

Computer Vision for Hockey Video Curation

HEMANTH PIDAPARTHY

A DISSERTATION SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

April 2022

© Hemanth Pidaparthi, 2022

Abstract

Over the last few years, computer vision has been increasingly applied to sports video analysis. With data becoming increasingly available, computer vision-based models are being actively investigated for tasks such as ball tracking, player detection and tracking, and extracting analytical insights into the game. Extracting meaningful information from the game is useful for both coaches and players to improve performance, and for broadcasters to enrich the viewing experience. This has created a lot of interest in applying computer vision based techniques for sports video analysis.

This thesis considers the general problem of hockey video analysis. Application of computer vision based techniques for hockey video analysis is challenging because of the small size of the puck, fast and non-smooth movement of the players, frequent occlusions and varying lighting conditions.

In this thesis, I present my research work on computer vision for hockey video curation. I discuss three problems: 1) automatic sports videography, 2) play segmentation of hockey videos and 3) automatic homography estimation.

Broadcast hockey videos are often recorded by professional cameramen using a PTZ camera. They constantly move their camera to follow the play. Professional videography is expensive for amateur games or minor league hockey games. This motivates the development of a low-cost solution for automatic videography of hockey games. This is feasible if we develop a system that can accurately predict the puck location on each frame. Then, centered around the predicted puck location, we can dynamically crop a wide-field video to generate a zoomed-in video following the play. We developed a system for predicting the puck location on each frame (with the video as input) using machine learning techniques. For this we need large training datasets that contain videos annotated with the ground truth puck location on each video frame. Due to the small size of the puck and motion blur, obtaining ground truth puck location from wide-field videos is very challenging. We used a method motivated by psychophysics to generate accurate ground truth of the puck location from wide-field video. This involved multiple human observers tracking the puck location on a video played back at different frame rates and determining the best

frame rate for ground truthing the videos. We use this ground truth data to train a novel deep network regressor to estimate the location of the puck on each frame. We present results that show the system achieves good accuracy and generalization across different camera viewpoints.

The automatic videography system delivers continuous video over the entire game. Typical hockey games that feature 40-60 minutes of active play are played over the course of 60-110 minutes with breaks in play due to warm-ups, fouls, etc. Automatic identification of these stoppages would allow for a temporally compressed video that is easier to watch and that may engage better with the audience. Further, identification of these stoppages allows for placement of ads and notices. We propose a novel solution for automatically identifying periods of play and no-play in hockey videos. We compute visual cues from the output of a deep neural network classifier trained to classify each frame as belonging to play or no-play. We compute the auditory cues from the referee’s whistle using a Wiener filter. We combine the visual cues with optional auditory cues using a hidden Markov model (HMM). We present results that show good generalization of the system across different games, cameras and lighting conditions. We also show that due to domain shift between training and test data, unsupervised domain adaptation is important to derive benefits from auditory cues.

In the work described in the two paragraphs above, the games are recorded from a stationary camera whose field-of-view was the entire ice rink. Since the cameras were stationary, the homography (i.e., the mapping between the image plane and the ice rink) did not change across different videos. This is not true for broadcast videos where professional videographers pan, tilt and zoom their cameras to record the game. As a result, the homography changes every frame. Knowing the homography between the camera plane and ice rink can be useful for hockey video analysis such as projection of graphics onto the ice surface. Current methods for homography estimation need large training datasets with detailed annotations. We propose an alternative approach that does not require any large annotated dataset. We exploit the consistency of colours used for markings in ice hockey. We model the colours as a multi-variate Gaussian and then use a two-step approach to search for the homography that aligns the colours of the template image to that of a test image. We show that this method for homography estimation delivers good performance.

Acknowledgement

I would like to thank everyone who helped me in my PhD journey and made this thesis possible.

First and foremost, I would like to thank my advisor Prof. James Elder for being an incredible advisor, mentor and teacher. His approach towards solving practical problems and his ability to provide meaningful criticisms have been extremely helpful during the course of my PhD.

To everyone in the Elderlab: Thank you for making my journey memorable with enriching discussions about work and life. To Bob, Kai and Anna: Thank you for being a sounding board and helping me with any issue I faced in the lab. To Kris and Shaiyan: Thank you for the brainstorming sessions and technical discussions. To Kartikeya: Thank you for the help in coding and building a practical system from the research. To Gong: Thank you for all the help with my research, coding, and the interesting discussions about world politics.

To Yogi, Rokhaya, Sweta, Fatima and Hanika: Thank you for making my stay in Canada memorable.

To my parents, Sastry and Padmaja, and my brother Vasanth: Thank you for the love and encouragement. To my wife Divya: Thank you for your patience, support and for making this journey wonderful!

Table of Contents

Abstract	ii
Acknowledgement	iv
Table of Contents	v
List of Tables	viii
List of Figures	xi
1 Introduction	1
2 Automatic Sports Videography	4
2.1 Prior work	5
2.1.1 Ball/puck tracking	6
2.1.2 Broadcast camera angle planning	13
2.2 Contributions	13
2.3 Dataset	14
2.3.1 Dataset 1	14
2.3.2 Dataset 2	15
2.4 Ground truthing	19
2.4.1 Apparatus	19
2.4.2 Procedure	20
2.4.3 Analysis	20
2.4.4 Systematic lag	21
2.5 Algorithm	23

2.5.1	Viewpoint invariance	24
2.5.2	Deep network regressor	25
2.5.3	Temporal smoothing	32
2.5.4	Dynamic cropping	32
2.5.5	Two-stage cascaded deep network regressor	33
2.6	Evaluation	34
2.6.1	Ablation study	35
2.6.2	Within-camera performance	36
2.6.3	Between-camera performance	39
2.6.4	Puck location visualizations	41
2.7	Conclusion & future work	45
3	Automatic Play Segmentation	47
3.1	Prior work	48
3.2	Contributions	49
3.3	Dataset	50
3.4	Method	53
3.4.1	Visual cues	54
3.4.2	Auditory cue	59
3.4.3	Hidden Markov model	64
3.4.4	Domain adaptation	71
3.5	Results	74
3.5.1	Within-camera performance (2-state HMM)	75
3.5.2	Between-camera performance (2-state HMM)	76
3.5.3	Within-camera performance (4-state HMM)	76
3.5.4	Domain adaptation performance	77
3.6	Conclusions & future work	78
4	Automatic Homography Estimation	91
4.1	Prior work	92
4.1.1	Traditional methods	93
4.1.2	Deep learning methods	94

4.2	Contributions	96
4.3	Dataset	96
4.4	Method	97
4.4.1	Manual rectification	98
4.4.2	Colour modelling	98
4.4.3	Post-processing techniques	99
4.4.4	Initial homography estimation	102
4.4.5	Final homography estimation	106
4.5	Run time analysis	109
4.6	Conclusions & future work	111
5	Conclusions and Future Work	113
	Bibliography	116
	Appendix A Deep network architecture optimization	130

List of Tables

2.1	Duration and resolution of the videos of each camera from Dataset 1. . . .	15
2.2	Duration and video resolution for each game across both cameras in Dataset 2.	19
2.3	Precision and recall of the various player detection models at the maximum F-score.	28
2.4	Optimal time constants λ_{opt} and σ_{opt} for causal and non-causal smoothing, respectively	32
2.5	Ablation study to evaluate the contribution of the three input feature maps.	37
2.6	Within-camera performance evaluated through leave-one-out cross-validation across all games.	37
2.7	Within-camera puck occupancy performance. The resolution of the zoomed-in video relative of the wide-field rectified image is indicated in columns 2 and 3. The accuracy is measured as the proportion of frames where the puck is visible in the zoomed-in video. Recall that the size of the rectified image is 6360×1080	39
2.8	Between-camera performance of the first-stage network with and without viewpoint invariance module.	40
2.9	Between-camera performance of the system using the 3-channel feature map of optic flow (O) and binary player mask (P).	40
2.10	Between-camera puck occupancy performance. In a similar manner to Table 2.7, the resolution of the zoomed-in video relative to the wide-field rectified image is indicated in columns 2 and 3, and the puck occupancy accuracy is measured. Recall that the size of the rectified image is 6360×1080	42

2.11	KL divergence of the APT1 and APT2 heatmaps with the ground truth heatmap.	42
2.12	Average error of puck zone occupancy for the APT1 and APT2 predictions.	44
3.1	Cropped video resolutions for all games in the input dataset. The vertical cropping for Camera 3 took two values depending on the camera tilt.	53
3.2	Duration of each game in the input dataset.	53
3.3	AUC scores for visual play/no-play classification.	59
3.4	The average cross-validated AUC score of the four whistle detectors for Camera 3.	63
3.5	Mean initial state probabilities for each camera.	68
3.6	Mean state transition probabilities for each camera.	69
3.7	Cross-validated AUC scores as a function of the number of GMM components. OF: maximum optic flow cue. DV: deep visual cue.	70
3.8	Histogram of the visual and auditory cues were computed for the specified number of bins and modelled using a Gaussian KDE. The AUC score for classifying play and no-play frames was computed. OF: maximum optic flow cue. DV: deep visual cue.	71
4.1	Reprojection error of the initial homography estimates for each possible combination of design parameters P1-P5.	105
4.2	Reprojection error of the final homography estimates for each possible combination of design parameters P1-P5.	107
4.3	Run time analysis of the system at various resolutions of the test image and template image.	109
A.1	Different convolutional layer architectures explored in the first phase of experiments.	131
A.2	Errors on the training, validation and test datasets for each network architecture at a given input data size, in the first phase of experiments.	132
A.3	Different convolutional layer architectures explored in the second phase of experiments.	132

A.4	The structure of the convolutional layers for each network in the second phase of experiments.	133
A.5	Errors on the training, validation and test datasets for each network architecture in the second phase of experiments, at input data sizes of 138×58 and 172×72	133
A.6	Errors on the training, validation and test datasets for each network architecture in the second phase of experiments, at input data sizes of 215×90 and 258×108	134
A.7	Different fully connected layer architectures explored in the second phase of experiments.	134
A.8	Errors on the training, validation and test datasets for different architectures of the fully connected layers.	135

List of Figures

1.1	With a quick glance at these images, we are able to interpret the scene and understand what is happening.	1
2.1	Automatic videography for ice hockey. Top: The 720p dynamic zoom view (blue rectangle) centered at the estimated puck location (pink dot). Bottom: Corresponding curated view.	6
2.2	Example video frames from Dataset 1.	16
2.3	Example frames from Dataset 2 after correction for radial distortion. . . .	17
2.4	Camera 2 view construction. (a) Example frames from two cameras. (b) Rink template. (c) Stitched image with homography control points.	18
2.5	Mean and standard error of within-observer and between-observer deviations as a function of playback frame rate.	21
2.6	Systematic lag in the observer trackpad data in (a) psychophysical time and (b) game time.	23
2.7	Pipeline of the automatic sports videography system. A 6-channel input feature map consisting of the RGB image, optic flow and binary player mask is passed to a deep network regressor that predicts the puck location on the input image. A crop of the high-resolution wide-field video centered at the prediction is used as the curated video following the play.	24
2.8	Rectification of the cameras to a known template viewpoint.	26
2.9	Horizontal (green) and vertical (red) components of the optic flow field. .	27
2.10	Faster RCNN experiments. Right image is a zoomed-in view of left image.	29
2.11	Detected players and the subsequent binary player mask that is one of the input feature maps for the deep network regressor.	30

2.12	Architecture of the deep network regressor.	31
2.13	RMS error on the training data with causal and non-causal temporal smoothing on the estimates from the deep network regressor.	33
2.14	Two-stage cascaded deep network regressor. Top: First-stage network. Bottom: Second-stage-network. The features from the final convolutional layer of the second stage network (green) are concatenated with features from the final convolutional layer of the first stage network (red). The size of the input data to the first-stage regressor is 240×60 , while that to the second-stage regressor is 160×90	35
2.15	Pipeline of the automatic sports videography system using the two-stage cascaded deep network regressor. A 6-channel input feature map consisting of the RGB image, optic flow and binary player mask is passed to the first-stage deep network regressor. A crop of the high-resolution wide-field video centered at the first-stage prediction forms the input for the second-stage network. The features from the final convolutional layers of the first-stage (red) are combined with the features from the final convolutional layer of the second-stage (green).	36
2.16	Left: Sample results for test images from Camera 2. Cyan and pink dots indicate ground truth puck location and the puck location estimated by APT2 + smoothing, respectively. The blue rectangle indicates the 1280×720 crop region. Right: Corresponding zoomed-in view.	38
2.17	Accuracy of the within-camera performance as a function of RMS error. Left: Camera 1. Right: Camera 2.	38
2.18	Accuracy of the between-camera performance as a function of RMS error. Left: Evaluating on Camera 2. Right: Evaluating on Camera 1.	41
2.19	Puck location heatmaps for all games recorded by Camera 1. Left: Ground truth data. Middle: APT1 predictions. Right: APT2 predictions. Units: per m^2	43
2.20	Puck location heatmaps for all games recorded by Camera 2. Left: Ground truth data. Middle: APT1 predictions. Right: APT2 predictions. Units: per m^2	44

2.21	Puck zone occupancy for all games recorded by Camera 1. Left: Ground truth data. Middle: APT1 predictions. Right: APT2 predictions.	45
2.22	Puck zone occupancy for all games recorded by Camera 2. Left: Ground truth data. Middle: APT1 predictions. Right: APT2 predictions.	46
3.1	Example frames from the dataset, after correction for radial distortion and vertical cropping to the rink boundaries.	51
3.2	Non-rink regions are manually masked out for all Camera 3 games. Left: images vertically cropped to the rink boundaries. Right: non-rink regions masked out.	52
3.3	Pipeline of our system for automatic play segmentation of hockey videos.	54
3.4	Example optic flow field within bounding boxes of detected players.	55
3.5	Error rate as a function of the L^p exponent used to aggregate the optic flow field.	55
3.6	Input data to deep visual play/no-play classifier.	57
3.7	Deep visual classifier trained to classify each frame as belonging to play or no-play classes.	57
3.8	Average power spectrum for whistle and non-whistle intervals. The right plot shows a zoom-in on the main frequency band of the whistle (2-3 kHz).	60
3.9	The Wiener filter response $H(f)$ for each of the three games. Right panels show a zoom-in of the response of a rectified Wiener filter in the main frequency range of 2-3 kHz.	62
3.10	State transition graphs for (a) 2-state and (b) 4-state HMMs.	69
3.11	Conditional probability densities for the maximum optic flow visual cue on all games across all three cameras.	72
3.12	Conditional probability densities for the deep visual cue on all games across all three cameras.	80
3.13	Conditional densities for the auditory cue of Wiener filter 3 detector on games from Camera 3.	81
3.14	Visual cue of maximum optic flow and auditory cue of Wiener filter 3 detector for an example video segment from Camera 3 Game 1.	81
3.15	Within-camera performance of the 2-state HMM (visual cue only).	82

3.16	Between-cameras performance compared to within-camera performance on all three cameras.	83
3.17	Performance of the 2-state HMM and 4-state HMM on Camera 3. The 4-state HMM combined visual and auditory cues, while the 2-state HMM used only the visual cues. OF: maximum optic flow cue; DV: deep visual cue.	84
3.18	Unconditional densities of the deep visual cue learnt from the training data shown on the test data histogram for each game recorded using Camera 3. Left: Before adaptation. Right: After adaptation.	85
3.19	Unconditional densities of the auditory cue learnt from the training data shown on the test data histogram for each game recorded using Camera 3. Left: Before adaptation. Right: After adaptation.	86
3.20	Performance of the 2-state HMM and 4-state HMM after domain adaptation on Camera 3. OF: maximum optic flow cue; DV: deep visual cue; DA: domain adaptation.	87
3.21	Unconditional densities of the deep visual cue learnt from the training data shown on the test data histogram for each game recorded using Camera 1. Left: Before adaptation. Right: After adaptation.	88
3.22	Unconditional densities of the deep visual cue learnt from the training data shown on the test data histogram for each game recorded using Camera 2. Left: Before adaptation. Right: After adaptation.	89
3.23	Performance of the 2-state HMM before and after domain adaptation on all games from Camera 1 and Camera 2. DV: deep visual cue; DA: domain adaptation.	90
4.1	(a) Template ice hockey rink and (b) sample test image from a moving camera broadcast hockey video.	92
4.2	Example key-frames from the dataset.	97
4.3	(a) Control points (black) on the template image. (b) Composed on the key-frame.	99
4.4	Colour markings on the template ice rink.	99
4.5	Image crops used for modelling the likelihood for each colour marking.	100

4.6	Left: Broadcast image. Right: Colour markings with the highest likelihood at each pixel. For visualization purposes, pixels with all colour likelihoods below $1e-16$ are labelled as black.	100
4.7	Faster RCNN player detections (left panels) and corresponding binary player masks (right panels).	101
4.8	Colour marking with the highest likelihood at each pixel of a broadcast image after applying post-processing techniques on the likelihoods of each colour marking. (a) Broadcast image. (b) Colour marking with highest likelihood at each pixel. (c) Image (b) masked by inverse binary player mask. (d) Median filtering applied on image (c). Black pixels indicate regions where all colour likelihoods are less than $1e-16$	102
4.9	Signed distance transform of the template image for each colour marking.	104
4.10	Left: Test images. Right: Projected template mask superimposed on test image based on the initial homography estimate.	108
4.11	Left: Initial homography estimate. Right: Final homography estimate. . .	110

Chapter 1

Introduction

The human visual system allows for a comprehensive understanding of the complex visual world around us. We can glance at a picture (Fig 1.1), and quickly separate foreground and background, decide what objects are present (classification), where they are (localization), and what information is being conveyed by the image (scene analysis). We are also able to quickly interpret videos and form an understanding of how objects are moving (tracking), how events are unfolding dynamically (event detection) and what actions are being performed (action recognition).



(a)



(b)

Figure 1.1: With a quick glance at these images, we are able to interpret the scene and understand what is happening.

Computer vision focuses on developing algorithms that emulate human visual performance. There are many problems in computer vision, such as classification, localization,

segmentation, action recognition and event detection [25, 65, 40, 111, 39]. There are applications where computer vision algorithms can be adopted for analysis of dynamically varying scenes. These applications include traffic analysis [77, 50], autonomous driving [55, 135] and sports video analysis [111, 76]. All these applications involve complex interactions of multiple objects. The effects of one object on its neighbours changes dynamically and these are important for understanding and interpreting the video. This is particularly relevant in the case of sports.

Sports video analysis is a topic of interest in current computer vision research. Computer vision techniques have been applied to several tasks such as ball tracking [29, 75, 86, 57], player detection and tracking [96, 73, 34, 74, 107, 109, 2, 128], player identification and team affiliation [69, 67, 14, 64], pose estimation [11, 9], action recognition [52, 10, 121] and event detection [91, 87, 4, 79, 117].

Automatic analysis of sports videos can assist players and coaches. Player tracking, detecting in-game events and analysing the movement of a group of players provides coaches with key insights into team tactics, roles, contributions & workload of players, and helps players improve themselves by analysing their strengths and weaknesses. Additionally, critical analysis of key events in a game can also significantly enhance the quality of a broadcast.

Due to the complex and dynamic nature of sports, automatic analysis is quite challenging. Accurate tracking of the ball/puck is challenging due to the small size and motion blur. In team sports, there are occlusions due to players moving and handling the ball/puck, which further complicates the problem. Player detection and tracking is challenging due to similarity of appearance: players on each team wear the same coloured jerseys. Players are also partially or fully occluded by equipment, obstacles on court or other players. The players are moving fast and performing multiple different actions simultaneously. Their posture varies significantly across different sports. This makes action recognition and event detection challenging. The scale and lighting of sports videos vary greatly. Also, for outdoor sports, weather conditions can vary greatly. Camera parameters also vary between sports and within a single sport, ranging from zoomed-in views of a single player to zoomed-out views of the entire field.

The goal of my PhD is to develop algorithms and systems to solve problems in hockey video analysis. Towards this end, I have worked on three main projects: 1) automatic

hockey videography [86], 2) automatic play segmentation of hockey videos [85] and 3) automatic homography estimation.

In Chapter 2, I discuss the work on tracking the puck from a single wide-field camera and show its application for automatic sports videography. After recording the video, the periods of inactivity can be removed, thus temporally compressing the video. This work has been discussed in Chapter 3. While all these applications assumed a single fixed camera, sports video is often recorded with dynamic cameras whose extrinsic (pan, tilt, roll) and intrinsic (focal length) parameters change over time. In these scenarios, meaningful interpretation of image observations depend upon accurate online estimation of the homographic mapping between the coordinates of the image and the coordinates of the playing surface. An approach to automatically estimate this homography is discussed in Chapter 4.

Chapter 2

Automatic Sports Videography

If you have a good seat at a hockey game, the rink may subtend around 120 deg of your field of view. This may sound great, but recall that your visual acuity falls off very rapidly with visual eccentricity. Letter acuity, for example, falls by a factor of two at only 1 deg eccentricity [99]. This means that to enjoy the game, you will be constantly shifting your gaze to keep your eyes on the action.

Most people watch hockey remotely on an electronic display. Typically, the image quality is far below a live experience, and the angular subtense of the display will be much less than if you were there in person. For example, a standard 10-inch iPad at a comfortable viewing distance of 60cm will subtend only around 24 deg.

To partially compensate for this mismatch between the live and remote experience, professional videographers employ longer focal lengths so that the camera captures only a fraction of the ice surface at any one time, and constantly pan the camera to keep it on the play. Unfortunately, this is too expensive for most amateur games or the minor leagues. This motivates the development of automated, attentive sports videography systems to follow the play.

Precisely defining what is meant by “the play” in any sport, including hockey, is tricky and involves domain-specific knowledge. How then do we train a computer vision system to follow the play if it can’t easily be defined? One possibility is to simultaneously record static wide-field video that covers the entire playing surface as well as dynamic professional videography using a longer focal length and then attempt to learn a regressor that

predicts the gaze parameters [17, 19]. However, this requires access to a sample of professional videographers, and it is not clear whether a system learnt using their behaviour will generalize across different levels of play and rink environments.

We propose an alternative approach, which is to use the puck as a surrogate for the location of play. This has the immense advantage that it is objectively well-defined, and therefore can serve as the basis for training a computer vision system, without requiring access to professional sports videographers.

How will we track the puck? Even for high resolution video, direct cues for puck location are weak. NHL rinks, for example, are 61m in length, while the puck is only 7.6cm in diameter. This means that the puck will subtend at most 11 pixels in an 8K video. Moreover, the motion of the puck reduces the effective contrast - it often appears as only a faint grey streak. This all seems to argue against trying to build an automatic system to track the puck. We hypothesize, however, that a number of more easily observed macroscopic visual cues such as the pattern of optic flow and player positions are statistically predictive of puck location. Our aim is to prove this hypothesis by building a system for Automatic Puck Tracking (APT) that uses these macroscopic cues to accurately track the puck.

To create APT, we recorded several amateur hockey games using two different stationary high resolution cameras systems. A sample image from one of the cameras is shown in Fig 2.1a. We then developed a novel psychophysical method to generate accurate ground truth for puck location on the wide-field image. The estimated ground truth was then used to train a deep network regressor employing the video imagery and optical flow data as well as automatically estimated player positions to predict puck location. The result is a system that accurately follows the play, allowing dynamic cropping of a zoomed-in view appropriate for viewing on a standard display (Fig 2.1b).

2.1 Prior work

Computer vision is playing an increasing role in the world of sports [111]. Applications range from autonomous camera systems [19] to player detection/tracking [70, 66] to free-viewpoint video generation [48] and forecasting of future events [35, 125].

The two areas of current research most relevant to our work are: (1) ball/puck tracking [29, 57] and (2) broadcast camera angle planning [111, 19].



(a)



(b)

Figure 2.1: Automatic videography for ice hockey. Top: The 720p dynamic zoom view (blue rectangle) centered at the estimated puck location (pink dot). Bottom: Corresponding curated view.

2.1.1 Ball/puck tracking

One of the most important problems in sports, is ball/puck tracking. In all sports involving a ball/puck, the players move relative to the location of the ball/puck and manipulate it in order to score points, goals or runs. Thus, automatic detection of the ball/puck is very useful and has a wide range of applications ranging from camera planning to coaching to real-time analytics, and is a well-studied problem [29, 111, 57]. It is a challenging problem due to the small size of the ball/puck and motion blur. In team sports such as basketball, soccer and volleyball, the ball is frequently occluded by players. In sports such as field hockey and ice hockey, the ball or puck may also be occluded by the sticks.

There are two distinct approaches among the traditional computer vision algorithms used for ball/puck tracking: 1) recognition methods and 2) trajectory estimation methods.

Most of the recent methods augment the traditional methods with deep learning-based features. Next, I will look at the prior work in ball/puck tracking using these approaches.

2.1.1.1 Recognition methods

Ball/puck recognition methods detect the ball/puck independently on each frame. Some of the early methods generated ball candidates using cues such as shape, size and colour. These were then input to a classifier which classified the candidates as positive samples (soccer ball) or negative samples (background) [1, 28] in soccer videos. The ball was detected independently on each frame.

These methods were sensitive to partial and full occlusions. Halbinger *et al.* [42] proposed an approach for ball detection in soccer videos in the event of occlusions. When the player is close to the ball or in the event of a partial occlusion, the foreground segments of the player and the ball are merged. The authors proposed a two-step approach to detect the ball in such scenarios. In the first step, circular regions are detected using a Hough transform. In the second step, the Freeman chain code [37] of these circular regions are computed to determine the ball location.

Deep learning-based features and models have been shown to be useful for various computer vision problems, including many in sports [40, 111, 76]. Taking advantage of this, instead of using hand-crafted features, many authors have used deep learning-based features and models for ball tracking. Reno *et al.* [95] trained a Convolutional Neural Network (CNN) for detecting tennis balls. Given a test image, 50×50 non-overlapping crops are comprehensively generated and the crop classified as containing the ball with the highest confidence was used as the ball location. This approach has two drawbacks: 1) negative examples for training CNN must be carefully chosen and 2) computation time per image is very high due to sliding-window approach. Komorowski *et al.* [63] proposed a deep network that works with a single pass of the entire image to detect the soccer balls. The authors designed a single-pass deep network for accurate ball detection similar to YOLO [92]. The output from each layer was concatenated and fed into the final output layer. This allowed the use of both low-level features from the first few layers and the high-level features from the subsequent layers. The output of the network was a confidence map of ball probabilities on the image.

2.1.1.2 Trajectory estimation methods

Instead of solely relying on frame-by-frame detections, which can be noisy, the robustness of ball tracking systems can be improved by incorporating temporal information. The ball candidates or detections in successive frames can be associated or linked together using past and/or future ball locations to generate the ball trajectory.

The simplest way to use temporal information is to search for objects of similar shape or size and match the template against the ball detections in the previous frames [127]. Miura *et al.* [78] combined appearance information with player position data to improve the robustness of ball tracking to partial occlusions in soccer videos.

2.1.1.2.1 Causal methods Appearance based methods are not robust to occlusions and are affected by background objects in the scene. Several approaches use causal methods such as Kalman filters and particle filters to predict future locations of the ball given past ball detections [111, 29, 130, 134, 116]. Typically, these follow a tracking-after-detection approach. Ball candidates are generated on each frame after applying background subtraction followed by filtering the foreground objects based on size and shape. The initial ball location is assigned based on heuristics. The history of the ball location is used to learn the motion model and predict a probable future ball location.

Ren *et al.* [93] extended this approach to multiple stationary overlapping cameras in order to tackle occlusions in soccer videos. For each camera view, the ball trajectory was estimated using a tracking-after-detection framework with a Kalman filter. These trajectories from each camera were refined using appearance-based template matching. The template was updated with the predicted ball locations for the previous 3 seconds. The cameras were calibrated and hence, the 3D ball location was estimated.

These methods are known to be sensitive to initial estimates. Hence some authors tried to estimate the initial ball location based on the intersection of the optical flow field on the first 5 frames in basketball videos [88], while others manually ground-truthed the initial ball location on the first frame [93].

2.1.1.2.2 Non-causal methods Causal methods can handle short term occlusions but are sensitive to initialization and can be affected by incorrect detections in the past. As

opposed to only using past information, non-causal methods use the location of ball candidates in the past and future to guide the estimation of the ball location on the current frame. It is more robust to noisy detections and initialization.

Liu *et al.* [71] used a weighted graph to improve the initial estimate for the ball location in soccer videos. On the first five frames, the ball candidates were generated after applying background subtraction and then filtering based on shape and size. These ball candidates were connected using a weighted graph, where each node represented a ball candidate and edge weights were a shape similarity score between the two candidates. Using dynamic programming, the path having highest similarity score was initialized as the ball trajectory. In the subsequent frames, a Kalman filter was used to estimate the ball trajectory.

Pallavi *et al.* [82] extended this weighted graph idea over an entire video to enable ball tracking in soccer videos. The initial ball location was manually marked on the first frame. A weighted graph connecting nodes (ball candidates) on frame f_t to that on frames t_{t+1} and f_{t+2} was generated. The edge weights were an appearance similarity score. Using dynamic programming, the path having the maximum sum of edge weights was determined as the ball trajectory.

Yan *et al.* [132] utilized a three-step data association scheme to track tennis balls. In the first step, ball candidates were estimated using background subtraction and then filtered based on size and shape. Nearby ball candidates were associated based on the Euclidean distance to form mini-tracklets of ball positions at frames $\{t - 1, t, t + 1\}$. These mini-tracklets were combined to form longer tracklets using a weighted graph. Each node of the graph was a mini-tracklet. The edge weights measured the compatibility between two nodes (mini-tracklets) based on the appearance similarity and the Euclidean distance between the ball locations. In the final step, the ball trajectory was estimated through dynamic programming.

2.1.1.2.3 Physics-based modelling Appearance-based methods can be combined with strong domain knowledge, such as physics-based motion models, to improve ball tracking. In all sports the ball has a well-defined motion path between collisions.

For example, when someone shoots a basketball, the ball follows a parabolic trajectory until it reaches the basket. Some authors took advantage of this to estimate the ball trajectories during shooting sequences in basketball videos [13, 16]. The authors fit a parabola

to either the 2D ball detections on the image [13] or the 3D ball detections [16] to estimate the ball trajectory. In addition to ball trajectory, the shooting location of the player on the floor can also be estimated [16].

Parisot *et al.* [83] tried to extend this approach for ball tracking in a basketball game by combining appearance features with physics-based modelling. Ball candidates were generated after applying background subtraction and filtering based on shape and size. Since multiple calibrated cameras were used to record the game, 3D ball positions were obtained from the 2D detections. Then the ball trajectory was estimated using a RANSAC based approach that found the best-fitting quadratic fit to the 3D ball candidates.

2.1.1.2.4 Player movement and interactions The ball has a pronounced parabolic trajectory mainly during shooting sequences in basketball videos. In many other sports such as soccer, field hockey and ice hockey the ball/puck is typically on the ground/ice. And in such team sports, tracking the ball/puck is very hard because players can occlude the ball/puck. However, the movement of players is very strategic and closely coupled with the location of the ball/puck. The players move depending on the ball/puck location and the relative positions of other players on the field. Thus, the interactions of players can be helpful for ball/puck tracking.

As opposed to using a tracking-after-detection framework, Wang *et al.* [124] track the ball in multiple sports by estimating which player is in possession of the ball at any time. The authors assume the ball location to be same as the location of the player possessing the ball. The ball is said to be in free motion (not possessed by any player) when passed between players, in which case the ball position is estimated through linear interpolation between the locations of the two players who passed the ball.

Players are tracked using the network flow algorithm proposed in [102] and their team affiliations are obtained through k-means clustering of the RGB colour histograms of the player detections. The player locations, velocities and team affiliations are input to a random forest that predicts the state of the game, i.e., which team is attacking. The ball candidates are generated after applying background subtraction and filtering based on shape and size. Each ball candidate is assigned a confidence score based on its appearance similarity with the ball template learnt from the training data. These ball candidates are projected onto the ground plane to generate a Ball Occupancy Map (BOM). A feature vector consist-

ing of the player trajectories, BOM and the state of the game are used to learn a Conditional Random Field (CRF) that predicts the player possessing the ball.

The above approach estimated the ball location to be either same as the location of the player possessing the ball or interpolated between two player locations. The authors did not impose any motion constraints on the ball. Thus, depending on the state of the ball, Maksai *et al.* [75] observed that some of the predicted trajectories were physically impossible. The authors improved upon the previous method by imposing physics-based motion constraints on the ball using the information of ball state. The ball moves differently at different states such as *flying*, *in-possession* and *rolling*. The authors formulated their problem in a similar manner to Wang *et al.* [124]. The main difference being that the feature vector used to estimate the ball state consisted of the ball location and the player trajectories. Depending on the predicted ball state, state-specific motion models were used to estimate the ball trajectories. Further, the authors used multiple calibrated cameras to determine the 3D positions of the ball. It helped in resolving ambiguities due to occlusions.

2.1.1.2.5 Deep learning-based methods Kamble *et al.* [58] combined deep features with causal methods for ball tracking in soccer. Ball candidates were generated by applying background subtraction and filtering based shape and size. The VGG-M network [15] was trained to accurately detect the ball from the ball candidates. The deep network was used to detect the ball on the first frame. In the subsequent frames, the ball candidate having the highest Intersection over Union (IOU)¹ with the ball detection in the previous frame was estimated as the ball location on the current frame. The deep network was only used to detect the ball on a frame if the IOU of all ball candidates with the ball location on the previous frame was less than a pre-defined threshold.

Instead of a simple IOU metric for linking past detections, Zhang *et al.* [137] combined a deep network detector with a Kalman filter in a two-stage model for tracking golf balls. In the first stage, the pre-trained Faster RCNN [94] was used to localize the ball on an image. In the second stage, a Kalman filter estimated the ball location on the next frame given the estimated ball location on the current frame. In the next frame, a neighbourhood

¹Intersection over Union (IOU) measures the closeness of two regions/bounding boxes in an image. The IOU between two regions A and B is defined as $IOU = \frac{|A \cap B|}{|A \cup B|}$, where $|A|$ represents area of region A. IOU takes values in $[0, 1]$, with 0 indicating disjoint regions and 1 indicating identical regions.

region around this estimated ball location was cropped and passed to the Faster RCNN which then predicted the exact ball location. This procedure was applied recursively.

Unlike most of the ball tracking algorithms that use either domain knowledge, player interactions or past and/or future predictions to improve the predictions on the current frame, Voeikov *et al.* [123] used the predictions on the current frame in a cascaded manner for improved ball tracking accuracy in table tennis. The input data to the first stage deep network was the RGB image from the previous 9 frames: $\{f_t, f_{t-1}, \dots, f_{t-8}\}$, where f_t is the current frame. This input data was passed through a pre-trained VGG network [105]. The output from the network was the x-y pixel location of the ball on the current frame f_t . In the second-stage, the input data was a zoomed-in crop around the predicted ball location from the first stage. This cascaded network improved performance since the resolution loss at the first stage was offset by the zoomed-in input at the second stage.

2.1.1.3 Puck tracking in ice hockey

There is not much literature on puck tracking for ice hockey. The small size of the puck and motion blur make it a very hard problem. The only two previous works for puck tracking in ice hockey videos that we are aware of, addressed it on zoomed-in broadcast videos. Yakut *et al.* [131] generate puck candidates after applying background subtraction and filtering based on size. The puck candidates are associated across frames based on velocity constraints.

