Automatic Acquisition of Motion Trajectories: Tracking Hockey Players

by

Kenji Okuma

BA (Computer Science) Hiram College, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

May 2003

Abstract

We address the problem of automatically analyzing hockey scenes by estimating the panning, tilting and zooming parameters of the broadcasting cameras, tracking hockey players in these scenes, and constructing a visual description of the scenes as trajectories of those players. Given quite fast and non-smooth camera motions to capture highly complex and dynamic scenes of hockey, tracking hockey players that are small blob-like, non-rigid and amorphous becomes an extremely difficult task. We suggest a new method of automatically computing the mappings to represent the globally consistent map of the hockey scenes by removing camera motions, and implement a color-based sequential Monte Carlo tracker to track hockey players to estimate their real world position on the rink. The result demonstrates a quite successful performance on both objectives. We make two new contributions in this research. First, we introduce a new model fitting algorithm to reduce projection errors. Second, we use an adaptive model to improve the current state-of-art color-based probablistic tracker. Our approach is also applicable for video annotation in other sports, surveillance, or many other situations that require obect tracking on a planar surface. Since there have not been any hockey annotation systems developed in the past, we hope that our system would become a stepping stone for automatic video annotation in hockey.

Keywords: RANSAC, KLT, Homography, sequential Monte Carlo

Contents

List of Figures

Acknowledgements

I would have never achieved this work without contributions by a number of individuals in academic circles. In particular, my appreciation is extended on helpful comments and meaningful discussions with my colleagues in the Laboratory of Computational Intelligence, namely Matthew Brown, Pantelis Elinas, Jesse Hoey, Fahong Li, Don Murray, and Michael Zhang. Since their contributions have been important in so many different ways, I would like to acknowledge the significance of their contributions in no other ways but alphabetically. Finally and more importantly, I would like to express special thanks to my supervisors, Dr. Little and Dr. Lowe for having numerous discussions with me and leading me to the right direction of research.

KENJI OKUMA

The University of British Columbia
May 2003

Chapter 1

Introduction

1.1 Motivation

Computer systems that have the capability of analyzing complex and dynamic scenes could play an essential role in video annotation. Scenes could contain many cluttered objects with different colors, shapes and sizes, and can be dynamic with multiple interactive moving objects and a constantly changing background. There are many scenes that are complex, dynamic, and challenging for computers to describe. Those scenes include games of sports, air traffic, car traffic, street intersections, and cloud transformations.

Our motivation arises in the challenge of inventing a descriptive computer system that analyzes scenes of hockey games where multiple moving players interact with each other on a constantly moving background due to camera motions. Ultimately, such a computer system should be able to acquire reliable data by extracting the players' motion as their trajectories, querying them by analyzing the descriptive information of data, and predicting the motions of some hockey players based on the result of the query.

Among these three major aspects of the system, we primarily focus on visual information of the scenes, that is, how to automatically acquire motion trajectories of hockey players from video. The source of our motivation is composed of one practical and two technical reasons.

First, hockey is one of the major sports in North America and there is a great

potential and space for the practical application of automatic video annotation. Although there have been video annotation systems in the domain of soccer [3, 7, 17, 38], basketball [29] and football [11], to the best of our knowledge, there have not been any automatic video annotation systems in hockey.

Second, the data acquisition part of the system we focus on is the heart of our hockey annotation system. For reliable performance of hockey analysis, it is crucial that we have a robust vision system to gain data for a further analysis. Our goal is to achieve the degree of automation and accuracy of the system which leads to our future development of the complete automatic hockey annotation system.

Finally, many technical challenges in vision make this problem worthwhile to tackle, which may be a reason why a complete annotation system for hockey has not been developed in the past. Unlike most major sports of North America including football, baseball, and basketball, hockey players move much faster and more unpredictably, which makes it extremely difficult to track them. The area of the hockey rink is smaller than the soccer or football field and the speed of a puck is much greater than that of a soccer ball, basketball or football. Therefore, broadcast hockey video often contains fast and non-smooth camera motions to capture fast-moving dynamic scenes of hockey games. Section 1.4 summarizes vision challenges we need to solve. Although there are many obstacles to overcome, our efforts and accomplishments would hopefully establish the infrastructure of the automatic hockey annotation system and contribute to research in automatic video annotation.

1.2 The Problem

With the advance of information technologies and the increasing demand of managing the vast amount of visual data in video, there is a great potential for developing reliable and efficient sports systems that are capable of understanding and describing various scenes of sports. Such systems would require the ability of analyzing the content of the data, which is the problem addressed in this thesis. The description of the problem is well articulated by Intille [11]:

> "Video annotation is the task of generating descriptions of video sequences

> that can be used for indexing, retrieval, and summarization. . . . one is primarily interested in what is happening in a scene, as opposed to what is in the scene."

The problem is not to build the system that performs high-level analysis of scenes, but to develop an automated video annotation system that provides the description of the scenes by performing low-level analysis of the scenes. More accurately, it is about automatically analyzing the hockey scenes by estimating parameters (i.e., pan, tilt, and zoom) of the broadcasting cameras, tracking hockey players in these scenes, and constructing a visual description of the scenes as trajectories of those players.

1.3 Hockey Annotation System

This section provides an overview of our system. As it is shown in Figure 1.1, our annotation system takes a TV sequence and automatically generates trajectories of tracked players as a visual description of the scenes. Two major major components of our system, namely the tracking system and rectification system, are explained in detail in the preceding chapters. Although our system could be described on a simple picture in Figure 1.1, building such a system requires to solve many vision problems that make this challenge difficult. The next section summarizes major vision problems we have encountered throughout the development of our system.

1.4 Vision challenges

In order to achieve a successful development of the vision system, we need to overcome various vision challenges. This section describes vision problems, including camera motions, low-resolution, deformable, non-smooth object to be tracked, and their rapid motions.

1.4.1 Camera motions

Since the source of our data is limited to video clips of broadcasting hockey games, there are various camera motions that we need to deal with or occasionally need to

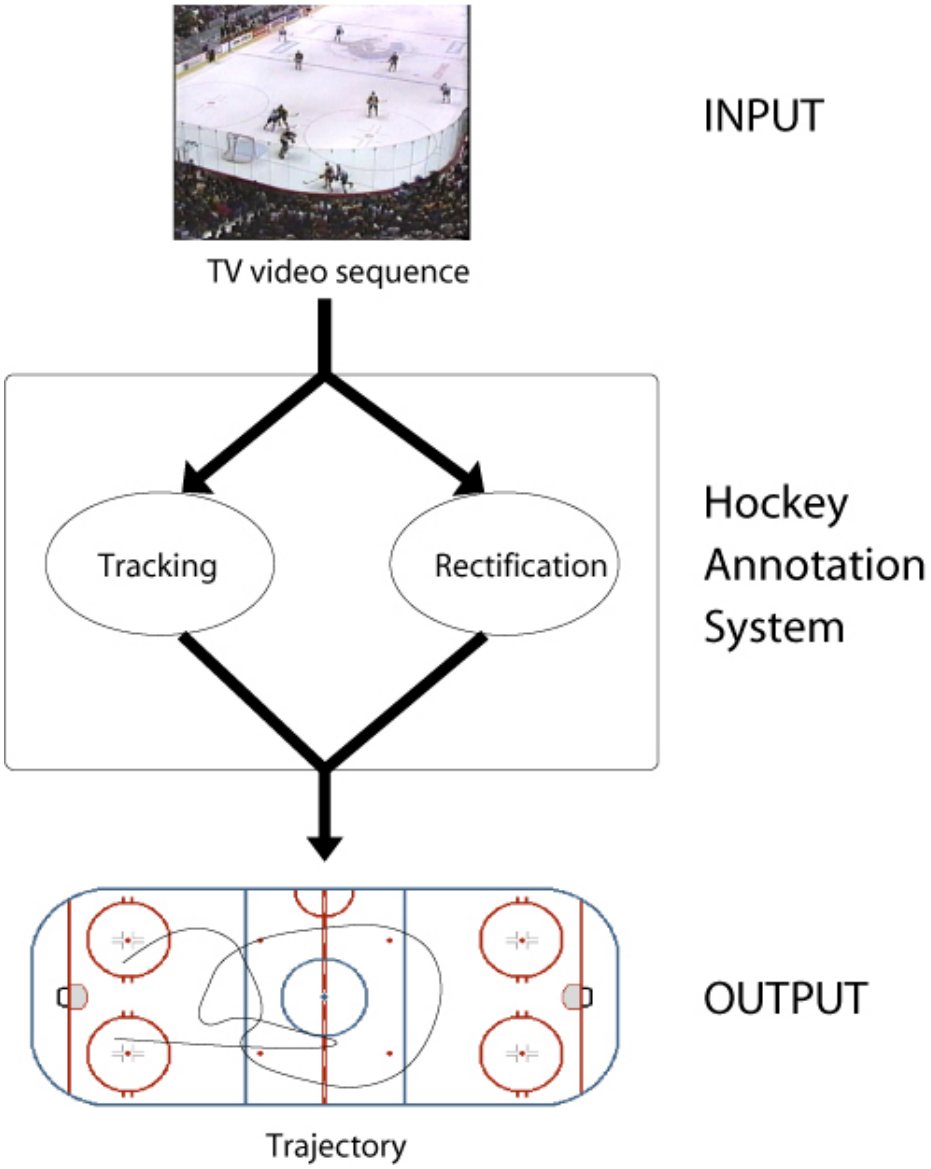
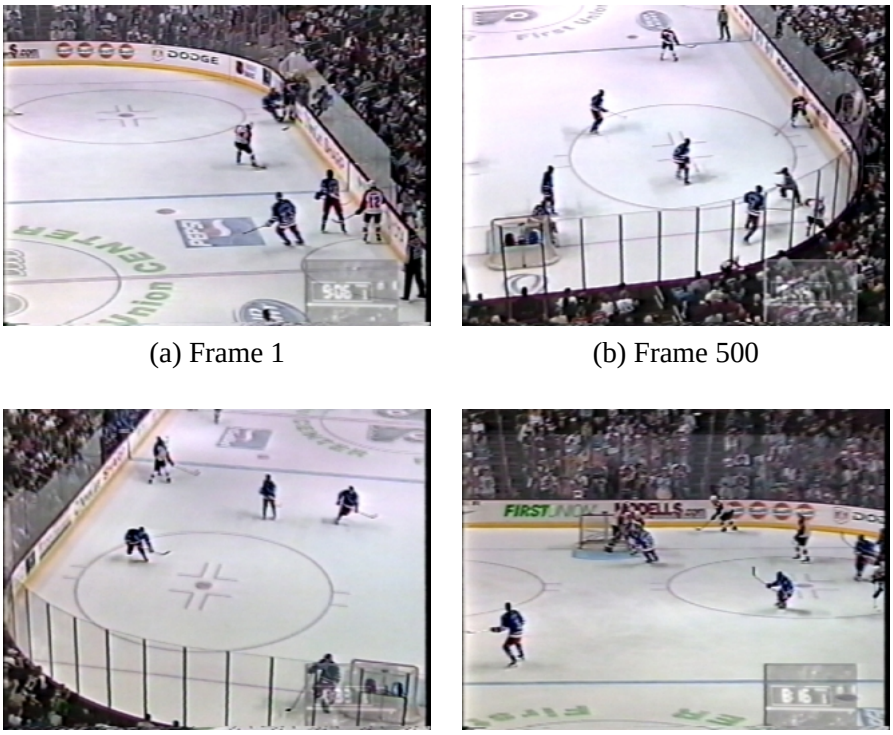


