HOCK dataset: Analyzing crowds at the stadium. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. → pages 18

[CSK12] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2–3):81–227, 2012. → pages 9

[CSM12a] Peter Carr, Yaser Sheikh, and Iain Matthews. Monocular object detection using 3D geometric primitives. In *European Conference on Computer Vision (ECCV)*. 2012. → pages 53, 54

[CSM12b] Peter Carr, Yaser Sheikh, and Iain Matthews. Point-less calibration: Camera parameters from gradient-based alignment to edge images. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2012. → pages 26

[CWH+13] Christine Chen, Oliver Wang, Simon Heinzle, Peter Carr, Aljoscha Smolic, and Markus Gross. Computational sports broadcasting: Automated director assistance for live sports. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2013. → pages 21, 70, 73, 74, 76, 78, 84, 85, 87

[CZL18] Jianhui Chen, Fangrui Zhu, and James J Little. A two-point method for PTZ camera calibration in sports. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018. → pages vi

[DBKK16] Andreas Doumanoglou, Vassileios Balntas, Rigas Kouskouridas, and Tae-Kyun Kim. Siamese regression networks with efficient mid-level feature extraction for 3D object pose estimation. *arXiv preprint arXiv:1607.02257*, 2016. → pages 45

[DC11] Fahad Daniyal and Andrea Cavallaro. Multi-camera scheduling for video production. In *Conference for Visual Media Production (CVMP)*, 2011. → pages 21, 70, 84

[DDG07] Anthony Dearden, Yiannis Demiris, and Oliver Grau. Learning models of camera control for imitation in football matches. In *4th*

International Symposium on Imitation in Animals and Artifacts, 2007. → pages 18

[DGFVG12] Matthias Dantone, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Real-time facial feature detection using conditional regression forests. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. → pages 54, 56, 57, 62, 63, 65

[DGSG04] Petr Doubek, Indra Geys, Tomas Svoboda, and Luc Van Gool. Cinematographic rules applied to a camera network. In *Proc. of Omnivis, The fifth Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras*, pages 17–30, 2004. → pages 20

[Die02] Thomas G Dietterich. Machine learning for sequential data: A review. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30. Springer, 2002. → pages 23

[DILM09] Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning*, 75(3):297–325, 2009. → pages 24

[DO04] Shinji Daigo and Shinji Ozawa. Automatic pan control system for broadcasting ball games based on audience's face direction. In *ACM International Conference on Multimedia (ACMMM)*, 2004. → pages 18

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. → pages 27

[DW08] Elan Dubrofsky and Robert J Woodham. Combining line and point correspondences for homography estimation. In *International Symposium on Visual Computing*, 2008. → pages 26

[ETM03] Ahmet Ekin, A Murat Tekalp, and Rajiv Mehrotra. Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing (TIP)*, 12(7):796–807, 2003. → pages 84

[FAH05] Aly A Farag and Alaa E Abdel-Hakim. Virtual forces for camera planning in smart vision systems. In *IEEE Workshops on Application of Computer Vision (WACV)*, 2005. → pages 20

[FB81] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. → pages 33

[FCL+13] Eric Foote, Peter Carr, Patrick Lucey, Yaser Sheikh, and Iain Matthews. One-man-band: A touch screen interface for producing live multi-camera sports broadcasts. In *ACM International Conference on Multimedia (ACMMM)*, 2013. → pages 84

[FH12] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, 8(19):415–428, 2012. → pages 42, 47

[FHS+15] Kazuki Fujisawa, Yuko Hirabe, Hirohiko Suwa, Yutaka Arakawa, and Keiichi Yasumoto. Automatic content curation system for multiple live sport video streams. In *IEEE International Symposium on Multimedia (ISM)*, 2015. → pages 22

[FKK+14] Sean Ryan Fanello, Cem Keskin, Pushmeet Kohli, Shahram Izadi, Jamie Shotton, Antonio Criminisi, Ugo Pattacini, and Tim Paek. Filter forests for learning data-dependent convolutional kernels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. → pages 54, 56, 57

[FMAJB16] Michael Firman, Oisin Mac Aodha, Simon Julier, and Gabriel J Brostow. Structured prediction of unobserved voxels from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. → pages 81

[GAG+17] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017. → pages 24

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press Cambridge, 2016. → pages 12

[Ged09] Suat Gedikli. *Continual and robust estimation of camera parameters in broadcasted sports games*. PhD thesis, München Technical University, 2009. → pages 31, 32