Vats *et al.* [120] trained a deep network for puck tracking in broadcast videos. The proposed network consists of 3 components: video branch, player branch, attention branch. The video branch takes as input the RGB image from the past 16 frames, $\{f_t, f_{t-1}, \dots, f_{t-15}\}$. This input is passed through the R(2+1)D network [119] to obtain the video features. The players are detected using Faster RCNN detector [94] and a player heatmap is passed through another network to obtain the player features. The attention branch concatenates the video features and player features in a squeeze and excitation network [51] and outputs a 2D probability map of the puck location.

Both these proposed methods are applied on zoomed-in broadcast videos where the puck is 20-30 times larger than in our wide-field videos. Additionally, they have been evaluated on short video clips of 2-6 seconds in duration.

No method has been proposed for puck tracking in hockey videos using a single wide-field camera. Additionally, none of the prior works evaluate their algorithm for an entire game. Problems such as occlusions due to the sideboards, movement of players and their sticks are exacerbated in longer videos as compared to the short 2-6 second clips.

2.1.2 Broadcast camera angle planning

Most prior work on camera planning for sports falls into one of two categories: (1) Direct regression methods and (2) Scene understanding methods.

Daigo *et al.* [24] used estimates of audience gaze directions and player locations to drive camera gaze for basketball. Others have directly regressed estimated PTZ parameters derived from professional videographers against estimated player trajectories [18, 19]. These approaches focus on generating smooth transitions as this has been shown to be important for the user experience [38].

Scene understanding methods for camera planning attempt to extract inferences about the state of play that can guide the selection of camera parameters. Ariki *et al.* [3] used estimated player and ball trajectories and a set of rules to determine smooth camera transitions and cropping of a wide angle video for soccer. Kim *et al.* [60] employed global motion features to predict regions of importance in soccer videos captured from multiple static calibrated cameras and later from moving cameras [61].

None of this prior work addresses the problem of camera planning for hockey. Our approach can be seen as a synthesis of direct regression and scene understanding approaches. We use a regression framework, but rather than regressing estimated camera parameters, we regress a ground-truthed scene variable: the puck.

2.2 Contributions

The contributions of this chapter are as follows:

1. Creation of a new dataset of hockey videos recorded using a single wide-field camera.

2. Design of a novel psychophysical method for ground-truthing the puck locations on a wide-field video.
3. Generation of ground truth puck location for all videos in the dataset.
4. Developing of a novel algorithm for puck tracking from single wide-field hockey videos.
5. Design of a pipeline for automatic sports videography.

We have published an earlier version of the work described in this chapter at the Winter Applications of Computer Vision (WACV) conference [86]. In that paper, we showed the feasibility of our puck tracking algorithm and its application towards automatic sports videography. We used a smaller dataset for that work. The work described in this chapter is an expanded and improved version of that work. Firstly, we have created a much larger and more comprehensive dataset. Secondly, we have significantly improved the algorithm by homographically normalizing each camera studied to a common camera frame and increasing the accuracy of puck tracking. Finally, we have comprehensively evaluated the performance of our system and shown generalization across cameras.

2.3 Dataset

As there were no public datasets of multiple hockey games recorded with fixed cameras, we created and labelled our own dataset. We recorded multiple games in two phases. We refer to the first phase of data as Dataset 1 and the second phase of data as Dataset 2. We describe the two datasets next.

2.3.1 Dataset 1

The first phase of data consisted of 8 games recorded at 4 different rinks using two different 4K 30fps wide-FOV cameras.

Camera 1 was a GoPro 5 camera with $3,840 \times 2,160$ pixel resolution, operated in wide-FOV mode. It was positioned in the stands along the centerline, about 7m above the ice and 8 meters away from the nearest sideboard. While the field-of-view of the camera spanned

the length of the rink, small portions of the rink at the corners were cut off. One game was recorded at Rink 1 using this camera. The camera tilt was varied between periods when recording this game (Fig 2.2a and Fig 2.2b).

Camera 2 was a proprietary system consisting of two cameras placed next to each other, with the two videos stitched together along a central vertical seam to yield a single wide-field video of $4,096 \times 832$ pixel resolution. The camera was positioned just behind the penalty box, displaced slightly to the right of the centerline. Since the camera was close to the rink, a small portion of the rink below the camera was cut off (Fig 2.2). The pose and camera parameters were not known for any of the recorded games. Three games were recorded at Rink 2, and two each at Rinks 3 and 4.

The videos were manually vertically cropped to remove the irrelevant regions above and below the rink². We manually extracted all video sequences of play lasting for at least 12 seconds. The resolution of the videos from each rink and their durations are shown in Table 2.1. The total duration of all games in the dataset is 59m 51s.

We used Dataset 1 for our WACV conference paper on this work in [86] and I will use parts of this dataset in Chapter 3 of this thesis.

Camera	Rink	# Games	Tilt	Resolution (pixels)	Duration (hh:mm:ss)
1	1	1	0°	3840×1020	00:08:10
1	1	1	12.2°	3840×1100	00:07:08
2	2	3		4096×747	00:21:05
2	3	2		4096×712	00:11:28
2	4	2		4096×742	00:12:00

Table 2.1: Duration and resolution of the videos of each camera from Dataset 1.

2.3.2 Dataset 2

The second phase of data consisted of 9 games recorded using two different high-resolution 30fps camera systems, placed in the stands, roughly aligned with the center line and about

²This one-time operation is performed when installing the camera at a rink, and could be automated in future work.



(a) Rink 1 (0 deg camera tilt) (Camera 1)



(b) Rink 1 (12.2 deg camera tilt) (Camera 1)



(c) Rink 2 (Camera 2)



(d) Rink 3 (Camera 2)



(e) Rink 4 (Camera 2)

Figure 2.2: Example video frames from Dataset 1.

10 meters horizontally and 8 meters vertically from the closest point on the ice surface.

Camera 1. Five games were recorded using a 4K Axis P1368-E camera (Fig 2.3a). The nonlinear distortions were removed using a standard calibration procedure [138]. We noticed that the field-of-view (FOV) of the camera did not cover the entire rink (corners are cut off) - this led us to develop Camera 2.

Camera 2. Four games were recorded using a proprietary camera system consisting of two 4K IP cameras with an inter-camera rotation of 75 deg (Fig 2.3b). Nonlinear distortions were removed using a standard calibration procedure [138]. Corresponding control points were manually marked on the ice surface on the images from each of the two cameras (Fig 2.4a) as well as a template of the ice rink (Fig 2.4b). The homography between the images and the template were estimated using OpenCV’s standard least squares reprojection minimization method *findHomography*. These homographies were used to reproject both cameras to a virtual cyclopean camera bisecting the two cameras, where the two images were stitched using a linear blending function (Fig 2.4c).



(a) Camera 1



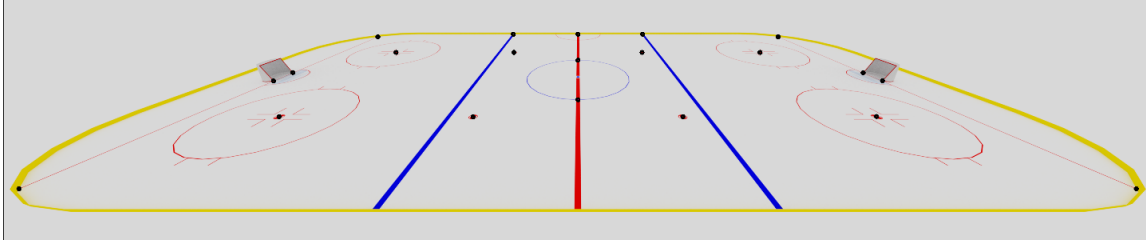
(b) Camera 2

Figure 2.3: Example frames from Dataset 2 after correction for radial distortion.

The videos were manually vertically close-cropped to the rink (Fig 2.3). All the 9 games recorded by both cameras were played at the same rink. We manually extracted all video sequences of play lasting for at least 12 seconds. The video resolution and duration



(a) Images from the two cameras



(b) Template image



(c) Stitched image

Figure 2.4: Camera 2 view construction. (a) Example frames from two cameras. (b) Rink template. (c) Stitched image with homography control points.

for each game in Dataset 2 is shown in Table 2.2. The duration of all games from Camera 1 is 2h11m27s and that of Camera 2 is 3h21m53s, resulting in a total dataset of 5h33m20s.

To evaluate the generalization across camera viewpoints, we experimented with rectifying all videos from Dataset 2 to a common viewpoint. This is described in Section 2.5.1. In order to rectify the data to a common viewpoint, it is important to know the camera parameters and correct for any nonlinear distortions. Since Dataset 2 is larger and more consistent than Dataset 1, we will rely exclusively on Dataset 2 for the remainder of this chapter, except where noted.

Camera	Game	Resolution (pixels)	Duration (hh:mm:ss)
1	1	$2,758 \times 500$	00:18:37
1	2	$2,758 \times 500$	00:29:08
1	3	$2,758 \times 500$	00:29:27
1	4	$2,758 \times 500$	00:24:41
1	5	$2,758 \times 500$	00:29:34
2	1	$5,960 \times 1,080$	00:52:32
2	2	$5,960 \times 1,080$	00:50:06
2	3	$5,960 \times 1,080$	00:45:44
2	4	$5,960 \times 1,080$	00:52:31

Table 2.2: Duration and video resolution for each game across both cameras in Dataset 2.

2.4 Ground truthing

At 4K resolution, the puck is roughly 5 pixels in diameter, making it very challenging to ground truth. Most of the time during a game, it appears as a grey streak due to motion blur. However, the puck can be directly visible to the eye at slow speeds. Also, human observers may use contextual cues to follow the puck while watching a game. Taking advantage of this, we used psychophysical methods to ground truth puck location in our videos as a single pixel location on each frame.

2.4.1 Apparatus

Human observers viewed videos on a 42-inch 4K 60Hz LCD display. We assessed the performance of observers using two input devices: 1) a standard desktop wireless mouse with a spatial resolution of 500 samples per inch, and 2) a 31.1×21.6 cm Wacom PTH860 tablet and pen with a spatial resolution of 5,080 lines per inch. Both devices were sampled at 125 Hz.

2.4.2 Procedure

We played back the video at a reduced frame rate and tasked human subjects with tracing the puck location across the entire video. We expected that more accurate ground truth would be obtained by slowing the frame rate, introducing a potential trade-off between data quality and the time required for ground truthing.

To quantify this trade-off, we conducted a preliminary study in which five observers ground-truthed a 28-second clip from a 4K 30 fps GoPro 5 camera (Dataset 1 Camera 1) at 5 different frame rates of 1, 2, 4, 8 and 16 fps (slowed by a factor of between 1.9 and 30), presented in random order. Each observer repeated the experiment twice, enabling an assessment of between-observer and within-observer consistency. The raw 28-second video clip was used for the initial study, i.e., the nonlinear distortions were not corrected.

2.4.3 Analysis

Since we do not have objective ground truth for puck location, we cannot definitively measure the accuracy of human puck tracking. As a surrogate, we assessed within-observer consistency (across the two trials for each observer) and between-observer consistency, under the assumption that observers are approximately unbiased.

We corrected for the nonlinear distortion in the video and then computed the homography between the video and the ice surface. This allowed us to express the within-observer and between-observer errors in meters. Fig 2.5 shows RMS deviation (along with standard error) within and between observers as a function of playback frame rate. We found that the deviations between observers were consistently about 15cm (about 20%) higher than within observers, indicating some degree of individual differences in tracking behaviour. We also found tracking with the trackpad to be more accurate than with the mouse. This could be due to the absolute coordinate frame provided by the trackpad, or differences in the pose of the hand or the haptic feedback from the device. We did not observe any increase in RMS error with increasing frame rate up to 8 fps. Thus, as a trade-off between accuracy and speed, we decided to ground truth the videos at 8 fps.

In Fig 2.5 it appears that a precision (between-observer accuracy) of around 80cm can be achieved at playback frame rates of up to 8 fps. However, this analysis does not rule out systematic bias that might further limit accuracy. One particular concern is temporal

lag in the human system, which we analyse next.

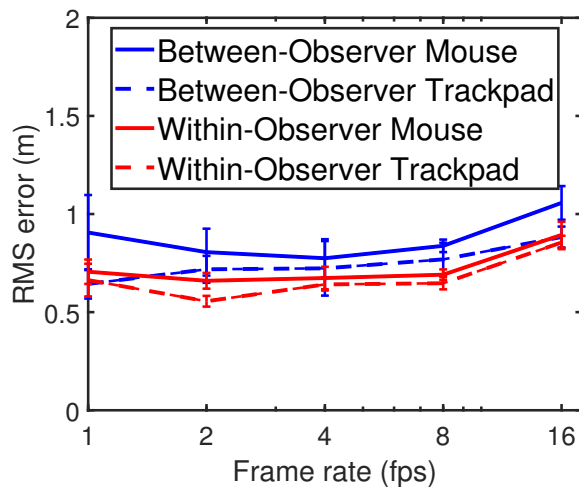


Figure 2.5: Mean and standard error of within-observer and between-observer deviations as a function of playback frame rate.

2.4.4 Systematic lag

Humans exhibit a lag of roughly 300-400 msec when manually tracking unpredictable (Brownian) 2D target motion [8]. Since the motion of the puck is predictable, we expect average lag to be much lower but still present. To estimate this lag, we can compare an observer’s trace of the puck at a higher frame rate with their trace at our lowest frame rate (1 fps).

Let $\mathbf{x}(s)$ represent the true 2D location of the puck in the image at time s seconds into the recording and let $\hat{\mathbf{x}}_f(t)$ be the observer’s trackpad coordinates at time t seconds into the psychophysical experiment when observing the video played back at a reduced frame rate of f fps. The original/native frame rate of the video is 30 fps. If we sync the two timescales so that $s = 0$ and $t = 0$ represent the same point in the video, then we expect that

$$\hat{\mathbf{x}}_f((t + t_0)/f) \approx \mathbf{x}(t/30) + \mathbf{n}(t) \quad (2.1)$$

where t_0 is the observer’s lag and $\mathbf{n}(t)$ is zero-mean iid noise.

We can use this identity to relate trackpad coordinates at higher frame rates f to the trackpad coordinates at 1 fps:

$$\hat{\mathbf{x}}_f((t + t_0)/f) \approx \hat{\mathbf{x}}_1(t) + \mathbf{n}(t) \quad (2.2)$$

We want to estimate the lag t_0 . Let $\hat{\mathbf{x}}'_f(i)$ represent the resampled trackpad data collected when the video was played back at f fps, where $f \in \{1, 2, 4, 8, 16\}$. The trackpad data was resampled at a sampling frequency of 100 samples per frame, regardless of the video playback frame rate. We achieved this by linearly interpolating the trackpad data $\hat{\mathbf{x}}_f$ for each video playback frame rate f . In continuous time, we can represent this interpolation as $\hat{\mathbf{x}}'_f(t) = \hat{\mathbf{x}}_f(t/(100f))$. $\hat{\mathbf{x}}'_f(i)$ and $\hat{\mathbf{x}}_1(i)$ are sampled versions of such continuous time signals. We can then rewrite Eq 2.2 as

$$\hat{\mathbf{x}}'_f(i + \Delta i) \approx \hat{\mathbf{x}}_1(i) + \mathbf{n}(i) \quad (2.3)$$

Now we can identify the relative lag Δi in samples between the two trackpad traces by minimizing the mean squared residual:

$$\Delta i = \arg \min_j \frac{1}{N - |j|} \sum_{\max(-j, 0)}^{\min(N, N-j)} \left\| \hat{\mathbf{x}}'_f(i + j) - \hat{\mathbf{x}}_1(i) \right\|^2 \quad (2.4)$$

which can be computed by cross-correlating the two traces, subtracting half the sum of the autocorrelations and then taking the max.

Finally, the observer lag in seconds t_0 can be recovered from the sample lag Δi :

$$t_0 = \Delta i / (100f) \quad (2.5)$$

We can express the lag/delay in two different ways: 1) psychophysical time and 2) game time. When the delay is computed according to the video playback frame rate, we say it is expressed in psychophysical time. Hence, in this scenario, f in Eq 2.5 takes one of the four values - $\{2, 4, 8, 16\}$. Alternatively, when the delay is computed according to the native/original video frame rate, we say it is expressed in game time. Since the videos were recorded at 30 fps, $f = 30$ in Eq 2.5, irrespective of the playback frame rate at which the trackpad data were recorded.

We found that the lag in psychophysical time was relatively constant with frame rate (Fig 2.6a), while the lag in game time increased with frame rate (Fig 2.6b). Expressing the lag in game time, at 8 fps, we found the mean and standard error of the lag over our 5 observers to be 30 ± 9 msec. This indicated that the ground truth should have been corrected by about 1 frame.

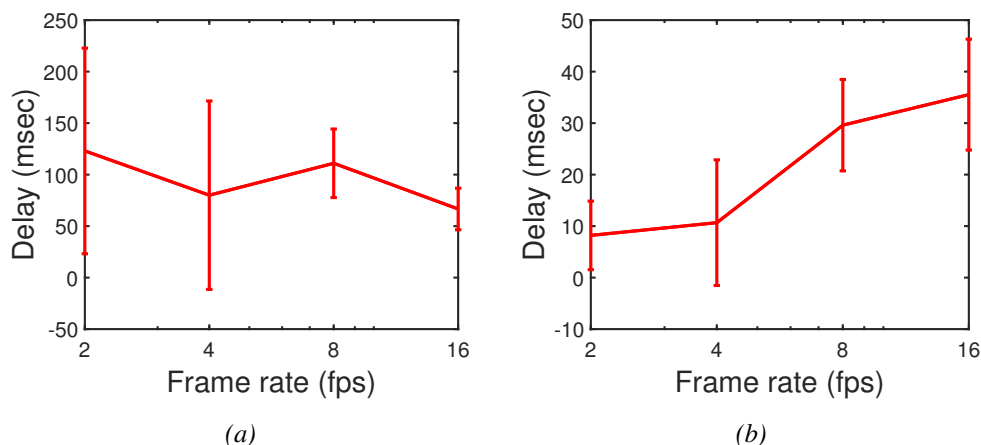


Figure 2.6: Systematic lag in the observer trackpad data in (a) psychophysical time and (b) game time.

To evaluate whether this lag was significant, we measured the RMS displacement of the puck over a 30 msec interval and found it to be 0.18 m. Given the RMS error of our APT system is about 4.5 m, we expect little benefit from correcting for the systematic lag in our ground truth.

2.5 Algorithm

A pipeline summarizing our algorithm to track the puck and enable automatic sports videography is shown in Fig 2.7. A 6-channel input feature map consisting of the RGB image, optic flow and binary player mask is passed to a deep network regressor, which predicts an x-y pixel location of the puck on the input image. These predicted puck locations are temporally smoothed using a non-causal Gaussian smoother. Centered around the smoothed puck estimates, the wide-field video is cropped to generate a zoomed-in

video following the play. In this section, I will detail each step and design choice of our algorithm.

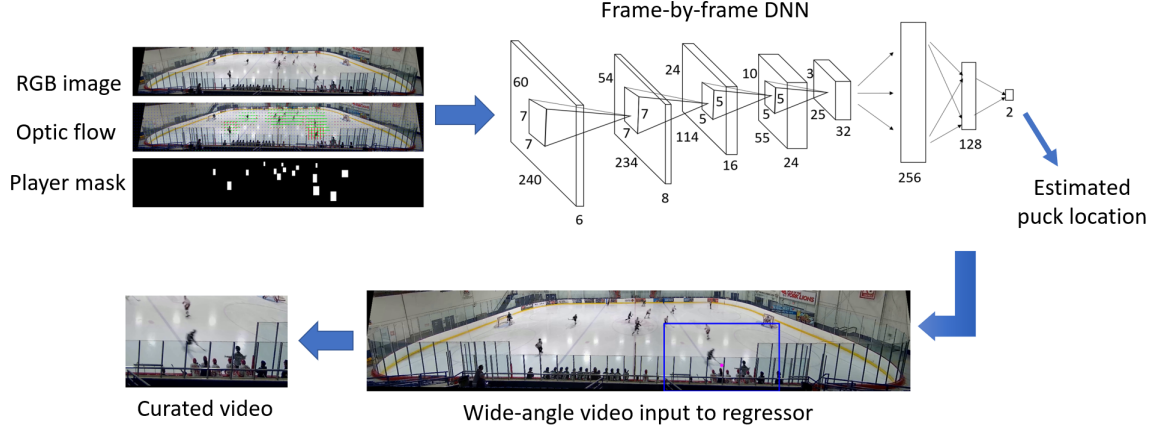


Figure 2.7: Pipeline of the automatic sports videography system. A 6-channel input feature map consisting of the RGB image, optic flow and binary player mask is passed to a deep network regressor that predicts the puck location on the input image. A crop of the high-resolution wide-field video centered at the prediction is used as the curated video following the play.

2.5.1 Viewpoint invariance

While the dimensions of the playing area of every ice rink is one of two standard sizes: NHL or international size, the arenas/stadiums can have vastly different designs. As a result, cameras across two different venues are rarely at identical locations with respect to the rink. This leads to different camera poses across different arenas. For a practical plug-and-play automatic sports videography system, it is important to be camera pose invariant. We incorporate viewpoint invariance into our system by rectifying each video in the dataset to a known template viewpoint.

We first created a geometric model of the ice rink and identified 19 control points at easily identifiable locations on the ice markings. We then identified the corresponding control points on images from the two cameras (Fig 2.8a, Fig 2.8b). This allowed us to estimate the homography relating each camera’s image plane to the ice surface, using OpenCV’s standard least squares reprojection minimization method, *findHomography*.

Next, we created a virtual camera positioned at the centerline, 15 meters horizontally and 10 meters vertically from the nearest point on the ice surface, and tilted down by 20 deg. We identified the corresponding control points on this virtual camera (Figure 2.8c). Finally, we reprojected the two real camera views through the ice surface and to the virtual camera plane (Fig 2.8d, Fig 2.8e).

The videos from Camera 1 and Camera 2 were rectified to this known template viewpoint. The deep network regressor was trained on the rectified data. At inference, we first rectified the image to the template viewpoint and then predicted the puck location on the rectified image.

2.5.2 Deep network regressor

We trained a deep network regressor to predict the puck location on each video frame (Fig 2.7). The input data to the network consisted of three feature maps: (1) RGB colour imagery, (2) Optic flow and (3) Player positions. In this section, I will describe how we computed the input feature maps and the architecture of the deep network regressor.

2.5.2.1 Input feature maps

As mentioned earlier, the input data to the deep network consisted of three feature maps: (1) RGB colour imagery, (2) Optic flow and (3) Player positions.

2.5.2.1.1 Optic flow The optic flow was computed using OpenCV’s implementation of Farneback’s dense optic flow algorithm [33]. We represented optic flow as a two-channel map containing horizontal (green) and vertical (red) components of the flow field (Fig 2.9).

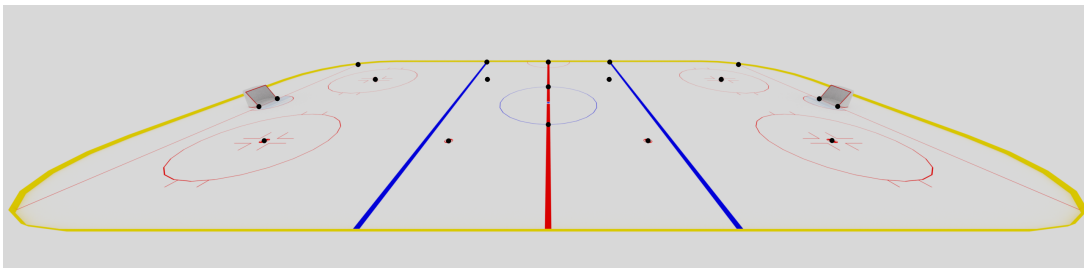
The size of the input data to the deep network regressor was 240×60 . We evaluated the influence of the image resolution on the optic flow contribution. We computed the optic flow at resolutions of 1, 2, 4 and 8 times the input image resolution. We trained and evaluated the deep network regressor with the input data consisting of the 2-channel optic flow feature maps computed at these various resolutions. The optic flow feature maps were computed at a higher resolution and resized to 240×60 using bilinear interpolation. We found that an image resolution of 4-times larger - 960×240 - resulted in the lowest error. Hence, the optic flow was computed at this resolution.



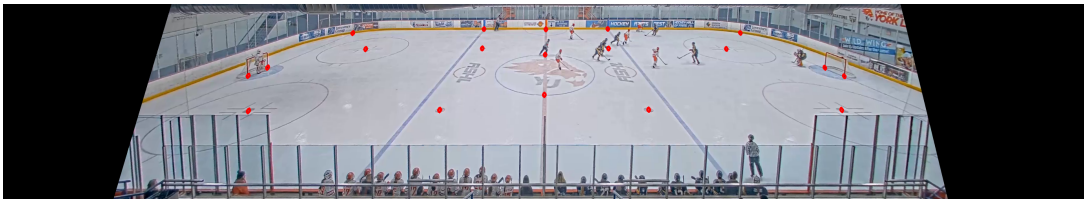
(a) Homography control points for Camera 1.



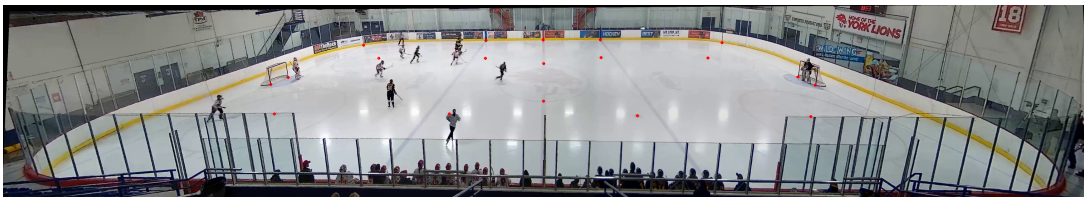
(b) Homography control points for Camera 2.



(c) Homography control points for template ice rink.



(d) Camera 1 rectified to the template rink.



(e) Camera 2 rectified to the template rink.

Figure 2.8: Rectification of the cameras to a known template viewpoint.

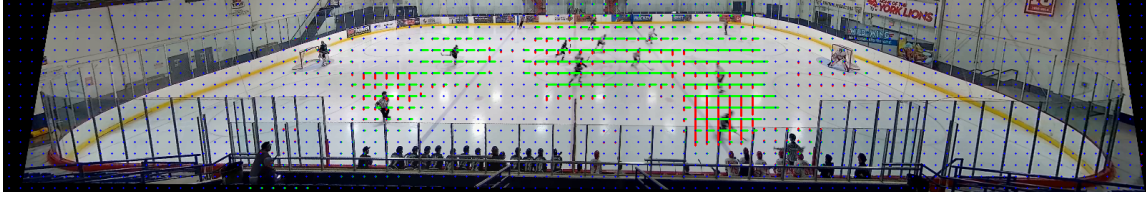


Figure 2.9: Horizontal (green) and vertical (red) components of the optic flow field.

2.5.2.1.2 Player detection We explored two algorithms for player detection - 1) HOG+SVM detector [25] and 2) Faster RCNN detector [94].

We created a dataset consisting of 12 games recorded using Camera 2 for training and evaluating the player detector. We selected every 30th frame to generate a dataset of 1,000 frames from each game. The test dataset consisted of 2,000 frames from 3 different games - 1,000 frames from one game and 500 frames each from the other two games. The validation dataset consisted of 1,000 frames from another game. The training dataset consisted of 8 different games. The ground truth player data were obtained through hand-labelled bounding boxes from human subjects on Amazon Mechanical Turk. We manually cleaned up the ground truth player data for both the validation and test datasets. We also manually cleaned up about 50% of the training data, but due to time constraints we did not correct all of it.

HOG+SVM detector: The training partitions were used to train OpenCV’s implementation of the Histogram of Oriented Gradients (HOG) classifier [25] for player detection in the luminance domain. The training dataset for the HOG+SVM model consisted of 1,000 frames each from three different games. The positive samples of player crops were augmented by left-right mirroring. For each positive sample, ten negative samples were generated from random locations on the same frame that did not overlap positive samples. The width and height of negative bounding boxes were drawn randomly from the range of widths and heights of positive examples. We used the standard two-pass training process: an initial linear SVM was trained on the training set which was then used to detect players in the training images. False positives from the training set were then added to the set of negative examples and the SVM was retrained. At inference, we used OpenCV’s *detectMultiScale* function which detected players at various resolutions using an image pyramid formed by continuously downsampling the input image. We created a 64-level

image pyramid where the image at the previous level was downsampled by 5% to obtain the image at the next level. We measured the F-score of the model on the test dataset for full ($5,930 \times 1,080$ pixel), half, third and quarter resolution. We found the highest F-score was obtained when the resolution was at half the original resolution - $2,960 \times 540$.

Faster RCNN detector: Faster RCNN is one of the state-of-art models for object detection [94]. We fine-tuned the pre-trained Faster RCNN model available in PyTorch on a training dataset consisting of either 1, 3, 5 or 8 games. We analysed the improvement in performance with increase in training data by measuring the accuracy on the test dataset.

The precision-recall curves of the various player detector models are shown in Fig 2.10. The precision and recall of the models at the maximum F-score is shown in Table 2.3. We observed that the performance of the pre-trained Faster RCNN was significantly better than the HOG+SVM detector. Also, we found that fine-tuning the pre-trained Faster RCNN model improved the performance of the player detector model. We did not find much gain in performance by increasing the number of training games to 5 or 8. Hence, for the subsequent experiments, the player detector model was trained on 3 games. These 3 games were then discarded from the training and test dataset.

Model name	Precision (%)	Recall (%)
HOG	74.4	54.2
Pre-trained Faster RCNN	90.3	81.4
Faster RCNN 1 game	98.4	91.9
Faster RCNN 3 games	98.6	92.6
Faster RCNN 5 games	98.6	92.5
Faster RCNN 8 games	98.5	93.0

Table 2.3: Precision and recall of the various player detection models at the maximum F-score.

At the threshold corresponding to the highest f-score, the precision of the player detector model was 98.6% and recall was 92.6%. The run time of the Faster RCCNN model was 26.8 fps at a resolution of $2,960 \times 540$ on a i7 processor and a single Nvidia 1080Ti GPU. Sample results from the player detector is shown in Fig 2.11a. Player positions were encoded as a single-channel binary mask, each player represented as a solid rectangle of 1s on a background of 0s (Fig 2.11b).

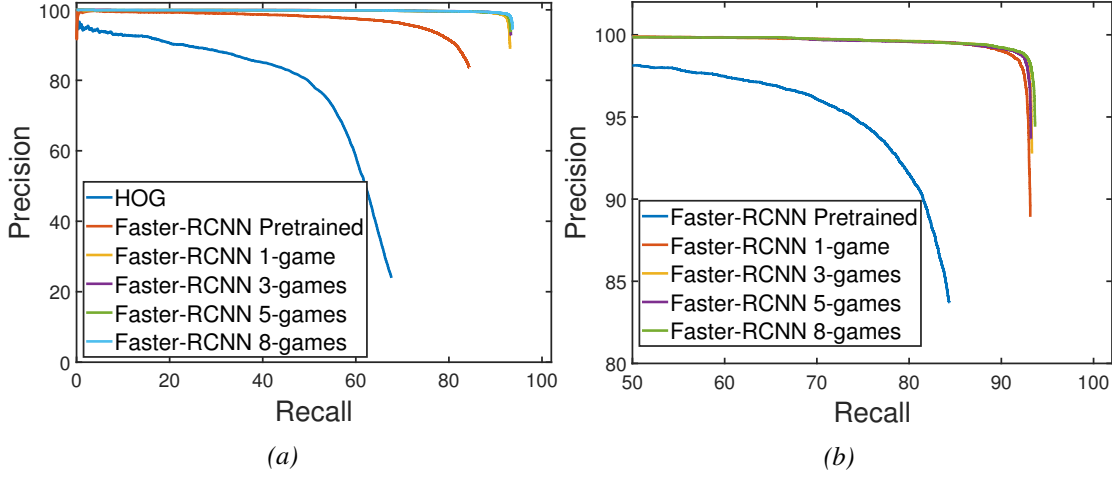


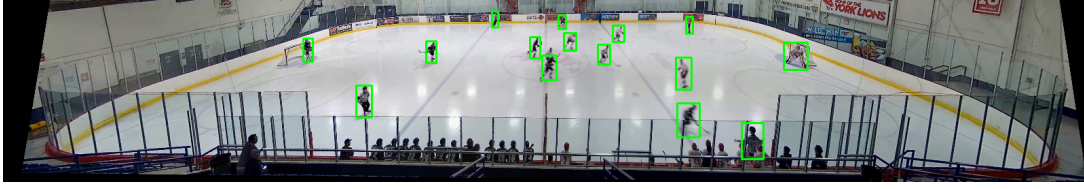
Figure 2.10: Faster RCNN experiments. Right image is a zoomed-in view of left image.

As described in Section 2.3, the video resolution of Camera 1 and Camera 2 were $2,758 \times 500$ and $5,960 \times 1,080$ pixels, respectively. These videos were then rectified to a template viewpoint of resolution $6,360 \times 1,080$, as described in Section 2.5.1. Empirically we found no difference between rectifying the binary player mask to the template viewpoint and detecting the players in the rectified template viewpoint. Hence, for the subsequent experiments, we detected players in the rectified viewpoint. The rectified image was resized to $2,960 \times 540$ and the players were detected using the Faster RCNN detector trained on 3 games.

2.5.2.2 Network architecture and training

Using the 6-channel input data consisting of the RGB image, horizontal and vertical optic flow maps and binary player mask, we trained a deep network regressor to predict the puck location. In this section, I describe the architecture and training details of the network.

The 6-channel input data was resized to 240×60 . Each feature map was normalized to have zero-mean and unit-variance. One of the standard techniques to reduce overfitting and improve generalization of deep networks is data augmentation. The training data is augmented by applying various image transformations in order to improve the diversity of training data. Data augmentation has been shown to work well for training deep learning



(a) Players detected using the fine-tuned Faster RCNN model



(b) Binary player mask

Figure 2.11: Detected players and the subsequent binary player mask that is one of the input feature maps for the deep network regressor.

models for various applications [84, 103]. Hence, for our application, we used 4 different types of data augmentation techniques:

1. Left-right mirroring of the input feature maps.
2. The HSV values of the RGB image were perturbed. The RGB image was converted to an HSV image and a constant value of $x_1 \in [-50, 50]$ was added to the hue channel, and $x_2 \in [-0.15, 0.15]$ was added to the saturation channel. x_1 and x_2 varied randomly for each frame. The perturbed image was then converted back to the RGB colour space.
3. Gaussian noise was added to the RGB image. The noise had zero-mean and the variance, $v \in [0.01, 0.04]$, varied randomly for each frame.
4. Gaussian noise was added to the horizontal and vertical optic flow feature maps. The noise had zero-mean and the variance, $v \in [0.01, 0.04]$, varied randomly for each frame.

The range of values for x_1 , x_2 and v were determined empirically. The four augmentation steps are independent and they can be applied sequentially. Thus, there are 16

possible combinations. For each frame of the training data, an integer between 0 and 16 is randomly generated. The binary representation of this number indicated the *ON/OFF* flag for each augmentation step. The steps are applied in the same sequence as listed above.

We used PyTorch to develop the deep network regressor. Our network architecture was loosely based on AlexNet [65]. We determined the various design parameters of the network architecture, such as the number of layers, the number of kernels, the sizes of the kernels, and the sizes of the fully connected layers, using cross-validation experiments on non-rectified videos from Camera 1 (Fig 2.3a). The details of the experiments to determine the optimal network architecture can be found in Appendix A.