Figure 1.1: **Hockey Annotation System**

This image explains how our hockey annotation system works. It takes a TV video sequence as an input and generates a description of the scenes as trajectories of the tracked players in the coordinate of the globally consistent map of the rink.

avoid. Figure 1.2 shows how much a camera moves over 1500 frames which is 50 seconds at the sampling rate of 30 frames per second. As it is shown, within only 50 seconds, a camera captures most of the rink surface. These camera motions include panning, tilting, and zooming in and out.



(a) Frame 1 (b) Frame 500

(c) Frame 1000 (d) Frame 1500

Figure 1.2: **Camera motions over 1500 frames**

The image of Frame 1, 500, 1000, and 1500 respectively from our hockey data. These four images show a different part of the rink captured by the same camera.

As it is demonstrated in later chapters, our vision system can deal with those motions. However, there are also camera motions we need to avoid. Those are occasional close-ups of players. The experimental data do not contain any close-ups of players because those scenes contain only information about players and no information on the rink, and there is no way to derive the transformation between the original data and the globally consistent rink map we use as a model. With data that contains no close-up shots of players, we find information about the camera pan, tilt, zoom by extracting visual features on static objects, such as standard rink elements (i.e., lines, circles, and

advertisements both on the rink and at the side of the rink) and audience, between two different images. Chapter 3 explains more details.

1.4.2 Camera flashes

During a hockey game, there are many camera flashes from fans in the audience. From a computer vision standpoint, camera flashes cause a drastic illumination change of the image. For example, Figure 1.3 shows a frame without a camera flash and one with a flash. It is clear that there is a noticeable change in the brightness of two different frames.

(a) a frame before a camera flashes (b) a frame with a camera flash

Figure 1.3: **Frame with or without a camera flash**

(a) is the image before a camera flashes. (b) is the next frame when a camera flashes. These two frames show how much camera flashes could affect the illumination condition of the image. Note the glass above the rink boards where we see the players' shadow and reflection.

Since camera flashes could result in failure of our vision system, we need to avoid them. In Chapter 3, we implement a detector to catch the intensity change of two different frames.

1.4.3 Attributes of target objects

For the successful development of the tracking system, it is crucial to have detailed information about the attributes of the targets. This section presents attributes of our target objects and issues specifically related to the tracking of hockey players.

Low spatial resolution

In our hockey data, players are small objects in low spatial resolution. They have the size of about 25×25 pixels at the highest resolution and 10×10 pixels at the lowest resolution. Figure 1.4 shows sixteen instances of such a small object.

Figure 1.4: **Low-resolution objects (i.e., hockey players)**

These people are some forwards, defensemen, and referees. All of them have low spatial resolution and it is hard to recognize them even by human vision.

Low-resolution objects limit the use of vision techniques that require high spatial resolution images. Furthermore, the role of players is not visible from their appearance, but only in relative position during play. Only goalie looks distinguishable from other players. However, it is important to note that the color property of the objects is still useful for tracking such objects in both high and low resolution images.

Deformable shapes

Target objects are not only in low spatial resolution, but also deformable. As it is shown in Figure 1.5, it is clear that a hockey player is a non-rigid, deformable object. In a video sampled at 30 frames per second, the object undergoes large shape changes.

Obviously, we cannot simply model the shape of the object. It is even harder to recognize distinctive visual features of the players such as numbers, logos on the uniform, or the color of their helmet. Although it seems quite difficult to track hockey players, it is not impossible to track hockey players. While Intille and Bobick [11] takes a non-model based approach on the tracking of similar objects (i.e., football players in their case) and obtains a successful tracking result of around 200 frames, we do not take the same approach as them. Chapter 4 explains, in detail, how we model low-resolution and deformable objects and how they are successfully tracked.

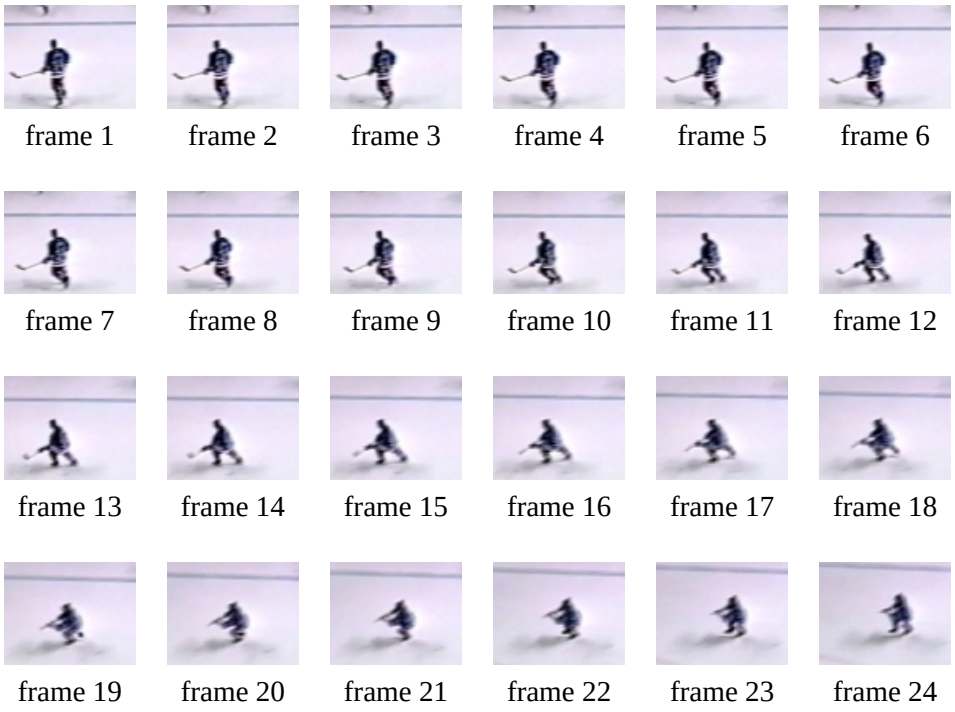


Figure 1.5: **How rapidly a hockey player changes his shape**

There are 24 frames (slightly less than a second) to display how the shape of a hockey player changes. There is a clear difference between the images in the first row and the last row.

The presence of similar objects

In hockey, there are six players in each team who play on the rink in a game. Therefore, there are many occasions when the players in the same team or in a different team come close or even collide and occlude each other. This makes the tracking task extremely difficult. Chapter 4 demonstrates the robustness of our tracking method in occlusions and cluttered scenes.

1.5 Outline of Thesis

Developing an automatic hockey annotation system is a challenging task. In the following chapters, we show that it is, however, possible to develop such a descriptive computer system that automatically analyzes hockey scenes. Our hockey system is composed of two major components: One is the rectification system for computing transformations between a broadcast video sequence and the globally consistent rink map. The other one is the tracking system for tracking hockey players and estimating their real world position on the rink.