[GEL+15] Vamsidhar Reddy Gaddam, Ragnhild Eg, Ragnar Langseth, Carsten Griwodz, and Pål Halvorsen. The cameraman operating my virtual

camera is artificial: Can the machine be as good as a human ? *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(4):56, 2015. → pages 22

[GHD09] Nicholas R Gans, Guoqiang Hu, and Warren E Dixon. Keeping multiple objects in the field of view of a single PTZ camera. In *American Control Conference (ACC)*, 2009. → pages 20

[GLW11] Ankur Gupta, James J Little, and Robert J Woodham. Using line and ellipse features for rectification of broadcast hockey video. In *Canadian Conference on Computer and Robot Vision (CRV)*, 2011. → pages 26

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. → pages 14

[GRG14] Vineet Gandhi, Remi Ronfard, and Michael Gleicher. Multi-clip video editing from a single viewpoint. In *Proceedings of the 11th European Conference on Visual Media Production*, 2014. → pages 1

[GZA12] Bernard Ghanem, Tianzhu Zhang, and Narendra Ahuja. Robust video registration applied to field-sports video analysis. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2012. → pages 26

[HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. → pages 44

[HF07] Robin Hess and Alan Fern. Improved video registration using non-distinctive local image features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. → pages 26

[HFU17] Namdar Homayounfar, Sanja Fidler, and Raquel Urtasun. Sports field localization via deep structured models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. → pages 26, 35, 36, 41, 42, 47, 48, 90

[HL17] Mir Rayat Imtiaz Hossain and James J Little. Exploiting temporal information for 3D pose estimation. *arXiv preprint arXiv:1711.08585*, 2017. → pages 24

[HLL+17] Hou-Ning Hu, Yen-Chen Lin, Ming-Yu Liu, Hsien-Tzu Cheng, Yung-Ju Chang, and Min Sun. Deep 360 pilot: Learning a deep agent for piloting through 360^o sports videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. → pages 19

[HPV04] Jean-Bernard Hayet, Justus Piater, and Jacques Verly. Robust incremental rectification of sports video sequences. In *British Machine Vision Conference (BMVC)*, 2004. → pages 27

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. → pages 24, 62, 74, 76

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Random forests. In *The elements of statistical learning*, pages 587–604. Springer, 2009. → pages 9

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. → pages 9, 26

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. → pages 12

[IKBL08] Hemant Ishwaran, Udaya B Kogalur, Eugene H Blackstone, and Michael S Lauer. Random survival forests. *The Annals of Applied Statistics*, pages 841–860, 2008. → pages 11, 12, 79, 82

[IZZE17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. → pages 15, 27, 49

[Jel97] Frederick Jelinek. *Statistical methods for speech recognition*. MIT press, 1997. → pages 23

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM international conference on Multimedia (ACMMM)*, 2014. → pages 13

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014. → pages 13

[Kel60] Henry J Kelley. Gradient theory of optimal flight paths. *American Rocket Society Journal*, 30(10):947–954, 1960. → pages 12

[KKK00] Daiichiro Kato, Tetsuo Katsuura, and Hideo Koyama. Automatic control of a robot camera for broadcasting based on cameramen's techniques and subjective evaluation and analysis of reproduced images. *Journal of physiological anthropology and applied human science*, 19(2):61–71, 2000. → pages 19

[KMO10] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010. → pages 89

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, 2012. → pages 12

[KYA+97] Daiichiro Kato, Mitsuho Yamada, K Abe, A Ishikawa, K Ishiyama, and M Obata. Analysis of the camerawork of broadcasting cameramen. *SMPTE journal*, 106(2):108–116, 1997. → pages 19, 20

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. → pages 12

[LC15] Christophe Lino and Marc Christie. Intuitive and efficient camera control with the toric space. *ACM Transactions on Graphics (TOG)*, 34(4):82, 2015. → pages 22

[LCCL13] Jingchen Liu, Peter Carr, Robert T Collins, and Yanxi Liu. Tracking sports players with context-conditioned motion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. → pages 53, 54, 93

[LCLJH17] Keyu Lu, Jianhui Chen, James Little J., and He Hangen. Light cascaded convolutional neural networks for accurate player detection. In *British Machine Vision Conference (BMVC)*, 2017. → pages 35

[Lin13] Christophe Lino. *Virtual camera control using dynamic spatial partitions*. PhD thesis, Université Rennes 1, 2013. → pages 18