The network consisted of four convolutional layers followed by two fully connected layers with ReLU non-linearities (Fig 2.12). We used 2×2 average pooling layers as empirically they achieved lower training error than 2×2 max pooling layers. Dropout was used between each fully connected layer. The output of the network was the estimated x-y pixel location of the puck. The loss function was the mean squared error between estimated and ground truth puck location, minimized using the Adam optimizer [62]. The model was trained for 50 epochs, with an initial learning rate of 0.001 and weight decay of 0.01. The learning rate was reduced by 50% every 8 epochs.

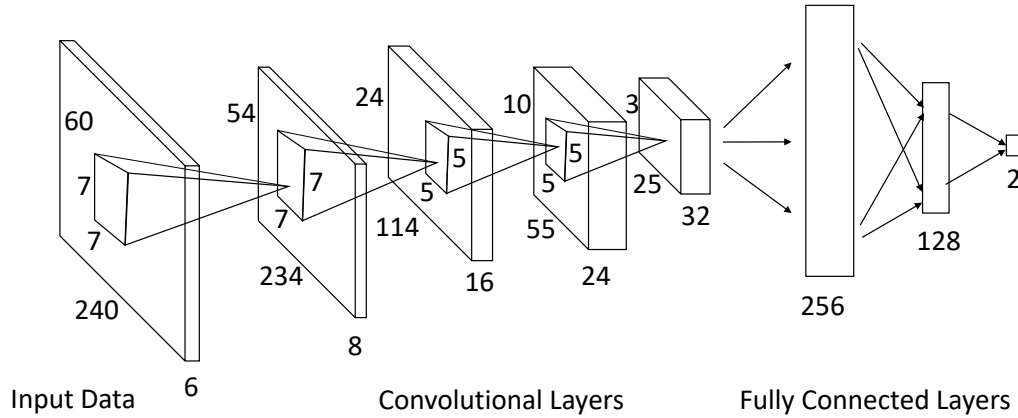


Figure 2.12: Architecture of the deep network regressor.

2.5.3 Temporal smoothing

The deep network regressor estimates the puck location in each frame independently. Given the high correlation in the ground truth puck location over successive frames, there is an opportunity to improve the estimates by temporal smoothing. Additionally, smoothing ensures smooth movement of the camera which has been shown to be important for aesthetically pleasing video recordings [38]. We evaluated both causal and non-causal temporal smoothers.

Causal Smoothing: We evaluated a recursive exponential causal smoother implementing the exponential IIR filter $h(t) = \frac{1}{\lambda}e^{-t/\lambda}$, varying the temporal smoothing constant λ .

Non-Causal Smoothing: We also evaluated the Gaussian non-causal smoother:

$$h(t) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{t^2}{2\sigma^2}}$$

truncated at $\pm 3\sigma$ and normalized to integrate to 1, varying the temporal smoothing constant σ .

Fig 2.13 shows the RMS error of the puck estimates from the deep network regressor on the training data with both causal and non-causal smoothers. The error was averaged across all training games for both Camera 1 and Camera 2. While both smoothers reduce the RMS error, non-causal smoothing yields superior results to causal smoothing.

Table 2.4 lists the optimal time constants for each camera. We used the non-causal smoother with optimal time constants for evaluating the model. Note that with a filter cutoff at $\pm 3\sigma$, non-causal smoothing entails a tape delay of roughly one second.

Camera	λ_{opt} (sec)	σ_{opt} (sec)
1	0.024	0.43
2	0.022	0.4

Table 2.4: Optimal time constants λ_{opt} and σ_{opt} for causal and non-causal smoothing, respectively

2.5.4 Dynamic cropping

To retarget the full rink wide-field video to a user’s display device, we extract a zoomed-in crop following the play. We reproject the final estimate of the puck location, obtained

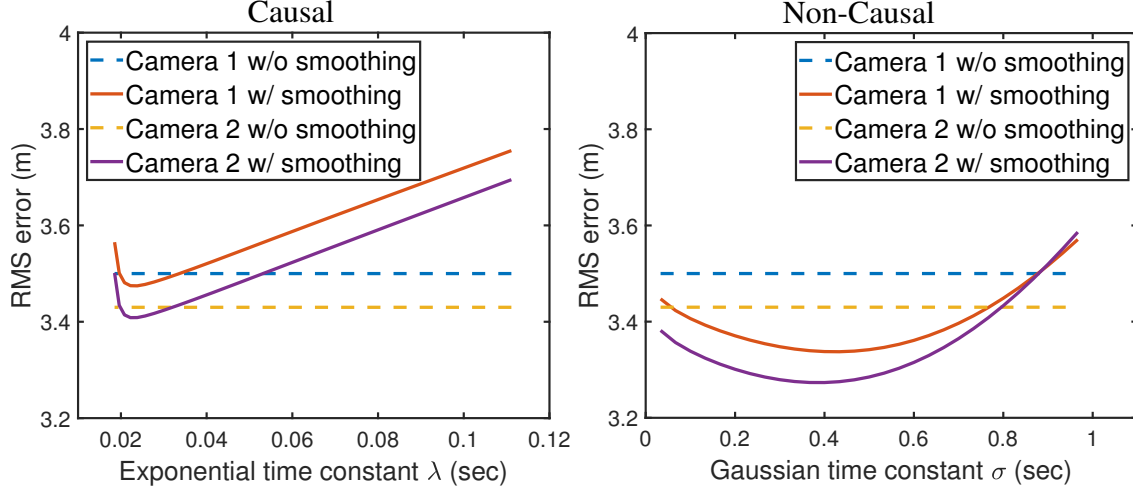


Figure 2.13: RMS error on the training data with causal and non-causal temporal smoothing on the estimates from the deep network regressor.

after non-causal smoothing on the estimates from the deep network regressor, from the template viewpoint to the original viewpoint of either Camera 1 or Camera 2. We then center a $1,280 \times 720$ pixel crop around the estimated puck location, while limiting the crop window to lie entirely within the FOV of the wide-field camera. This play following video is shown on the user’s display device. The size and shape of the crop is a design parameter that could be optimized for individual devices. For this work, we focus on 16:9 format devices.

2.5.5 Two-stage cascaded deep network regressor

The rectified input video to the deep network regressor is at the resolution of 6, $360 \times 1,080$ pixels. The puck is 11 pixels wide at this resolution. The input data to the network is the full wide-field video, downsampled by a factor of 27 along the width and a factor of 18 along the height. At this resolution, the puck is not visible.

Hence, to improve upon the accuracy of the predicted puck location from the deep network regressor, we introduce a cascaded two-stage deep network regressor. The first-stage deep network regressor is described in Section 2.5.2.2. The second-stage network is cascaded with the first-stage network in a similar manner to the idea proposed in [123]. It

processes a zoomed-in input image in which the puck will be more visible and this could allow for a more accurate prediction. In the second-stage, we extracted a $1,280 \times 720$ crop of the wide-field input image centered around the predicted puck location from the first-stage. We then crop the 6-channel input feature maps appropriately and pass it to the second-stage deep network regressor (Fig 2.14 (bottom)). Additionally, the features from the final convolutional layer of the first-stage network are concatenated with the features from the final convolutional layer of the second-stage network, and then passed to the fully connected layers of the second-stage network. The pipeline for automatic sports videography using the two-stage cascaded deep network regressor is seen in Fig 2.15.

The cropped 6-channel input feature maps are resized to 160×90 . The second-stage network is identical to the first-stage network except that it has three convolutional layers (Fig 2.14 (bottom)). The loss function, optimizer and other training parameters were same as that of the first-stage network.

The network architecture for the second stage was not fully optimized. We leave this as future work.

2.6 Evaluation

As there are no prior attempts to track the puck from a static wide-field camera view of a hockey game, we benchmark our system APT (Automatic Puck Tracking), against a baseline algorithm MPP, which uses the median player position from our player detector as an estimate of puck location.

We evaluate the performance of our system in several ways. First, we evaluate the contribution of each feature map in an ablation study. Next, we evaluate within-camera performance for generalization across games. Then, we evaluate the benefit of rectifying the images for viewpoint invariance. Next, we evaluate between-camera performance for generalization across cameras. Finally, we show two visualizations - heatmaps of the puck locations and puck occupancy maps.

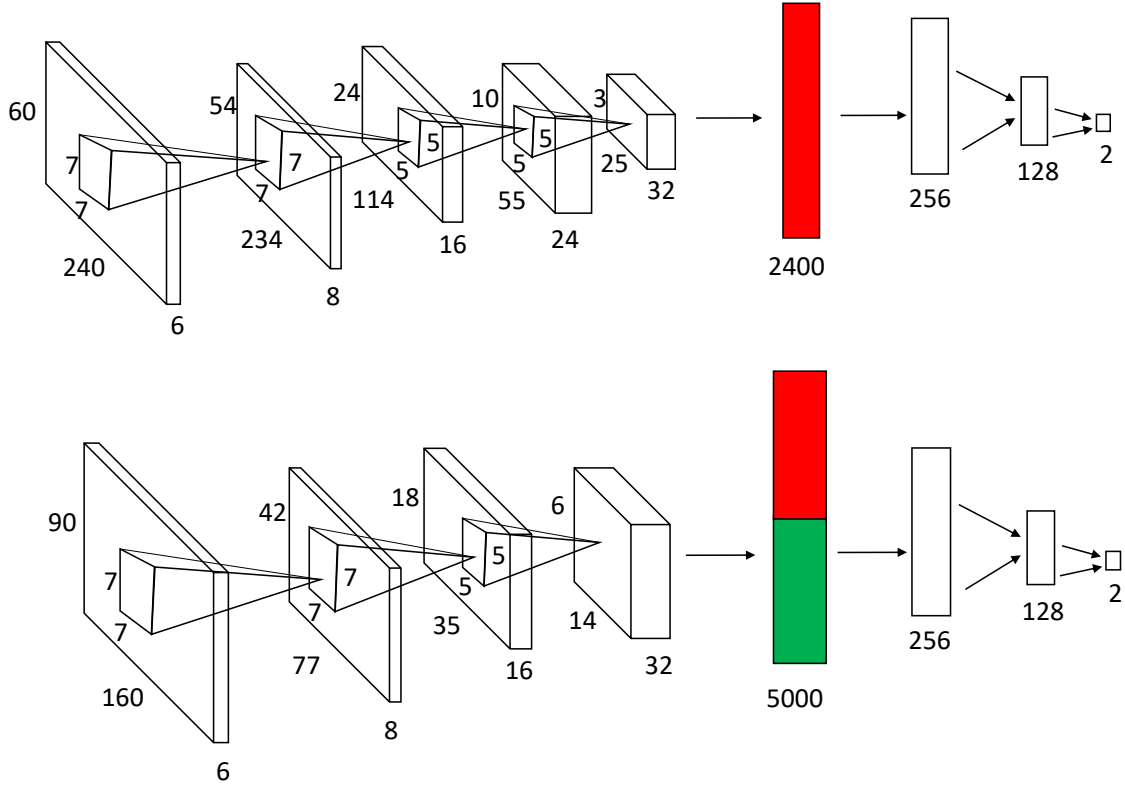


Figure 2.14: Two-stage cascaded deep network regressor. Top: First-stage network. Bottom: Second-stage-network. The features from the final convolutional layer of the second stage network (green) are concatenated with features from the final convolutional layer of the first stage network (red). The size of the input data to the first-stage regressor is 240×60 , while that to the second-stage regressor is 160×90 .

2.6.1 Ablation study

We conducted an ablation study to understand the relative contribution of the three input feature maps to the performance of APT. We train the model on the first 4 games from Camera 1 and evaluate it on the 5th game. We measure the performance of the first-stage deep network regressor (APT1). The ground truth and predicted puck locations are back-projected to ice coordinates (in meters). Table 2.5 shows the results of the ablation study.

The colour imagery, player positions and optic flow are all seen to be useful features

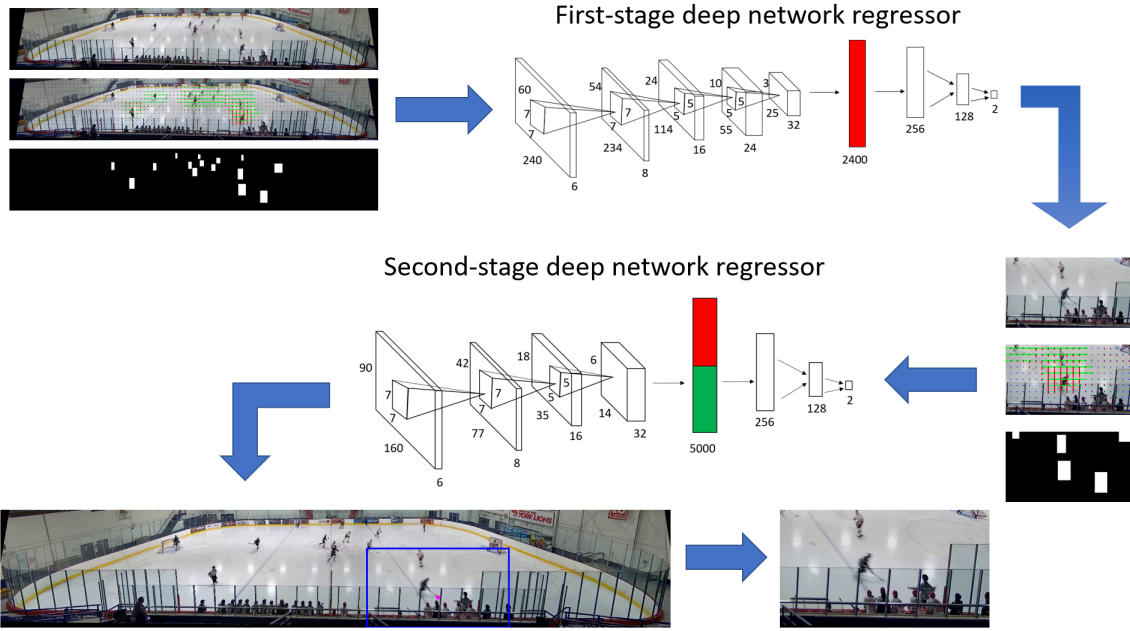


Figure 2.15: Pipeline of the automatic sports videography system using the two-stage cascaded deep network regressor. A 6-channel input feature map consisting of the RGB image, optic flow and binary player mask is passed to the first-stage deep network regressor. A crop of the high-resolution wide-field video centered at the first-stage prediction forms the input for the second-stage network. The features from the final convolutional layers of the first-stage (red) are combined with the features from the final convolutional layer of the second-stage (green).

as the model performance using any of the three features is much better than the baseline MPP model that uses median player position. APT1 achieves best performance using all three feature maps, and it achieves a reduction in error of 40.5% relative to the MPP baseline.

2.6.2 Within-camera performance

We evaluate the generalization of the system across games by measuring the within-camera performance through leave-one out cross-validation for each camera. We evaluated the performance of the system using only the first-stage network (APT1) and the full system (APT2 - both first and second stages). The results are shown in Table 2.6. We find that

Model	Input features maps	RMS error (m)
MPP	Median player position	8.4
C	RGB image	7.0
P	Binary player mask	6.4
O	Optic flow	5.9
CP	RGB image + Binary player mask	6.1
CO	RGB image + Optic flow	5.4
PO	Binary player mask + Optic flow	5.5
APT1	All Features (CPO)	5.0

Table 2.5: Ablation study to evaluate the contribution of the three input feature maps.

our system generalizes well across games for both cameras. APT2 improves upon APT1 by about 10% for both cameras, showing the benefit of the cascaded deep network. The benefit of temporal smoothing is also observed as APT2 + smoothing further reduces the error of APT2. APT2 + smoothing achieves a total reduction in error of 52% and 56% relative to the MPP baseline on Camera 1 and Camera 2, respectively. Some example predictions and the resulting zoomed-in view (curated images) are shown in Fig 2.16.

Camera	RMS Error (m)			
	APT1	APT2	APT2 + Smoothing	MPP
1	5.1 ± 0.25	4.6 ± 0.22	4.4 ± 0.25	9.1 ± 0.48
2	5.2 ± 0.25	4.7 ± 0.24	4.5 ± 0.25	10.2 ± 0.20

Table 2.6: Within-camera performance evaluated through leave-one-out cross-validation across all games.

In addition to the results in Table 2.6, we show the benefit of our two-stage cascaded deep network regressor by measuring the accuracy of the predicted puck locations at various error tolerance thresholds. We threshold the RMS error and compute the number of frames where the RMS error is less than the threshold. We evaluated the within-camera performance for both APT1 and APT2 (Fig 2.17). Across both Camera 1 and 2, we found that APT2 performs better than APT1, thus showing the benefit of the cascaded architec-



Figure 2.16: Left: Sample results for test images from Camera 2. Cyan and pink dots indicate ground truth puck location and the puck location estimated by APT2 + smoothing, respectively. The blue rectangle indicates the 1280×720 crop region. Right: Corresponding zoomed-in view.

ture. The benefit of non-causal Gaussian smoothing is also seen.

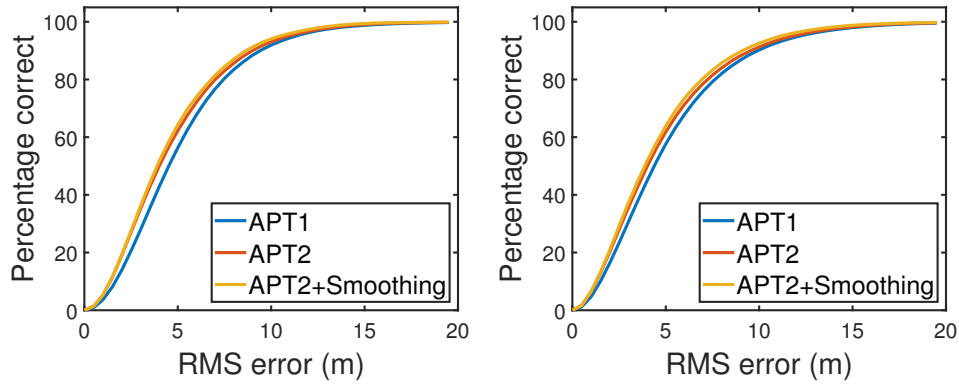


Figure 2.17: Accuracy of the within-camera performance as a function of RMS error. Left: Camera 1. Right: Camera 2.

For a sports videography system, it is essential to not miss the play. Since the play revolves around the puck in ice hockey, we can measure the performance of a videography system as the proportion of frames that the puck is visible in the zoomed-in video following the play (puck occupancy). We measure this metric at various resolutions of the zoomed-in video, as shown in Table 2.7. While the videos recorded by the two cameras in the datasets are not the standard 16:9 aspect ratio, the zoomed-in play following video output by our system is fixed to have a 16:9 aspect ratio. We find that at a resolution of 1280×720 , the puck is visible for 97.6% and 95.0% of the frames for Camera 1 and Camera 2,

respectively.

Zoomed-in video resolution (pixels)	% of rectified image width	% of rectified image height	Accuracy (%)	
			Camera 1	Camera 2
640×360	10.1	33.3	78.9	73.1
960×540	15.1	50.0	92.3	88.5
1280×720	20.1	66.7	97.6	95.0
1600×900	25.2	83.3	99.4	97.8
1920×1080	30.2	100.0	99.8	99.0

Table 2.7: Within-camera puck occupancy performance. The resolution of the zoomed-in video relative of the wide-field rectified image is indicated in columns 2 and 3. The accuracy is measured as the proportion of frames where the puck is visible in the zoomed-in video. Recall that the size of the rectified image is 6360×1080 .

2.6.3 Between-camera performance

We computed the generalization of the system across two different cameras. We trained the model on all the games of one camera and evaluated it on all the games from the other camera. Since the vantage point of the two cameras were different, we measured the benefit of the viewpoint invariance module by measuring the error with and without it. The results are seen in Table 2.8.

We found the generalization of the first-stage network with 6-channel input feature map (APT1) was very poor across cameras. The RMS error was significantly worse than the baseline MPP. This is because the lighting conditions between the two cameras are very different (Fig 2.3). We found the performance of the system improved after discarding the RGB image and using a 3-channel input feature map consisting of optical flow (O) and binary player mask (P). As seen in Table 2.8, irrespective of viewpoint invariance, we found significant improvement in performance of the first-stage network using the 3-channel input feature map (PO1) over the performance of the network using the 6-channel input feature map (APT1).

We also observed the benefit of the viewpoint invariance module. Rectifying the two cameras to a common template viewpoint corrected for any vantage point discrepancies

and hence, the RMS error of the first-stage network using the 3-channel input feature map (PO1) was significantly lower using the viewpoint invariance module. We also observed a significant improvement over the baseline MPP.

Training camera	Test Camera	RMS Error (m)				
		Without viewpoint invariance		With viewpoint invariance		MPP
		APT1	PO1	APT1	PO1	
1	2	16.4 ± 0.16	9.6 ± 0.05	14.6 ± 0.24	6.4 ± 0.22	10.2 ± 0.20
2	1	14.5 ± 0.47	12.8 ± 0.28	11.2 ± 0.19	5.9 ± 0.13	9.1 ± 0.48

Table 2.8: Between-camera performance of the first-stage network with and without viewpoint invariance module.

Using the 3-channel input feature map, we evaluated the generalization of the cascaded two-stage deep network across cameras. The results are seen in Table 2.9. We found that the performance of the second-stage network (PO2) improved upon that of the first-stage network (PO1). We also observed a reduction in error after applying non-causal Gaussian smoothing. PO2+Smoothing achieved a total reduction in error of 44% relative to the baseline MPP for each camera, thus showing generalization of the system across cameras.

Training camera	Test Camera	RMS Error (m)			
		PO1	PO2	PO2 + Smoothing	MPP
1	2	6.4 ± 0.23	6.1 ± 0.24	5.7 ± 0.23	10.2 ± 0.20
2	1	5.9 ± 0.13	5.7 ± 0.16	5.1 ± 0.13	9.1 ± 0.48

Table 2.9: Between-camera performance of the system using the 3-channel feature map of optic flow (O) and binary player mask (P).

In Table 2.9, we observe that the error of the model when evaluating on Camera 2 is slightly higher than that when evaluating on Camera 1. This could be due to one of two factors. The first one is the difference in the field-of-view (FOV) of the two cameras (Fig 2.3). The FOV for Camera 1 does not span the entire rink, while the FOV of Camera 2 spans the entire rink. The second factor could be the size and diversity of the datasets. As seen in Table 2.2, the dataset for Camera 2 is 1.5 times larger than that for Camera 1. This increase in training data may have led to better generalization of the model.

We computed the accuracy of the predicted puck locations at various error tolerance thresholds. We evaluated the between-camera performance for both the first-stage (PO1) and second-stage (PO2) networks (Fig 2.18). We found that PO2 was slightly better than PO1. Also, we observed significant benefit using the non-causal Gaussian smoothing (PO2+Smoothing).

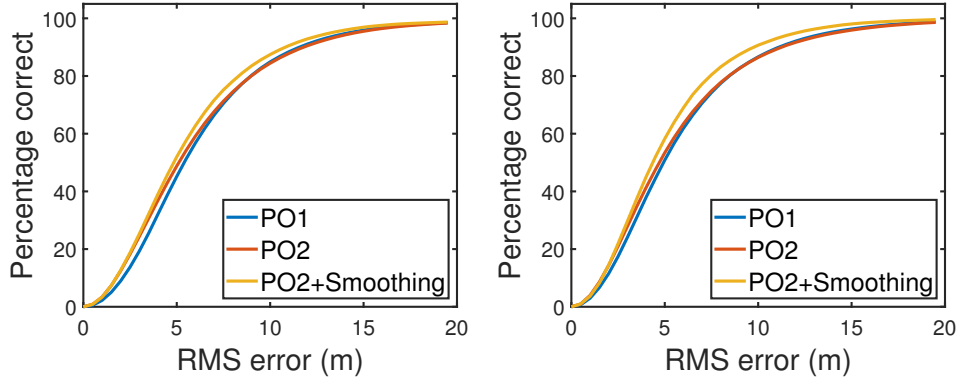


Figure 2.18: Accuracy of the between-camera performance as a function of RMS error. Left: Evaluating on Camera 2. Right: Evaluating on Camera 1.

In a similar manner to Table 2.7, we computed the puck occupancy accuracy at different resolutions of the zoomed-in video, as seen in Table 2.10. We found that at a resolution of 1280×720 , the puck is visible for 94.2% and 88.0% of the frames for Camera 1 and Camera 2, respectively.

2.6.4 Puck location visualizations

In this section, we show two visualizations of the puck location - heatmaps and puck occupancy maps. We computed the homography between the two cameras and the ice surface. The puck location heatmaps were estimated for both the ground truth puck location and predicted puck location (APT1 and APT2) using Gaussian kernel density estimation. The bandwidth was calculated using Silverman’s rule [104], and was approximately 27, 26, and 25 pixels for the ground truth, APT1, and APT2 predictions, respectively, for a 610×300 pixel rink template. The heatmap visualizes the puck location density per m^2 for each game from Camera 1 (Fig 2.19) and Camera 2 (Fig 2.20).

Zoomed-in video resolution (pixels)	% of rectified image width	% of rectified image height	Accuracy (%)	
			Evaluating on Camera 1	Evaluating on Camera 2
640×360	10.1	33.3	74.1	61.2
960×540	15.1	50.0	87.8	78.1
1280×720	20.1	66.7	94.2	88.0
1600×900	25.2	83.3	97.1	93.6
1920×1080	30.2	100.0	98.4	96.5

Table 2.10: Between-camera puck occupancy performance. In a similar manner to Table 2.7, the resolution of the zoomed-in video relative to the wide-field rectified image is indicated in columns 2 and 3, and the puck occupancy accuracy is measured. Recall that the size of the rectified image is 6360×1080 .

We measured the KL divergence between APT1 and APT2 heatmaps with the ground truth heatmap as

$$D(P||Q) = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right) \quad (2.6)$$

where, $P(x)$ and $Q(x)$ are the normalized heatmaps for the predicted puck location (APT1 or APT2) and ground truth puck location, respectively.

We averaged the KL divergence across all games for each camera. We found the KL divergence was lower for APT2 predictions (Table 2.11).

Camera	KL Divergence	
	APT1	APT2
1	0.218 ± 0.014	0.088 ± 0.010
2	0.136 ± 0.017	0.067 ± 0.006

Table 2.11: KL divergence of the APT1 and APT2 heatmaps with the ground truth heatmap.

We also computed the puck occupancy in different zones of the ice rink. We quantized the ice rink into three zones - 1) left of the left blue line, 2) between the two blue lines and 3) right of the right blue line. We computed the percentage of frames that the puck

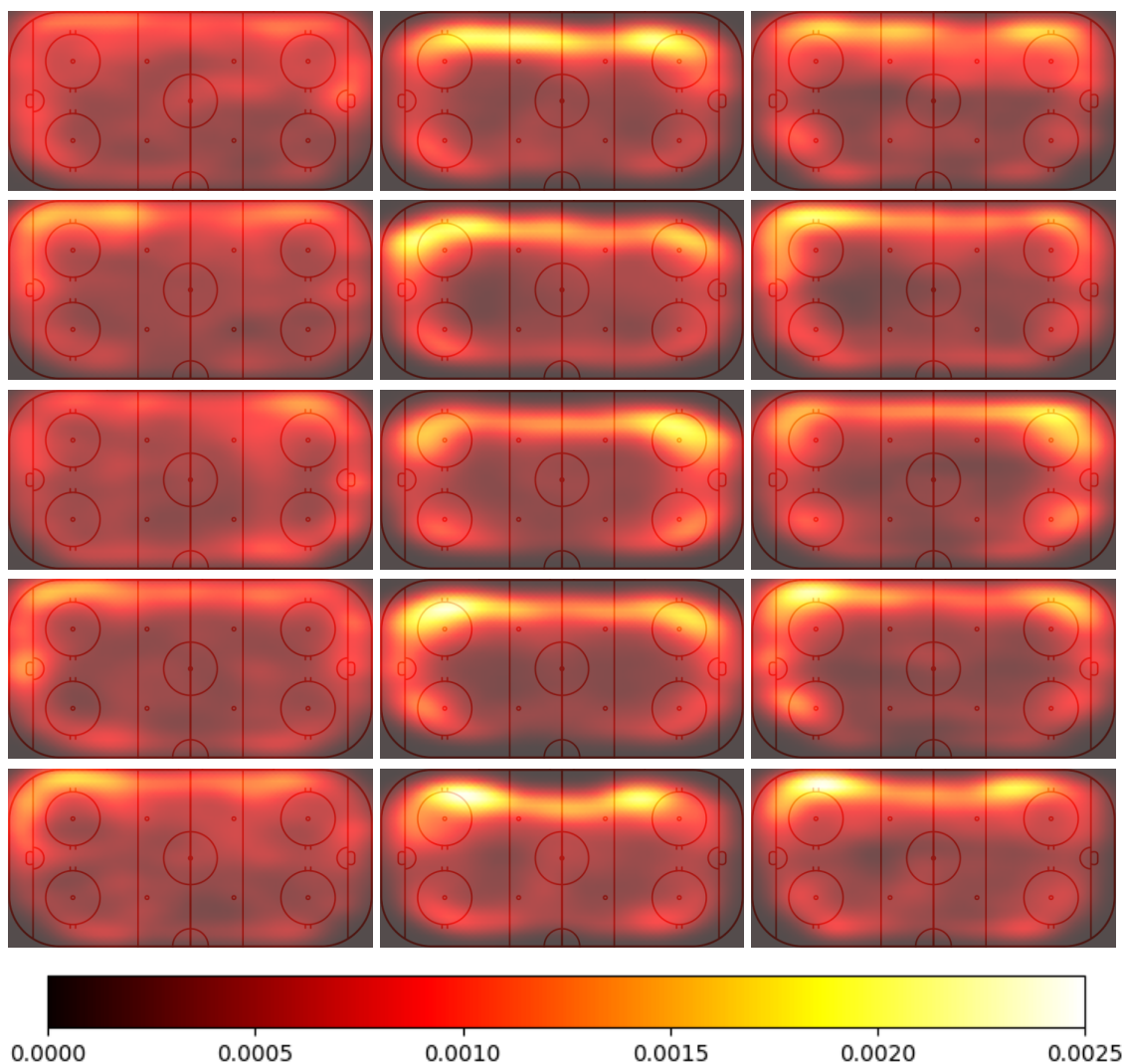


Figure 2.19: Puck location heatmaps for all games recorded by Camera 1. Left: Ground truth data. Middle: APT1 predictions. Right: APT2 predictions. Units: per m^2 .

lies in the specified zone for the ground truth puck locations, APT1 predictions and APT2 predictions. The results for Camera 1 and Camera 2 are seen in Fig 2.21 and Fig 2.22, respectively. We then measured the error of APT1 and APT2 predictions from the ground truth as the sum of the mean absolute difference across all three zones. The error was averaged across all games for each camera. While we found the error was lower using

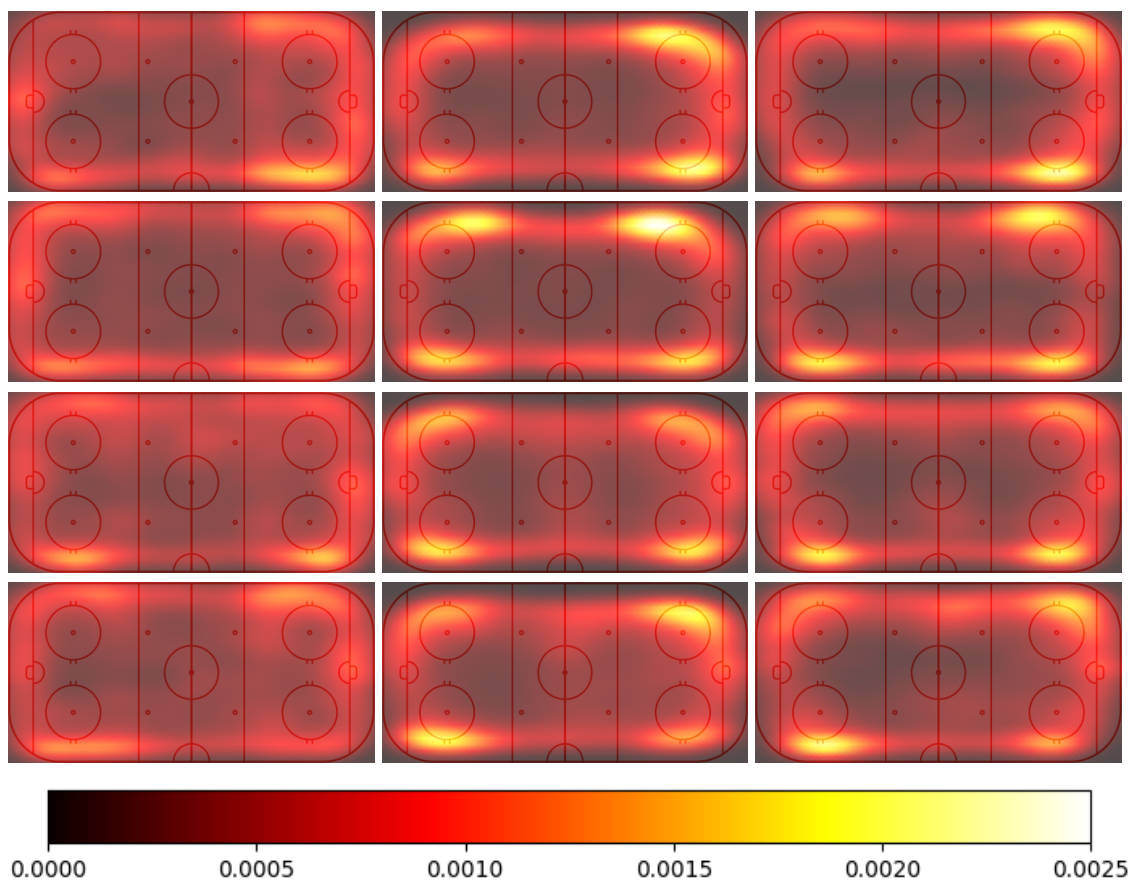


Figure 2.20: Puck location heatmaps for all games recorded by Camera 2. Left: Ground truth data. Middle: APT1 predictions. Right: APT2 predictions. Units: per m^2 .

APT2 predictions (Table 2.12), this improvement was not statistically significant.

Camera	Mean Absolute Difference (%)	
	APT1	APT2
1	2.62 ± 0.63	2.58 ± 0.43
2	2.31 ± 0.59	2.16 ± 0.65

Table 2.12: Average error of puck zone occupancy for the APT1 and APT2 predictions.



Figure 2.21: Puck zone occupancy for all games recorded by Camera 1. Left: Ground truth data. Middle: APT1 predictions. Right: APT2 predictions.