The next chapter explores the previous work in the development of sports system in various other fields of sports, introducing a brief background of tracking techniques, and justifies our choice of the tracker for tracking small objects (i.e., hockey players) in complex environments. Chapter 3 explains the process of the rectification system and show how the system automatically registers the sequence. We present a improved version of the Kanade-Lucas-Tomasi(KLT) tracker by predicting the current camera motion based on the previous camera motion and introduce a new model fitting algorithm to reduce projection errors over time. Chapter 4 describes the tracking system for hockey players. We improve the current state-of-art color-based probabilistic method by applying adaptation to the color-based likelihood model. Along with the tracking results by our color-based sequential Monte Carlo tracker, we also present trajectories of hockey players by transforming real world positions of hockey trackers to the globally consistent rink map. In the last chapter, we summarize our thesis and show future directions for the further development of our annotation system.

Chapter 2

Previous Work

This chapter presents a brief review of the literature on sports systems and techniques of visual tracking, and justifies our approaches taken to develop our annotation system.

2.1 Sports systems

Video annotation for sports has been one of the major research areas for which vision techniques are essential. This section is about the brief literature survey of past video annotation systems in various sports.

2.1.1 Soccer

Soccer is known as the most popular sport in the world. Due to a great potential and an expected high market value for the practical application of soccer annotation systems, many annotation systems have been developed in the past. In this section, we present only selected ones that have techniques related to our hockey annotation system: two representatives for low-level analysis such as the tracking of players and a ball, and one for high-level analysis on the classification of the scenes.

Yamada *et. al* [38] develop a system to analyze soccer game scenes by tracking players and a ball for estimating their 3D positions. Their system estimates parameters (i.e., pan, tilt, and zoom) of a camera in each frame and maps positions in the image to 3D positions on the globally consistent map of the soccer field. A TV broadcast video sequence is used for their low-level content analysis. Assuming that soccer

players have smooth and constant motion within a short time period, the positional information of players is used to estimate their current position. The color information of the uniform is used to deal with occlusions. As for tracking a ball, the detection is performed by extracting the white regions that are neither players nor lines, and the position of the ball is estimated based on the previous position, gravity and air friction. For the 3D position of players and a ball, the rotational parameters (i.e., the pan and tilt angles) and the focal length of the camera are estimated by matching extracted line and circle regions to the model of the soccer field. In the initial frame of the sequence, the camera parameters are estimated by maximizing the number of pairs of the matching points and estimated again for the second frame by searching around the initial parameters. Assuming the smooth motion of broadcasting cameras, the system estimates the camera parameters for subsequent frames by extrapolating the previous two positions of the camera. The 3D position of players and a ball is now gained based on the estimation.

The system developed by Yow *et. al* [3] explores techniques to analyze soccer games by extracting the soccer highlights and presenting their content as the panoramic views of trajectories of players and a ball. Their system uses a luminance model based on a second-order Taylor series expansion for motion estimation between frames. The panning parameters of a camera are estimated first for initialization. The system then applies block matching on texture rich regions of the field to estimate the global motion parameters of the camera. For detection of the ball, the template-based "intra-frame" approach is used by searching the possible motions of the ball among all motions detected based on the difference image of two frames. Considering the previous and next position of the ball based on estimated camera parameters, the tracking of the ball is performed.

Those two systems are, however, not appropriate for hockey games due to the following reasons: First, their assumption of smooth and constant motion of players and a ball does not hold in hockey because the motion of hockey players is much faster and less smooth than soccer players and the speed of the puck is much greater than that of the soccer ball. Second, in hockey, the camera motion is required to be fast and non-smooth for capturing much faster motions of players and the puck where their

method of estimating camera parameters has most likely failed to work. Third, their systems may not be robust because there is no evidence that proves the robustness of those systems for more than 70 frames, that is, a bit over two seconds of the scenes at the sample rate of 30 frames per second. Finally, it is extremely difficult to predict the position and motion of a puck due to its tiny size and quite fast and unpredictable motion. While the detection and tracking of the puck is one of the hardest vision problems remained to be unsolved, it is certainly a key to the success of our future annotation system that eventually combines low-level and high-level analysis of hockey.

For high-level content analysis of soccer, an automatic parsing system on TV broadcasting video is introduced by Gong *et. al* [7]. The goal of their system is to extract semantic information of the soccer games and categorize it into various highlights such as shooting scenes, and corner kick scenes. Their parsing system takes a model based approach which uses the soccer field, a ball, players and camera motions to recognize the soccer scenes. The detection of the field features such as lines, circles, and the goal posts is performed to recognize where the play is happening. The shape and color of the ball is used for the detection of a ball to figure out the content of the play (i.e., shootings, corner kicks, goals and etc) by finding out where the play is happening on the soccer field and the color of players' uniform is used to further refine their recognition accuracy. Their system provides accurate high-level analysis of soccer scenes without tracking players or a ball. However, now that their system requires the detection of a ball, it does not work in hockey because there is no vision technique to accurately detect and track a puck. With the detection and tracking of a puck, their approach is a possible future direction of our annotation system.

2.1.2 Other sports

There are also annotation systems developed in other sports. For instance, Saur *et. al* [29] develop a system for automated analysis and annotation of basketball video. Their system provides fast and efficient analysis of basketball video by using only information in the MPEG bit stream without full decompression. There are three different frames of MPEG standard such as Intra-pictures, Predicted pictures and Bi-directional predicted pictures. In any one frame, they are limited to the following information:

macroblock motion vectors, intra-coded block positions and count, block discrete cosine transform coefficients and residual error. Using motion vectors and counting intra-coded blocks, the panning parameters of camera motions are estimated for high-level analysis of the scenes including wide-angle and close-up views, fast breaks, steals, potential shots, number of possessions and possession times. They do not use low-level content of video such as colors, object shapes, motions of players, and image texture. Since our hockey annotation system focuses on low-level content of video, their method does not directly apply to our system. Their result on a minute long video sequence are fairly successful since they simplify their analysis by not considering the tilting parameters of camera motions and camera motions are reasonably smooth and constant in basketball. Although their method may not work well in highly complex and dynamic scenes such as ones in hockey, their approach indicates a possible future extension of our annotation system that may require the combination of low-level content of video and camera motions for accurate high-level analysis.

Intille and Bobick [11] develop the-state-of-art automatic annotation system for American football footage and become the pioneer of automatic video annotation research in the domain. Their system extracts camera motions by tracking features on the football field and registering the original football video sequence to the globally consistent map of the football field. With an accurate registration of the globally consistent football sequence, they conduct "closed-world" analysis to track football players in complex scenes where "closed-world" is defined as "a region of space and time in which the specific context is adequate to determine all possible objects present in that region." In other words, given the model of the football field that is composed of all field features including turf, lines, hash-marks, number, arrows, and logos, they extract blobs of moving objects (i.e., players) by taking the difference image between two frames. Only player's pixels are then extracted from those motion blobs by "closed-world" analysis and the tracking is performed on those player's pixels. Their system is currently the-state-of-art annotation system since they prove the robustness and reliability of their system by showing the result of around 200 frames that is longer than any other annotation systems for low-level content analysis can give. Therefore, their approach becomes the model of our system.

As a consequence of surveying video annotation systems in various sports, it is evident that soccer, basketball, and football are easier domains for automatic video annotation than hockey from a vision standpoint. In the following chapters, we show that our annotation system has a similar framework as Intille and Bobick's system, yet uses completely different tracking approaches and gives much better performance.

2.2 Tracking techniques

Object tracking requires the efficient tracking of visual features in complex environments and applies to many applications. There is, of course, a substantial literature on tracking for various purposes including sports tracking [22, 11, 17, 20, 38], smart environments [12, 37], human figure tracking [15, 32, 35, 24], video surveillance [26], face tracking [15, 14, 28], among many others.

As Intille and Bobick [11] point out, low spatial resolution, non-rigid, and deformable objects such as football players or hockey players in broadcast video sequences limit the choice of techniques applicable. In fact, Intille and Bobick avoid the modeling of football players and invent a non-model-based tracking method by extracting players' pixels and tracking them. Their system removes camera motions from the original video sequence taped by broadcast cameras, transforming the original sequence to the globally consistent football field map (i.e., world coordinates), and then performs the tracking of football players on the rectified sequence. Their tracking approach has, however, two drawbacks for visual tracking applications such as the tracking of hockey players. Principally, their tracking results depend highly on the accuracy of the rectification process and limit their tracking on the fixed background. Secondly, and more importantly, their non-model based tracking method does not incorporate useful properties of objects such as the color information and dynamics of their motions. Therefore, we choose to use a model-based approach for tracking hockey players and overcome these drawbacks.