[LMNF09] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision (IJCV)*, 81(2):155, 2009. → pages 9

[LMP01] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, 2001. → pages 23

[LMPDB16] Giuseppe Lisanti, Iacopo Masi, Federico Pernici, and Alberto Del Bimbo. Continuous localization and mapping of a pan tilt zoom camera for wide area tracking. *Machine Vision and Applications*, 27(7):1071–1085, 2016. → pages 93

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. → pages 26

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. → pages 13, 15

[LTLM13] Wei-Lwun Lu, J-A Ting, James J Little, and Kevin P Murphy. Learning to track and identify players from broadcast sports videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(7):1704–1716, 2013. → pages 83

[LW+02] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002. → pages 11

[LZHT15] Yunting Li, Jun Zhang, Wenwen Hu, and Jinwen Tian. Method for pan-tilt camera calibration using single control point. *Journal of the Optical Society of America (JOSA) A*, 32(1):156–163, 2015. → pages 31, 32

[MAP+10] Aditya Mavlankar, Piyush Agrawal, Derek Pang, Sherif Halawa, Ngai-Man Cheung, and Bernd Girod. An interactive region-of-interest video streaming system for online lecture viewing. In *IEEE International Packet Video Workshop (PV)*, pages 64–71, 2010. → pages 17

[MCT+17] Lili Meng, Jianhui Chen, Frederick Tung, James Little J., Julien Valentin, and Clarence Silva. Backtracking regression forests for accurate camera relocalization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017. → pages 11, 33

[MFP00] Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML*, 2000. → pages 23

[MJ99] Larry Medsker and Lakhmi C Jain. *Recurrent neural networks: design and applications*. CRC press, 1999. → pages 23

[MS99] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999. → pages 23

[Mur02] Kevin P Murphy. Hidden semi-markov models (hsmms). Technical report, Computer Science Department, University of British Columbia, 2002. → pages 65

[NAK15] Prabhu Natarajan, Pradeep K Atrey, and Mohan Kankanhalli. Multi-camera coordination and control in surveillance systems: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(4):57, 2015. → pages 21, 23

[NMD+17] Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics (TOG)*, 36(4):132, 2017. → pages 19

[NO04] Dennis Nieuwenhuisen and Mark H Overmars. Motion planning for camera movements. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2004. → pages 20

[NVS06] Mukund Narasimhan, Paul Viola, and Michael Shilman. Online decoding of markov models under latency constraints. In *International Conference on Machine Learning (ICML)*, 2006. → pages 60

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006. → pages 71

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision (ECCV)*, 2016. → pages 93

[OIF09] Makoto Okuda, Seiki Inoue, and Mahito Fujii. Machine learning of shooting technique for controlling a robot camera. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2009. → pages 18

[OLL04] Kenji Okuma, James J Little, and David G Lowe. Automatic rectification of long image sequences. In *Asian Conference on Computer Vision (ACCV)*, 2004. → pages 26

[Owe15] Jim Owens. *Television sports production*. CRC Press, 2015. → pages 79

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Neural Information Processing Systems (NIPS) Autodiff Workshop*, 2017. → pages 13

[PGvG12] Viorica Pătrăucean, Pierre Gurdjos, and Rafael Grompone von Gioi. A parameterless line segment and elliptical arc detector with enhanced ellipse fitting. In *European Conference on Computer Vision (ECCV)*. 2012. → pages 27

[PMKS10] Derek Pang, Sameer Madan, Serene Kosaraju, and Tarun Vir Singh. Automatic virtual camera view generation for lecture videos. Technical report, Stanford University, 2010. → pages 17, 18

[PP95] Claudio S Pinhanez and Alex Paul Pentland. Intelligent studios: Using computer vision to control tv cameras. In *IJCAI Workshop on Entertainment and AI/Alife*, 1995. → pages 16, 17

[Pri17] PricewaterhouseCoopers. At the gate and beyond: Outlook for the sports market in North America through 2021. Technical report, PricewaterhouseCoopers, United States, 2017. → pages 1

[PZVP11] Jens Puwein, Remo Ziegler, Julia Vogel, and Marc Pollefeys. Robust multi-view camera calibration for wide-baseline camera networks. In *IEEE Workshop on Applications of Computer Vision (WACV)*, 2011. → pages 26