2.7 Conclusion & future work

I have described a system that tracks the puck location in a game of hockey based on a single wide-field camera, allowing for automatic dynamic cropping of the wide-field video to a curated, zoomed-in view suitable for delivery to a remote viewing device such as an iPad. I ground truthed the puck location by having human users trace the puck location using a trackpad and stylus in a video played back at reduced frame rate. Then, I learnt a cascaded two-stage deep network regressor to estimate the puck location on each frame. I

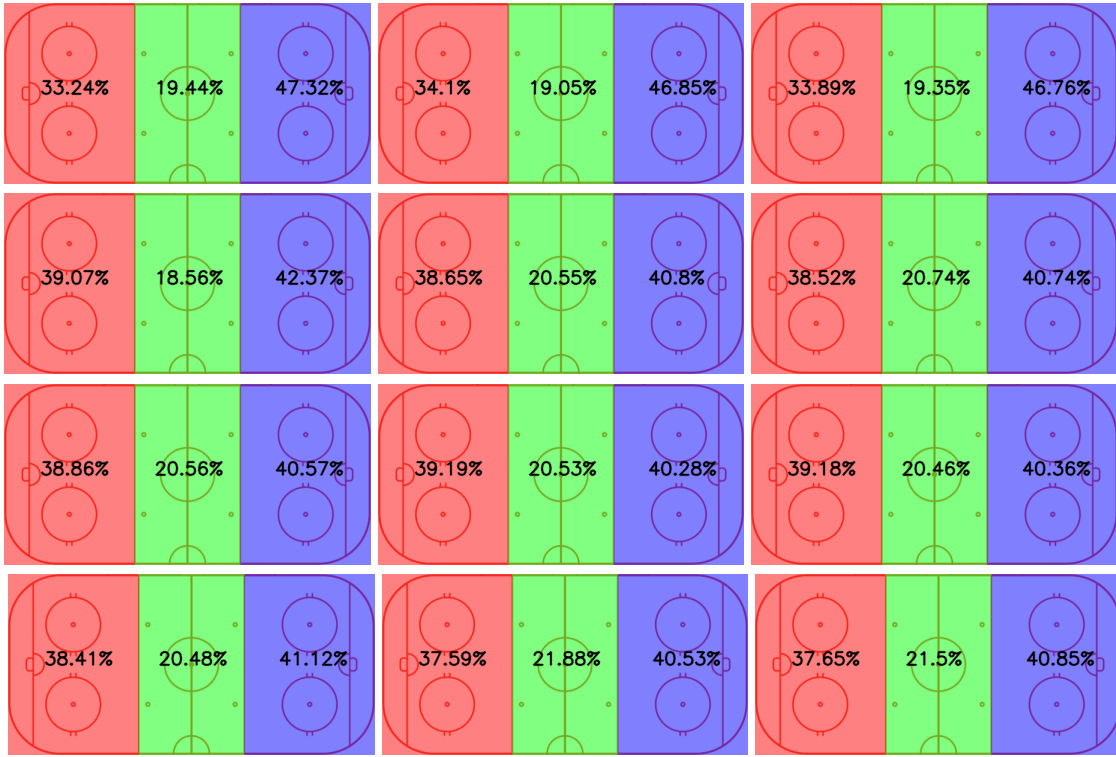


Figure 2.22: Puck zone occupancy for all games recorded by Camera 2. Left: Ground truth data. Middle: APT1 predictions. Right: APT2 predictions.

found the RMS error of the system to be 4.5 meters, which is significantly better than the baselines.

Currently, the system only pans or tilts the camera to follow the play. In future work, a zoom component can be incorporated by modelling the uncertainty in predictions. When the model predictions have high confidence, the system could further zoom-in to the play, if desired.

Chapter 3

Automatic Play Segmentation

Automation of sports videography has the potential to provide professional-level viewing experiences at a cost that is affordable for amateur sport. Autonomous camera planning systems have been proposed [18, 19, 86], however, these systems deliver continuous video over the entire game. Typical amateur ice hockey games feature between 40 and 60 minutes of actual game play. However, these games are played over the course of 60 to 110 minutes, with downtime due to the warm-up before the start of a period and the breaks between plays when the referee collects the puck and the players set up for the ensuing face-off. Also, there is a 15 minute break between periods for ice re-surfacing. Automatic identification of these stoppages would allow abbreviation/temporal compression of the video.

We propose a novel system that uses visual cues from a single wide-field camera and optional auditory cues. We explore two different visual cues. The first is based on the optic flow - players tend to move faster during play than breaks. However, we find that this cue is fallible: motion on the ice can sometimes be substantial during breaks and sometimes quite limited during periods of play. This motivates the development of a more complex deep visual classifier that takes not only the optic flow but also the RGB image and detected player positions as input.

We also explore the utility of auditory cues - more specifically, the referee whistle that starts and stops play. While not directly informative of the current state, the whistle does serve to identify the timing of state transitions, and thus can potentially contribute to

performance.

To take into account temporal dependencies, we employ a hidden Markov model (HMM), which, while simplifying modelling through conditional independence approximations, allows (1) optimal probabilistic integration of these noisy cues and (2) an account of temporal dependencies captured through the state transition matrix.

We also propose a method for unsupervised domain adaptation of the HMM, iteratively updating emission and/or transition probability distributions at inference, and show that this is critical to benefitting from auditory cues.

3.1 Prior work

One of the active areas of computer vision research that has been applied to sports is semantic analysis [39, 125, 91, 35]. Using ball detections and player tracking data, meaningful insights about individual players and teams can be potentially extracted. These insights can be used to understand the actions of a single player or a group of players and detect events in the game [4, 87, 52, 79].

Another form of semantic analysis is video summarization. Some have analysed broadcast video clips to stitch together a short video of highlights [114, 7]. However, this summarized video is short for consumption and cannot be used for tagging of in-game events, analysis of team tactics, etc. To achieve this, the summary video should retain all the active periods of play. Sports such as soccer, ice hockey and basketball have many stoppages during the game. Thus, we would have to divide the game into segments of active play and no-play, known as play-break segmentation.

Several approaches have been proposed to address play-break segmentation. Some use play-break segmentation for automatic highlight generation [32, 112] or event detection [108], while others use event detection to guide play-break segmentation [12]. Some of the initial methods for play-break segmentation used rule-based approaches that combined text graphics on a broadcast feed with audio cues from the crowd and commentator [113] or the type of broadcast camera shot [32, 112, 129, 108]. Ekin *et al.* [32] and Tjondronegoro *et al.* [112] selected the play frames as key frames for automatic highlights from broadcast soccer videos, while Tavassolipour *et al.* [108] used the play frames to detect events in broadcast soccer videos. The authors in [108] used the type of broad-

cast camera shot as the observation cue to an HMM that classified each frame for play segmentation.

All of these approaches used broadcast cues (camera shot type) or production cues (graphics and commentary) for play-break segmentation, and thus are not directly relevant to unedited amateur sport video recorded automatically with fixed cameras. We are aware of only one prior paper that addresses a similar problem for unedited videos. Carbonneau *et al.* [12] detected three in-game events (face-off, line change and play) in ice hockey and used the rules of the sport to determine segments of play and no-play. They trained an SVM on Bag-of-Words features to detect in-game events in video snippets. At inference, an event was predicted for each video snippet and it was classified as play or no-play segments using the rules of the sport. The method was trained and evaluated on disjoint intervals of a single hockey game recorded by two different cameras.

Our research goes beyond this prior work in several important ways. First, our approach classifies frames as play and no-play without requiring the detection of finer-grain events. Second, we show how the temporal dependencies between states can be captured and integrated within an HMM framework. (While HMMs have previously been used for broadcast video [129, 108] they have not been applied to unedited video.) Third, we propose a method for handling auditory domain shift that is critical for integration of auditory cues with visual cues. Finally, we go beyond this prior work by showing generalization across games, rinks and viewing parameters, which is critical for successful deployment.

3.2 Contributions

The contributions of this chapter are:

1. Creation of a dataset of multiple games recorded using multiple cameras. Generated play segmentation ground truth for all games.
2. Design of a deep neural network classifier trained for play/no-play classification based on visual input.
3. Design of a system for obtaining auditory cues from the referee’s whistle using a Wiener filter.

4. Integration of the visual and auditory cues into an HMM framework for play segmentation and an approach for unsupervised domain adaptation of the system.

In the work reported in this chapter, the code for implementing the HMM framework was done in collaboration with undergraduate summer intern Michael Dowling.

3.3 Dataset

The dataset consisted of 12 games. Four games were recorded using Camera 1 and five games were recorded using Camera 2. The details about the two cameras can be found in Section 2.3.2. We also recorded 3 games using a 4K wide-FOV GoPro 5 camera (Camera 3), which included synchronized audio at 48kHz. Cameras 1 and 2 did not record audio. The details about Camera 3 can be found in Section 2.3.1 (under Camera 1).

The nonlinear distortions in the videos were removed using a standard calibration procedure [138] and the videos were manually vertically cropped¹ to the rink boundaries (Fig 3.1). To evaluate the generalization of the play segmentation system over camera parameters, we varied the roll and tilt of Camera 3 by roughly ± 10 deg between games and periods. We manually masked out the non-rink regions for all games. The results for Camera 3 are seen in Fig 3.2. Unlike the dataset created in the previous chapter, the videos in this dataset were not rectified to a common viewpoint.

We manually vertically cropped all videos to the rink after recording; due to variation in camera roll, two different vertical crops were employed for Camera 3, depending upon the game and period. The final resolutions of all videos in the input dataset are listed in Table 3.1.

The 9 games recorded by Cameras 1 and 2, and Games 2 and 3 recorded by Camera 3 were all played at the same rink, while Game 1 recorded by Camera 3 was played at a different rink. All games recorded by Cameras 1 and 2 were captured in their entirety, while only the first two of three periods in each game were captured by Camera 3. The duration of each game in the dataset is shown in Table 3.2.

We manually ground-truthed all 12 games by marking the start and end of play intervals. For Cameras 1 and 2, we marked the start of play as the time instant when the referee

¹This is a one-time operation for each camera position.



(a) Camera 1



(b) Camera 2



(c) Camera 3 Game 1 Period 1



(d) Camera 3 Game 2 Period 1



(e) Camera 3 Game 3 Period 1

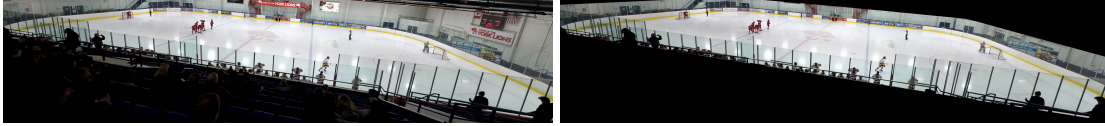
Figure 3.1: Example frames from the dataset, after correction for radial distortion and vertical cropping to the rink boundaries.



(a) Game 1 Period 1



(b) Game 1 Period 2



(c) Game 2 Period 1



(d) Game 2 Period 2



(e) Game 3 Period 1



(f) Game 3 Period 2

Figure 3.2: Non-rink regions are manually masked out for all Camera 3 games. Left: images vertically cropped to the rink boundaries. Right: non-rink regions masked out.

dropped the puck during a face-off and the end of play by when the referee was seen to blow the whistle. Since we had audio for Camera 3, we instead identified state changes by the auditory whistle cue, marking both the beginning and end of whistle intervals, which were found to average 0.73 sec in duration.

Camera	Horizontal	Vertical
1	2,758	500
2	5,930	1,080
3	3,550	630 or 780

Table 3.1: Cropped video resolutions for all games in the input dataset. The vertical cropping for Camera 3 took two values depending on the camera tilt.

Camera	Game	# frames	Duration (hh:mm:ss)
1	1	120,692	01:07:03
1	2	185,825	01:43:14
1	3	161,825	01:29:54
1	4	167,225	01:32:54
2	1	185,282	01:42:56
2	2	158,792	01:28:19
2	3	159,992	01:28:53
2	4	197,942	01:49:58
2	5	163,292	01:30:43
3	1	63,249	00:35:08
3	2	103,685	00:57:36
3	3	89,737	00:49:51

Table 3.2: Duration of each game in the input dataset.

3.4 Method

The pipeline of our system for automatic play segmentation of hockey videos is shown in Fig 3.3. Our system uses visual cues from a single wide-field camera and combines it with auditory cues, when available. We combined the visual and auditory cues in a hidden Markov model [90] framework which allows for probabilistic integration of both these cues. We use a method for unsupervised adaptation of the emission distributions of the HMM at inference, and show that this is important to derive benefit from the auditory

cues. The HMM predicts the state on each frame. Based on this prediction, we determine whether to *keep* or *cut* the frame from the game recording.

In this section, I describe how we computed the visual and auditory cues, formulated the HMM and the method for unsupervised domain adaptation.

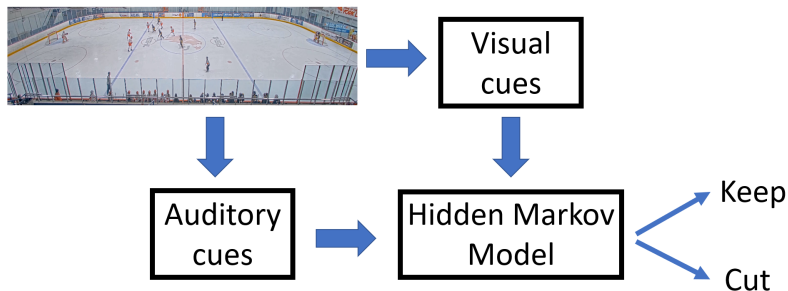


Figure 3.3: Pipeline of our system for automatic play segmentation of hockey videos.

3.4.1 Visual cues

We explored the utility of two different visual cues: 1) maximum optic flow and 2) deep visual cue.

3.4.1.1 Maximum optic flow

Players tend to move faster during play than breaks in ice hockey games. Hence, we explored the utility of the visual cue from optical flow.

We used the OpenCV implementation of Farneback’s dense optic flow algorithm [33] and selected the flow fields lying within bounding boxes of players detected using the fine-tuned Faster RCNN player detector that has been discussed in Section 2.5.2.1.2. Fig 3.4 shows the optic flow field contained within the bounding box of the detector players on a sample image.

Motion energy is generally higher during periods of play than during breaks, but given the sparse nature of the flow it is not immediately clear how to optimally aggregate the flow signal to create the strongest classifier. To explore this, we calculated the generalized mean of the flow vectors for Game 1 recorded using Camera 3.

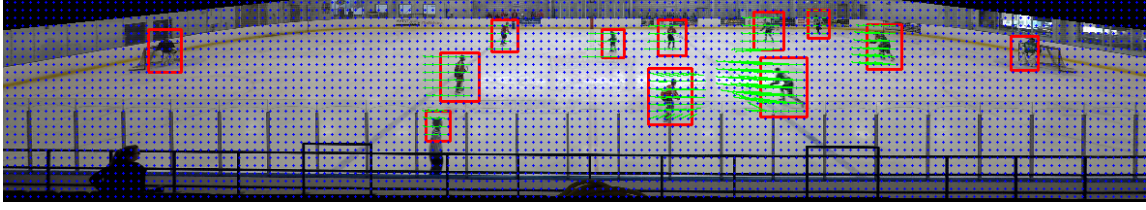


Figure 3.4: Example optic flow field within bounding boxes of detected players.

Let M_p be the generalized mean for an exponent p . M_p is formulated as

$$M_p(x_1, \dots, x_n) = \left(\frac{1}{n} \sum_{i=1}^n x_i^p \right)^{\frac{1}{p}} \quad (3.1)$$

For each proposed value of the exponent p , we identified the threshold that minimized classification error in distinguishing play and no-play frames. Interestingly, we found that error rate was lowest for very high exponents (Fig 3.5). This led to a very simple and computationally efficient visual cue: the L^∞ norm of the optic flow, i.e., the maximum flow vector magnitude within detected player bounding boxes.

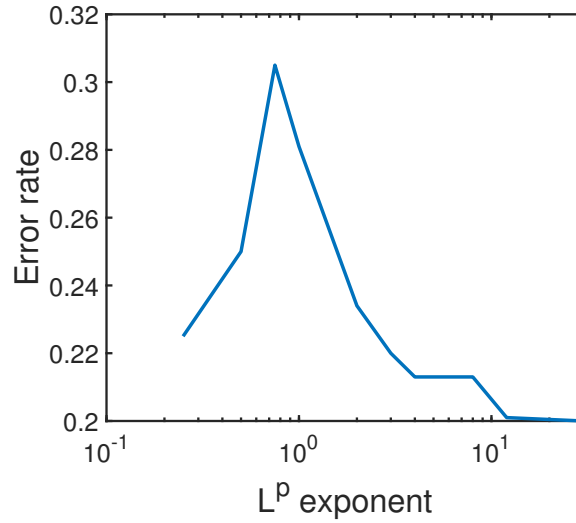


Figure 3.5: Error rate as a function of the L^p exponent used to aggregate the optic flow field.

3.4.1.2 Deep visual cue

We found the maximum optic flow visual cue is fallible: motion on the ice can sometimes be substantial during breaks and sometimes quite limited during periods of play. This motivated the development of a more complex visual cue formed from the logits of a deep network classifier.

A CNN (deep network classifier) was trained to classify each frame as belonging to play or no-play classes² (Fig 3.7). The 6-channel input feature map (Fig 3.6) to the CNN consisted of a) the RGB image, b) horizontal and vertical optic flow maps and c) binary player position mask. All feature maps were normalized to have zero-mean and unit-variance, and resized to 150×60 pixels. The training dataset was augmented by left-right mirroring.

We used PyTorch to develop the network for play/no-play classification, which was loosely based on the network used for tracking the puck (Section 2.5.2.2). As shown in Fig 3.7, the network consisted of two *conv-pool* modules followed by two fully connected layers. A max pooling layer followed each convolution layer and dropout was used between every fully connected layer. Cross-entropy loss between the predicted class and ground truth class was minimized using a SGD optimizer. The model was trained for 20 epochs with an initial learning rate of 0.01 and weight decay of 0.01. The learning rate was decreased by 50% every 5 epochs.

The pre-softmax (logit) layer output difference of the trained model was used as the *deep visual cue*. We trained a separate model for each camera. For Cameras 1 and 2, one game was used for validation and one for test, and the remaining games used for training. For Camera 3, one game was used for test, one period from one of the other games was used for validation, and the remaining data was used for training.

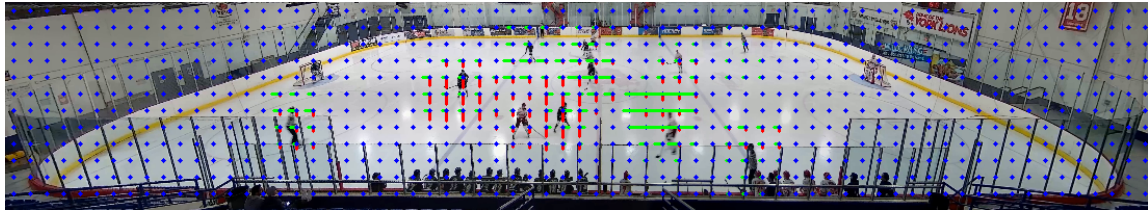
3.4.1.3 Evaluation

We compared our two visual classifiers (described in Section 3.4.1.1 and Section 3.4.1.2) against two baseline deep visual classifiers. The input data to the two baseline classifiers was the 512-dimensional output from the final fully connected layer of the ResNet18 network [46] pre-trained on the ImageNet dataset [27].

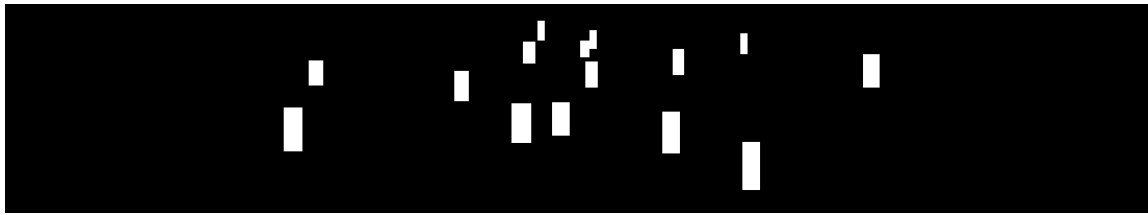
²For Camera 3, whistle frames were included in the play intervals.



(a) RGB image



(b) Horizontal (green) and vertical (red) optic flow



(c) Binary player position mask

Figure 3.6: Input data to deep visual play/no-play classifier.

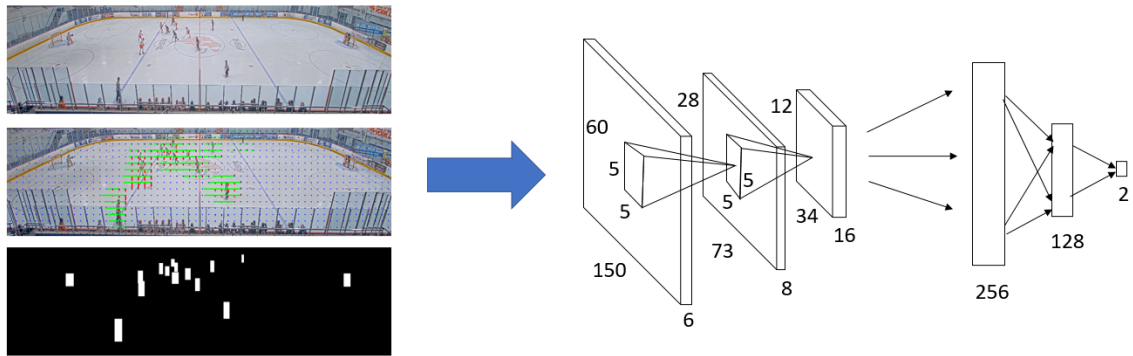


Figure 3.7: Deep visual classifier trained to classify each frame as belonging to play or no-play classes.

The first classifier was a deep network consisting of two fully connected layers of dimensions 128 and 64, followed by a play/no-play softmax layer. The learning rate for this network was 0.001, weight decay was 0.01 and it was trained for 10 epochs. The second classifier was a Support Vector Machine (SVM) [23] using an RBF kernel.

To determine the optimal visual cue, we evaluated the performance of the four visual classifiers in classifying each frame as belonging to play and no-play. We measured the performance of the classifier in terms of the Area Under Curve (AUC) score. The AUC score measures the area under the Receiver Operating Characteristic (ROC) curve. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) for different thresholds. It measures the ability of a classifier to distinguish between classes at a given threshold. The AUC score summarizes the performance of a classifier across all thresholds. AUC score takes values in $[0, 1]$, with 0 indicating a classifier that classifies all positive examples as negative and all negative examples as positive, and 1 indicating a classifier that correctly classifies all positive and negative samples.

For each camera, the AUC score was measured through leave-one-out cross validation, and was averaged across all cross-validation folds. The results are shown in Table 3.3. The AUC scores of all four visual classifiers are good across all cameras, indicating that these cues/classifiers are good at differentiating play and no-play frames. Across all cameras, the performance of the baseline classifier with a deep network (ResNet18 + FC) was better than that of the baseline classifier with SVM (ResNet18 + SVM). The performance of all classifiers are worse on Camera 3 than Cameras 1 and 2. This was because the roll and tilt varied across different games recorded using Camera 3, while Cameras 1 and 2 were fixed stationary cameras.

The performance of the maximum optic flow visual cue is worse than the baselines on Cameras 1 and 2. However, on Camera 3, the AUC score is significantly better. Since the camera roll is varied across different games, maximum optic flow cue is less affected by these changes than the ResNet18 model whose input is the RGB image. Across all cameras, the best performance was obtained using our deep visual cue.

	AUC scores		
	Camera 1	Camera 2	Camera 3
ResNet18 + FC	0.923 ± 0.018	0.907 ± 0.052	0.598 ± 0.03
ResNet18 + SVM	0.884 ± 0.009	0.844 ± 0.014	0.545 ± 0.01
Maximum optic flow	0.885 ± 0.011	0.818 ± 0.008	0.799 ± 0.028
Deep visual cue	0.977 ± 0.004	0.966 ± 0.005	0.819 ± 0.053

Table 3.3: AUC scores for visual play/no-play classification.

3.4.2 Auditory cue

In ice hockey, referees blow their whistles to start and stop play. We therefore explored the utility of auditory cues for classifying play and no-play frames. While not directly informative of the current state, the whistle can serve as an indicator of the transitions between the play state and no-play state.

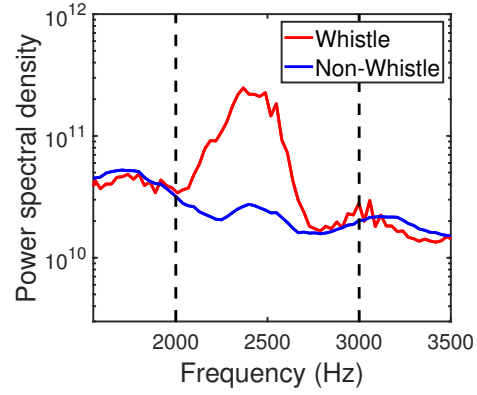
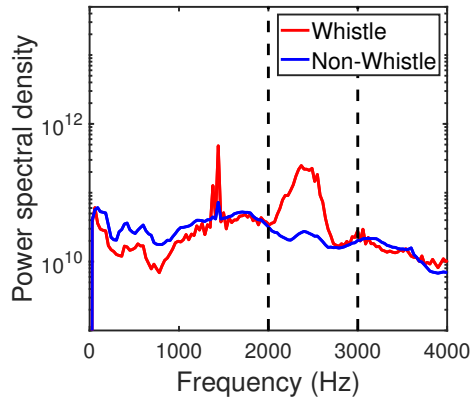
As described in Section 3.3, the three games recorded using Camera 3 also recorded synchronized audio at 48 kHz. Since the video was recorded at 30 fps, we partitioned the audio signal into 33 msec intervals, temporally aligned with the video frames. As the audio was sampled at 48 kHz, each interval consisted of 1,600 samples. The audio samples in each interval were normalized to have zero-mean. We then calculated the power spectrum density (PSD) for each interval as

$$P(f) = S(f) S^*(f) \quad (3.2)$$

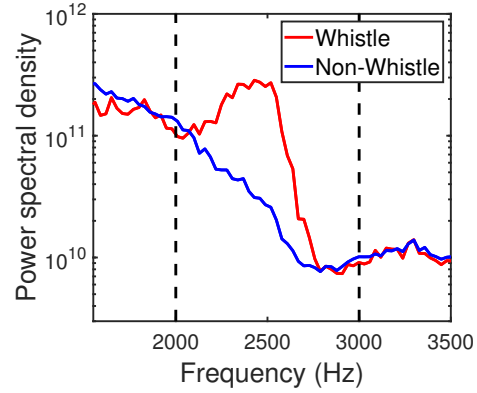
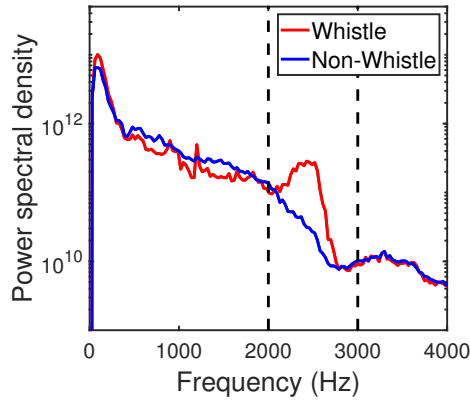
where $S(f)$ and $S^*(f)$ are the Fourier Transform and conjugate Fourier Transform of an interval of audio samples at the frequency f .

Fig 3.8 shows the PSD averaged over whistle and non-whistle intervals for all three games. These plots reveal two important facts. First, the overall volume of sound varies widely from game to game: While Game 1 is relatively quiet, Games 2 and 3 are quite noisy, with a lot of power in the low frequencies. Second, most of the whistle power lies in the 2-3 kHz range, however that power is not distributed evenly and the power of that signal and hence the signal-to-noise ratio varies widely from game to game.

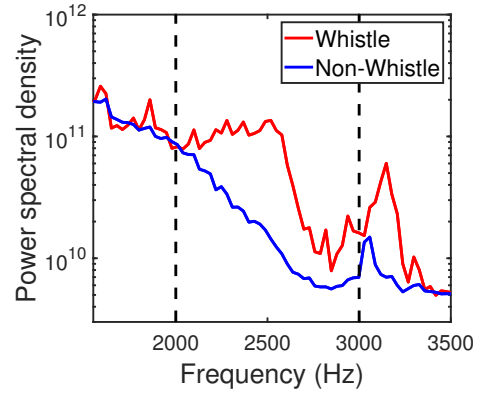
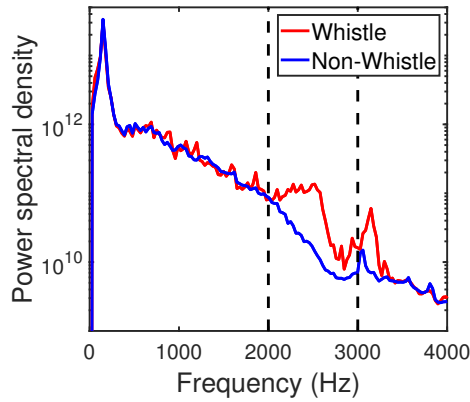
To form a decision variable for each interval, we considered two candidate detectors: 1) Bandpass filter and 2) Wiener filter.



(a) Game 1



(b) Game 2



(c) Game 3

Figure 3.8: Average power spectrum for whistle and non-whistle intervals. The right plot shows a zoom-in on the main frequency band of the whistle (2-3 kHz).

3.4.2.1 Bandpass filter

We computed the integral of the PSD over the 2-3 kHz band. This is probabilistically optimal if both the signal and noise are additive, stationary, white Gaussian processes and the PSDs are identical outside this band.

3.4.2.2 Wiener filter

Fig 3.8 shows that in fact the signal and noise are not white. Relaxing the condition that the PSDs be white and identical outside the 2-3 kHz band, for longer intervals (many samples), it can be shown (Section 5.5 in [59]) that probabilistically near-optimal detection is achieved by taking the inner product of the stimulus PSDs with the Wiener filter

$$H(f) = \frac{P_{ss}(f)}{P_{ss}(f) + P_{nn}(f)} \quad (3.3)$$

where $P_{ss}(f)$ and $P_{nn}(f)$ are the PSD of the signal (whistle) and noise, respectively, as a function of frequency f .

In our case, we do not have direct knowledge of the whistle and noise PSDs and so must estimate them from the training data. In our videos, we know that

$$P_{ss}(f) \approx P_W(f) - P_{NW}(f) \quad (3.4)$$

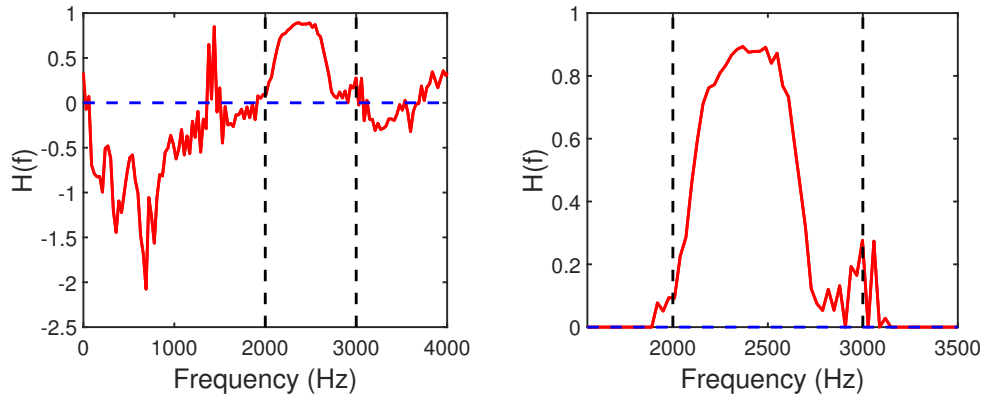
$$P_{nn}(f) \approx P_{NW}(f) \quad (3.5)$$

where $P_W(f)$ and $P_{NW}(f)$ are the average PSDs over whistle and non-whistle training intervals, respectively. Substituting Eq 3.4 and Eq 3.5 in Eq 3.3, we have the response of the Wiener filter as

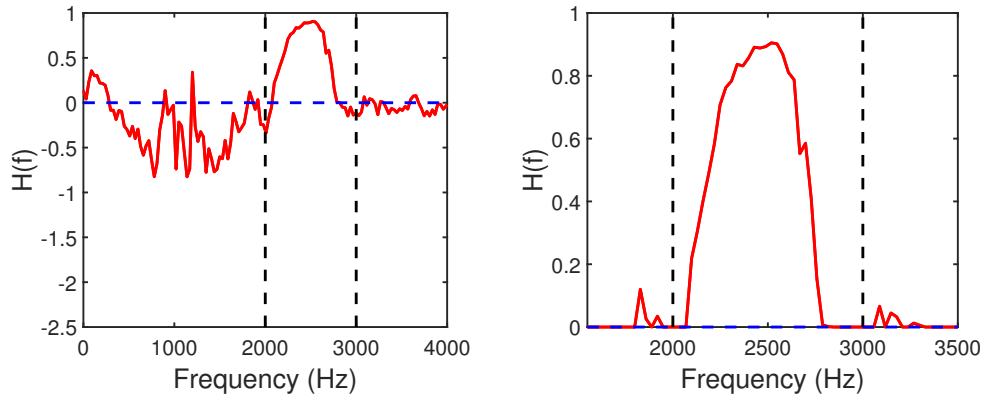
$$H(f) \approx \frac{P_W(f) - P_{NW}(f)}{P_W(f)} \quad (3.6)$$

$$= 1 - \frac{P_{NW}(f)}{P_W(f)} \quad (3.7)$$

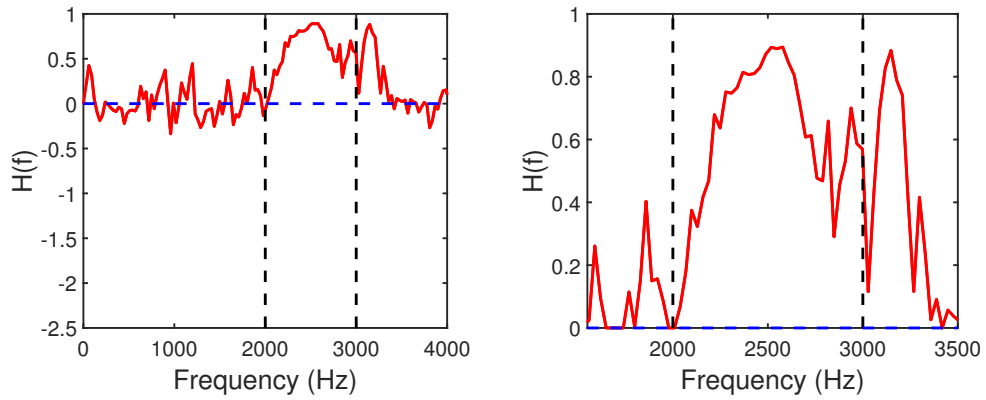
Fig 3.9 show the resulting Wiener filters $H(f)$ estimated for each of the three games recorded by Camera 3. The filter is largely positive in the 2-3 kHz range but can become negative outside this range. This suggests that in fact the signals are not exactly stationary and/or additive. This could be due to two possibilities: 1) some acoustic signals are more



(a) Game 1



(b) Game 2



(c) Game 3

Figure 3.9: The Wiener filter response $H(f)$ for each of the three games. Right panels show a zoom-in of the response of a rectified Wiener filter in the main frequency range of 2-3 kHz.

likely to occur in non-whistle than in whistle intervals and 2) when the whistle is blown, auto-gain circuitry in the camera attenuates energy outside the whistle band.

To handle these deviations from our assumptions, we evaluated three versions of the Wiener filter:

1. **Wiener filter 1:** Take the inner product of the stimulus with the estimated Wiener filter over the entire frequency range, including negative values.
2. **Wiener filter 2:** Take the inner product of the stimulus with the rectified Wiener filter (negative values clipped to 0). Shown in Fig 3.9 (right panels).
3. **Wiener filter 3:** Take the inner product of the stimulus with the rectified Wiener filter (negative values clipped to 0), only over the 2-3 kHz range.