2.2.1 Probabilistic models for tracking

Among many model-based tracking approaches, the probabilistic approach becomes attractive due to its systematic handling of uncertainty and ability to incorporate fusion of information. Given the state sequence $\mathbf{x}_{0:t} \in \mathbb{R}^{n_x}$ and observation history $\mathbf{y}_{0:t} \in \mathbb{R}^{n_y}$ from time 0 to time t where n_x and n_y denotes the dimension of $\mathbf{x}$ and $\mathbf{y}$, a prior probability density, $p(\mathbf{x}_t|\mathbf{y}_{0:t-1})$, propagates the past state into future before new observation is made. Given the prior, a posterior distribution, $p(\mathbf{x}_t|\mathbf{y}_{0:t})$, can be estimated for $\mathbf{x}_t$ based on the past and current observations, which is also known as the process of "filtering". For different applications, the state sequence $\mathbf{x}_{0:t}$ and observation history $\mathbf{y}_{0:t}$ can represent different entities. However, regardless of applications, the general state-space model for the object tracking problems can be represented as a state transition (i.e., the system dynamics) and state measurement model (i.e., the system observation):

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}): \quad \mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{v}_{t-1})$$
$$p(\mathbf{y}_t|\mathbf{x}_t): \quad \mathbf{y}_t = \mathbf{h}(\mathbf{u}_t, \mathbf{x}_t, \mathbf{w}_t)$$

where, in this case, $\mathbf{v}_t \in \mathbb{R}^{n_v}$ and $\mathbf{w}_t \in \mathbb{R}^{n_w}$ are the process noise and observation noise respectively. $\mathbf{u}_t \in \mathbb{R}^{n_u}$ denotes the input observations. Under a circumstance which two mappings $\mathbf{f}: \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \mapsto \mathbb{R}^{n_x}$ and $\mathbf{f}: \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \mapsto \mathbb{R}^{n_x}$ are linear and Gaussian, the posterior density, $p(\mathbf{x}_t|\mathbf{y}_{0:t})$, has an analytic solution obtained by the Kalman filter [27]. Unfortunately, tracking objects in real world including hockey players rarely satisfy Gaussian assumptions required by the Kalman filter because background clutter may resemble a part of foreground features, which makes the density for $\mathbf{x}_t$ to be multi-modal and non-Gaussian, and the system dynamics and the system observation are usually highly non-linear.

CONDENSATION [13], also known as Temporal Bayesian filtering, is one of the most successful object tracking approaches for non-linear and/or non-Gaussian reality. Following the pioneering work of CONDENSATION, the vision community has drawn a considerable attention to techniques based on Monte Carlo simulations whose idea is about selecting statistic random samples (i.e., particles) to estimate the posteriors. Recently, sequential Monte Carlo methods [4, 18, 25, 36, 23, 19] have become

popular and catch our attention due to their flexibility, easy implementation, and wide applicability.

2.2.2 Learning models

For modeling complex and dynamic objects, learning the appearance and motion of such objects is common [15, 32, 35, 24]. One of the most successful probabilistic tracking approaches with learned models is the system developed by Toyama and Black [35]. They apply a new examplar-based tracking method with probabilistic mechanisms that enable fusion of information. Their learning algorithm, referred as "Metric Mixture" approach, is about learning complex object models based on exemplars that represent different configurations of objects as probabilistic mixture distributions. With valuable properties of their "Metric Mixture" approach such as the use of metrics without a vector space embedding, incorporation of a noise model from the training data, and avoidance of probabilistic pixelwise independence, they demonstrate the effectiveness and robustness of their probabilistic tracking method for the tracking of walking people and mouth tracking. However, their approach requires the fixed background for the automatic generation of exemplars and therefore it is not applicable for the tracking of hockey players with a constantly changing background. And more importantly, Bayesian probabilistic method with learned dynamic models are not easily extended to the multiple object tracking since learning is required offline for a model of each object.

2.2.3 Color-based tracking methods

Instead of learning the parameters of an object, the idea of using a color model becomes our interest since color-based tracking methods have been proved robust, versatile and computationally efficient [2, 23, 25]. Comaniciu *et. al* [2] introduce the real-time color-based tracking system for non-rigid objects using the mean shift procedure. Given the global reference color models, the best target candidate is selected based on the statistical similarity measure of a metric expressed with the Bhattacharyya coefficient. Excellent tracking results on complex scenes are presented by their method. However, their method also reveals a weakness of their deterministic search that fails if

the object is occluded for a while or the background has similar colors. For improved handling of such situations, Nummiaro *et. al* [23] and Peréz *et. al* [25] introduce the same idea of using color-based model tracking but embedded in the framework of sequential Monte Carlo methods. The main difference between their methods is the color space they use for color models. While Nummiaro *et. al* [23] uses the RGB color space for their color models, Peréz *et. al* [25] uses the Hue-Saturation-Value (HSV) color space for their color models. Since the HSV color space is known to be insensitive to illumination changes, we choose to implement the system developed by Peréz *et. al* [25], and add an improvement and extension to their method for realizing the tracking of hockey players.

2.3 Summary

In our literature survey, we do not find any hockey annotation systems developed in the past. There are many annotation systems developed in other sports including soccer, basketball, and football. The framework of our annotation system is based on that of the football annotation system developed by Intille and Bobick [11]. However, we use totally different approaches for both the rectification of the original sequence to the globally consistent map and the object tracking. While the attributes of the object such as non-rigidity, deformable shape, and low-resolution, limit the choice of tracking techniques applicable, we implement an improved and extended version of the color-based sequential Monte Carlo tracker developed by Peréz *et. al* [25]. The following chapters explain in detail the major components of our annotation system and show successful performance results.

Chapter 3

Automatic Computation of Homography

3.1 Introduction

This chapter presents a major component of our hockey annotation system: how to capture camera motions and transform the original sequence in broadcast video to the globally consistent map of the hockey rink. We first introduce the theoretical background of a planar projective transformation (i.e., homography) [9, 16, 34, 39] and a detailed framework of the Kanade-Lucas-Tomasi (hereafter KLT) tracking system [1, 30, 31, 33]. Our algorithm, also described later in this chapter, for automatically computing homographies uses the KLT system to track features over a sequence of frames and RANSAC [6] to gain reliable features (i.e., inliers) to compute a homography between images by the normalized direct linear transformation. The homography is then refined by iteratively discarding outliers based on the symmetric transfer error on each correspondence. Our model fitting algorithm reduces projection errors of the homography by matching a projected image to our model of the globally consistent rink map. At the end of the chapter, our rectification results demonstrate the robustness of our algorithm by successfully eliminating camera motions in images, namely panning, tilting, and zooming.

3.2 Notation

Vectors and matrices are represented by bold-face letters where bold lower case letters, such as $\mathbf{v}$, are vectors and bold upper case letters, such as $\mathbf{M}$, are matrices. We write $\mathbf{v} = (x, y)^\top$, which means that both sides of this equation represent column vectors. Vectors are considered as being column vectors unless explicitly transposed. Therefore, $\mathbf{v}^\top$ is a row vector. Vectors are multiplied as if they were matrices. In particular, for two vectors $\mathbf{u}$ and $\mathbf{v}$, the product $\mathbf{u}^\top\mathbf{v}$ represents the inner product, whereas $\mathbf{u}\mathbf{v}^\top$ is a matrix. The cross product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is expressed as $\mathbf{u} \times \mathbf{v}$. The norm of a vector, $\|\mathbf{v}\|$, is the Euclidean length of the vector, which is obtained by the square root of the sum of squares of its entries.

3.3 Homography

Within this section, the theoretical background of homography (also known as a plane projective transformation, or collineation) is described. The definition of homography (or more generally *projectivity*) is given in [9] as an invertible mapping of points and lines on the projective plane $\mathbb{P}^2$:

DEFINITION 1 *A projectivity is an invertible mapping $h : \mathbb{P}^2 \rightarrow \mathbb{P}^2$, such that three points $\mathbf{x_1}$, $\mathbf{x_2}$ and $\mathbf{x_3}$ lie on the same line if and only if mapped points $h(\mathbf{x_1})$, $h(\mathbf{x_2})$ and $h(\mathbf{x_3})$ also lie on the same line.*

This gives homography two useful properties. For a stationary camera with its fixed center of projection, it does not depend on the scene structure (i.e., depth of the scene points) and it applies even if the camera ”pans and zooms”, which means to change the focal length of the camera while it is rotating about its center. With these properties, we apply homography under the circumstance which the camera pans, tilts, zooms and rotates about its center. Given a sequence of images acquired by such a camera, the objective is to specify a point-to-point planar homography map in order to remove the camera motion in images. Although, in our case, the camera center moves by a negligible amount due to the small offset from the rotational center of the camera, the distance between the scene and the camera is much bigger than the amount so that we

can still assume that we have a fixed camera center.

3.3.1 Representation of Homography

We use homogeneous representation for a point $\mathbf{x} = (x, y, w)^\top$, which is a 3-vector, representing a point $(x/w, y/w)^\top$ in Euclidean 2-space $\mathbb{R}^2$. As homogeneous vectors, points are also elements of the projective space $\mathbb{P}^2$. It is helpful to consider the inhomogeneous coordinates of a pair of matching points in the world and image plane as $(x/w, y/w)^\top$ and $(x'/w', y'/w')^\top$, because points are measured in the inhomogeneous coordinates directly from the world plane. According to [39], a homography is a linear transformation of $\mathbb{P}^2$, which is expressed in inhomogeneous form as:

$$x'/w' = \frac{Ax + By + C}{Px + Qy + R} \quad , \quad y'/w' = \frac{Dx + Ey + F}{Px + Qy + R} \tag{3.1}$$

where we define vectors $\mathbf{x}$ and $\mathbf{x}'$ in homogeneous form, and a transformation matrix $\mathbf{M}$ as:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ w \end{pmatrix} \qquad \mathbf{x}' = \begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} \qquad \mathbf{M} = \begin{bmatrix} A & B & C \\ D & E & F \\ P & Q & R \end{bmatrix}$$

where we have a pair of 2D point correspondences, $\mathbf{x} \leftrightarrow \mathbf{x}'$. Normally we choose the scale factor w in such a way that x/w and y/w have order of 1, so that we can avoid numerical instability.