[QT07] Faisal Z Qureshi and Demetri Terzopoulos. Surveillance in virtual reality: System design and multi-camera control. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. → pages 23

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. → pages 12

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015. → pages 15, 49

[RGB11] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. → pages 25

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986. → pages 12

[RKSCS18] Konstantinos Rematas, Ira Kemelmacher-Shlizerman, Brian Curless, and Steve Seitz. Soccer on your tabletop. In *CVPR*, 2018. → pages 93

[RM51] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. → pages 13

[Ros13] Stéphane Ross. *Interactive learning for sequential decisions and predictions*. PhD thesis, Carnegie Mellon University, 2013. → pages 25

[RVRK16] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision (ECCV)*, pages 102–118, 2016. → pages 94

[SBGJ18] Rahul Anand Sharma, Bharath Bhat, Vineet Gandhi, and CV Jawahar. Automated top view registration of broadcast football videos. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018. → pages 27, 41, 46, 47, 48

[Sca11] Davide Scaramuzza. 1-point-RANSAC structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints. *International Journal of Computer Vision (IJCV)*, 95(1):74, 2011. → pages 33

[SEZ+13] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. → pages 13

[SG64] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964. → pages 50, 54, 60

[SG17] Yu-Chuan Su and Kristen Grauman. Making 360^o video watchable in 2D: Learning videography for click free viewing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. → pages 19, 22

[SJG16] Yu-Chuan Su, Dinesh Jayaraman, and Kristen Grauman. Pano2vid: Automatic cinematography for watching 360^o videos. In *Asian Conference on Computer Vision (ACCV)*, 2016. → pages 19

[SLHS17] Paul Hongsuck Seo, Andreas Lehrmann, Bohyung Han, and Leonid Sigal. Visual reference resolution using attention memory for visual dialog. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. → pages 94

[SLS16] Alireza Shafaei, James J Little, and Mark Schmidt. Play and learn: Using video games to train computer vision models. In *British Machine Vision Conference (BMVC)*, 2016. → pages 94

[SO02] Rares Stanciu and Paul Y Oh. Designing visually servoed tracking to augment camera teleoperators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002. → pages 20

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. → pages 62

[SZS12] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012. → pages 73

[TBF+15] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3D convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. → pages 73, 85, 87

[TCNS15] Masaki Takahashi, Simon Clippingdale, Masahide Naemura, and Masahiro Shibata. Estimation of viewers ratings of tv programs based on behaviors in home environments. *Multimedia Tools and Applications*, 74(19):8669–8684, 2015. → pages 22

[Tho07] Graham Thomas. Real-time camera tracking using sports pitch markings. *Journal of Real-Time Image Processing*, 2(2-3):117–132, 2007. → pages 5, 26, 27, 29

[TI17] Fei Tang and Hemant Ishwaran. Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2017. → pages 82

[TJHA05] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(September):1453–1484, 2005. → pages 23

[TK91] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, School of Computer Science, Carnegie Mellon University, 1991. → pages 26

[TNN+98] Kumiko Tanaka, Hideyuki Nakashima, Itsuki Noda, Kôiti Hasida, Ian Frank, and Hitoshi Matsubara. MIKE: An automatic commentary system for soccer. In *International Conference on Multi Agent Systems*, pages 285–292, 1998. → pages 94

[VAHR98] Dirk Voelz, Elisabeth André, Gerd Herzog, and Thomas Rist. Rocco: A RoboCup soccer commentator system. In *Robot Soccer World Cup*, 1998. → pages 94

[VFSH01] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich. Viewpoint selection using viewpoint entropy. In *International Symposium on Vision, Modeling and Visualization (VMV)*, 2001. → pages 23

[VVA15] H Cevallos Valdiviezo and Stefan Van Aelst. Tree-based prediction on incomplete data using imputation or surrogate decisions. *Information Sciences*, 311:163–181, 2015. → pages 81

[WCW+16] Pei-Chih Wen, Wei-Chih Cheng, Yu-Shuen Wang, Hung-Kuo Chu, Nick C Tang, and Hong-Yuan Mark Liao. Court reconstruction for camera calibration in broadcast basketball videos. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 22(5):1517–1526, 2016. → pages 26, 41

[WHE+18] Xueting Wang, Kensho Hara, Yu Enokibori, Takatsugu Hirayama, et al. Personal viewpoint navigation based on object trajectory distribution for multi-view videos. *IEICE Transactions on Information and Systems*, 101(1):193–204, 2018. → pages 22