3.4.2.3 Evaluation

To determine the optimal auditory cue, we evaluated the performance of four detectors (including Bandpass filter) in classifying whistle and non-whistle frames/instances. The performance was evaluated through leave-one-out cross validation and we measured the average AUC score across all cross-validation folds. The results are shown in Table 3.4.

Wiener filter 3 performs better than Wiener filter 1 and Wiener filter 2. This could be because of considering the response only in the 2-3 kHz range. The PSD is vastly different across games, outside this frequency band. The advantage of Wiener filter 3 over the bandpass filter could be because of its ability to weight the input by the non-uniform SNR within the 2-3 kHz band. Overall, Wiener filter 3 performs the best and we use this auditory cue for all experiments in this chapter.

	AUC score
Bandpass filter	0.919 ± 0.039
Wiener filter 1	0.779 ± 0.105
Wiener filter 2	0.809 ± 0.093
Wiener filter 3	0.943 ± 0.028

Table 3.4: The average cross-validated AUC score of the four whistle detectors for Camera 3.

3.4.3 Hidden Markov model

Visual cues are seen to be useful for classifying video frames individually as play/no-play and auditory cues are useful for detecting the whistle. In order to put these cues together and reliably excise periods of non-play from the entire video, we need a model that captures statistical dependencies over time. We employed a hidden Markov model (HMM) [90].

3.4.3.1 Background

A Markov chain is a model of a stochastic dynamical system that evolves in discrete time over a finite state space, and that follows the Markov property or assumption. The Markov property states that when conditioned on the state at time t , the state at time $t + 1$ is independent of all other past states. Thus, when predicting the future, the past does not matter, only the present matters.

Consider a sequence of observations $O = \{o_1, o_2, \dots, o_T\}$ and a state sequence $Q = \{q_1, q_2, \dots, q_T\}$. The Markov property is mathematically represented as:

$$P(q_i | q_1, \dots, q_{i-1}) = P(q_i | q_{i-1}) \quad (3.8)$$

The Markov chain is specified by two components: 1) initial probability distribution over the states and 2) state transition probabilities.

A model that is built upon Markov chains and is very popular in many machine learning applications is the Hidden Markov Model (HMM) [90]. A Markov chain is useful when we need to compute the probability for a sequence of observable states. However, sometimes the states of interest are hidden, such as play and no-play states in hockey videos. An HMM is a model that consists of a Markov chain whose state at any given time is not observable; however, at each instant the system emits a symbol whose distribution depends on the state. Hence, the model is useful for capturing distribution of the hidden states in terms of observable quantities known as symbols/observations. In addition to the Markov property given by Eq 3.8, the HMM has an extra assumption that given the state at that instant, the probability of the emitted symbol/observation is independent of any other states and any other observations. This is mathematically represented as:

$$P(o_i | q_1, \dots, q_i, \dots, q_T, o_1, \dots, o_i, \dots, o_T) = P(o_i | q_i) \quad (3.9)$$

An HMM is specified by the following parameters:

1. **Initial probability** distribution over states, π_i , such that $\sum_{i=1}^N \pi_i = 1$.
2. **State transition probability** matrix A , where each element a_{ij} represents the probability of moving from state i to state j , such that $\sum_{j=1}^N a_{ij} = 1 \forall i$.
3. **Emission probabilities** $B = b_i(o_t)$, which indicates the probability of an observation o_t being generated from state i .

An HMM is characterized by three learning problems [90]:

1. **Likelihood:** Given an HMM $\lambda = (A, B)$ and an observation sequence O , determine the likelihood of $P(O|\lambda)$.
2. **Decoding:** Given an HMM $\lambda = (A, B)$ and an observation sequence O , what is the best sequence of hidden states Q .
3. **Learning:** Given an observation sequence O and the set of possible states in the HMM, learn the HMM parameters A and B .

For our application of play segmentation of hockey videos, we want to determine if a given frame belongs to a play segment or a no-play segment, and the observations emitted are the visual cue and optional auditory cue. After learning the model, given the sequence of visual and optional auditory observations, we estimate whether each frame belongs to play or no-play states.

Since we have the sequence of states available for the training data, estimating the state transition probability matrix is straightforward and is a maximum likelihood estimate. In a similar manner, the observation likelihoods can be modelled from the training data. We explored two different methods to model the likelihoods: 1) Gaussian Mixture Models (GMMs) and 2) Kernel Density Estimation (KDE).

A Gaussian Mixture Model (GMM) is a probabilistic model that fits a finite number of Gaussian distributions with unknown parameters to a set of data points. The GMM is parameterized by the means and variances of the components and the mixture coefficients.

For a GMM with K components, the i^{th} component has a mean μ_i , variance σ_i^2 and component weight of ϕ_i . The probability density function, $f(x)$, of a such a GMM is given as:

$$f(x) = \sum_{i=1}^K \phi_i \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(x - \mu_i)^2}{2\sigma_i^2}\right) \quad (3.10)$$

The mixing/component weights ϕ_i satisfy the constraint $\sum_{i=1}^K \phi_i = 1$.

If the number of components in the GMM is known, the model parameters can be estimated using the Expectation Maximization (EM) algorithm [26].

An alternative non-parametric approach to modelling the likelihoods is Kernel Density Estimation (KDE). Gaussian KDE approximates the probability density at a point as the average of Gaussian kernels centered at observed values. The probability density function, $f(x)$, for Gaussian KDE is given as:

$$f(x) = \frac{1}{N} \sum_{j=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - x_j)^2}{2\sigma^2}\right) \quad (3.11)$$

where N is the total number of data points.

Although KDE is expressed as a Gaussian mixture, there are two major differences to the GMM density in Eq 3.10. First, the number of Gaussian components in Eq 3.11 is N (the number of data points), which is typically significantly more than the M components in a GMM (Eq 3.10). Second, the variance, σ^2 , is the same for all components in Eq 3.11. The only parameter to be estimated for KDE is the variance, σ^2 . We estimate it using Silverman's rule [104].

The learnt state transition matrix and the emission probabilities are used at inference to estimate the sequence of states. A popular approach to determine the optimal sequence of hidden states is the Viterbi algorithm [122]. It determines the maximum a posteriori sequence of hidden states, i.e., the most probable state sequence. As a result, it is difficult to tune the system to control type 1 and type 2 errors. Instead, we estimate the marginal posteriors at each time instant. We can then adjust a threshold to achieve the desired balance of type 1 and type 2 errors.

Let $O = \{o_1, o_2, \dots, o_T\}$ be the sequence of observations and $Q = \{q_1, q_2, \dots, q_T\}$ be a sequence of hidden states. $q_t \in \{1, 2, \dots, N\}$, where N is the number of states;

$N = 2$ in our application. T is the number frames in the video. The maximum posterior of marginals (MPM) returns the state sequence $\hat{\mathbf{Q}}$, where

$$\hat{\mathbf{Q}} = \{\arg \max_{q_1} P(q_1|o_1, \dots, o_T), \dots, \arg \max_{q_T} P(q_T|o_1, \dots, o_T)\} \quad (3.12)$$

Let $\lambda = (A, B)$ be an HMM model with state transition matrix A and emission probabilities B . The posterior probability of being in state j at time t is given as

$$\begin{aligned} \gamma_t(j) &= P(q_t = j|O, \lambda) \\ &= \frac{P(q_t = j, O|\lambda)}{P(O|\lambda)} \end{aligned} \quad (3.13)$$

The forward probability, $\alpha_t(j)$, is defined as the probability of being in state j after seeing the first t observations, given the HMM λ . The value of $\alpha_t(j)$ is computed by summing over the probabilities of all paths that could lead to the state j at time t . It is expressed as

$$\begin{aligned} \alpha_t(j) &= P(o_1, o_2, \dots, o_t, q_t = j|\lambda) \\ &= \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t) \end{aligned} \quad (3.14)$$

where a_{ij} is the state transition probability from previous state $q_{t-1} = i$ to current state $q_t = j$. $\alpha_{t-1}(i)$ is the forward probability of being in state i at time $t - 1$, and it can be recursively computed.

The backward probability, $\beta_t(j)$, is defined as the probability of seeing the observations from time $t + 1$ to T , given that we are in state j at time t and given the HMM λ . It is expressed as

$$\begin{aligned} \beta_t(j) &= P(o_{t+1}, o_{t+2}, \dots, o_T|q_t = j, \lambda) \\ &= \sum_{i=1}^N a_{ji} b_j(o_{t+1}) \beta_{t+1}(i) \end{aligned} \quad (3.15)$$

where $\beta_{t+1}(i)$ is the backward probability being in state i at time $t + 1$, and it can be computed recursively.

Putting the forward probability ($\alpha_t(j)$) and backward probability ($\beta_t(j)$) in Eq 3.13, we have the posterior probability $\gamma_t(j)$ as

$$\gamma_t(j) = \frac{\alpha_t(j) \beta_t(j)}{\sum_{i=1}^N \alpha_t(i) \beta_t(i)} \quad (3.16)$$

Finally, the state sequence maximizing the posterior marginals (MPM) is computed as

$$\hat{\mathbf{Q}} = \{\arg \max_j \gamma_1(j), \arg \max_j \gamma_2(j), \dots, \arg \max_j \gamma_T(j)\} \quad (3.17)$$

In our application, mislabelling a play state as a no-play state might be more serious than mislabelling a no-play state as a play state, as the former could lead to the viewer missing a key part of the game, whereas the latter would just waste a little time. Thus, rather than selecting the MPM solution, we adjust the threshold on the posterior to achieve the desired trade-off between hits and misses.

3.4.3.2 HMM framework and observation likelihoods

The visual cues described in Section 3.4.1 and the auditory cue described in Section 3.4.2 were used as the observations to the HMM. Since Cameras 1 and 2 did not record audio, only the visual cue was available. Hence, we employed a 2-state model (play/no-play) (Fig 3.10a). As Camera 3 recorded audio, we employed a 4-state model (Fig 3.10b). The initial state probabilities were computed from the training data as the percentage of frames belonging to either Play or No-play state across all games for each camera. The results are seen in Table 3.5.

Camera	Initial probabilities (π)	
	Play	No-play
1	0.629	0.371
2	0.656	0.344
3	0.699	0.301

Table 3.5: Mean initial state probabilities for each camera.

Similarly, the probability of transitioning between states is computed from the training data as the proportion of frames where the desired transition happens. For example, the

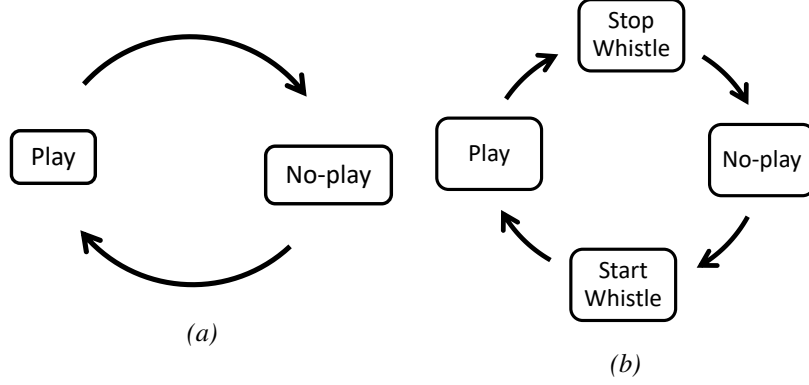


Figure 3.10: State transition graphs for (a) 2-state and (b) 4-state HMMs.

transition probability of going from No-play state to Play state is computed as the fraction of No-play frames where the next state was Play. The results are seen in Table 3.6.

Camera	Transition	Probability
1	No-play→Play	0.00100
1	Play→No-play	0.00053
2	No-play→Play	0.00092
2	Play→No-play	0.00054
3	No-play→Start Whistle	0.00117
3	Start Whistle→Play	0.04973
3	Play→Stop Whistle	0.00050
3	Stop Whistle→No-play	0.04709

Table 3.6: Mean state transition probabilities for each camera.

The auditory and visual cues were normalized to have zero-mean and unit-variance. The two features were assumed to be conditionally independent. Hence, the observation likelihoods were modelled separately.

In order to model the auditory and visual cues using a GMM, we have to determine the optimal number of components. We varied the number of components and measured the AUC score for classifying play and no-play frames. The GMM model was learnt on the training games. Given a test game, the ratio of the likelihoods of play and no-play

states was used to compute the AUC score for that game. The AUC score was averaged across all games for each camera through leave-one-out cross validation. The results are shown in Table 3.7. We found that the discriminative power of the deep visual cue was superior to that of the maximum optic flow cue. The 3-component GMM achieved the best results for both 2-state and 4-state HMM using either visual cue. For the 4-state model, the likelihoods of the whistle states were added to the likelihood of the play state.

# GMM components	2-state HMM		4-state HMM	
	OF	DV	OF+Audio	DV+Audio
1	0.8394	0.9149	0.7366	0.7337
2	0.8398	0.9150	0.7378	0.7349
3	0.8399	0.9152	0.7433	0.7454
4	0.8374	0.9151	0.7369	0.7346
5	0.8387	0.9150	0.7378	0.7363
7	0.8387	0.9143	0.737	0.7362
10	0.8379	0.9145	0.7374	0.7368

Table 3.7: Cross-validated AUC scores as a function of the number of GMM components. OF: maximum optic flow cue. DV: deep visual cue.

Since the KDE models a Gaussian for each data point, it is computationally expensive for long sequences/videos. We computed the histogram of the visual and auditory cues for a specified number of bins and then modelled the histogram of the observations using a Gaussian KDE. In a similar manner to the analysis for the optimal number of GMM components, we computed the AUC score to determine the optimal number of histogram bins. The results are seen in Table 3.8. The discriminative power of the deep visual cue was superior to that of the maximum optic flow cue. The best results were obtained when the observation was a 32-bin histogram.

As seen in Table 3.7 and Table 3.8, the AUC score was better when modelling the likelihoods using a GMM rather than KDE. Hence, in the subsequent experiments, we modelled the likelihoods using a 3-component Gaussian Mixture Model (GMM).

The conditional densities for the maximum optic flow cue and deep visual cue are shown in Fig 3.11 and Fig 3.12, respectively. The conditional densities for the auditory

# histogram bins	2-state HMM		4-state HMM	
	OF	DV	OF+Audio	DV+Audio
8	0.8221	0.8984	0.6704	0.6661
16	0.8345	0.9099	0.6967	0.6952
32	0.8376	0.9143	0.6986	0.7008
64	0.8376	0.9142	0.6904	0.6881
128	0.8373	0.9141	0.675	0.6747
256	0.8372	0.9136	0.6598	0.6603
512	0.8367	0.9133	0.6442	0.6471
1024	0.8360	0.9126	0.629	0.6338

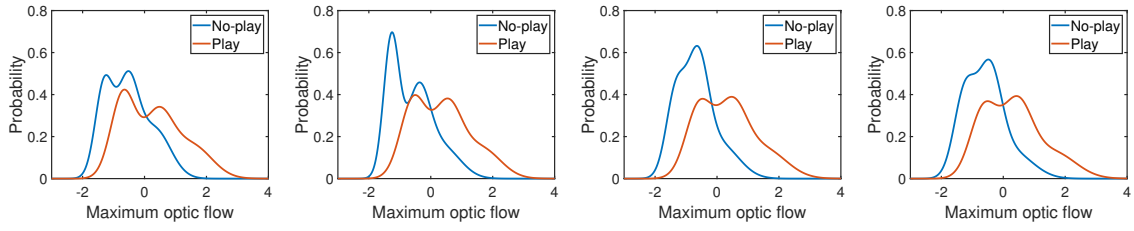
Table 3.8: Histogram of the visual and auditory cues were computed for the specified number of bins and modelled using a Gaussian KDE. The AUC score for classifying play and no-play frames was computed. OF: maximum optic flow cue. DV: deep visual cue.

cue of Wiener filter 3 detector on games from Camera 3 are shown in Fig 3.13. Recall that only Camera 3 was recorded with audio. Hence, four conditional densities are shown for Camera 3, including the distributions for start and stop whistles. The two whistle states are considered to be a part of play when reducing the 4-state HMM to a 2-state HMM.

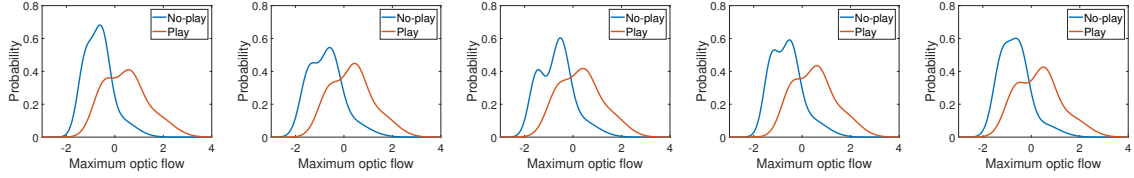
3.4.4 Domain adaptation

A fundamental part of machine learning is the problem of generalization, that is, how to make sure that a trained model performs well on unseen data. If the unseen data has a different distribution, i.e., a domain shift exists, the problem is significantly more difficult [89, 118].

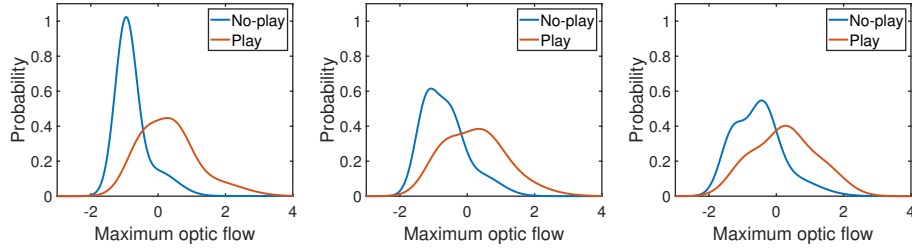
We learn the emission probabilities by modelling the observation likelihoods using a 3-component GMM on the training games. If the observation distribution is different between the training and test games, then there is a risk that the emission probabilities on the test game are wrong, and this will affect the estimated state sequence. One possible solution for this is to adapt the emission probabilities of the HMM at inference to accommodate these domain shifts.



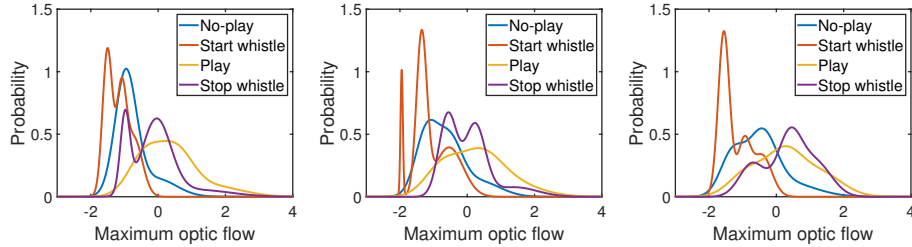
(a) Camera 1



(b) Camera 2



(c) Camera 3 - 2-state HMM



(d) Camera 3 - 4-state HMM

Figure 3.11: Conditional probability densities for the maximum optic flow visual cue on all games across all three cameras.

The standard approach to unsupervised HMM parameter learning is the Baum-Welch algorithm [5], which is a special case of the EM algorithm [26]. The Baum-Welch algorithm allows us to learn both the state transition probabilities A and the emission proba-

bilities B . This is the third problem (learning) that is characterized by using an HMM, as described in Section 3.4.3.1. We use the forward and backward probabilities to learn the state transition and emission probabilities.

Let $O = \{o_1, o_2, \dots, o_T\}$ be the sequence of observations and $Q = \{q_1, q_2, \dots, q_T\}$ be a sequence of hidden states. Let $\alpha_t(j)$ be the probability of being in state j after seeing the first t observations. Let $\beta_t(j)$ be the probability of seeing the observations from time $t + 1$ to T , given that we are in state j at time t . Let $\gamma_t(j)$ be the probability of being in state j at time t , given all observations.

First, we look at how to estimate the state transition probabilities A . We define $\hat{a}_{ij}$ as

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} \quad (3.18)$$

The probability of being in state i at time t and state j at time $t + 1$, given the observation sequence O and HMM $\lambda = (A, B)$, is given as

$$\begin{aligned} \xi_t(i, j) &= P(q_t = i, q_{t+1} = j | O, \lambda) \\ &= \frac{P(q_t = i, q_{t+1} = j, O | \lambda)}{P(O | \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{k=1}^N \alpha_t(k) \beta_t(k)} \end{aligned} \quad (3.19)$$

The expected number of transitions from state i to state j can be obtained by summing $\xi_t(i, j)$ over all frames t . Using Eq 3.19, we can rewrite Eq 3.18 as

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)} \quad (3.20)$$

As described in Section 3.4.3.2, the observation likelihoods were modelled using a 3-component GMM. Thus, the probability of seeing observation o_t in state j is given as

$$b_j(o_t) = \sum_{k=1}^M \phi_{kj} \mathcal{N}(o_t; \mu_{kj}, \sigma_{kj}^2) \quad (3.21)$$

where ϕ_{kj} , μ_{kj} and σ_{kj}^2 are the weight, mean and variance of the k^{th} component of the GMM of state j , and $\mathcal{N}$ is the Gaussian distribution with mean μ_{kj} and variance σ_{kj}^2 .

If we know the state for each observation sample, then estimating the emission probabilities B is trivial. The posterior probability $\gamma_t(j)$ gives the probability that observation o_t came from state j . In a similar manner to the EM algorithm, the Baum-Welch algorithm updates the weights, means and variances of the GMM as

$$\hat{\phi}_{kj} = \frac{\sum_{t=1}^T P_j(k|o_t, \Phi) \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad (3.22)$$

$$\hat{\mu}_{kj} = \frac{\sum_{t=1}^T o_t P_j(k|o_t, \Phi) \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad (3.23)$$

$$\hat{\sigma}_{kj}^2 = \frac{\sum_{t=1}^T (o_t - \hat{\mu}_{kj})^2 P_j(k|o_t, \Phi) \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad (3.24)$$

where Φ represents the current set of GMM parameters. $P_j(k|o_t, \Phi)$ is the probability that the observation o_t was from the k^{th} component of the GMM of state j . It is given as

$$P_j(k|o_t, \Phi) = \frac{\phi_{kj} \mathcal{N}(o_t; \mu_{kj}, \sigma_{kj}^2)}{\sum_{m=1}^M \phi_{mj} \mathcal{N}(o_t; \mu_{mj}, \sigma_{mj}^2)} \quad (3.25)$$

Thus, we can estimate the state transition probabilities A using Eq 3.20, and the emission probabilities B using Eq 3.22, Eq 3.23 and Eq 3.24. The iterative Baum-Welch algorithm is as follows. First, we initialize the state transition probabilities A and emission probabilities B . Next, we use Eq 3.16 to estimate $\gamma_t(j)$ given the state transition matrix A and emission probabilities B . Then, we use $\gamma_t(j)$ to update the state transition probabilities A and emission probabilities B , as shown above. We repeat this procedure iteratively until the difference in the log-likelihood between five successive iterations is less 0.1.

3.5 Results

Fig 3.14 shows an example of how the visual cue of maximum optic flow and auditory cue of Wiener filter 3 detector vary over time within each game state, for a 160-second sample video from Game 1 recorded using Camera 3. We observe that Wiener filter 3 has a sharp response during whistle periods. Our hypothesis of players moving faster during play than breaks (no-play) is backed up by the large values of the maximum optic flow cue during play frames and lower values during no-play frames.

Using the forward-backward algorithm, we compute the probability of being in state j at time t , $\gamma_t(j)$, for each state across all frames of the video. To temporally compress the video, frames are cut if $P(\text{no-play})$ exceeds a threshold η_0 . We define precision, recall and compression as

$$\text{Precision} = \frac{\# \text{ play \& whistle frames retained}}{\# \text{ frames retained}} \quad (3.26)$$

$$\text{Recall} = \frac{\# \text{ play \& whistle frames retained}}{\# \text{ play \& whistle frames in video}} \quad (3.27)$$

$$\text{Compression} = 1 - \frac{\# \text{ frames retained}}{\# \text{ frames in video}} \quad (3.28)$$

Varying η_0 sweeps out a precision-recall curve. Since no audio was available for Cameras 1 and 2, we evaluated the precision and recall for retaining play frames only. For Camera 3, as audio was available, we evaluated the precision and recall for retaining both play and whistle frames. We also evaluated the %compression at each rate of recall.

We evaluated both within-camera and between-camera performance. We also show the need for domain adaptation to benefit from the audio cues.

3.5.1 Within-camera performance (2-state HMM)

We evaluated the generalization of the system across different games for each camera by measuring the within-camera performance through leave-one-out cross validation. For each camera, the precision, recall and compression were measured through leave-one out cross validation across all games. These were then averaged across all three cameras.

The within-camera performance of the 2-state HMM (using visual cue only) is shown in Fig 3.15. We compared against two baselines: 1) Random - the lower bound baseline that randomly removes frames and 2) Ideal - the upper bound of an ideal model that accurately removes all no-play frames, before beginning to remove play frames.

We compared the within-camera performance using both the maximum optic flow cue and deep visual cue. Both cues were found to be significantly better than lower bound baseline (Random). The performance of the deep visual cue was significantly better than the maximum optic flow cue.

3.5.2 Between-camera performance (2-state HMM)

We evaluated the generalization of the system across different cameras by measuring the between-camera performance. We trained the 2-state HMM on all games from two cameras and then evaluated it on the games from the third camera. For example, a model was trained on all games from Cameras 1 and 2 and then evaluated on all games from Camera 3. We compared the between-camera performance to the within-camera performance on the third camera, as shown in Fig 3.16.

We found between-camera performance to be very similar to the within-camera performance across all cameras. Our model is able to generalize to different games, rinks and lighting conditions. The performance was worse on Camera 3 as compared to Cameras 1 and 2. Since Camera 3 was positioned closer to the ice surface as compared to Cameras 1 and 2, the fans are more visible and cause more occlusions in the video recording. Hence, the performance of the player detector could have been poorer on Camera 3, leading to less discriminative deep visual cues. In addition to occlusions, if the fans were moving during periods of no-play, this would also make the deep visual cue less discriminative.

3.5.3 Within-camera performance (4-state HMM)

The previous results were using the 2-state HMM with only the visual cues. Next, we evaluate the performance of the 4-state HMM that combines visual and auditory cues.

Three games were recorded with audio using Camera 3. We evaluated the performance of the 4-state HMM on these three games through leave-one-out cross validation. The precision, recall and compression were averaged across all three games. The results are seen in Fig 3.17.

We compared the performance of the 4-state HMM combining the visual cue and auditory cues with that of the 2-state HMM using only the visual cues. Combining auditory cues with the maximum optic flow cue significantly improved performance. However, no benefit was observed upon integration of the deep visual cue with the auditory cue (Fig 3.17).

3.5.4 Domain adaptation performance

We failed to observe any benefit of integrating the visual and auditory cues for Camera 3 once we used the strong deep visual cue. While the deep visual cues generalized well across cameras (Fig 3.16), the emission distributions of the auditory cues for Camera 3 seem to vary substantially across games (Fig 3.13). This could indicate a domain shift between the training and test data for the auditory cues.

We examine this domain shift by analysing the fit of the unconditional emission distribution learnt from the training data on the test data. We computed the unconditional emission distribution as

$$f(x) = \sum_{i=1}^N f_i(x)P(i) \quad (3.29)$$

where $f_i(x)$ and $P(i)$ are the emission distribution and prior for state i , respectively. N is the number of states; $N = 2$ or $N = 4$ in our system.

Fig 3.18 and Fig 3.19 visualizes the unconditional densities learnt from the training data on the histogram of the test data. We observed a slight domain shift in the emission distribution for the deep visual cue on Game 3. For the auditory cue, we observed a substantial domain shift for Games 1 and 2.

Domain shift can be overcome by adapting the HMM to the test data at inference. We use the Baum-Welch algorithm for unsupervised HMM parameter learning. As described in Section 3.4.4, both the emission probabilities and the state transition probabilities can be updated. We measured the %change in the values of the state transition matrix A , between the training and test games for Camera 3. We averaged the change across all three cross-validations folds. We found the average to be 4.48%. This is a small change which we did not anticipate influencing the model performance. Empirically, we found that updating the transition probabilities did not make any difference in the model performance. Hence, we only update the emission probabilities.

The emission distributions for deep visual cue and auditory cue after adaptation is shown in Fig 3.18 (right panels) and Fig 3.19 (right panels), respectively. The performance of the model before and after domain adaption is seen in Fig 3.20. We found no improvement in the performance of the 2-state HMM (visual cue only). However, there was a dramatic improvement in the performance of 4-state HMM (visual and auditory cue)

after domain adaptation.

In a similar manner, we compared the performance of the 2-state HMM (visual cue only) before and after domain adaptation on Cameras 1 and 2. The unconditional densities before and after domain adaptation are shown in Fig 3.21 and Fig 3.22 for Camera 1 and Camera 2, respectively. We found that the emission distributions for the deep visual cue learnt on the training data, modelled the test data distributions well. Hence, we found no benefit with domain adaptation, as seen in the precision-recall performance plots in Fig 3.23.

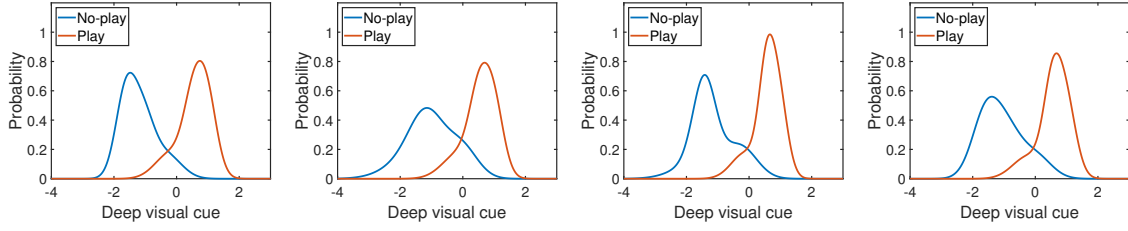
3.6 Conclusions & future work

We have developed a novel system for automatic play-break segmentation for hockey, and have shown its utility in temporally compressing hockey videos while maintaining high recall for periods of active play. We trained a deep network for play/no-play classification and found the pre-softmax (logit) output difference to be a much stronger visual cue than the maximum optic flow cue. Incorporation of an HMM framework accommodates statistical dependencies over time, allowing effective play/break segmentation and temporal video compression. We found that integration of auditory (whistle) cues could boost segmentation performance, but only by incorporating unsupervised adaptation of emission distribution models to accommodate domain shift. Our system was found to achieve temporal compression rates of 20-50% at a recall of 96%.

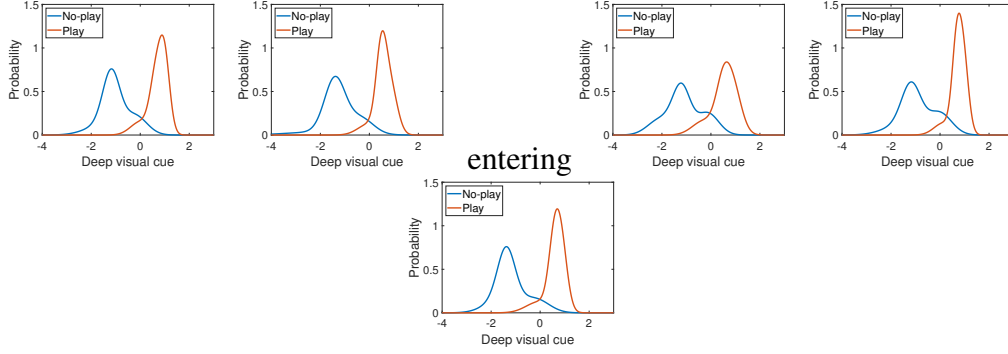
Some ideas that could be explored in the future are:

1. The auditory background noise is very high in Games 2 and 3 recorded by Camera 3. An improved whistle detector that is robust to strong background noise could further improve the performance of the model on Camera 3 games.
2. Currently, we have not analysed the minimum recall that does not affect the viewing experience of the temporally compressed video. This can be done through a user validation study of the temporally compressed videos at various values of recall, to determine the optimal recall.
3. Instead of separately learning the visual and auditory cues, a possible future idea is

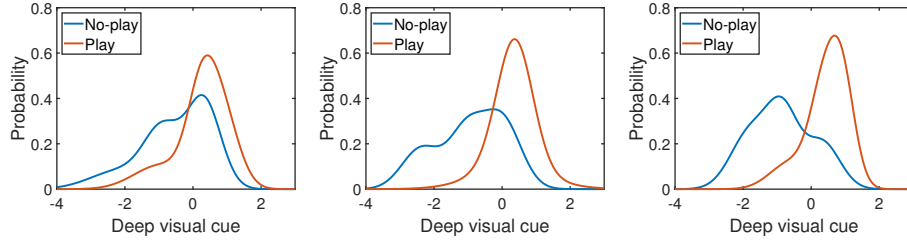
to fuse the audio and visual input in a deep network that can be trained to predict all four states (play, no-play, whistle ON and whistle OFF).



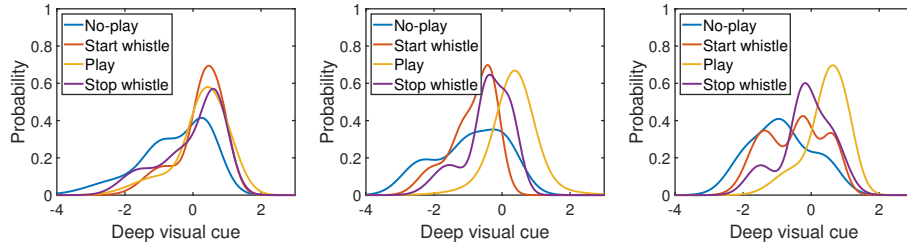
(a) Camera 1



(b) Camera 2



(c) Camera 3 - 2-state HMM



(d) Camera 3 - 4-state HMM

Figure 3.12: Conditional probability densities for the deep visual cue on all games across all three cameras.

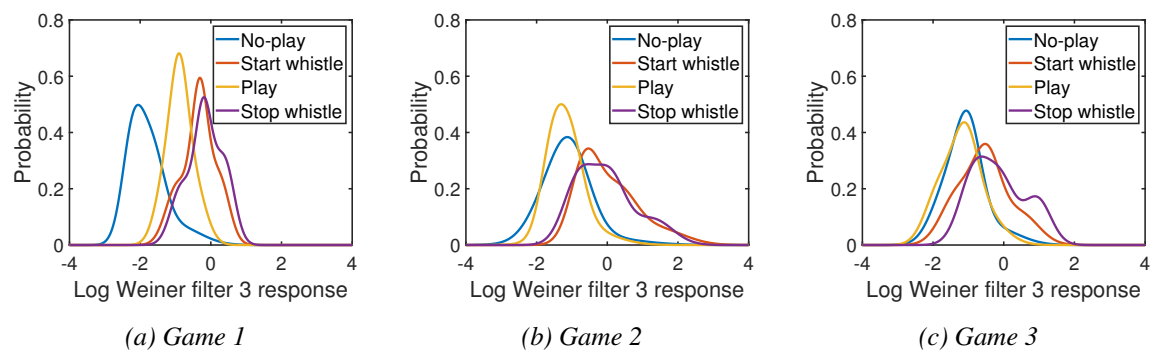


Figure 3.13: Conditional densities for the auditory cue of Wiener filter 3 detector on games from Camera 3.