Now, Eq. (3.1) can be written as:

$$\mathbf{x}' = c\mathbf{M}\mathbf{x} \tag{3.2}$$

where c is an arbitrary nonzero constant. Homographies and points are defined up to a nonzero scalar c, and thus we have 8 degrees of freedom for homography. Often we take $R = 1$ and the scale factor $w = 1$. Eq. (3.2) can now be written simply as:

$$\mathbf{x}' = \mathbf{H}\mathbf{x}$$

where $\mathbf{H}$ is 3×3 matrix called homography. Every correspondence ($\mathbf{x}$, $\mathbf{x}'$) gives two equations (3.1). Therefore, computing a homography requires at least four correspondences.

3.3.2 Homography Computation

The Direct Linear Transformation algorithm

Given a set of n 2D to 2D point correspondences, $\mathbf{x}_i \leftrightarrow \mathbf{x'}_i$ ($n \geq 4$ and $i = 1 \ldots n$), we can determine homography, $\mathbf{H}$, by a simple linear algorithm, the Direct Linear Transformation (DLT) algorithm, introduced in [9].

If the transformation is given by the equation $\mathbf{x'}_i = \mathbf{H}\mathbf{x}_i$, which involves homogeneous vectors, then the 3-vectors $\mathbf{x'}_i$ and $\mathbf{H}\mathbf{x}_i$ are not necessarily equal. They have the same direction but may differ in magnitude by a non-zero scalar factor. In order to enable a simple linear solution for $\mathbf{H}$ to be derived, the equation may simply be expressed in terms of the vector cross product as $\mathbf{x'}_i \times \mathbf{H}\mathbf{x}_i = \mathbf{0}$.

Then we may write:

$$\mathbf{H}\mathbf{x}_i = \begin{pmatrix} h_1 x_i + h_2 y_i + h_3 \\ h_4 x_i + h_5 y_i + h_6 \\ h_7 x_i + h_8 y_i + h_9 \end{pmatrix}$$

where we define 3-vectors $\mathbf{x}_i$ and $\mathbf{x'}_i$, and homography $\mathbf{H}$ as follows:

$$\mathbf{x}_i = \begin{pmatrix} x_i \\ y_i \\ w_i \end{pmatrix} \qquad \mathbf{x'}_i = \begin{pmatrix} x'_i \\ y'_i \\ w'_i \end{pmatrix} \qquad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{3.3}$$

where we choose $w_i = w'_i = 1$ so that (x_i, y_i) and (x'_i, y'_i) are the coordinates measured in the image. This avoids the case when one of the points, $\mathbf{x'}_i$, is the ideal point with $w'_i = 0$. If one of the points $\mathbf{x'}_i$ is the ideal point, then Eq. (3.5) below collapses into a single equation. Meanwhile, the cross product may be expressed as:

$$\mathbf{x'}_i \times \mathbf{H}\mathbf{x}_i = \begin{pmatrix} y'_i(h_7 x_i + h_8 y_i + h_9) - (h_4 x_i + h_5 y_i + h_6) \\ (h_1 x_i + h_2 y_i + h_3) - x'_i(h_7 x_i + h_8 y_i + h_9) \\ x'_i(h_4 x_i + h_5 y_i + h_6) - y'_i(h_1 x_i + h_2 y_i + h_3) \end{pmatrix} \tag{3.4}$$

Eq. (3.4) may be written in a simpler form:

$$\begin{bmatrix} 0 & 0 & 0 & -x_i & -y_i & -1 & y'_i x_i & y'_i y_i & y'_i \\ x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ -y'_i x_i & -y'_i y_i & -y'_i & x'_i x_i & x'_i y_i & x'_i & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{pmatrix} = \mathbf{0} \quad (3.5)$$

For simplicity, this can be written as:

$$\mathbf{T}_i \mathbf{h} = \mathbf{0} \quad (3.6)$$

where $\mathbf{T}_i$ becomes the 3×9 transformation matrix and $\mathbf{h}$ is the 9-vector made up of all the entries of $\mathbf{H}$ in Eq. (3.3). It is important to note that in Eq. (3.5), only two among three equations are linearly independent since the the third row is obtained, up to scale, and can be omitted. Without the equation in the third row, Eq. (3.5) becomes:

$$\begin{bmatrix} 0 & 0 & 0 & -x_i & -y_i & -1 & y'_i x_i & y'_i y_i & y'_i \\ x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \end{bmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{pmatrix} = \mathbf{0} \quad (3.7)$$

where $\mathbf{T}_i$ can now become the 2×9 matrix in Eq. (3.6).

If a set of n 2D to 2D point correspondences, $\mathbf{x}_i \leftrightarrow \mathbf{x'}_i$ ($n \geq 4$ and $i = 1 \ldots n$), then the set of $2n - 1$ equations $\mathbf{Th} = \mathbf{0}$ can be derived from Eq. (3.7). Although the set of $2n$ equations can also be derived from Eq. (3.5) as well, in either case, $\mathbf{T}$ has

rank 8, a one dimensional null-space, and a solution. If the position of the points is exact, then the matrix $\mathbf{T}$ will still have rank 8 with a one dimensional null-space, and there is an exact solution for $\mathbf{h}$. However, this is hardly the case because, in practice, the measurement of image coordinates is inexact due to noise. If there is no exact solution, then we need an approximate solution for the system of $\mathbf{T}_i\mathbf{h} = \mathbf{0}$. Although $\mathbf{h} = \mathbf{0}$ is clearly a solution, that is not what we want. In order to avoid the zero solution, we can add a constraint of the norm, $\|\mathbf{h}\| = 1$. Since $\mathbf{H}$ is only defined up to scale, the value of the norm can be any number except 0.

Now, the problem naturally becomes the minimization of the norm $\|\mathbf{T}\mathbf{h}\|$. As shown in detail in [9], the solution is the (unit) eigenvector of $\mathbf{T}^\top\mathbf{T}$ with least eigenvalue, which is equivalent to the unit singular vector corresponding to the smallest singular value of $\mathbf{T}$. The summary of this DLT algorithm is shown in Algorithm 1.

Algorithm

Given a set of n 2D to 2D point correspondences, $\mathbf{x}_i \leftrightarrow \mathbf{x'}_i$ ($n \geq 4$ and $i = 1 \ldots n$), determine the 2D homography $\mathbf{H}$ such that $\mathbf{x}_i = \mathbf{H}\mathbf{x'}_i$.

(1) **Derivation of T_i:**
For each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x'}_i$, compute the matrix $\mathbf{T}_i$ from either Eq. (3.5) or Eq. (3.7).

(2) **$\mathbf{T}_i \rightarrow \mathbf{T}$:**
Assemble the n 2×9 matrices $\mathbf{T}_i$ into a single $2n \times 9$ matrix $\mathbf{T}$.

(3) **SVD:**
Perform SVD on $\mathbf{T}$ and obtain the unit singular vector corresponding to the smallest singular value, which is the solution, the 9-vector $\mathbf{h}$. That is, if $\mathbf{T} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ with $\mathbf{D}$ diagonal with positive diagonal entries, arranged in descending order down the diagonal, then $\mathbf{h}$ is the last column of $\mathbf{V}$.

(4) **$\mathbf{h} \rightarrow \mathbf{H}$:**
Obtain the matrix $\mathbf{H}$ by rearranging all entries of $\mathbf{h}$ as in Eq. (3.3).

Algorithm 1: **The DLT algorithm for 2D homographies excerpted from Hartley and Zisserman [9]**

The Normalized Direct Linear Transformation algorithm

Since the DLT algorithm is not invariant for the correspondences in different image coordinate systems, Hartley and Zisserman [9] develop the normalized DLT algorithm that works under different image coordinate systems. In our algorithm described later in Section 3.6, we use the normalized DLT algorithm.

There are two benefits of data normalization before carrying out the DLT algorithm. One is that normalizing point correspondences into the same coordinate system gives a better solution because the algebraic minimization takes place in a fixed canonical frame rather than an arbitrary one. As a result, normalization determines the errors to be minimized, so that correspondences are measured in the image coordinate. This effect is related to the condition of the system of the DLT equations. The condition is represented as the ratio of d_1/d_{n-1} of the first to the second last singular value of the matrix $\mathbf{T}$, and it should be made small for $\mathbf{T}$ to be well-conditioned [8, 9]. The other is that normalization process makes an algorithm invariant to similarity transformations of the image where there are translation and scaling changes of image coordinates. While the complete description of the normalized DLT algorithm is found in [9], the resulting algorithm is also summarized in Algorithm 2 and the implementation result is shown in Figure 3.1.

3.4 KLT: Kanade Lucas Tomasi tracker

In this section, we present a well-known feature tracker, called KLT [1, 30, 31, 33]. Although the objective of KLT is to track features, their approach is unique in such a way that, instead of tracking single pixels that individually represent a certain feature, they track “windows” of pixels that contain sufficient texture. We can summarize the framework of KLT by explaining models of image motion, how to compute image motion based on its models, and the process of selecting features that KLT can track well.