[WL15] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3D pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. → pages 14, 42

[WM08] Stefan Winkler and Praveen Mohandas. The evolution of video quality measurement: From PSNR to hybrid metrics. *IEEE Transactions on Broadcasting*, 54(3):660–668, 2008. → pages 21

[WMHM14] Xueting Wang, Yuki Muramatu, Takatsugu Hirayama, and Kenji Mase. Context-dependent viewpoint sequence recommendation system for multi-view video. In *IEEE International Symposium on Multimedia (ISM)*, 2014. → pages 21, 70, 84, 90

[WRKS16] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. → pages 93

[WXC+08] Jinjun Wang, Changsheng Xu, Engsiong Chng, Hanqing Lu, and Qi Tian. Automatic composition of broadcast sports video. *Multimedia Systems*, 14(4):179–193, 2008. → pages 21

[XSJL17] Fanyi Xiao, Leonid Sigal, and Yong Jae Lee. Weakly-supervised visual grounding of phrases with linguistic structures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. → pages 94

[YF05] Takao Yokoi and Hironobu Fujiyoshi. Virtual camerawork for generating lecture video from high resolution images. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2005. → pages 17, 19

[YJEP08] Chun-Nam John Yu, Thorsten Joachims, Ron Elber, and Jaroslaw Pillardy. Support vector training of protein alignment models. *Journal of Computational Biology*, 15(7):867–880, 2008. → pages 23

[Yoo09] Byung-Jun Yoon. Hidden Markov models and their applications in biological sequence analysis. *Current genomics*, 10(6):402–415, 2009. → pages 23

[Yu10] Shun-Zheng Yu. Hidden semi-markov models. *Artificial intelligence*, 174(2):215–243, 2010. → pages 65

[Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(11):1330–1334, 2000. → pages 9, 31

[ZRCH08] Cha Zhang, Yong Rui, Jim Crawford, and Li-Wei He. An automated end-to-end lecture capture and broadcasting system. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 4(1):6, 2008. → pages 20

Appendix A

Two-point Calibration Method

A.1 Focal length from two points

$$f^2 = \frac{2(d^2ab - c^2)}{2c - d^2(a+b) + \sqrt{(d^2(a+b) - 2c)^2 - 4(d^2ab - c^2)(d^2 - 1)}} \tag{A.1}$$

where

$$a = \mathbf{x}_1^\mathsf{T}\mathbf{x}_1 \quad b = \mathbf{x}_2^\mathsf{T}\mathbf{x}_2$$
$$c = \mathbf{x}_1^\mathsf{T}\mathbf{x}_2 \quad d = \bar{\mathbf{X}}_1^\mathsf{T}\bar{\mathbf{X}}_2$$

where $\bar{\mathbf{X}} = \frac{\mathbf{X}-\mathbf{C}}{\|\mathbf{X}-\mathbf{C}\|}$, $\mathbf{X}$ are 3D world points, $\mathbf{x}$ are pixel locations and $\mathbf{C}$ is the camera center.

A.2 Single point pan-tilt camera calibration

Recall our camera model:

$$\mathtt{P} = \mathtt{K}\mathtt{Q}_\phi\mathtt{Q}_\theta\mathtt{S}[\mathtt{I}|-\mathbf{C}], \tag{A.2}$$

where $\mathbf{C}$, $\mathtt{S}$ and $\mathtt{K}$ are known for a pan-tilt camera.

There is one 3D point $P_w = [X,Y,Z]^\mathsf{T}$ in the world coordinate and its projection p_i in the image. We have

$$(X,Y,Z) = \mathtt{S}(P_w - \mathbf{C})$$

and

$$(U,V,1)^{\mathsf{T}} = (\mathtt{K}^{-1}p_i)^{\mathsf{T}}.$$

As a result, we have

$$\begin{bmatrix} U \\ V \\ 1 \end{bmatrix} = \mathtt{Q}_\phi \mathtt{Q}_\theta \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} c_p & 0 & -s_p \\ s_t s_p & c_t & s_t c_p \\ c_t s_p & -s_t & c_t c_p \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{A.3}$$

where s_p, c_p, s_t and c_t are short for $\sin(\theta), \cos(\theta), \sin(\phi), \cos(\phi)$. Because the cross product of two sides of (A.3) is a zero vector (i.e., cross product of a vector with itself), we have:

$$\begin{bmatrix} A_{pan} & B_{pan} \end{bmatrix} \begin{bmatrix} c_t \\ s_t \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{A.4}$$

where

$$A_{pan} = \begin{bmatrix} VXs_p + VZc_p - Y & -VY - Xs_p - Zc_p \\ UXs_p + UZc_p & -YU \end{bmatrix}$$
$$B_{pan} = \begin{bmatrix} 0 \\ Zs_p - Xc_p \end{bmatrix}$$