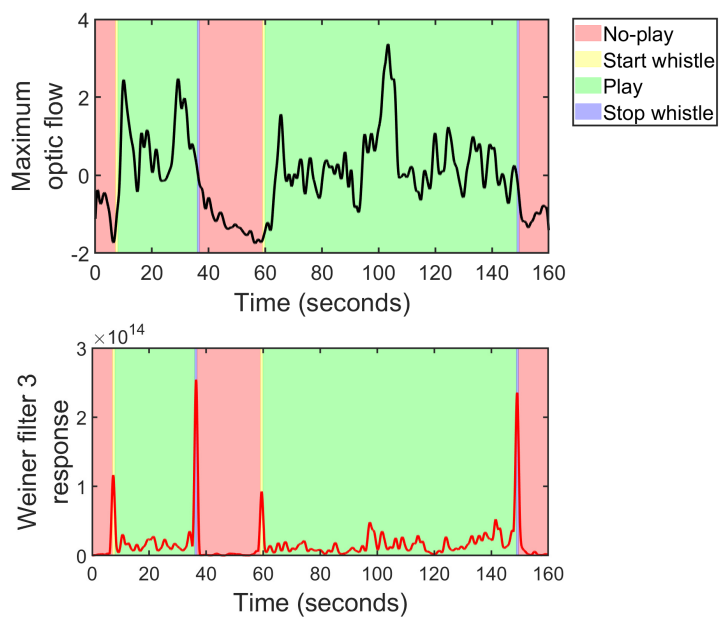


Figure 3.14: Visual cue of maximum optic flow and auditory cue of Wiener filter 3 detector for an example video segment from Camera 3 Game 1.

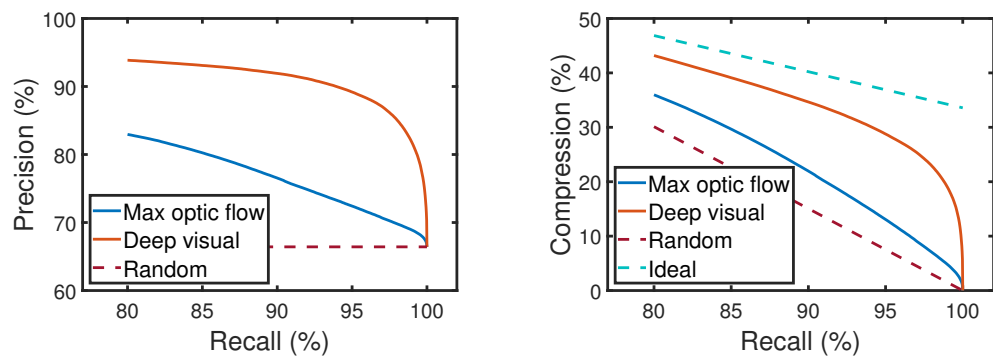
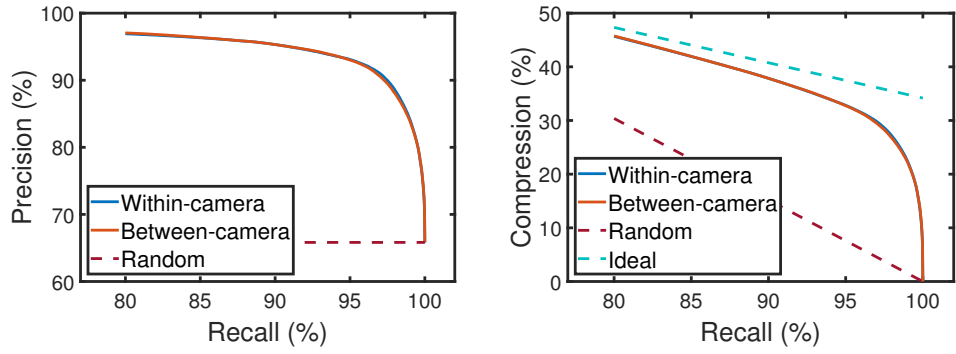
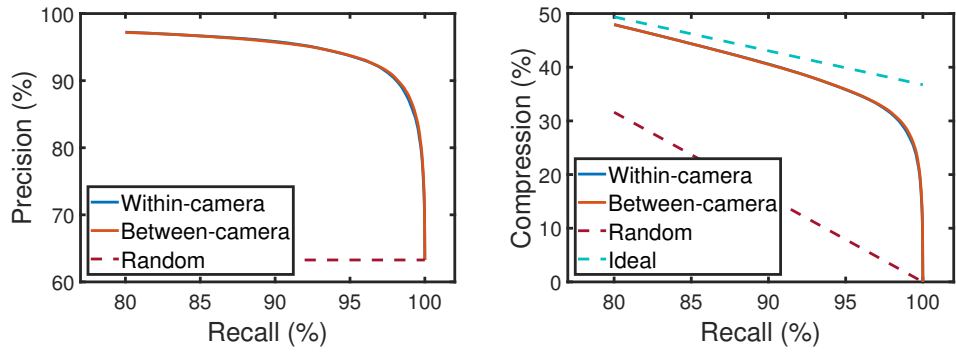


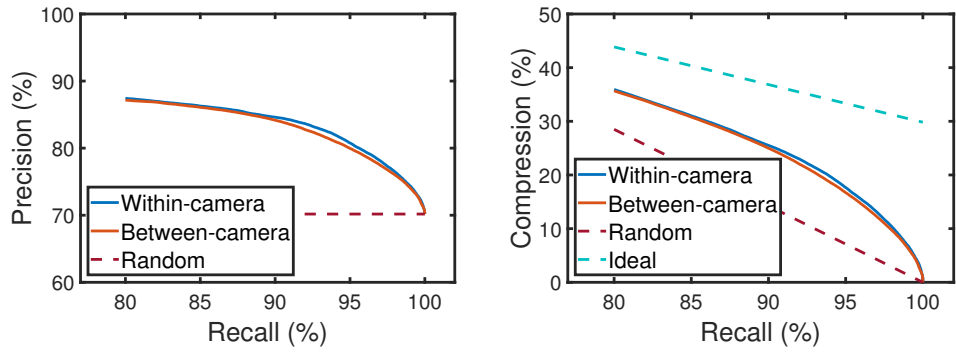
Figure 3.15: Within-camera performance of the 2-state HMM (visual cue only).



(a) Camera 1



(b) Camera 2



(c) Camera 3

Figure 3.16: Between-cameras performance compared to within-camera performance on all three cameras.

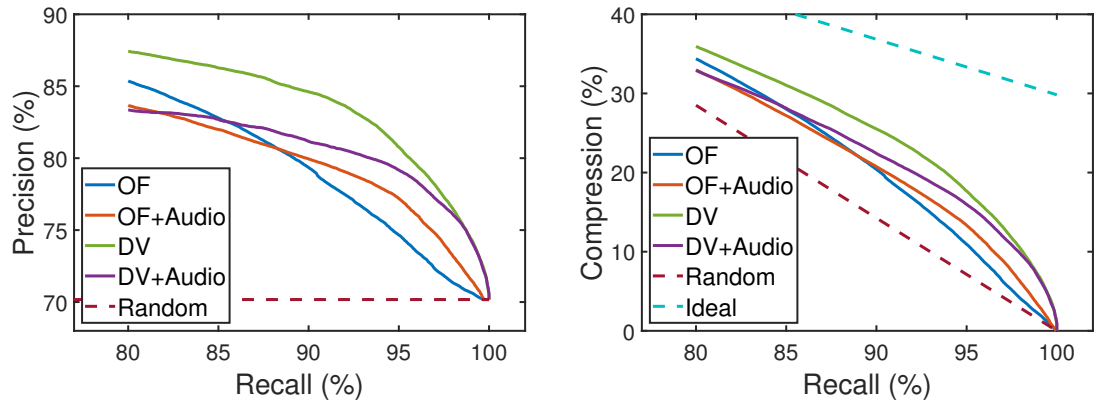
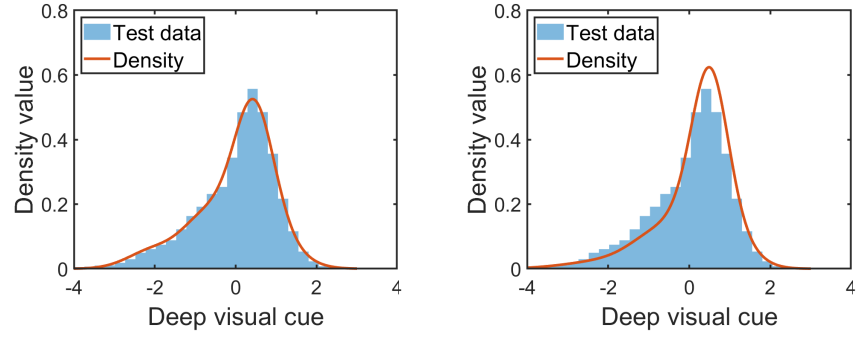
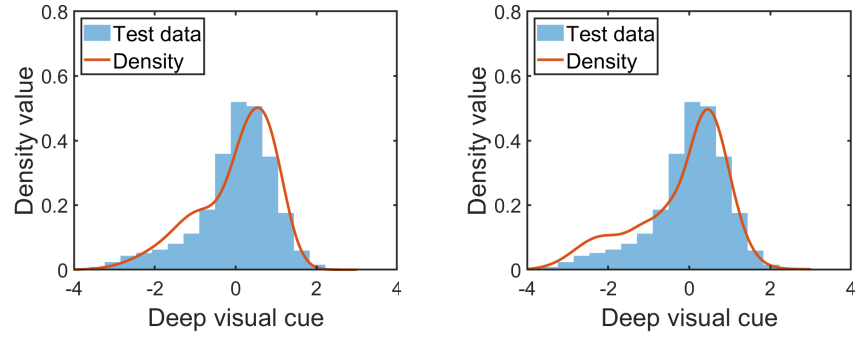


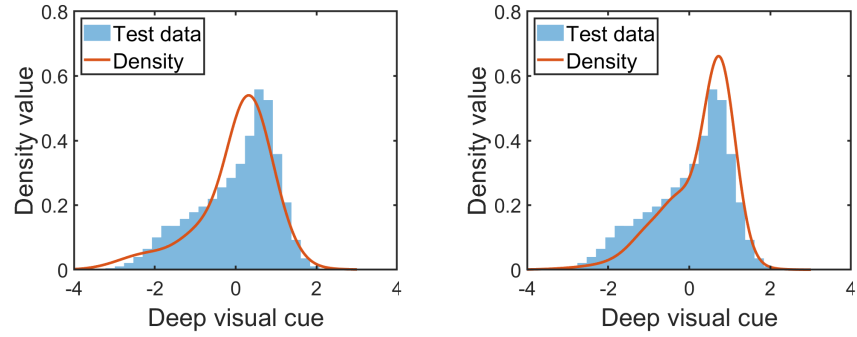
Figure 3.17: Performance of the 2-state HMM and 4-state HMM on Camera 3. The 4-state HMM combined visual and auditory cues, while the 2-state HMM used only the visual cues. OF: maximum optic flow cue; DV: deep visual cue.



(a) Game 1

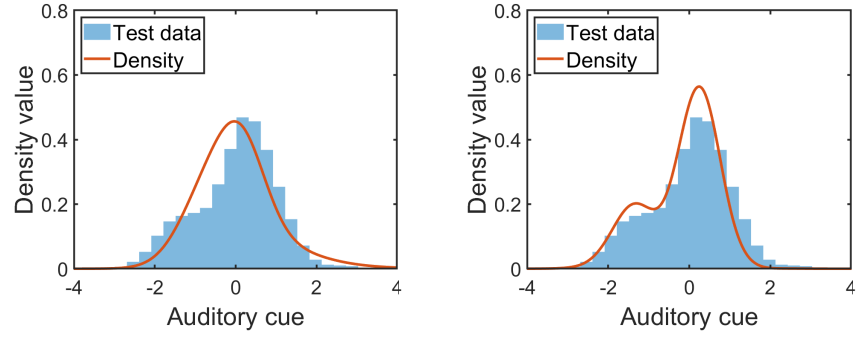


(b) Game 2

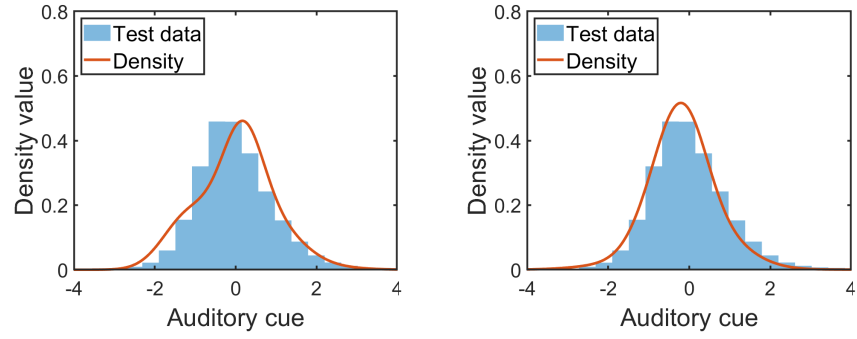


(c) Game 3

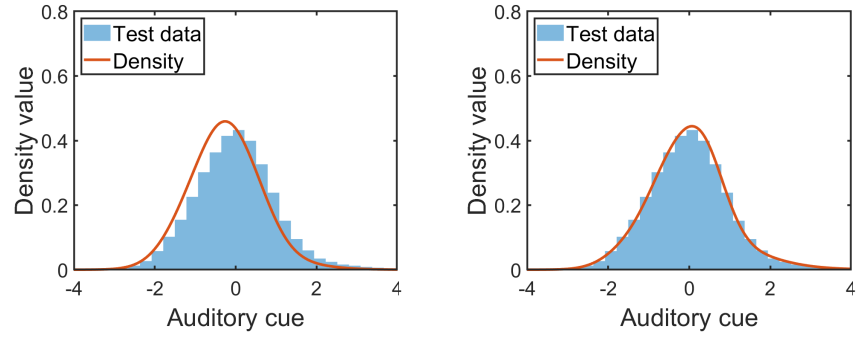
Figure 3.18: Unconditional densities of the deep visual cue learnt from the training data shown on the test data histogram for each game recorded using Camera 3. Left: Before adaptation. Right: After adaptation.



(a) Game 1



(b) Game 2



(c) Game 3

Figure 3.19: Unconditional densities of the auditory cue learnt from the training data shown on the test data histogram for each game recorded using Camera 3. Left: Before adaptation. Right: After adaptation.

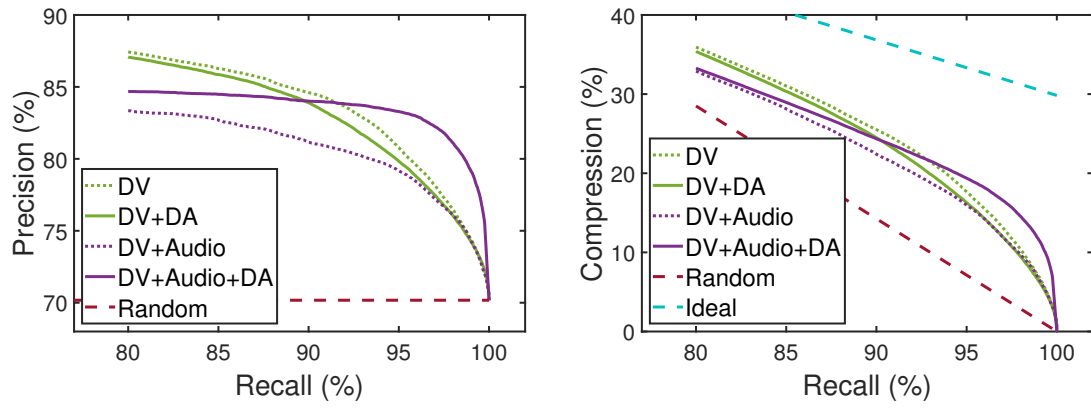
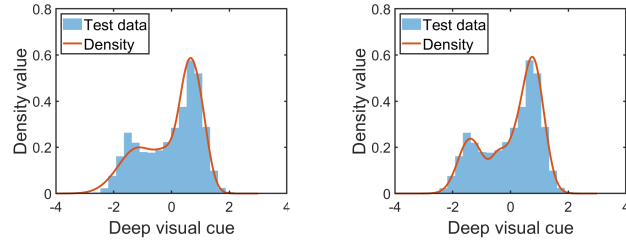
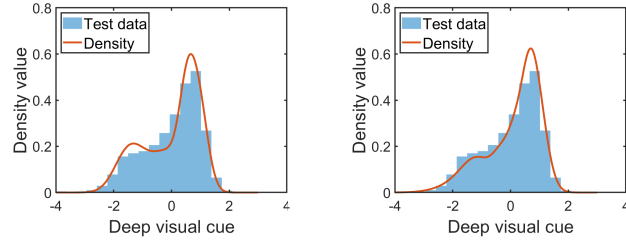


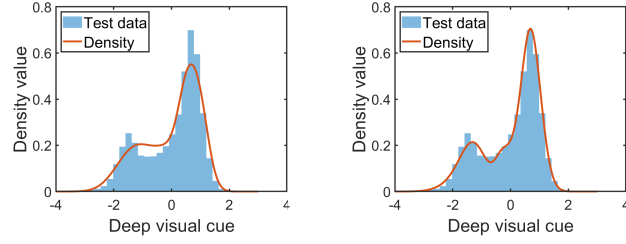
Figure 3.20: Performance of the 2-state HMM and 4-state HMM after domain adaptation on Camera 3. OF: maximum optic flow cue; DV: deep visual cue; DA: domain adaptation.



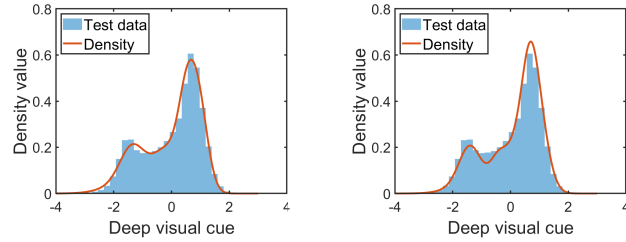
(a) Game 1



(b) Game 2

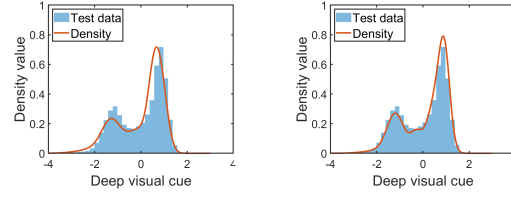


(c) Game 3

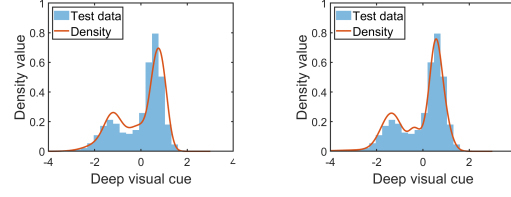


(d) Game 4

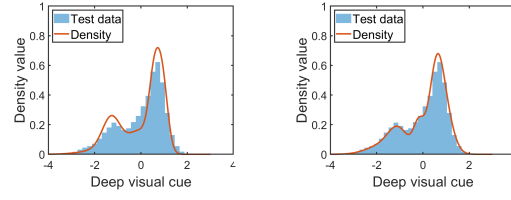
Figure 3.21: Unconditional densities of the deep visual cue learnt from the training data shown on the test data histogram for each game recorded using Camera 1. Left: Before adaptation. Right: After adaptation.



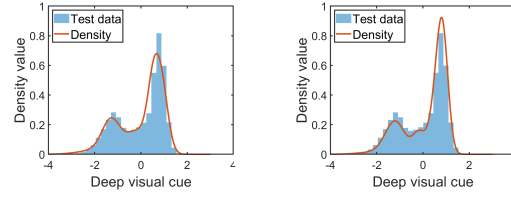
(a) Game 1



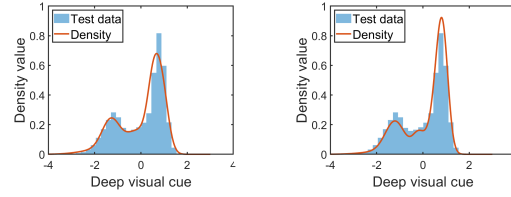
(b) Game 2



(c) Game 3

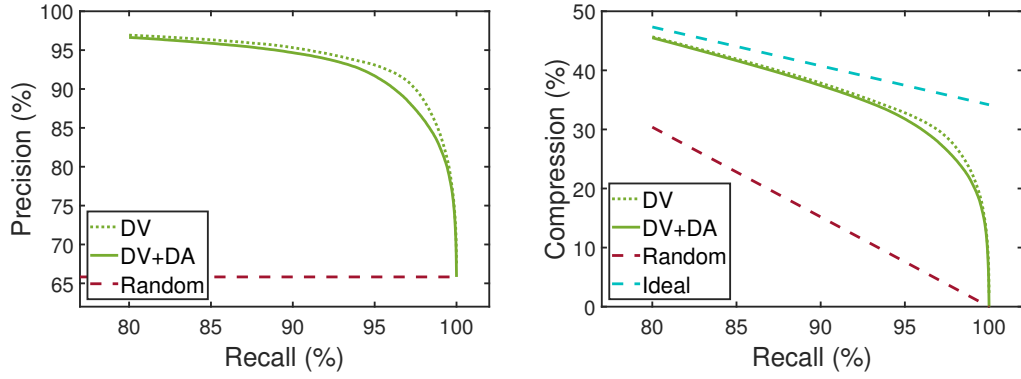


(d) Game 4

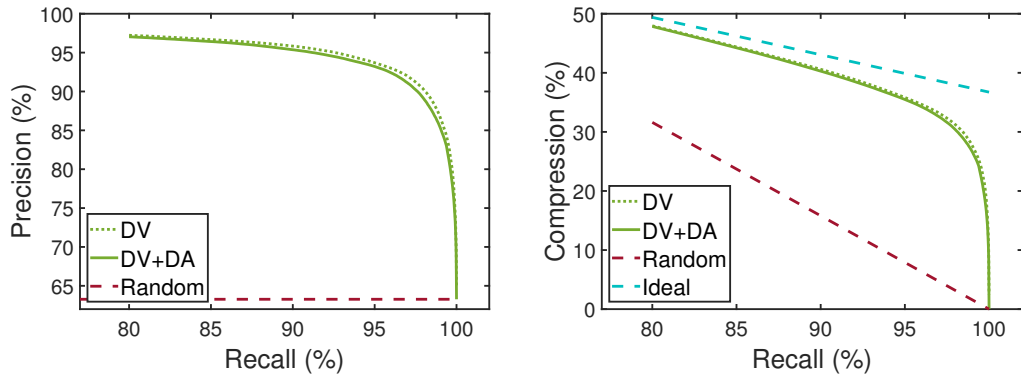


(e) Game 5

Figure 3.22: Unconditional densities of the deep visual cue learnt from the training data shown on the test data histogram for each game recorded using Camera 2. Left: Before adaptation. Right: After adaptation.



(a) Camera 1



(b) Camera 2

Figure 3.23: Performance of the 2-state HMM before and after domain adaptation on all games from Camera 1 and Camera 2. DV: deep visual cue; DA: domain adaptation.

Chapter 4

Automatic Homography Estimation

In Chapter 2 and Chapter 3 we have described the problem of automatically tracking the play to create a curated, zoomed-in video recording, and temporally compressing this recording to include all periods of active play. The next logical step in hockey video analysis is extracting analytical and statistical insights from the recorded game. These allow for coaches and players to make more informed decisions both during and after the game. Additionally, the quality of the broadcast can be greatly improved with analytical insights.

An important step for reliable statistical analysis is homography estimation, which determines the planar transform between camera and ice coordinates (Fig 4.1). Knowing this homography could allow projection of images and statistics to a common virtual camera, thus improving registration across cameras, rinks and games, and therefore improving the reliability of the visual features like those used in Chapter 2 and Chapter 3. Additionally, it could allow the statistics of the game to be expressed in real ice coordinates and projection of graphics onto the ice.

The work discussed in Chapter 2 and Chapter 3 used games recorded from a stationary camera system whose field-of-view was the entire ice rink. Since the camera was stationary, the homography does not change.

However, this is not true with broadcast video generated by professional videographers, who pan, tilt and zoom their camera to record the game. Since the camera is moving continuously, the homography changes almost every frame. This necessitates an algorithm

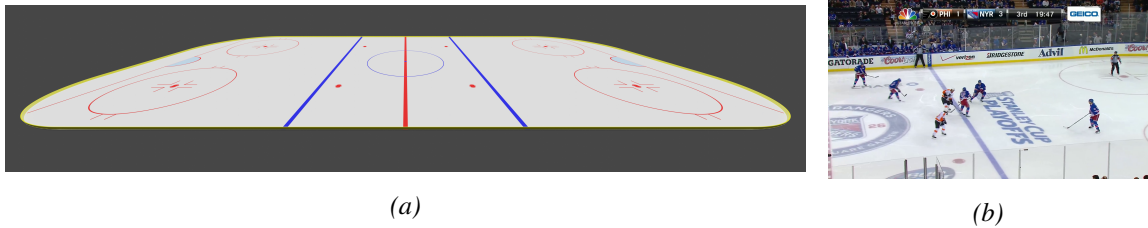


Figure 4.1: (a) Template ice hockey rink and (b) sample test image from a moving camera broadcast hockey video.

to automatically estimate the homography on each frame.

Typically, the homography is estimated in a two-step process. An initial estimate is first computed and subsequently refined to obtain the final estimate. The initial estimate can be obtained by estimating corresponding keypoints between a template image (Fig 4.1a) and test image (Fig 4.1b). Keypoint detectors can be hand-crafted [47, 44, 41] or learnt from labelling training data [22, 80]. Depending upon the number of initial estimates and the required accuracy, refinement may not be necessary [101, 20].

While deep learning-based methods have been shown to work well, they require large amount of training data and detailed annotations. We propose an alternative approach that does not require a large-scale annotated dataset. We exploit the consistency of colours used for markings on the ice surface (Fig 4.1a). We model these colours as multi-variate Gaussian in RGB space. We then search for the homography that aligns the colours of the template to the observed colours in the image. We find that the accuracy of our method is comparable to the other methods trained on large datasets.

4.1 Prior work

Automatically estimating the homography in sports videos, also known as sports field localization, is a well-studied problem in computer vision [111]. The most popular approach for automatic homography estimation is a two-step process. Given a test image, the initial homography is estimated on the first step. The initial estimate is refined in the second step.

Most of the early methods used traditional computer vision algorithms to detect keypoints and line segments in an image for homography estimation. However, most of the

more recent methods used deep networks to segment the playing surface.

4.1.1 Traditional methods

Some of the early approaches detected keypoints and matched the correspondences between frames for homography estimation. Some of the methods for keypoint detection were the Scale-Invariant Feature Transform (SIFT) [72], Speeded Up Robust Features (SURF) [6] and Scale-Invariant Feature Operator [36]. The initial homography was either manually obtained [81] or estimated using the correspondence between the keypoints detected on the test image and pre-selected keyframes [47, 45, 44, 41]. Once the homography was initialized, a KLT tracker [115] was typically used to track the keypoints between frames and estimate the homography between video frames in American football [47], ice hockey [41] and soccer [45, 44]. Gupta *et al.* [41] extended this approach by combining point-based matches with correspondences between the lines and circles on the ice surface.

Instead of using keypoints, others solve for homography estimation through line detection and matching. In [110, 133], the line markings in soccer were detected using the Hough transform and the homography was estimated by minimizing the squared error between the detected lines and the lines in a known field template. Dubrofsky *et al* [31] extended the Direct Linear Transform (DLT) algorithm [43] to use both points and line correspondences for homography estimation in ice hockey. Skinner *et al.* [106] extended this work for homography estimation of rugby games recorded by any spectator with a smartphone in the stands. They used the Hough transform to extract line segments and matched the vertical lines to a pre-defined field template. Once the homography estimate is accurate, each user could be targeted with augmented reality content aligning with the field.

Wen *et al.* [126] detected SURF features to stitch images and generate a panoramic view of a basketball court from a given moving camera video. The dominant colour in the panoramic image was used to extract the field markings. The four corners of the court were used to estimate the homography. Zeng *et al.* [136] used a similar approach for field hockey.

The main drawback of these methods is that most keypoints are detected in the stands or on the players, rather than on the playing area. Detecting keypoints on the playing

area is challenging because the field markings provide little texture, and are often noisy with ads or other graphics. Furthermore, the players often occlude the landmarks for keypoints. Hence, these methods tend to perform only under controlled conditions and do not translate well to changes in player appearance, lighting and playing surface.

4.1.2 Deep learning methods

Instead of handcrafted features, most of the recent approaches have used deep learning-based features and models for extracting the field markings and estimating the homography.

Homayounfar *et al.* [49] detect lines and circles with a VGG16 semantic segmentation network [105] in soccer and ice hockey videos. The vanishing points at the intersection of all parallel lines on the field are used to limit the search space for the homography matrix. The semantic cues are used to learn the potentials of a Markov Random Field and the homography is estimated via branch and bound.

Instead of using the full semantic information of the field some others used only the field markings. Sharma *et al.* [101] and Chen *et al.* [20] extracted the field markings in broadcast soccer videos using the pix2pix network [53]. They created a comprehensive dictionary of 100,000 camera poses from the real and synthetic training data. Sharma *et al.* use nearest neighbour search with HOG features [25] to estimate the camera pose while Chen *et al.* show that deep features obtained through training a Siamese network [30] are more robust, and they refine the initial pose through gradient descent. The authors reported a run time of 0.5 seconds on a CPU. These methods generate a comprehensive dictionary of all camera poses. The queried initial homography is often very similar to the final desired homography, and requires minimal refinement, but they require a large training set representative of all the views in the test dataset. Additionally, large dictionaries significantly increase inference time.

Sha *et al.* [100] combine the two previous approaches for homography estimation in soccer and basketball videos. The field area was extracted using the UNet semantic segmentation network [97] and classified into zones to generate a semantic map. A dictionary of 450 camera poses was created from the training data. A Siamese network [30] was trained to compute the distance between the semantic maps of a training image and each

image in the dictionary, and classify them as being similar and dissimilar. At inference, the Siamese network retrieves the dictionary image closest to a given test image, thus initializing the camera pose (homography). Then, a Spatial Transformer Network (STN) [54] is used to refine the initial homography estimate. The segmentation network, Siamese network and STN are jointly trained in an end-to-end manner. The authors report a very fast inference time of 4 msec on a GPU.

A few methods have trained models to detect the location of control points on an image. Jiang *et al.* [56] proposed a deep network framework that iteratively optimizes the homography for soccer and ice hockey videos. The proposed system consists of two networks. The first network regresses four homography control points on the playing area to estimate the initial homography. A template image of the playing area (field/rink) is warped according to the initial estimate. The second network computes the registration error of the warped image and backpropagates the error to improve the initial homography estimation. This process is repeated iteratively until convergence.

Citraro *et al.* [22] estimated the location of multiple keypoints based on the intersections of the lines on the playing surface in basketball, soccer and volleyball videos. The UNet semantic segmentation network [97] was trained to extract the field markings. The intersection points between various markings were selected as the keypoints. A particle filter was then used to predict keypoint locations in future frames. The authors report an inference time of 8 fps. This method might not work on sports such as ice hockey which have very few lines and a sparse number of keypoints.

A limitation of using pre-selected fixed keypoints is that they are sparse in cases of zoomed-in broadcast video and fail to estimate the homography. Hence, Nie *et al.* [80] select a uniform grid of points on the playing surface, including those on the texture-less parts of the field, and compute dense features for field localization in basketball, American football, ice hockey, soccer and tennis. They train a deep network to predict the keypoints on a given image and compute the dense features. The initial homography is estimated using the DLT algorithm. The initial homography estimate is refined using the keypoints and dense features on the current and previous frames. The authors demonstrate state-of-the-art performance of their algorithm on various sports with a run time between 25-45 fps.

A drawback of the deep learning based methods is the need for a large and diverse

training dataset. Additionally, there is also a need for detailed and extensive annotations of either the field boundaries and zones on the field [49, 100, 101, 21] or keypoints on the playing field [22, 80], both of which are challenging and expensive.

The wide variety of methods described in this section shows that there are a lot of differences between sports and not all methods generalize well to different sports. Most methods trained and evaluated their algorithms on private datasets. Hence, there are no publicly available datasets for ice hockey.

The markings on the rink in ice hockey are of specific colours. We present an alternative approach that models each of the colour markings separately for accurate homography estimation. We show the feasibility of our approach when there is a limited amount of training data. Our proposed system does not require any training. It works on uncalibrated moving camera broadcast videos and the camera parameters do not have to be estimated. We estimate the initial homography and then refine it using gradient descent. We show that our model is able to compensate for large errors in the initial homography estimate.

4.2 Contributions

The contributions of this work are:

1. Creation of a dataset of 29 images with ground truth control points annotated on each image.
2. Design of a method for automatic homography estimation without requiring any training.

4.3 Dataset

None of the ice hockey datasets used in prior works are publicly available. Hence we have created and labelled a dataset consisting of 29 images from a 5-minute NHL broadcast video obtained from Evertz Microsystems. The video resolution was $1,920 \times 1,080$ pixels and it was recorded using a single PTZ camera placed in the stands, along the center-line. 16 key-frames representative of the range of homographies encountered in broadcast

videos were selected from the first 2 minutes of the video. A few example images are shown in Fig 4.2. 13 frames selected from the last one minute of the video were used as the test dataset.



Figure 4.2: Example key-frames from the dataset.

4.4 Method

The task in automatic homography estimation is to estimate the homography between two planar surfaces, in this case, the moving camera broadcast video frame (Fig 4.1b) and the ice surface (Fig 4.1a).

Let $p_i = [u \ v \ 1]^T$ and $p_r = [x \ y \ 1]^T$ be corresponding points on the broadcast image and the rink, respectively. The homography matrix H for projection of the ice surface to the image is given as

$$p_r = H * p_i \quad (4.1)$$

$$s * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (4.2)$$

where s is the scale factor. The homography matrix H has 8 degrees of freedom as h_{33} can be set to 1.

To estimate the homography between the rink and the camera, we propose to exploit the spatial structure of the markings on the ice rink, which are known in advance (Fig 4.1a) and are the same across rinks¹. There may be additional markings on the actual ice rink

¹There are two different rink geometries: NHL size and Olympic size. The game used in this chapter was recorded at an NHL size rink, while the games used in Chapter 2 and Chapter 3 were recorded at an Olympic size rink.

(Fig 4.2), but the standard markings are common to all ice rinks and the colours of these markings are always the same (Fig 4.1a).

We estimate the homography by aligning the coloured markings detected in the test image with the projected markings of the template image. The projections are learnt on the key-frames by manually locating control points. The coloured markings are detected on the test image and the homography is initialized to the key-frame that maximizes the alignment of the coloured markings. Finally, the alignment is refined using gradient descent. Each step is detailed in the subsequent sections.

4.4.1 Manual rectification

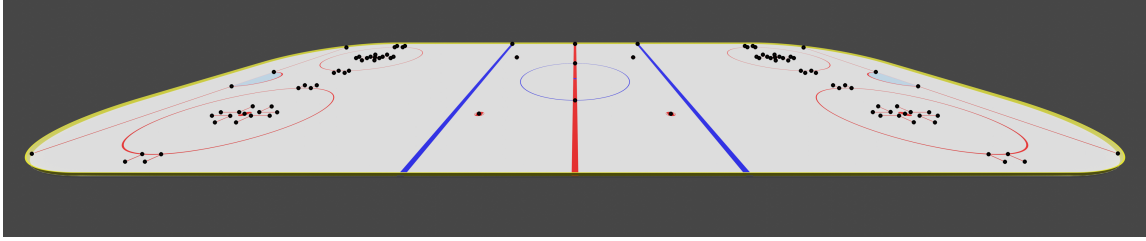
We manually annotated 102 control points on the ice surface in the template image (Fig 4.3a). Since we have a mathematical model of the ice rink, in the future, we can automatically generate the control points on the template image. We also manually annotated all corresponding control points in the field-of-view of each of the 16 key-frames (Fig 4.3b). We estimated homographies for each of these 16 key-frames using OpenCV's standard least squares reprojection minimization method, *findHomography* (Fig 4.3c).