Algorithm

Given a set of n 2D to 2D point correspondences, $\mathbf{x}_i \leftrightarrow \mathbf{x'}_i$ ($n \geq 4$ and $i = 1 \ldots n$), determine the 2D homography $\mathbf{H}$ such that $\mathbf{x}_i = \mathbf{H}\mathbf{x'}_i$.

(1) **Normalization of $\mathbf{x}_i$:**
Compute a similarity transformation $\mathbf{S}$ that transforms points $\mathbf{x}_i$ to a new set of points $\widetilde{\mathbf{x}}_i$. The transformation consists of the following translation and scaling:

- **Translation**:
  the centroid of $\mathbf{x}_i$ is translated to the origin $(0, 0)^\top$
- **Scaling**:
  $\mathbf{x}_i$ are scaled so that the average distance from the origin is $\sqrt{2}$

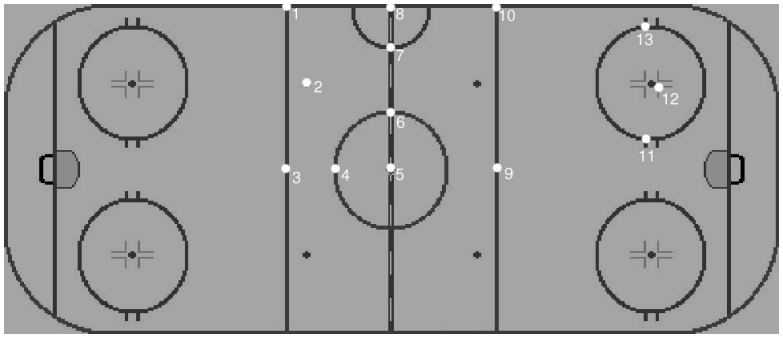
(2) **Normalization of $\mathbf{x'}_i$:**
Compute a similarity transformation $\mathbf{S'}$ for the points $\mathbf{x'}_i$ in the second image independently from the first one, and transform $\mathbf{x'}_i$ to $\widetilde{\mathbf{x}}'_i$.

(3) **DLT:**
Perform the DLT algorithm on $\widetilde{\mathbf{x}}_i \leftrightarrow \widetilde{\mathbf{x}}'_i$ to obtain a homography $\widetilde{\mathbf{H}}$.

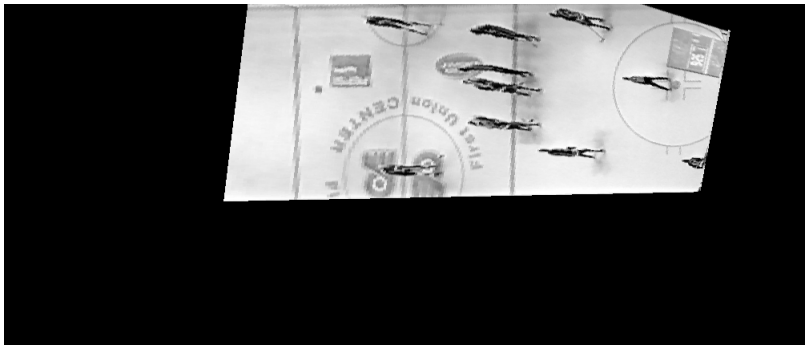
(4) **Denormalization:**
$\mathbf{H} = \mathbf{S'}^{-1}\widetilde{\mathbf{H}}\mathbf{S}$

Algorithm 2: **The normalized DLT algorithm for 2D homographies excerpted from Hartley and Zisserman [9]**

(a) The original image



(b) The correspondences on the rink map



(c) The transformation result

Figure 3.1: **Homography Transformation**

This is the demonstration of Algorithm 2 on hockey data. (a) is the original image (320×240) to be transformed. Those points beside the numeric number are the manually selected points that are corresponding to those on the rink in (b). The correspondences are paired up by the numeric number. (c) is the result (1000×425) of the transformation by Algorithm 2, which shows a successful transformation of homography given a set of manually selected 13 point correspondences.

3.4.1 Modeling Image Motion

The model of image motion which KLT is based upon plays a crucial role of how good the quality of tracking is. Tomasi and Kanade [33] formalize the complex pattern of image intensities by the following property:

$$I(x + \xi(x, y, t, \tau), y + \eta(x, y, t, \tau), t + \tau) = I(x, y, t) \tag{3.8}$$

where the space variables, x and y, and the time variable, t, are discrete and suitably bounded. In other words, Eq. (3.8) represents the correlation of images taken within short time intervals, and captures the same scene from slightly different view points. A later image taken at time $t + \tau$ differs from an image taken at time t by the amount of motion depicted by $\delta = (\xi, \eta)^\top$, which is also referred as the "displacement" of the point at $\mathbf{x} = (x, y)^\top$ between time instants, t and $t + \tau$.

It is known, however, that the property of the Eq. (3.8) does not hold well even in a static environment under constant lighting. This is due to frequent changes of points at occluding boundaries (e.g. disappearance and reappearance of points) and the photometric changes over the appearance of a visible surface when reflectivity is a function of the viewpoint. Those difficulties are reduced at surface markings that are away from occluding boundaries where the invariant of Eq. (3.8) is by and large satisfied. At these locations, it is still difficult to track a single pixel due to abrupt changes of the image intensity with x and y, and confusing adjacent pixels with noise.

In order to overcome these problems, it naturally makes sense to track "windows" of pixels instead of a single pixel. Considering the fact that the amount of motion, δ, is a function with respect to the image position $\mathbf{x}$, for given time t and τ, it is still a constant function of $\mathbf{x}$. Since there are different displacements within the same window, a better description of motion needs to be defined that could associate a set of different velocities at different points within the window. Shi and Tomashi [30, 31] introduce the *affine motion* model, which compromises well between simplicity and flexibility, expressed as follows:

$$\delta = \mathbf{D}\mathbf{x} + \mathbf{d}$$

where

$$\mathbf{D} = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \qquad \mathbf{d} = \begin{pmatrix} d_x \\ d_y \end{pmatrix}$$

$\mathbf{D}$ is a deformation matrix, and $\mathbf{d}$ is the translation of the feature window center. If the image coordinates $\mathbf{x}$ are measured with respect to the center of the window, then a point $\mathbf{x}$ in the first image moves to point $\mathbf{Ax} + \mathbf{d}$ in the second image J, such that:

$$\mathbf{A} = \mathbf{1} + \mathbf{D}$$

where $\mathbf{1}$ is the 2×2 identity matrix. Now, the model can be simply written as:

$$J(\mathbf{Ax} + \mathbf{d}) = I(\mathbf{x}) \tag{3.9}$$

Between two given images, I and J, tracking means to solve Eq. (3.9) for the deformation matrix $\mathbf{D}$ and displacement vector $\mathbf{d}$. Tomasi and Kanade [33] discover that, for tracking, smaller windows produce the better result because they are less likely to straddle at a depth discontinuity. However, the deformation matrix $\mathbf{D}$ becomes harder to estimate when the size of the window is small where the variations of motion are small and less reliable. As a consequence, a simpler model of motion, the *pure translation* model, is introduced in [30, 31, 33]:

$$J(\mathbf{x} + \mathbf{d}) = I(\mathbf{x}) \tag{3.10}$$

where the deformation matrix $\mathbf{D}$ is assumed to be zero. Here, tracking means simply to determine two parameters of the displacement vector $\mathbf{d}$ of the feature window center. The system of KLT, which uses small windows for tracking, applies the *pure translation* model under the assumption which the inter-frame displacement is sufficiently small with respect to the texture fluctuations within the window. In our implementation, the size of the feature window is set to 7×7.

3.4.2 Computing Image Motion

With the *pure translation* model of motion, we can now derive and compute the equation to solve for tracking. In Appendix C of [1], Birchfield describes the derivation of the symmetric KLT tracking equation that is originally proposed by Tomasi. As it

is described in the previous subsection 3.4.1, tracking is to solve Eq. (3.10) for the displacement vector $\mathbf{d} = (d_x, d_y)^\top$, which minimizes the following dissimilarity ϵ of two feature windows in given images, I and J:

$$\epsilon = \int\int_W [J(\mathrm{x} + \frac{\mathbf{d}}{2}) - I(\mathbf{x} - \frac{\mathbf{d}}{2})]^2 w(\mathbf{x})\, d\mathbf{x}$$

where w is a weighting function. Simply, w could be set to 1 or alternatively be a Gaussian-like function to emphasize the central area of the window. In our implementation of KLT, we set $w = 1$.

The truncated Taylor series expansion of an image J about a point $\mathbf{p} = (p_x, p_y)^\top$ gives the following linear term:

$$J(\boldsymbol{\xi}) \approx J(\mathbf{p}) + (\xi_x - p_x)\frac{\partial J}{\partial x}(\mathbf{p}) + (\xi_y - p_y)\frac{\partial J}{\partial y}(\mathbf{p}) \tag{3.11}$$

where $\boldsymbol{\xi} = (\xi_x, \xi_y)^\top$.