From (A.4), we have

$$c_t = \frac{(VY + Xs_p + Zc_p)(Xc_p - Zs_p)}{\det(A_{pan})} \tag{A.5}$$

and

$$s_t = \frac{(VXs_p + VZc_p - Y)(Xc_p - Zs_p)}{\det(A_{pan})} \tag{A.6}$$

where $det_{Apan} = U(Y^2 + (Zc_p + Xs_p)^2)$. Because $c_t^2 + s_t^2 = 1$, we have a quadratic equation of t_p (short for $\tan(\theta)$) by

$$a{t_p}^2 + bt_p + c = 0 \tag{A.7}$$

where

$$a = (V^2+1)Z^2 - U^2(X^2+Y^2)$$
$$b = -2XZ(U^2+V^2+1)$$
$$c = (V^2+1)X^2 - U^2(Y^2+Z^2).$$

From (A.7), we can calculate *pan* angle up to 2 solutions. We can eliminate one of them by setting the valid range to $(-90^o, 90^o)$. The tilt angle can be calculated from (A.5) and (A.6).

Appendix B

Overlapped hidden Markov model

B.1 Parameters of OHMM

Window size. We first fix the overlap ratio as $r = \frac{w-1}{w}$ (fully overlapped) and test the influence of different window sizes.

Fig. B.1 shows the results with different window sizes from 64 to 320 frames. Comparing with the ground truth, we found that both $w = 192$ and $w = 256$ achieve satisfactory results. $w = 192$ is better in the velocity prediction and $w = 256$ produces smoother camera trajectories. We further study how well the two error measurements describe errors. We plot the corresponding $RMSE_p$, $RMSE_v$ in Fig. B.2. Comparing Fig. B.1 with Fig. B.2, we found that $RMSE_p$ and $RMSE_v$ are reasonable but not the perfect error measurements. It is not surprising because the quantitative measurement is still an opening problem in camera planning. Relatively, $RMSE_v$ is better at deciding which signal achieves the best result.

Overlap ratio. The computational complexity increases when the overlap ratio approaches 1.0. One concern is the OHMM method cannot give a quick response. However, we found that with the proper setting of the overlap ratio, our method can achieve real-time response without sacrificing prediction quality.

We evaluate the influence of overlap by fixing other parameters as $N_h = 786$, $N_{tr} = 5$ and $w = 256$. Fig. B.3 shows the prediction result with different overlap ratios. When the overlap is 0.95, the predictions are almost the same as the fully overlapping predictions ($RMSE_p$ = 0.28 and $MRSE_v$ = 0.01). However, the

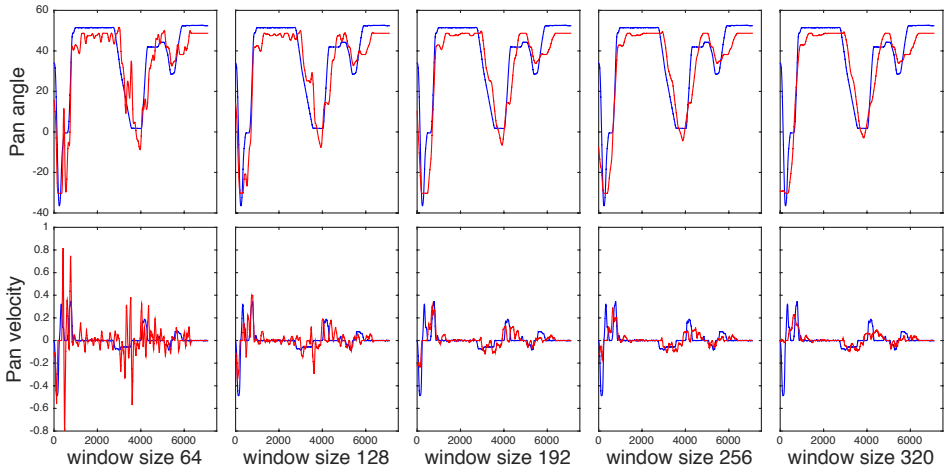


Figure B.1: OHMM result with different window sizes. By visually comparing the prediction (red) and the ground truth (blue), the window size 256 gives the best result.