The accuracy of the homography was computed as the average reprojection error of the control points on the ice surface. The average reprojection error across all images was 0.05 m.

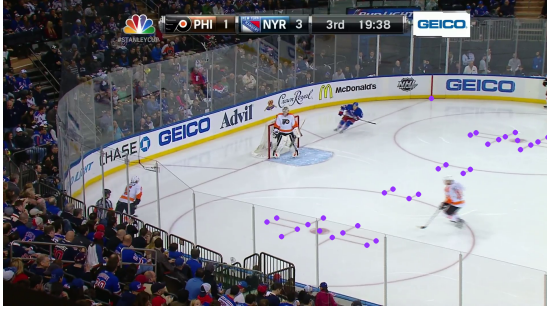
4.4.2 Colour modelling

There are four standard colours on the ice surface: red, blue, yellow and white/grey (ice) (Fig 4.1a). The colour markings on the template ice rink is shown in Fig 4.4.

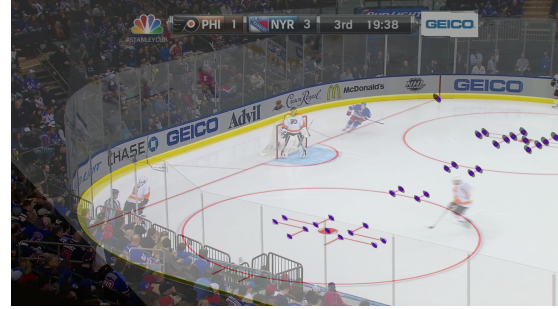
To estimate the markings on the images from the broadcast video, we modelled the pixel intensities of each marking as multi-variate Gaussian in RGB space, learning the parameters from crops of the markings on the key-frames. A few sample crops are shown in Fig 4.5. At each pixel, we estimated the likelihood of each colour marking. Fig 4.6 shows the colour marking with the highest likelihood at each pixel of a broadcast image.



(a)

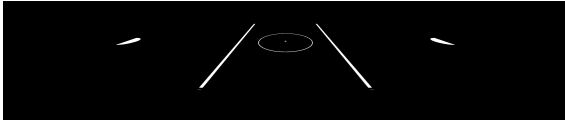


(b)



(c)

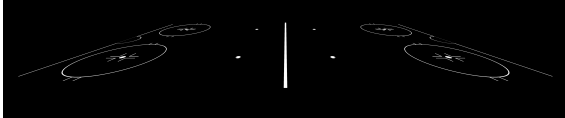
Figure 4.3: (a) Control points (black) on the template image. (b) Control points (purple) on the key-frame. (c) Projected template superimposed on the key-frame.



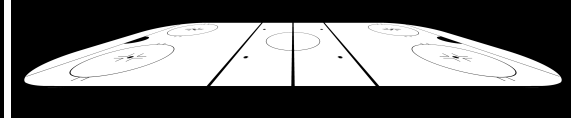
(a) Blue



(c) Yellow



(b) Red



(d) White

Figure 4.4: Colour markings on the template ice rink.

4.4.3 Post-processing techniques

Player jerseys often have similar intensities as the markings, as do the ads on the sideboards. These add to the noise in the estimated likelihoods. For example, in Fig 4.6 (right),

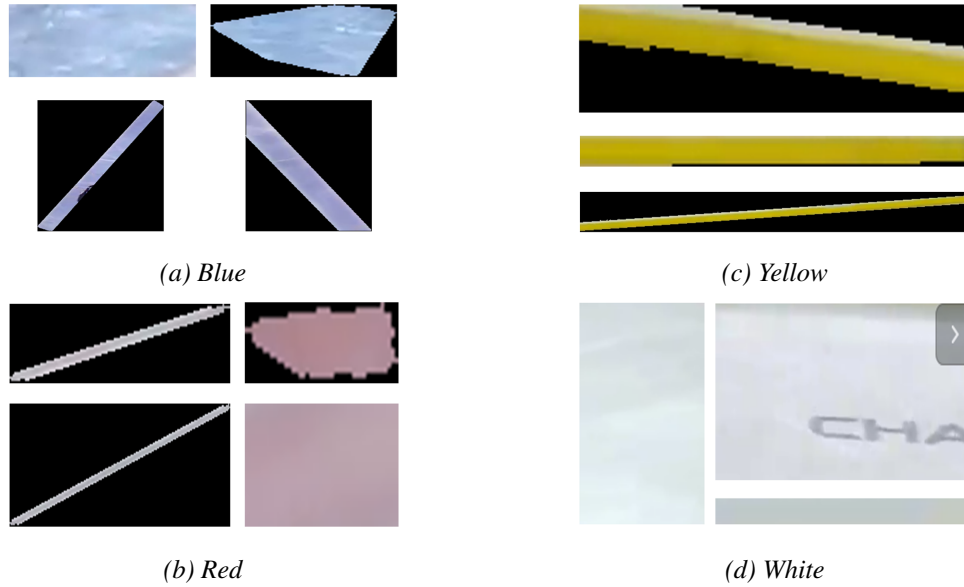


Figure 4.5: Image crops used for modelling the likelihood for each colour marking.

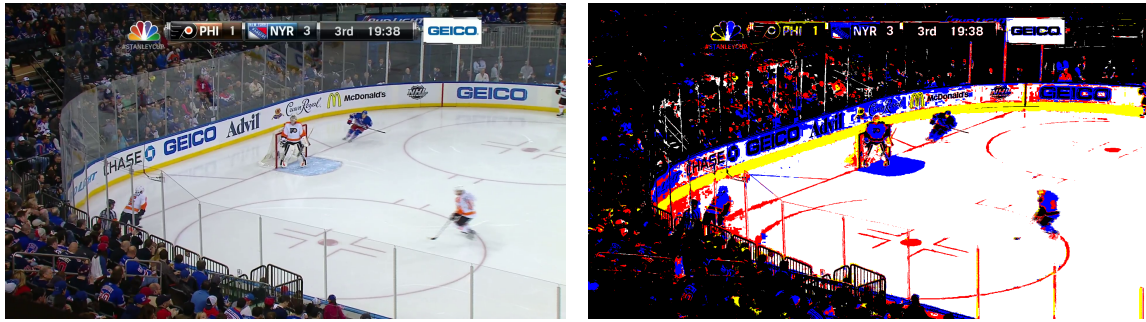


Figure 4.6: Left: Broadcast image. Right: Colour markings with the highest likelihood at each pixel. For visualization purposes, pixels with all colour likelihoods below $1e-16$ are labelled as black.

we see that the blue player jerseys and blue colour ads on the sideboards are detected as blue ice markings. We explored two post-processing steps to reduce the noise:

1. We detected the players using a Faster RCNN network [94] pre-trained on the MS-COCO dataset [68]. Example predictions from the detector are seen in Fig 4.7 (left panels). We set a very low threshold as our aim is to detect all the players on the ice

surface. We generated a binary mask of the player positions (Fig 4.7 (right panels)). We multiplied the estimated likelihoods with the inverse binary player mask so that positive pixels in the binary player mask are set to zero, as indicated by black pixels in (Fig 4.8c).

2. In addition to noise due to players, we found small outlier patches of false detections on the ads on the sideboards. To remove these, we applied median filtering with an empirically determined square kernel size of 7. The results are shown in Fig 4.8d.

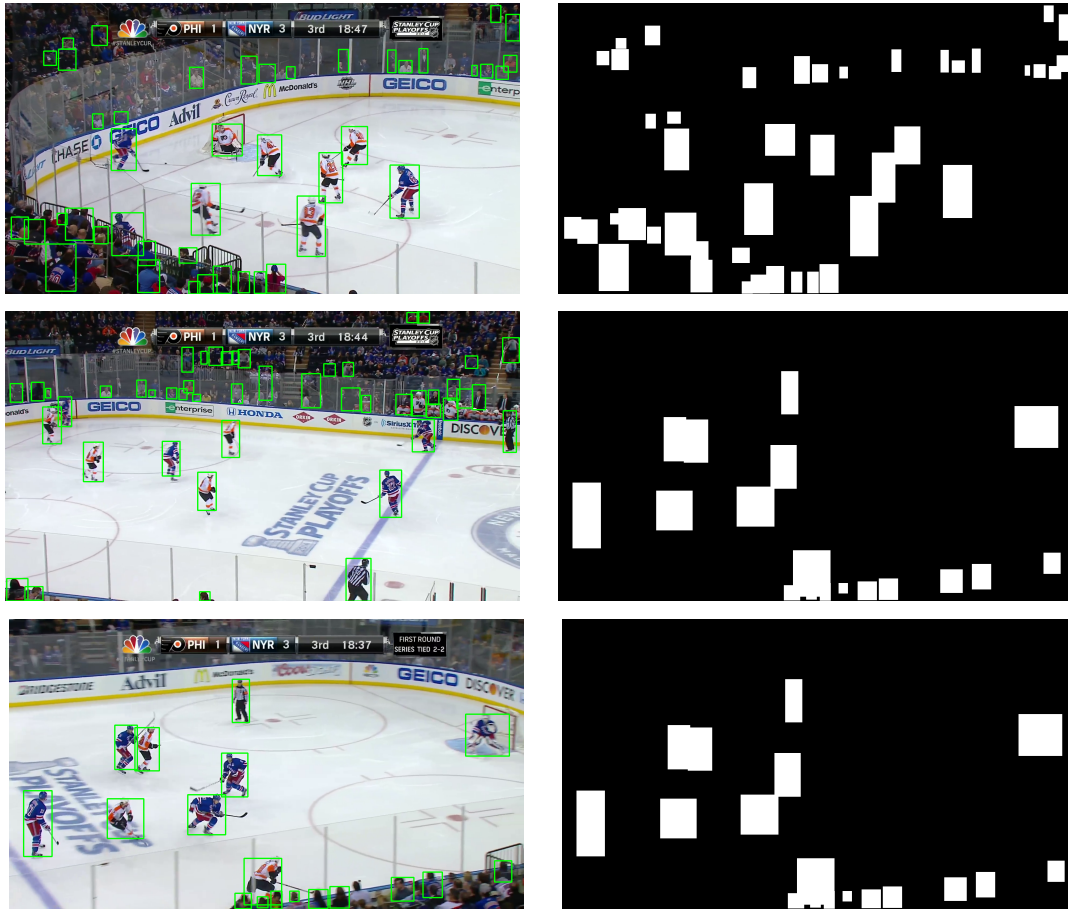


Figure 4.7: Faster RCNN player detections (left panels) and corresponding binary player masks (right panels).

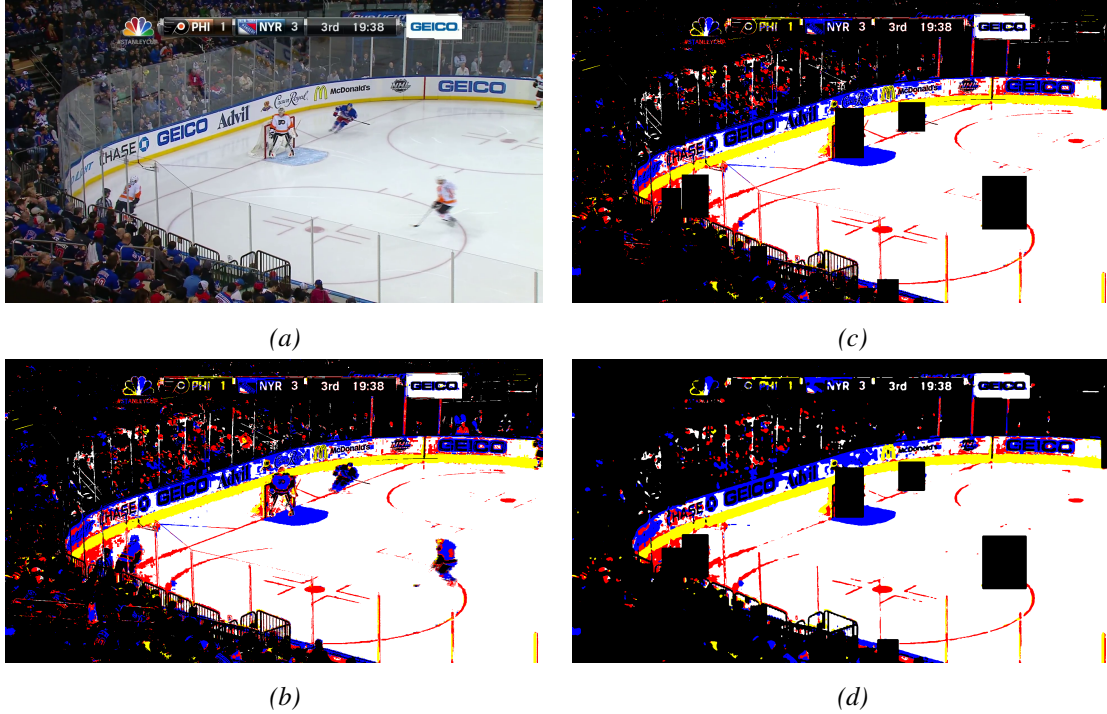


Figure 4.8: Colour marking with the highest likelihood at each pixel of a broadcast image after applying post-processing techniques on the likelihoods of each colour marking. (a) Broadcast image. (b) Colour marking with highest likelihood at each pixel. (c) Image (b) masked by inverse binary player mask. (d) Median filtering applied on image (c). Black pixels indicate regions where all colour likelihoods are less than $1e-16$.

4.4.4 Initial homography estimation

Given a test image, we first estimate the likelihood maps for each of the four colours and then apply the two post-processing steps. The objective is then to find the key-frame maximizing alignment with the test image. To do this, we could simply project the template to the image according to the key-frame homography and then simply sum the appropriate colour likelihoods of the pixels within each projected template region. However, this approach only works for small misalignments. For larger misalignments, there may be little overlap between corresponding regions of the image and the projected template.

To solve this problem, instead of projecting the template colour regions, we project

signed distance transforms [98] for each of the colour regions. This allows us to express the misalignment at each pixel as the average distance to each colour region in the template, weighted by the colour likelihood. This has the advantage that even for large misalignments, the gradient of the objective function is informative. The signed distance transform for each colour of the template mask is shown in Fig 4.9. Distances within the regions of interest (markings) are negative and those outside are positive.

The key-frame k maximizing alignment with the test image is found by minimizing

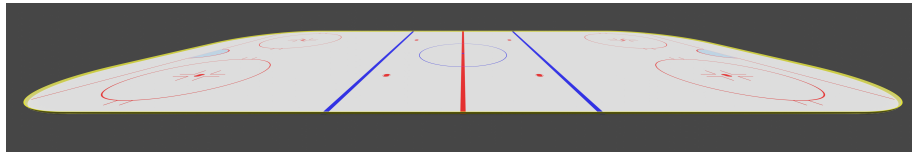
$$\sum_c \sum_x \sum_y \rho_c L_{kc}[x, y] \tilde{D}(T_{kc})[x, y] \quad (4.3)$$

where L_{kc} is the likelihood and $\tilde{D}(T_{kc})$ is the signed distance transform for the projected template mask for colour c of the k^{th} key-frame. $\rho_c = \frac{1}{P(c)}$ is the inverse prior for each colour c . The prior is computed as the proportion of pixels corresponding to each colour marking across all training images.

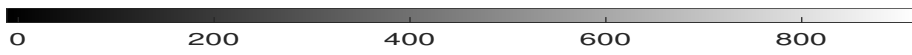
The objective function will decline as pixels in the image approach template colour regions for which they have high likelihood. For each test image, the homography was initialized to the homography of the key-frame minimizing this objective. We evaluated the effect of five different design parameters:

1. P1 - Toggles whether to use 3 (R,Y,B) or 4 (R,Y,B,W) colours.
2. P2 - Ignore player regions
3. P3 - Median filtering of the colour likelihoods L_c
4. P4 - Normalize likelihoods L_c to sum to 1 over pixels. This will serve to up-weight colours that are rarer in a given image.
5. P5 - Weight the objective function by the inverse prior ρ_c . Since the pixels corresponding to non-white colour markings are sparse, this up-weights the influence of non-white colour markings and down-weights the influence of white markings.

There are 5 design parameters resulting in 32 different settings. The five steps are applied sequentially. We evaluated the accuracy of the initial homography estimate by



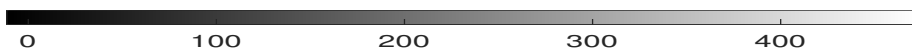
(a) *Template image*



(b) *Blue*



(c) *Red*



(d) *Yellow*



(e) *White*

Figure 4.9: Signed distance transform of the template image for each colour marking.

Exp #	P1	P2	P3	P4	P5	Reprojection error (m)
1	✗	✗	✗	✗	✗	5.69 ± 1.88
2	✗	✗	✗	✗	✓	6.35 ± 1.91
3	✗	✗	✗	✓	✗	6.34 ± 1.87
4	✗	✗	✗	✓	✓	8.18 ± 2.54
5	✗	✗	✓	✗	✗	2.83 ± 0.29
6	✗	✗	✓	✗	✓	2.75 ± 0.29
7	✗	✗	✓	✓	✗	5.32 ± 1.67
8	✗	✗	✓	✓	✓	4.99 ± 1.70
9	✗	✓	✗	✗	✗	2.83 ± 0.29
10	✗	✓	✗	✗	✓	2.75 ± 0.29
11	✗	✓	✗	✓	✗	5.32 ± 1.67
12	✗	✓	✗	✓	✓	4.44 ± 1.60
13	✗	✓	✓	✗	✗	2.83 ± 0.29
14	✗	✓	✓	✗	✓	2.75 ± 0.29
15	✗	✓	✓	✓	✗	3.55 ± 0.73
16	✗	✓	✓	✓	✓	2.68 ± 0.24
17	✓	✗	✗	✗	✗	14.57 ± 3.40
18	✓	✗	✗	✗	✓	12.37 ± 3.48
19	✓	✗	✗	✓	✗	5.44 ± 1.84
20	✓	✗	✗	✓	✓	8.18 ± 2.54
21	✓	✗	✓	✗	✗	14.57 ± 3.40
22	✓	✗	✓	✗	✓	12.37 ± 3.48
23	✓	✗	✓	✓	✗	5.51 ± 1.89
24	✓	✗	✓	✓	✓	4.99 ± 1.70
25	✓	✓	✗	✗	✗	14.57 ± 3.40
26	✓	✓	✗	✗	✓	11.52 ± 3.49
27	✓	✓	✗	✓	✗	2.69 ± 0.24
28	✓	✓	✗	✓	✓	4.44 ± 1.60
29	✓	✓	✓	✗	✗	14.57 ± 3.40
30	✓	✓	✓	✗	✓	11.52 ± 3.49
31	✓	✓	✓	✓	✗	2.69 ± 0.24
32	✓	✓	✓	✓	✓	2.68 ± 0.24

Table 4.1: Reprojection error of the initial homography estimates for each possible combination of design parameters P1-P5.

computing the average reprojection error of the control points on the ice surface. The mean and standard error across all test images are reported in Table 4.1.

There are a few important observations from the results. While using three colour classes (RBY) vs four colour classes (RBYW) yields similar results when other choices are optimized, overall, it seems that using only three classes yields more stable results as other choices are varied. While ignoring player regions (P2) or performing median filtering (P3) both lower the error, these steps seem to be largely redundant: doing both leads to only a modest reduction in error compared to doing only one of the two. This could be because players generate many of the higher likelihoods for our colour classes that bias our objective function. Normalizing the likelihood (P4) and weighting by the inverse prior (P5) lead to only a modest reduction in error.

The lowest reprojection error of 2.68 ± 0.24 m is obtained for the parameter settings in experiments 16 and 32. Since using only three colour classes seems to yield more stable results when other parameters are varied, we used the parameter setting of experiment 16 for all further experiments. Some example initializations on the test images are shown in Fig 4.10.

4.4.5 Final homography estimation

In the second-stage of the computation, the initial homography estimate is iteratively refined to further reduce the objective function (Eq 4.3) using MATLAB's *fminsearch* function until convergence, to obtain the final homography estimate. Empirically, we observed that stopping after 100 iterations produced similar results to running gradient descent until convergence. Hence, for all experiments in this chapter, we show the results after 100 iterations.

We again varied the five design parameters (P1 - P5) and analysed the effect on the reprojection errors of the final homography estimate. The mean and standard error across all test images is reported in Table 4.2.

Again, we find that using only three colour classes (RBY) yields the most stable results. Applying player masks (P2) and median filtering (P3) and normalizing likelihoods (P4) all seem to modestly improve performance, while applying the inverse prior does not. Our best result is with System 15, which yields a final average reprojection error of 48cm.

Exp #	P1	P2	P3	P4	P5	Reprojection error (m)
1	✗	✗	✗	✗	✗	1.52 ± 0.35
2	✗	✗	✗	✗	✓	1.69 ± 0.37
3	✗	✗	✗	✓	✗	0.73 ± 0.11
4	✗	✗	✗	✓	✓	0.84 ± 0.13
5	✗	✗	✓	✗	✗	0.99 ± 0.25
6	✗	✗	✓	✗	✓	1.52 ± 0.38
7	✗	✗	✓	✓	✗	0.63 ± 0.09
8	✗	✗	✓	✓	✓	0.67 ± 0.10
9	✗	✓	✗	✗	✗	0.86 ± 0.25
10	✗	✓	✗	✗	✓	0.96 ± 0.26
11	✗	✓	✗	✓	✗	0.60 ± 0.12
12	✗	✓	✗	✓	✓	0.56 ± 0.10
13	✗	✓	✓	✗	✗	0.71 ± 0.16
14	✗	✓	✓	✗	✓	1.00 ± 0.30
15	✗	✓	✓	✓	✗	0.48 ± 0.09
16	✗	✓	✓	✓	✓	0.54 ± 0.10
17	✓	✗	✗	✗	✗	13.89 ± 0.70
18	✓	✗	✗	✗	✓	9.11 ± 0.78
19	✓	✗	✗	✓	✗	0.81 ± 0.22
20	✓	✗	✗	✓	✓	0.80 ± 0.11
21	✓	✗	✓	✗	✗	14.25 ± 0.43
22	✓	✗	✓	✗	✓	9.17 ± 0.99
23	✓	✗	✓	✓	✗	0.64 ± 0.13
24	✓	✗	✓	✓	✓	0.80 ± 0.14
25	✓	✓	✗	✗	✗	13.81 ± 0.62
26	✓	✓	✗	✗	✓	9.33 ± 0.99
27	✓	✓	✗	✓	✗	0.86 ± 0.20
28	✓	✓	✗	✓	✓	0.60 ± 0.09
29	✓	✓	✓	✗	✗	14.09 ± 0.47
30	✓	✓	✓	✗	✓	9.38 ± 0.87
31	✓	✓	✓	✓	✗	0.75 ± 0.21
32	✓	✓	✓	✓	✓	0.52 ± 0.09

Table 4.2: Reprojection error of the final homography estimates for each possible combination of design parameters P1-P5.

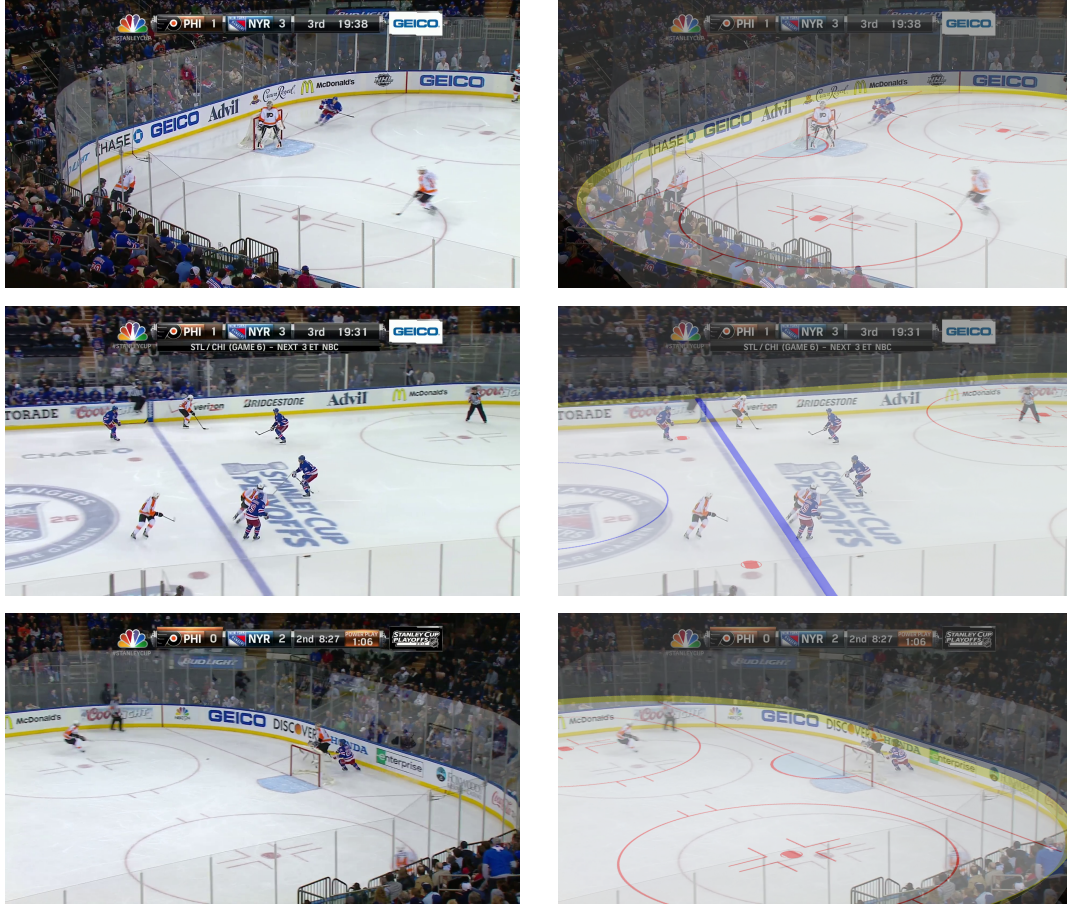


Figure 4.10: Left: Test images. Right: Projected template mask superimposed on test image based on the initial homography estimate.

Using only three colour classes (RBY) rather than all four colour classes (RBYW) yielded more stable results for both the initial and final homography estimates. This could be because of the noise induced into the likelihood estimates for the white colour markings by the graphics and ads on the ice surface.

Most of the prior works measure the performance of their system in terms of the Intersection Over Union (IOU) between the ice surface planes of the projected template image and the test image (IOU_{part}). The average IOU of our system was 0.967, which is comparable to the performance of most of the prior works on their private ice hockey

datasets [20, 56, 101].

Some example results of the final homography estimations on the test images are shown in Fig 4.11. We found that despite large errors in the initial homography estimates, the final homography estimate was very accurate.

4.5 Run time analysis

In the experiments described in Section 4.4.4 and Section 4.4.5, the test image and template image were processed at their native resolution of $1,920 \times 1,080$ pixels and $7,680 \times 4,320$ pixels, respectively. The rectification of the template image to the test image is computed at every iteration. As a result, the run time is very high at native resolution. We analysed the trade-off between run time and reprojection error for various resolutions of the test image and the template image, as seen in Table 4.3.

Test image resolution	Template image resolution	Initial homography estimation		Final homography estimation		Total processing time per image(sec)
		Time per image (sec)	Reprojection error (m)	Time per image (sec)	Reprojection error (m)	
1920×1080	7680×3240	0.98	2.68 ± 0.24	11.49	0.48 ± 0.09	12.47
1920×1080	3840×2160	0.97	2.68 ± 0.24	10.03	0.43 ± 0.05	11.00
1920×1080	1920×1080	0.97	2.68 ± 0.24	10.39	0.44 ± 0.06	11.36
960×540	3840×2160	0.25	2.68 ± 0.25	2.61	0.41 ± 0.05	2.86
960×540	1920×1080	0.25	2.68 ± 0.24	2.60	0.49 ± 0.08	2.85
960×540	960×540	0.25	2.68 ± 0.25	2.61	0.54 ± 0.08	2.86
480×270	1920×1080	0.07	2.68 ± 0.24	0.72	0.51 ± 0.09	0.79
480×270	960×540	0.07	2.68 ± 0.25	0.73	0.48 ± 0.05	0.80
480×270	480×270	0.07	2.68 ± 0.25	0.72	0.74 ± 0.11	0.79
240×135	960×540	0.02	2.94 ± 0.25	0.20	0.67 ± 1.00	0.22
240×135	480×270	0.02	2.94 ± 0.32	0.21	0.79 ± 0.14	0.23
240×135	240×135	0.02	21.33 ± 4.86	0.18	22.97 ± 5.29	0.20

Table 4.3: Run time analysis of the system at various resolutions of the test image and template image.

For these experiments, the parameter setting for the initial homography was that of experiment 16 in Table 4.1 and the parameter setting for the final homography estimation

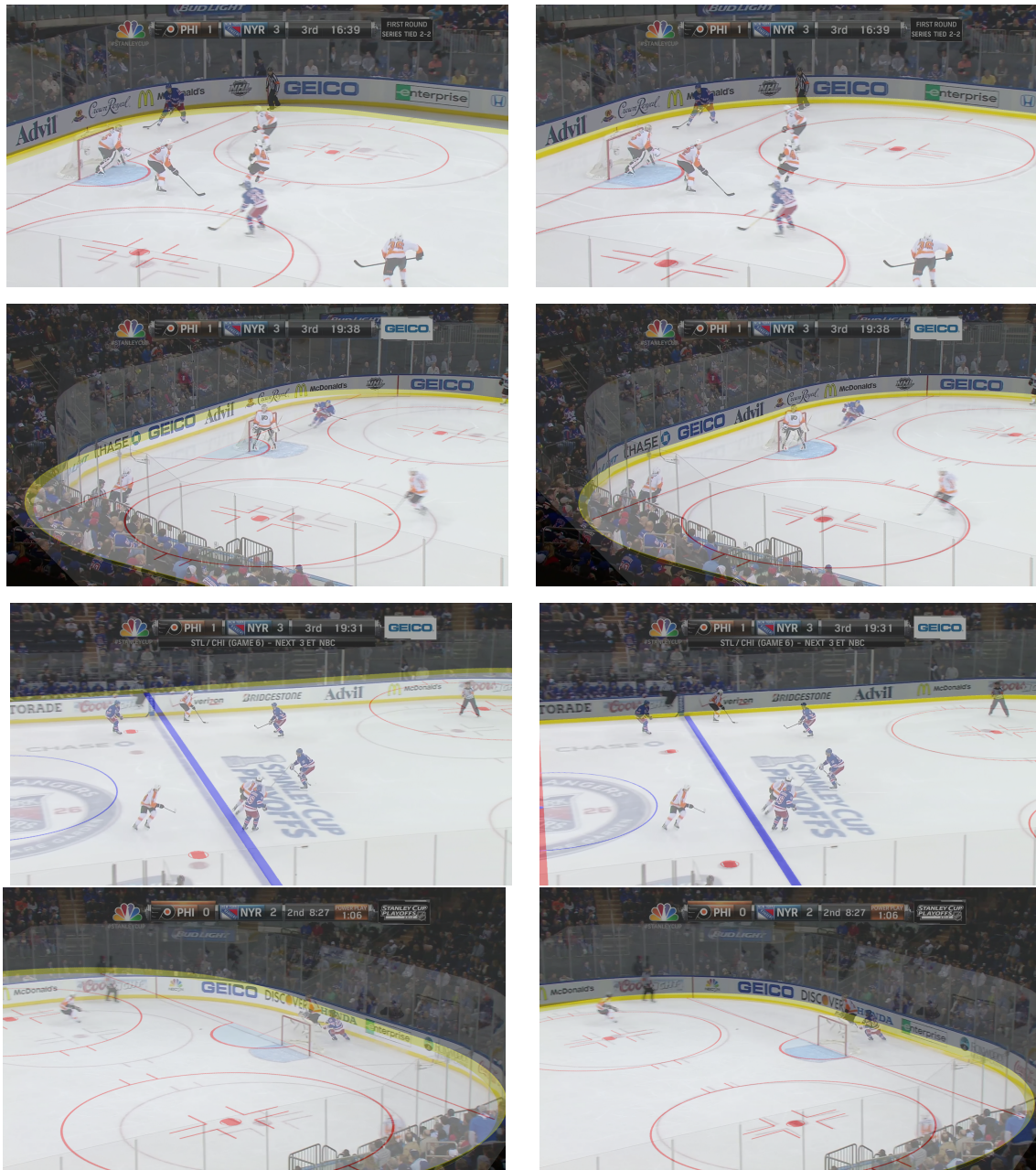


Figure 4.11: Left: Initial homography estimate. Right: Final homography estimate.

was that of experiment 15 in Table 4.2. Median filtering and player detection were performed at the native resolutions of the test image and template image, and then the images were appropriately resized. Median filtering was applied with a square kernel of size 7.

We found that only downsampling the test image reduces the computation time. This is reasonable because projection of the template to the image is performed for each pixel of the image, and this is independent of the number of pixels in the template. The time for the initial homography estimate can be ignored for a continuous video, for which the homography on the current frame would be initialized to the final estimated homography on the previous frame. We found the lowest error at a test image resolution of 960×540 and template image resolution of 3840×2160 . However, for these resolutions the computation time for the final homography estimate is 2.61 seconds per image. Thus, in my opinion, the best configuration is at a test image resolution of 480×270 and template image resolution of 960×540 , where the reprojection error for the final estimated homography is 0.48 ± 0.05 m. This is not statistically different from the lowest error attained at higher resolution (0.41 m). The computation time for the final homography estimate at this resolution is only 0.73 seconds per image.

4.6 Conclusions & future work

I have designed and evaluated a novel method for automatically estimating homographies on video frames from uncalibrated moving camera broadcast hockey video. We modelled each of the four standard colours used in the ice markings using a multi-variate Gaussian in RGB space, learning the parameters from a few pre-selected key-frames. We employed a novel objective function that used a distance transform to smooth the objective, allowing convergence of an iterative solver to the correct solution despite substantial error in the initial guess.

Our system requires minimal ground truthing. The accuracy of our system on our dataset is comparable to that of other methods on their private datasets. We have shown the feasibility of our approach for estimating the homography with limited training data.

Some ideas that could be explored in the future are:

1. Create and evaluate the method on a larger dataset consisting of multiple games,

recorded at different venues, and consisting of many diverse camera viewpoints.

2. Develop an online variant for continuous video that uses the homography estimate on the previous frame as an initial guess.
3. Improving the run time for the final homography estimation. This can be done by limiting transfer of the template distance transform to only those pixels with high likelihoods for one of the three colour classes (RBY).

Chapter 5

Conclusions and Future Work

In this thesis, I have discussed my work on exploring computer vision methods for hockey video curation. The focus of my PhD has been to build a practical system for hockey video curation. Due to the complex and dynamic nature of sports, automatically extracting information such as the location of the puck or state of play, from hockey videos is very challenging. Due to the small size of the puck and motion blur, it often appears as a grey streak. It is frequently occluded by players or equipment such as sticks. Further, player poses vary drastically and they are often occluded by the sideboards or other players. The pose and location of the camera, and the lighting conditions can vary drastically across different games.