According to the derivation in [30, 31], if we let $\mathbf{x} = \mathbf{p}$ and $\mathbf{x} \pm \frac{\mathbf{d}}{2} = \xi$, then Eq. (3.11) may be written as:

$$J(\mathbf{x} + \frac{\mathbf{d}}{2}) \approx J(\mathbf{x}) + \frac{d_x}{2}\frac{\partial J}{\partial x}(\mathbf{x}) + \frac{d_y}{2}\frac{\partial J}{\partial y}(\mathbf{x}) \tag{3.12}$$

similarly for an image I:

$$I(\mathbf{x} - \frac{\mathbf{d}}{2}) \approx I(\mathbf{x}) - \frac{d_x}{2}\frac{\partial I}{\partial x}(\mathbf{x}) - \frac{d_y}{2}\frac{\partial I}{\partial y}(\mathbf{x}) \tag{3.13}$$

with Eq. (3.12) and (3.13):

$$\begin{aligned}\frac{\partial \epsilon}{\partial \mathbf{d}} &= 2\int\int_W [J(\mathbf{x} + \frac{\mathbf{d}}{2}) - I(\mathbf{x} - \frac{\mathbf{d}}{2})][\frac{\partial J(\mathbf{x} + \frac{\mathbf{d}}{2})}{\partial \mathbf{d}} - \frac{\partial I(\mathbf{x} - \frac{\mathbf{d}}{2})}{\partial \mathbf{d}}] w(\mathbf{x})\, d\mathbf{x} \\ &\approx \int\int_W [J(\mathbf{x}) + I(\mathbf{x}) + \frac{1}{2}\mathbf{g}(\mathbf{x})^\top \mathbf{d}]\mathbf{g}(\mathbf{x}) w(\mathbf{x})\, d\mathbf{x}\end{aligned} \tag{3.14}$$

where the image gradient vector $\mathbf{g}$ is defined as:

$$\mathbf{g}(\mathbf{x}) = \begin{pmatrix} \frac{\partial}{\partial x}(I + J) \\ \frac{\partial}{\partial y}(I + J) \end{pmatrix}$$

In order to determine the displacement vector $\mathbf{d}$, Eq. (3.14) is simply set to be zero:

$$\frac{\partial \epsilon}{\partial \mathbf{d}} \approx \int\int_W [J(\mathbf{x}) + I(\mathbf{x}) + \frac{1}{2}\mathbf{g}(\mathbf{x})^\top \mathbf{d}]\mathbf{g}(\mathbf{x}) w(\mathbf{x})\, d\mathbf{x} = 0$$

terms may be rearranged to perform one iteration of Newton-Raphson minimization as follows:

$$\begin{aligned}\int\int_W [J(\mathbf{x}) + I(\mathbf{x})]\mathbf{g}(\mathbf{x})w(\mathbf{x})\,d\mathbf{x} &= -\int\int_W \frac{1}{2}\mathbf{g}(\mathbf{x})^\top \mathbf{d}\mathbf{g}(\mathbf{x})w(\mathbf{x})\,d\mathbf{x} \\ &= -\frac{1}{2}[\int\int_W \mathbf{g}(\mathbf{x})\mathbf{g}(\mathbf{x})^\top w(\mathbf{x})\,d\mathbf{x}]\mathbf{d}\end{aligned} \tag{3.15}$$

for simplicity, Eq. (3.15) is expressed as:

$$\mathbf{Z}\mathbf{d} = \mathbf{e} \tag{3.16}$$

where the 2×2 matrix $\mathbf{Z}$ and the error vector $\mathbf{e}$ are defined as:

$$\begin{aligned}\mathbf{Z} &= \int\int_W \mathbf{g}(\mathbf{x})\mathbf{g}(\mathbf{x})^\top w(\mathbf{x})\,d\mathbf{x} \\ \mathbf{e} &= 2\int\int_W [I(\mathbf{x}) - J(\mathbf{x})]\mathbf{g}(\mathbf{x})w(\mathbf{x})\,d\mathbf{x}\end{aligned}$$

Tracking is complete after repeatedly solving Eq. (3.16) and shifting feature windows by the displacement $\mathbf{d}$.

3.4.3 Feature Selection Criterion

For successful tracking, ”good” features are required in order for Eq. (3.16) to be numerically well-conditioned. Tomasi and Kanade [33] define the quality of the feature based on the tracking method they use. That is, ”good” features are ones that can be tracked well by the method of tracking. In terms of the notion of KLT tracker, ”good” features are ”windows” of pixels that can be tracked well.

In order to select ”good” feature windows, two requirements need to be satisfied. The 2×2 coefficient matrix $\mathbf{Z}$ in the system of Eq. (3.16) must meet the image noise criterion and be well-conditioned. In other words, the noise requirement has two eigenvalues of $\mathbf{Z}$ to be large, and the conditioning requirement disallows them to be differed by several orders of magnitude.

According to [33], in practice, the matrix $\mathbf{Z}$ is well-conditioned if a smaller eigenvalue of $\mathbf{Z}$ is sufficiently large to meet the noise criterion. Since the intensity variations in a window are bounded by the maximum allowable pixel value, the greater eigenvalue cannot be large. Two large eigenvalues imply a ”good” feature, such as

corners, salt-and pepper texture or any other pattern that can be tracked reliably. Consequently, the following criterion can be formulated:

$$\min(\lambda_1, \lambda_2) > \lambda_{thresh} \tag{3.17}$$

where λ_1 and λ_2 are two eigenvalues of Z and λ_{thresh} is a predefined threshold.

The lower bound of λ_{thresh} is determined by measuring the eigenvalues for images of a region under approximately uniform brightness, while the upper bound is determined by selecting a set of strong features, such as corners or highly textured region. In practice, the two bounds are found to be separate enough that choosing the value of λ_{thresh} between two bounds is trivial. Based on our experiments, we set $\lambda_{thresh} = 1$ to discard feature windows that have a smaller eigenvalue less than 1. The chosen value works well because not only does it keep reliable features but also discards unreliable features that are hard for the KLT tracker to track. After all, the feature selection criterion of KLT is optimal by construction because of its foundation based on how the tracker works and features that correspond to the points in the real world [30, 31, 33]. The algorithm of KLT is summarized in Algorithm 3.

In Figure 3.2, (c) is the image contains 761 KLT features to be successfully tracked and 439 features to be lost a track of from Frame 58 to Frame 62. The majority of features is detected and tracked on texture rich regions. As is shown in the figure, there are many features that are distributed uniformally on a feature-less region of the rink (i.e., the white surface of the rink). This raises the question of whether these features provide reliable tracking or not. In our experiments, we test our rectification system with various numbers of features ranging from 700 to 1200. We determine the number of features as 1200 because it provides the best result. In order to have features that are spatially well spread out over the entire region of the image, we need to have a large number of KLT feature extracted. Since the KLT tracker ranks features based on their trackability (i.e, how easy it is for the tracker to track), most of the features that are in a high rank are concentrated on texture-rich regions of the image such as players, a scoring board on a TV, or audience. Therefore, if we extract a small number of KLT features, then we do not gain many KLT features from the rink features such as logos, lines , and cirlces. For some frames, we don't need to extract 1200 features to have a spatially well distributed set of KLT features. However, for other frames

Algorithm

Given two images, $\{I(\mathbf{x}_i)\}_{i=1\cdots N}$ and $\{J(\mathbf{x_i}+\mathbf{d_i})\}_{i=1\cdots N}$ such that $\mathbf{x}_i = (x_i, y_i)^\top$ and $\mathbf{d}_i = (d_{i,x}, d_{i,y})^\top$, taken at near time instants, the objective is to determine the displacement vector $\{\mathbf{d}_i\}_{i=1\cdots N}$ for each of N feature windows.

(1) **Selection of features:**
The following steps select N best feature windows from an image I.

- **Computing image gradients**:
  - Place the center of the feature window in I and, at each location, compute image gradients in both x and y direction to determine the matrix $\mathbf{Z}$ in Eq. (3.16).
- **Apply the feature selection criterion**:
  - Compute trackability of each image pixel as the minimum of two eigenvalues of the matrix $\mathbf{Z}$.
  - Sort all the feature window based on trackability and discard ones that do not satisfy the minimum trackability in Eq.(3.17), or ones close by the better window.

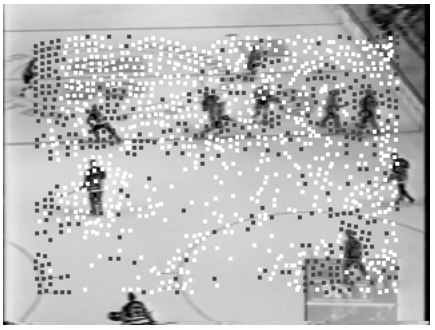
(2) **Tracking:**
Tracking means determining two parameters for the displacement vector $\{\mathbf{d}_i\}_{i=1\cdots N}$.

- **Computing image motion**:
  - Compute the error vector $\{\mathbf{e}_i\}_{i=1\cdots N}$ in Eq. (3.16) for N selected feature windows.
  - Solve Eq. (3.16) for the displacement vector $\{\mathbf{d}_i\}_{i=1\cdots N}$.
- **Shifting feature windows**:
  - Shift N feature windows in I by the displacement $\mathbf{d}$ and start the next search at the new locations in J.