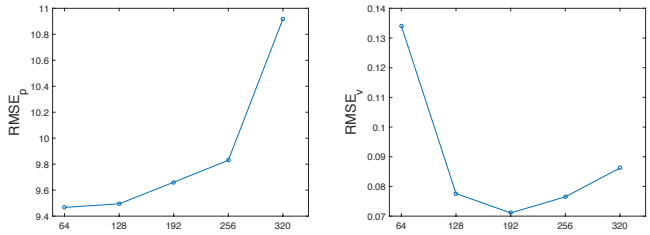


Figure B.2: Quantitative errors with different windows sizes. $RMSE_p$ grows when the window size grows. $RMSE_v$ suggests that $w = 192$ achieve the best result, which approximately agrees with the visual inspection in Fig. B.1.

speed is about 10 times faster (2.94 vs. 33.59 ms/frame) than the fully overlapping method. It means that the OHMM method can achieve real-time response without sacrificing prediction quality. In the following experiment, we fix the overlap ratio at 0.95.

Overlap vs low-pass filter. To demonstrate the importance of OHMM on smoothing signals, we compare the results with and without overlap. Fig.B.4 shows the $RMSE_v$ errors of using the combination of the overlap and the Savitzky-Golay (SG)

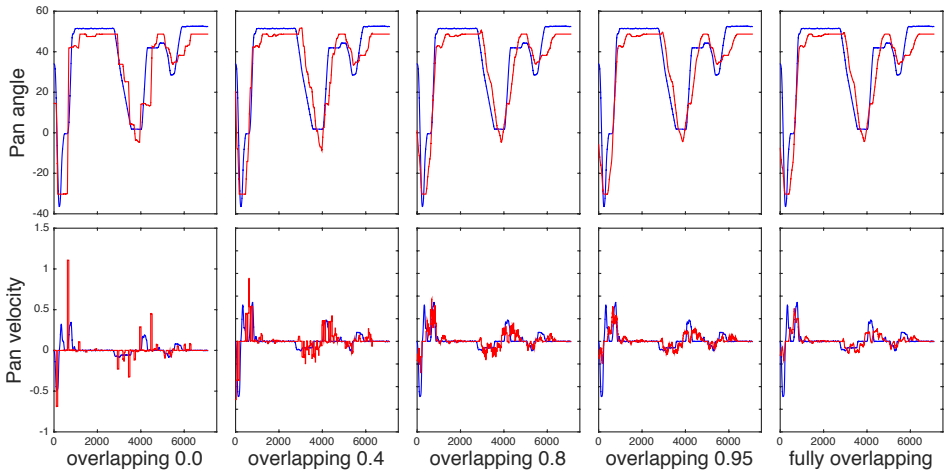


Figure B.3: OHMM result with different overlap ratios. Blue is the ground truth and red is the prediction. When the overlap ratio is 0.95, the prediction is unnoticeable from the result with fully overlapping.

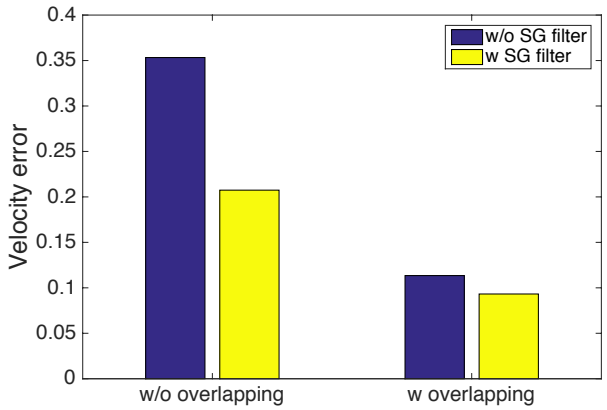


Figure B.4: $RMSE_v$ with and without the overlap and the SG filter. The overlap plays the dominant role in decreasing $RMSE_v$ error and smoothing the signal.

filter. The error drops from 0.35 to 0.11 by using overlap only. The error further drops from 0.11 to 0.09 by using the SG filter further. It means the overlap alone removes 92.3% of the total dropped error. On the other hand, using the SG filter alone removes about 56.1% of the total dropped error. It shows that the overlap plays the dominant role in smoothing the signal.