In the first part of my thesis, I discussed a method for accurately tracking the puck. We proposed a novel method that enabled accurate ground truthing of the puck location in wide-field videos. Using this ground truth data, we trained a novel deep neural network regressor to estimate the puck location on each video frame. We have presented extensive empirical results to demonstrate the effectiveness of the puck tracking algorithm. Using the puck tracking algorithm, we developed a system to automatically record the game. We achieved this by dynamically cropping the wide-field video centered around the estimated puck location, thus allowing the system to automatically follow the play. This automatic sports videography system has been successfully deployed to record amateur hockey games at two Canadian ice rinks.

Next, I discussed a novel algorithm to segment the game video into video sequences

of play and no-play. We combined the deep visual cues obtained from the output of a trained deep neural network classifier with optional auditory cues obtained from the referee's whistle using a Wiener filter, in a hidden Markov model (HMM) framework, to classify each frame as belonging to play or no-play. This classification is used to temporally compress the game video by removing no-play frames. We showed the need for unsupervised domain adaptation to derive maximum benefit from the auditory cues.

Finally, I showed the feasibility of an automatic homography estimation algorithm that exploited the structure of markings normally found in an ice hockey rink. Specifically, we used the known colour information of different markings. We modelled the colours as a multi-variate Gaussian in the RGB colour space and used a two-step approach to search for the homography that aligned the colours of the template image with the test image. This allows the system to operate across changes in camera parameters and also allows for projection of game-related graphics, ads and notices.

There are some natural directions in which the work discussed in this thesis may be extended. The puck tracking algorithm can be utilized for various analyses of the game. For example, the predicted puck locations can be combined with player detection and/or player tracking data to automatically generate game statistics such as puck possession, zone entry, zone exit, etc. Our automatic play segmentation algorithm could benefit from an improved whistle detector that is more robust to background noise. Another interesting direction to extend this work would be to fuse the audio and visual inputs into a deep network trained for play classification. Additionally, transfer learning can be explored for extending both the puck tracking system and play segmentation algorithm to different sports. One can explore accuracy and run time trade-offs for the automatic homography estimation algorithm. The run time for the final homography estimate can be improved by limiting the transfer of the template distance transform to only those pixels with high likelihoods for one of the three colour classes (RBY). In the future, an online variant of the automatic homography estimation algorithm for continuous video can be explored.

While this thesis is focused on hockey video analysis, the problems and approaches explored in the thesis are more generic and they can be applied to different sports. In any team sport where the ball is fast moving such as basketball and soccer, the approaches discussed in this thesis can be used to accurately ground truth the ball location on video sequences and train a model to estimate the ball location on each frame. Sports such as

basketball and American football have many stoppages during the play. Thus, the play segmentation algorithm can be extended to those sports to obtain a temporally compressed video of play sequences.

Computer vision is being increasingly applied to provide many sports video analyses. Using ball location and player tracking data, many in-game statistics could be generated automatically. In the future, computer vision techniques can be used to automatically record and summarize games. The computer vision and AI algorithms could be utilized to deliver curated statistics to individual players and teams, giving possible advantages over competitors. These automatically generated analytical insights about the game could also increase fan engagement in sports.

Bibliography

- [1] N. Ancona, G. Cicirelli, E. Stella, and A. Distanto. Ball detection in static images with support vector machines for classification. *Image and Vision Computing*, 21(8):675–692, 2003.
- [2] A. Arbués-Sangüesa, C. Ballester, and G. Haro. Single-camera basketball tracker through pose and semantic feature fusion. *arXiv preprint arXiv:1906.02042*, 2019.
- [3] Y. Ariki, S. Kubota, and M. Kumano. Automatic production system of soccer sports video by digital camera work based on situation recognition. In *IEEE International Symposium on Multimedia*, pages 851–860, 2006.
- [4] L. Ballan, M. Bertini, A. Del Bimbo, and G. Serra. Action categorization in soccer videos using string kernels. In *International Workshop on Content-Based Multimedia Indexing*, pages 13–18. IEEE, 2009.
- [5] L. E. Baum et al. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3(1):1–8, 1972.
- [6] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer, 2006.
- [7] V. Bettadapura, C. Pantofaru, and I. Essa. Leveraging contextual cues for generating basketball highlights. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 908–917, 2016.
- [8] K. Bonnen, A. Huk, and L. Cormack. Dynamic mechanisms of visually guided 3D motion tracking. *Journal of Neurophysiology*, 118:1515–1531, 2017.

- [9] L. Bridgeman, M. Volino, J.-Y. Guillemaut, and A. Hilton. Multi-person 3d pose estimation and tracking in sports. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.
- [10] Z. Cai, H. Neher, K. Vats, D. A. Clausi, and J. Zelek. Temporal hockey action recognition via pose and optical flows. In *Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.
- [11] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.
- [12] M. A. Carbonneau, A. J. Raymond, E. Granger, and G. Gagnon. Real-time visual play-break detection in sport events using a context descriptor. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2808–2811. IEEE, 2015.
- [13] B. Chakraborty and S. Meher. A real-time trajectory-based ball detection-and-tracking framework for basketball video. *Journal of Optics*, 42(2):156–170, 2013.
- [14] A. Chan, M. D. Levine, and M. Javan. Player identification in hockey broadcast videos. *Expert Systems with Applications*, 165:113891, 2021.
- [15] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.
- [16] H. Chen, M. Tien, Y. Chen, W. Tsai, and S. Lee. Physics-based ball tracking and 3d trajectory reconstruction with applications to shooting location estimation in basketball video. *Journal of Visual Communication and Image Representation*, 20(3):204–216, 2009.
- [17] J. Chen and P. Carr. Autonomous camera systems: A survey. In *AAAI Conference on Artificial Intelligence*, volume 2, 2014.
- [18] J. Chen and P. Carr. Mimicking human camera operators. In *IEEE Winter Conference on Applications of Computer Vision*, pages 215–222, 2015.
- [19] J. Chen, H. M. Le, P. Carr, Y. Yue, and J. J. Little. Learning online smooth predictors for realtime camera planning using recurrent decision trees. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4688–4696, 2016.

- [20] J. Chen and J. J. Little. Sports camera calibration via synthetic data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.
- [21] J. Chen, K. Lu, S. Tian, and J. Little. Learning sports camera selection from internet videos. In *Winter Conference on Applications of Computer Vision*, pages 1682–1691. IEEE, 2019.
- [22] L. Citraro, P. Márquez-Neila, S. Savare, V. Jayaram, C. Dubout, F. Renaut, A. Hasfura, H. B. Shitrit, and P. Fua. Real-time camera pose estimation for sports fields. *Machine Vision and Applications*, 31(3):1–13, 2020.
- [23] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [24] S. Daigo and S. Ozawa. Automatic pan control system for broadcasting ball games based on audience’s face direction. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pages 444–447, 2004.
- [25] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.
- [26] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–22, 1977.
- [27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.
- [28] T. D’Orazio, C. Guaragnella, M. Leo, and A. Distanto. A new algorithm for ball recognition using circle Hough transform and neural classifier. *Pattern Recognition*, 37(3):393–408, 2004.
- [29] T. D’Orazio and M. Leo. A review of vision-based systems for soccer video analysis. *Pattern recognition*, 43(8):2911–2926, 2010.

- [30] A. Doumanoglou, V. Balntas, R. Kouskouridas, and T.-K. Kim. Siamese regression networks with efficient mid-level feature extraction for 3d object pose estimation. *arXiv preprint arXiv:1607.02257*, 2016.
- [31] E. Dubrofsky and R. J. Woodham. Combining line and point correspondences for homography estimation. In *International Symposium on Visual Computing*, pages 202–213. Springer, 2008.
- [32] A. Ekin and M. Tekalp. Generic play-break event detection for summarization and hierarchical sports video analysis. In *International Conference on Multimedia and Expo (ICME)*, volume 1, pages I–169. IEEE, 2003.
- [33] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image analysis*, pages 363–370, 2003.
- [34] H. Faulkner and A. Dick. AFL player detection and tracking. In *Digital Image Computing: Techniques and Applications*, pages 1–8. IEEE, 2015.
- [35] P. Felsen, P. Agrawal, and J. Malik. What will happen next? forecasting player moves in sports videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3342–3351, 2017.
- [36] W. Förstner, T. Dickscheid, and F. Schindler. Detecting interpretable and accurate scale-invariant keypoints. In *International Conference on Computer Vision*, pages 2256–2263. IEEE, 2009.
- [37] H. Freeman. Computer processing of line-drawing images. *ACM Computing Surveys*, 6(1):57–97, 1974.
- [38] V. R. Gaddam, R. Eg, R. Langseth, C. Griwodz, and P. 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*, 11(4):56, 2015.
- [39] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.
- [40] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

- [41] A. Gupta, J. J. Little, and R. J. Woodham. Using line and ellipse features for rectification of broadcast hockey video. In *Canadian Conference on Computer and Robot Vision*, pages 32–39. IEEE, 2011.
- [42] J. Halbinger and J. Metzler. Video-based soccer ball detection in difficult situations. In *International Congress on Sports Science Research and Technology Support*, pages 17–24. Springer, 2013.
- [43] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004.
- [44] J.-B. Hayet and J. Piater. On-line rectification of sport sequences with moving cameras. In *Mexican International Conference on Artificial Intelligence*, pages 736–746. Springer, 2007.
- [45] J.-B. Hayet, J. Piater, and J. Verly. Robust incremental rectification of sports video sequences. In *British Machine Vision Conference*, pages 687–696, 2004.
- [46] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [47] R. Hess and A. Fern. Improved video registration using non-distinctive local image features. In *Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.
- [48] A. Hilton, J. Guillemaut, J. Kilner, O. Grau, and G. Thomas. Free-viewpoint video for TV sport production. In *Image and Geometry Processing for 3-D Cinematography*, pages 77–106. 2010.
- [49] N. Homayounfar, S. Fidler, and R. Urtasun. Sports field localization via deep structured models. In *Computer Vision and Pattern Recognition*, pages 5212–5220, 2017.
- [50] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, and F. Yu. Joint monocular 3d vehicle detection and tracking. In *International Conference on Computer Vision*, pages 5390–5399, 2019.
- [51] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018.

- [52] M. S. Ibrahim, S. Muralidharan, Z. Deng, A. Vahdat, and G. Mori. A Hierarchical Deep Temporal Model for Group Activity Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1971–1980, 2016.
- [53] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.
- [54] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.
- [55] J. Janai, F. Güney, A. Behl, A. Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3):1–308, 2020.
- [56] W. Jiang, J. C. G. Higuera, B. Angles, W. Sun, M. Javan, and K. M. Yi. Optimizing through learned errors for accurate sports field registration. In *Winter Conference on Applications of Computer Vision*, pages 201–210, 2020.
- [57] P. R. Kamble, A. G. Keskar, and K. M. Bhurchandi. Ball tracking in sports: A survey. *Artificial Intelligence Review*, 52(3):1655–1705, 2019.
- [58] P. R. Kamble, A. G. Keskar, and K. M. Bhurchandi. A deep learning ball tracking system in soccer videos. *Opto-Electronics Review*, 27(1):58–69, 2019.
- [59] S. M. Kay. *Fundamentals of statistical signal processing: detection theory*. Prentice Hall PTR, 1998.
- [60] K. Kim, M. Grundmann, A. Shamir, I. Matthews, J. Hodgins, and I. Essa. Motion fields to predict play evolution in dynamic sport scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 840–847, 2010.
- [61] K. Kim, D. Lee, and I. Essa. Detecting regions of interest in dynamic scenes with camera motions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1258–1265, 2012.
- [62] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [63] J. Komorowski, G. Kurzejamski, and G. Sarwas. Deepball: Deep neural-network ball detector. *arXiv preprint arXiv:1902.07304*, 2019.

- [64] M. Koshkina, H. Pidaparthi, and J. H. Elder. Contrastive learning for sports video: Unsupervised player classification. In *Computer Vision and Pattern Recognition*, pages 4528–4536, 2021.
- [65] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [66] M. Leo, N. Mosca, P. Spagnolo, P. L. Mazzeo, T. D’Orazio, and A. Distanti. Real-time multiview analysis of soccer matches for understanding interactions between ball and players. In *International Conference on Image and Video Retrieval*, pages 525–534, 2008.
- [67] G. Li, S. Xu, X. Liu, L. Li, and C. Wang. Jersey number recognition with semi-supervised spatial transformer network. In *Computer Vision and Pattern Recognition Workshops*, pages 1783–1790, 2018.
- [68] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.
- [69] H. Liu and B. Bhanu. Pose-guided R-CNN for jersey number recognition in sports. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020.
- [70] J. Liu, P. Carr, R. T. Collins, and Y. Liu. Tracking sports players with context-conditioned motion models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1830–1837, 2013.
- [71] Y. Liu, D. Liang, Q. Huang, and W. Gao. Extracting 3D information from broadcast soccer video. *Image and Vision Computing*, 24(10):1146–1162, 2006.
- [72] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [73] K. Lu, J. Chen, J. J. Little, and H. He. Lightweight convolutional neural networks for player detection and classification. *Computer Vision and Image Understanding*, 172:77–87, 2018.

- [74] W.-L. Lu, K. Okuma, and J. J. Little. Tracking and recognizing actions of multiple hockey players using the boosted particle filter. *Image and Vision Computing*, 27(1-2):189–205, 2009.
- [75] A. Maksai, X. Wang, and P. Fua. What players do with the ball: a physically constrained interaction modeling. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 972–981, 2016.
- [76] M. Manafifard, H. Ebadi, and H. A. Moghaddam. A survey on player tracking in soccer videos. *Computer Vision and Image Understanding*, 159:19–46, 2017.
- [77] A. Mhalla, T. Chateau, S. Gazzah, and N. E. B. Amara. An embedded computer-vision system for multi-object detection in traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 20(11):4006–4018, 2018.
- [78] J. Miura, T. Shimawaki, T. Sakiyama, and Y. Shirai. Ball route estimation under heavy occlusion in broadcast soccer video. *Computer Vision and Image Understanding*, 113(5):653–662, 2009.
- [79] V. Morariu and L. Davis. Multi-agent event recognition in structured scenarios. In *Computer Vision and Pattern Recognition*, pages 3289–3296. IEEE, 2011.
- [80] X. Nie, S. Chen, and R. Hamid. A robust and efficient framework for sports-field registration. In *Winter Conference on Applications of Computer Vision*, pages 1936–1944, 2021.
- [81] K. Okuma, J. J. Little, and D. G. Lowe. Automatic rectification of long image sequences. In *Asian Conference on Computer Vision*, volume 9, 2004.
- [82] V. Pallavi, J. Mukherjee, A. K. Majumdar, and S. Sural. Ball detection from broadcast soccer videos using static and dynamic features. *Journal of Visual Communication and Image Representation*, 19(7):426–436, 2008.
- [83] P. Parisot and C. De Vleeschouwer. Consensus-based trajectory estimation for ball detection in calibrated cameras systems. *Journal of Real-Time Image Processing*, pages 1–16, 2016.
- [84] L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

- [85] H. Pidaparthi, M. Dowling, and J. H. Elder. Automatic play segmentation of hockey videos. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2021.
- [86] H. Pidaparthi and J. H. Elder. Keep your eye on the puck: automatic hockey videography. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1636–1644. IEEE, 2019.
- [87] A. Piergiovanni and M. S. Ryoo. Early detection of injuries in MLB pitchers from video. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.
- [88] F. Poiesi, F. Daniyal, and A. Cavallaro. Detector-less ball localization using context and motion flow analysis. In *International Conference on Image Processing*, pages 3913–3916. IEEE, 2010.
- [89] J. Quiñonero-Candela, M. Sugiyama, N. D. Lawrence, and A. Schwaighofer. *Dataset shift in machine learning*. MIT Press, 2009.
- [90] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [91] V. Ramanathan, J. Huang, S. Abu-El-Haija, A. Gorban, K. Murphy, and L. Fei-Fei. Detecting events and key actors in multi-person videos. In *Computer Vision and Pattern Recognition*, pages 3043–3053, 2016.
- [92] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017.
- [93] J. Ren, J. Orwell, G. A. Jones, and M. Xu. Tracking the soccer ball using multiple fixed cameras. *Computer Vision and Image Understanding*, 113(5):633–642, 2009.
- [94] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.
- [95] V. Reno, N. Mosca, R. Marani, M. Nitti, T. D’Orazio, and E. Stella. Convolutional Neural Networks based ball detection in tennis games. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1758–1764, 2018.

- [96] V. Reno, N. Mosca, M. Nitti, T. D’Orazio, D. Campagnoli, A. Prati, and E. Stella. Tennis player segmentation for semantic behavior analysis. In *International Conference on Computer Vision Workshops*, pages 1–8. IEEE, 2015.
- [97] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 234–241. Springer, 2015.
- [98] A. Rosenfeld and J. L. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1(1):33–61, 1968.
- [99] E. Rossi and A. Roorda. The relationship between visual resolution and cone spacing in the human fovea. *Nature Neuroscience*, 13(2):156–157, 2010.
- [100] L. Sha, J. Hobbs, P. Felsen, X. Wei, P. Lucey, and S. Ganguly. End-to-end camera calibration for broadcast videos. In *Computer Vision and Pattern Recognition*, pages 13627–13636. IEEE, 2020.
- [101] R. A. Sharma, B. Bhat, V. Gandhi, and C. Jawahar. Automated top view registration of broadcast football videos. In *Winter Conference on Applications of Computer Vision*, pages 305–313. IEEE, 2018.
- [102] H. B. Shitrit, J. Berclaz, F. Fleuret, and P. Fua. Multi-commodity network flow for tracking multiple people. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1614–1627, 2013.
- [103] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.
- [104] B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.
- [105] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [106] P. Skinner and S. Zollmann. Localisation for augmented reality at sport events. In *International Conference on Image and Vision Computing New Zealand*, pages 1–6. IEEE, 2019.

- [107] K. Soomro, S. Khokhar, and M. Shah. Tracking when the camera looks away. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 25–33, 2015.
- [108] M. Tavassolipour, M. Karimian, and S. Kasaei. Event detection and summarization in soccer videos using Bayesian network. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(2):291–304, 2013.
- [109] R. Theagarajan and B. Bhanu. An automated system for generating tactical performance statistics for individual soccer players from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 632–646, 2020.
- [110] G. Thomas. Real-time camera tracking using sports pitch markings. *Journal of Real-Time Image Processing*, 2(2):117–132, 2007.
- [111] G. Thomas, R. Gade, T. B. Moeslund, P. Carr, and A. Hilton. Computer vision for sports: Current applications and research topics. *Computer Vision and Image Understanding*, 159:3–18, 2017.
- [112] D. Tjondronegoro, Y. P. Chen, and B. Pham. The power of play-break for automatic detection and browsing of self-consumable sport video highlights. In *ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 267–274.
- [113] D. Tjondronegoro, Y. P. Chen, and B. Pham. Sports video summarization using highlights and play-breaks. In *ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 201–208, 2003.
- [114] D. Tjondronegoro, X. Tao, J. Sasongko, and C. H. Lau. Multi-modal summarization of key events and top players in sports tournament videos. In *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 471–478. IEEE, 2011.
- [115] C. Tomasi and T. Kanade. Detection and tracking of point. *International Journal of Computer Vision*, 9:137–154, 1991.
- [116] X. F. Tong, H. Q. Lu, and Q. S. Liu. An effective and fast soccer ball detection and tracking method. In *International Conference on Pattern Recognition*, volume 4, pages 795–798. IEEE, 2004.

- [117] M. R. Tora, J. Chen, and J. J. Little. Classification of puck possession events in ice hockey. In *2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pages 147–154. IEEE, 2017.
- [118] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1521–1528. IEEE, 2011.
- [119] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.
- [120] K. Vats, M. Fani, D. A. Clausi, and J. Zelek. Puck localization and multi-task event recognition in broadcast hockey videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4567–4575, 2021.
- [121] K. Vats, H. Neher, D. A. Clausi, and J. Zelek. Two-stream action recognition in ice hockey using player pose sequences and optical flows. In *Computer and Robot Vision (CRV)*, pages 181–188. IEEE, 2019.
- [122] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.
- [123] R. Voeikov, N. Falaleev, and R. Baikulov. TTNNet: Real-time temporal and spatial video analysis of table tennis. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020.
- [124] X. Wang, V. Ablavsky, H. B. Shitrit, and P. Fua. Take your eyes off the ball: Improving ball-tracking by focusing on team play. *Computer Vision and Image Understanding*, 119:102–115, 2014.
- [125] X. Wei, P. Lucey, S. Vidas, S. Morgan, and S. Sridharan. Forecasting events using an augmented hidden conditional random field. In *Asian Conference on Computer Vision*, pages 569–582. Springer, 2014.
- [126] P.-C. Wen, W.-C. Cheng, Y.-S. Wang, H.-K. Chu, N. C. Tang, and H.-Y. M. Liao. Court reconstruction for camera calibration in broadcast basketball videos. *IEEE Transactions on Visualization and Computer Graphics*, 22(5):1517–1526, 2015.

- [127] L. Wu, X. Meng, X. Liu, and S. Chen. A new method of object segmentation in the basketball videos. In *International Conference on Pattern Recognition*, pages 319–322. IEEE, 2006.
- [128] J. Xiao, R. Stolkin, and A. Leonardis. Multi-target tracking in team-sports videos via multi-level context-conditioned latent behaviour models. In *British Machine Vision Conference*, 2014.
- [129] L. Xie, P. Xu, S. Chang, A. Divakaran, and H. Sun. Structure analysis of soccer video with domain knowledge and hidden markov models. *Pattern Recognition Letters*, 25(7):767–775, 2004.
- [130] M. Xu, J. Orwell, and G. Jones. Tracking football players with multiple cameras. In *International Conference on Image Processing*, pages 2909–2912, 2004.
- [131] M. Yakut and N. Kehtarnavaz. Ice-hockey puck detection and tracking for video highlighting. *Signal, Image and Video Processing*, 10(3):527–533, 2016.
- [132] F. Yan, W. Christmas, and J. Kittler. Ball tracking for tennis video annotation. In *Computer Vision in Sports*, pages 25–45. 2014.
- [133] Q. Yao, A. Kubota, K. Kawakita, K. Nonaka, H. Sankoh, and S. Naito. Fast camera self-calibration for synthesizing free viewpoint soccer video. In *International Conference on Acoustics, Speech and Signal Processing*, pages 1612–1616. IEEE, 2017.
- [134] X. Yu, C. Xu, H. W. Leong, Q. Tian, Q. Tang, and K. W. Wan. Trajectory-based ball detection and tracking with applications to semantic analysis of broadcast soccer video. In *ACM International Conference on Multimedia*, pages 11–20. ACM, 2003.
- [135] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.
- [136] R. Zeng, R. Lakemond, S. Denman, S. Sridharan, C. Fookes, and S. Morgan. Calibrating cameras in poor-conditioned pitch-based sports games. In *International Conference on Acoustics, Speech and Signal Processing*, pages 1902–1906. IEEE, 2018.

- [137] X. Zhang, T. Zhang, Y. Yang, Z. Wang, and G. Wang. Real-time golf ball detection and tracking based on convolutional neural networks. In *International Conference on Systems, Man, and Cybernetics*, pages 2808–2813. IEEE, 2020.
- [138] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(11):1330–1334, 2000.

Appendix A

Deep network architecture optimization

As described in Section 2.5.2.2, the architecture of the first-stage deep network regressor was optimized on the non-rectified videos (games) from Camera 1 (Fig 2.3a). The video resolution of Camera 1 was $3,840 \times 2,160$ pixels. After correcting for non-linear distortions and manually vertically cropping the video to the ice rink, the video resolution reduced to $2,758 \times 500$. Five games were recorded using Camera 1. The duration of each game is listed in Table 2.2. The dataset was split such that 3 games were used for training, one for validation and one for test. This resulted in 10 different combinations of games for training. The training, validation and test errors were averaged across all 10 combinations.

The input data consisted of three feature maps: 1) RGB image, 2) horizontal and vertical optic flow, and 3) binary player positions mask. The optic flow was computed at an image resolution of 688×288 . The players were detected using Faster RCNN detector at an input image resolution of 1920×360 . These input feature maps were normalized to have zero-mean and unit-variance. The training dataset was augmented in four different ways, as described in Section 2.5.2.2.

To explore the optimal deep network architecture, we started off with a small network and progressively increased the size of the network. Initially, we explored several different architectures as listed in Table A.1. Network A22 consisted of two convolution layers followed by two fully connected layers. The first convolution layer had 8 channels, while the second layer had 16 channels. Both layers used square kernels of size 11×11 ¹. The

¹As only square kernels were used for all experiments, the kernel size for each layer is expressed as a

Network	# Conv layers	# FC layers	Channels	Kernels	FC layer size
A11	1	1	8	11	512
B11	1	1	8	11	256
A12	1	2	8	11	512, 256
B12	1	2	8	11	256, 128
A21	2	1	8, 16	11, 11	512
B21	2	1	8, 16	11, 11	256
A22	2	2	8, 16	11, 11	512, 256
B22	2	2	8, 16	11, 11	256, 128

Table A.1: Different convolutional layer architectures explored in the first phase of experiments.

first fully connected layer had 512 units, while the second fully connected layer had 256 units. For all these networks, each convolutional layer was followed by a 2×2 max pooling layer. Dropout was used between each fully connected layer. The output of the network was the estimated x-y pixel location of the puck. The loss function was the mean squared error between estimated and ground truth puck location, minimized using the Adam optimizer [62]. The model was trained until the validation error converged. The initial learning rate was 0.01 and weight decay was 0.01.

The 6-channel input feature maps were resized before being passed to the deep network. Each network was trained using input data resized to three different sizes: 1) 98×40 , 2) 115×48 , and 3) 138×58 . For each input data size, for a given network architecture, the mean and standard error for the training, validation and test datasets are reported, as seen in Table A.2. As there were 10 different combinations of training games, the mean and standard error across all 10 combinations was reported. We found that the validation error was higher for smaller networks and it decreased as the network size increased. The validation error was the least for networks A22 and B22 which consist of two convolution layers followed by two fully connected layers. For the most part, the errors were lower for the larger input data size of 138×58 . Hence, in the next phase of experiments, we explored larger input data sizes and larger network architectures.

We followed a two-step approach to exploring larger networks and increasing the size

scalar.

Network	Error (m)								
	Input data size: 98×40			Input data size: 115×48			Input data size: 138×58		
	Training	Validation	Test	Training	Validation	Test	Training	Validation	Test
A11	3.1 ± 0.04	6.38 ± 0.21	6.26 ± 0.21	2.97 ± 0.04	6.51 ± 0.28	6.43 ± 0.27	2.69 ± 0.04	6.47 ± 0.25	6.46 ± 0.29
B11	3.45 ± 0.03	6.34 ± 0.22	6.3 ± 0.24	3.2 ± 0.04	6.31 ± 0.25	6.27 ± 0.26	2.94 ± 0.04	6.32 ± 0.23	6.36 ± 0.24
A12	2.32 ± 0.01	5.72 ± 0.15	5.75 ± 0.17	2.17 ± 0.02	5.73 ± 0.17	5.74 ± 0.15	2.07 ± 0.02	5.73 ± 0.17	5.74 ± 0.18
B12	2.8 ± 0.02	5.72 ± 0.15	5.74 ± 0.16	2.64 ± 0.02	5.71 ± 0.15	5.74 ± 0.16	2.53 ± 0.02	5.77 ± 0.17	5.74 ± 0.16
A21	4.02 ± 0.05	5.85 ± 0.17	5.86 ± 0.18	3.7 ± 0.04	5.74 ± 0.15	5.67 ± 0.17	3.05 ± 0.1	5.92 ± 0.22	6.06 ± 0.26
B21	4.15 ± 0.04	5.92 ± 0.15	5.83 ± 0.16	3.85 ± 0.04	5.79 ± 0.15	5.69 ± 0.17	3.4 ± 0.03	5.96 ± 0.25	5.95 ± 0.26
A22	3.56 ± 0.03	5.47 ± 0.14	5.41 ± 0.14	2.8 ± 0.02	5.38 ± 0.12	5.39 ± 0.13	2.19 ± 0.03	5.33 ± 0.13	5.32 ± 0.13
B22	3.87 ± 0.05	5.56 ± 0.13	5.5 ± 0.13	3.28 ± 0.03	5.48 ± 0.13	5.44 ± 0.16	2.71 ± 0.02	5.31 ± 0.14	5.34 ± 0.14

Table A.2: Errors on the training, validation and test datasets for each network architecture at a given input data size, in the first phase of experiments.

of the network. In the first step, we explored the optimal architecture for the convolution layers while fixing the architecture of the fully connected layers. In the second step, the architecture of the convolutional layers was fixed to be that of the optimal architecture from the first step, and the optimal architecture for the fully connected layers was explored.

In the first step, all networks consisted of 3 fully connected layers such that the first layer had 512 units, the second layer had 256 units and the third layer had 128. The different architectures for the convolutional layers explored in the first step are listed in Table A.3. The kernel sizes were chosen such that all networks can be explored for all four input data sizes.

Network	# Conv layers	# FC layers	Channels	Kernels	FC layer size
A2	2	3	8, 16	11, 11	512, 256, 128
A3	3	3	8, 16, 32	11, 7, 7	512, 256, 128
A4	4	3	8, 16, 24, 32	7, 7, 5, 5	512, 256, 128
A5	5	3	8, 16, 24, 32, 48	7, 7, 5, 5, 3	512, 256, 128
A6	6	3	8, 16, 24, 32, 48, 64	7, 7, 5, 5, 3, 3	512, 256, 128

Table A.3: Different convolutional layer architectures explored in the second phase of experiments.

The structure of the convolutional layers and pooling layers for each network in the second phase of experiments is shown in Table A.4. For all networks, 2×2 max pooling layers are used. The networks A2 and A3, consists of two and three *conv-pool* modules, respectively. These are modules where a convolutional layer is immediately followed by

a pooling layer. The first module of network A4 is a *conv-conv-pool* module, where the first convolutional layer is followed by a second convolution layer before a pooling layer. The subsequent layers are *conv-pool* modules. Networks A5 and A6, have two or three *conv-conv-pool* modules. The learning rate, optimizer and loss function are retained to be the same as that in the first phase of experiments.

Network	Layer structure
A2	conv-pool – conv-pool
A3	conv-pool – conv-pool – conv-pool
A4	conv-conv-pool – conv-pool – conv-pool
A5	conv-conv-pool – conv-conv-pool – conv-pool
A6	conv-conv-pool – conv-conv-pool – conv-conv-pool

Table A.4: The structure of the convolutional layers for each network in the second phase of experiments.

In the second phase of experiments, four different input data sizes were explored: 1) 138×58 , 2) 172×72 , 3) 215×90 , and 4) 258×108 . For each input data size, for a given network architecture, the mean and standard error for the training, validation and test datasets are reported, as seen in Table A.5 and Table A.6. We found that the validation error was least for network A4 and the lowest error was achieved at an input data size of 172×72 .

Network	Error (m)					
	Input data size: 138×58			Input data size: 172×72		
	Training	Validation	Test	Training	Validation	Test
A2	2.46 ± 0.03	5.37 ± 0.16	5.24 ± 0.19	2.17 ± 0.02	5.26 ± 0.18	5.2 ± 0.18
A3	2.52 ± 0.04	5.24 ± 0.17	5.26 ± 0.17	2.15 ± 0.03	5.18 ± 0.18	5.17 ± 0.16
A4	3.22 ± 0.03	5.19 ± 0.18	5.2 ± 0.18	2.48 ± 0.02	5.06 ± 0.16	5.07 ± 0.12
A5	3.15 ± 0.03	5.44 ± 0.15	5.3 ± 0.18	2.41 ± 0.02	5.25 ± 0.19	5.12 ± 0.12
A6	3.38 ± 0.04	5.42 ± 0.16	5.31 ± 0.24	2.56 ± 0.03	5.38 ± 0.14	5.2 ± 0.18

Table A.5: Errors on the training, validation and test datasets for each network architecture in the second phase of experiments, at input data sizes of 138×58 and 172×72 .

Network	Error (m)					
	Input data size: 215×90			Input data size: 258×108		
	Training	Validation	Test	Training	Validation	Test
A2	2.21 ± 0.03	5.28 ± 0.18	5.27 ± 0.18	2.57 ± 0.03	5.32 ± 0.18	5.28 ± 0.18
A3	2.55 ± 0.03	5.24 ± 0.18	5.24 ± 0.22	2.65 ± 0.03	5.27 ± 0.18	5.26 ± 0.22
A4	2.69 ± 0.03	5.20 ± 0.20	5.22 ± 0.21	2.77 ± 0.03	5.23 ± 0.20	5.25 ± 0.21
A5	2.61 ± 0.05	5.19 ± 0.17	5.27 ± 0.22	2.71 ± 0.05	5.23 ± 0.17	5.29 ± 0.22
A6	2.64 ± 0.03	5.33 ± 0.18	5.34 ± 0.18	2.75 ± 0.03	5.35 ± 0.18	5.39 ± 0.18

Table A.6: Errors on the training, validation and test datasets for each network architecture in the second phase of experiments, at input data sizes of 215×90 and 258×108 .

In the first step, the optimal architecture for the convolutional layer was found to be the network architecture A4 and the best input data size was found to be 172×72 . In the second step, these configurations were fixed and the optimal architecture for the fully connected layer was explored. The different architectures are listed in Table A.7. Network 4FC2 has 4 fully connected layers with the first, second, third and fourth layers having 2048, 1024, 256, 128 units, respectively.

Network	# FC layers	FC layer size
4FC1	4	2048, 1024, 256, 128
4FC2	4	784, 512, 128, 128
3FC1	3	1024, 512, 256
3FC2	3	512, 256, 128
2FC1	2	512, 256
2FC2	2	256, 128

Table A.7: Different fully connected layer architectures explored in the second phase of experiments.

In a similar manner to the previous experiments, we averaged the training, validation and test error across all combinations of training games. The errors for various fully connected layer architectures are listed in Table A.8. We found the errors were least for networks with 2 fully connected layers, with the least error achieved by the network 2FC2.

Network	Training	Validation	Test
4 FC1	2.22 ± 0.03	5.39 ± 0.21	5.37 ± 0.17
4 FC2	2.79 ± 0.05	5.53 ± 0.19	5.54 ± 0.19
3 FC1	2.42 ± 0.02	4.95 ± 0.14	5.06 ± 0.14
3 FC2	2.50 ± 0.03	5.03 ± 0.13	5.08 ± 0.14
2 FC1	2.21 ± 0.01	4.99 ± 0.16	5.00 ± 0.16
2 FC2	2.29 ± 0.02	4.97 ± 0.14	4.98 ± 0.17

Table A.8: Errors on the training, validation and test datasets for different architectures of the fully connected layers.

We found the optimal architecture for the first-stage deep network to have four convolutional layers (A4) followed by two fully connected layers (2FC2). The layer structure of the convolutional layers is: *conv-conv-pool* module followed by two *conv-pool* modules. The number of channels for each convolutional layer was [8, 16, 24, 32] with the kernel sizes being [7,7,5,5]. All layers used square kernels. The first layer had 8 channels and used a 7×7 kernel. 2×2 max pooling layer was used to aggregate the features learned from the convolutional layers. The optimal network consisted of two fully connected layers such that the first layer had 256 units while the second layer had 128 units. Dropout was used between each fully connected layer.