Algorithm 3: **KLT**

(a) Frame 58

(b) Frame 62



(c) KLT features to be tracked or to be lost a track of from frame 58 to 62

Figure 3.2: **KLT tracking result**

The KLT tracker is used to track KLT features from the image in (a) to the one in (b). (c) is the result of the tracker. Lighter dots are features successfully being tracked and darker dots are failed to be tracked. These three images are in the size of 320×240.

with many players or a large audience in the scene, we need to extract 1200 features to gain a spatially well distributed set of KLT features. As a result, we choose to extract 1200 features to make our rectification system so that we can constantly gain a set of features well spread out on the entire region of the image. The answer to the previously addressed question is no. For some frames, we extract KLT features that are on uniform parts of the rink and they do not provide us reliable tracking. Since we need to extract 1200 features for always acquiring a spatially well distributed set of features, we determine to extract 1200 features and discard these features on uniform regions of the rink whenever we get them.

3.5 Further improvement

There are some conditions when KLT tracker works poorly. Those specific conditions are often due to rapid camera motions and occasional camera flashes. In order to make KLT tracker more robust to rapid motions of camera and drastic intensity changes by camera flashes, we introduce a method of "predicting" camera motions and detecting camera flashes by observing the amount of intensity changes in each frame.

3.5.1 Camera flash detection

As Chapter 1 introduces the camera flash as one of vision problems, it causes a drastic intensity change over all pixels in an image and is preferred to be avoided if necessary. Figure 3.3 shows the average intensity of each frame over 2300 frames. The vertical axis indicates the number of the intensity ranging from 0 to 255 where 0 indicates a dark pixel and 255 is a bright pixel, and the horizontal axis is the number of the frame. In the graph, there are several sudden spikes which indicate that there is a camera flash for that particular frame. One can also observe that each spike makes a jump that results in increasing the average intensity by at least 15.

With our observation of camera flashes, we formulate a simple flash detection method by taking the difference of the average intensity from two separate images.

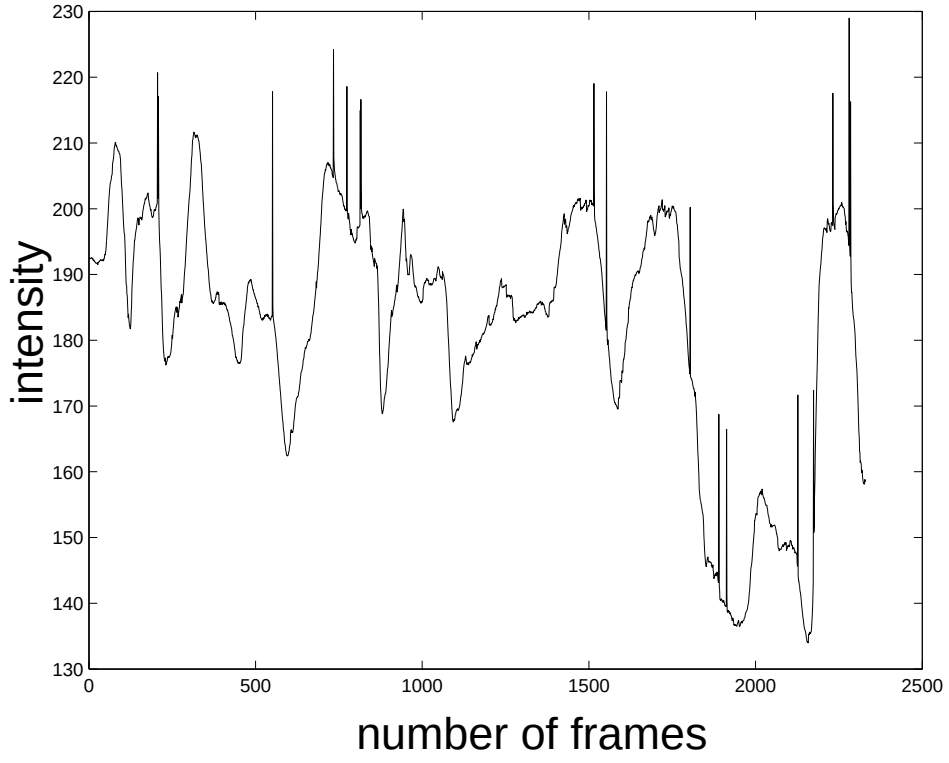


Figure 3.3: **the average intensities over 2300 frames**

Given two images, I and J, we compute the average intensity of these two images:

$$\bar{I} = \frac{\sum_{x=1}^{width} \sum_{y=1}^{height} I(x,y)}{width \times hight}$$
$$\bar{J} = \frac{\sum_{x=1}^{width} \sum_{y=1}^{height} J(x,y)}{width \times hight}$$

where $I(x,y)$ means the intensity of the pixel located at (x,y). Now we define a simple binary function Φ:

$$\Phi = \begin{cases} 0 & \text{if } |\bar{I} - \bar{J}| > 15, \\ 1 & \text{otherwise} \end{cases}$$

where if $\Phi = 0$, then a camera flash is detected. Although this detection scheme is quite simple, it works well for our purpose of avoiding occasional flashes.

3.5.2 Prediction

We can improve KLT feature tracker by *predicting* camera motion and make the tracker even better. We define the *prediction* by using the previous camera motion to estimate the current camera motion. For instance, given a frame-to-frame homography $\mathbf{H}_{1,2}$ that depicts the camera motion from Frame 1 to Frame 2, we use $\mathbf{H}_{1,2}$ as the estimation of $\mathbf{H}_{2,3}$ so that we apply KLT feature tracker on Frame 2'(i.e., Frame 2 transformed by the inverse of $\mathbf{H}_{1,2}$) and Frame 3 instead of Frame 2 and Frame 3. That is, we have the following assumption:

$$\mathbf{H}_{\mathrm{n,n-1}} \approx \mathbf{H}_{\mathrm{n+1,n}}$$

where we process every single frame without skipping any and $\mathbf{H}_{\mathrm{n-1,n}}$ means a homography from Frame n-1 to Frame n. This assumption holds if we don't skip many frames. Since we skip at most 5 frames to process data sampled at 30 frames per second, there is hardly any change in a motion of a camera within one sixth of a second and thus we could hold the assumption. This *prediction* step decreases the amount of motion between frames and helps KLT feature tracker to track on moving features. KLT feature tracker is, after all, a simple correlation tracker and thus, by controlling the amount of motion of moving objects, we can easily make the performance of the tracker better.

Figure 3.4 shows the number of KLT features successfully being tracked with and without *prediction*. The original number of KLT features chosen at each frame is 1200. We set the frame step as 4, that is, the original sequence is processed every fourth frame at a time. For example, Frame 35 means the 35th frame to be transformed from the original sequence. Therefore, data in Figure 3.4 has the record of 200 transformed frames computed from about 800 frames of the original sequence. In order to avoid any confusions, we refer to the frame number of the transformed sequence for the rest of this section. As it is shown in the figure, in the beginning of the sequence, the detection of many features shows a small camera motion between frames. After Frame 15, there are constant camera motions because the number of features found varies between 700 and 950. The most important remark on this graph is that, on Frame 35, there is a huge gap between the number of features found with and without prediction.

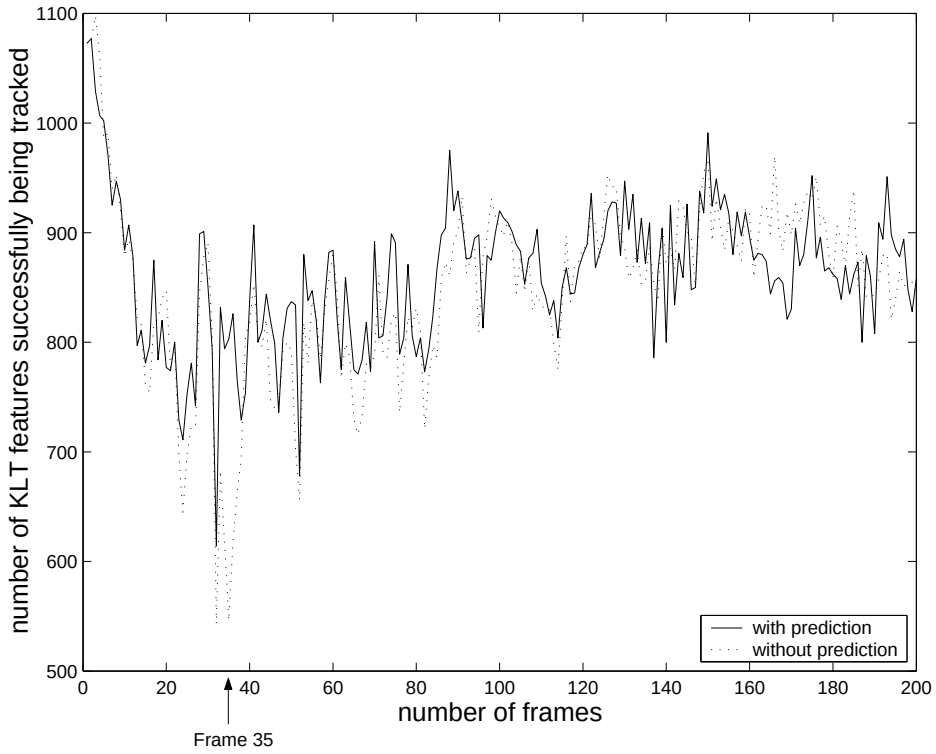
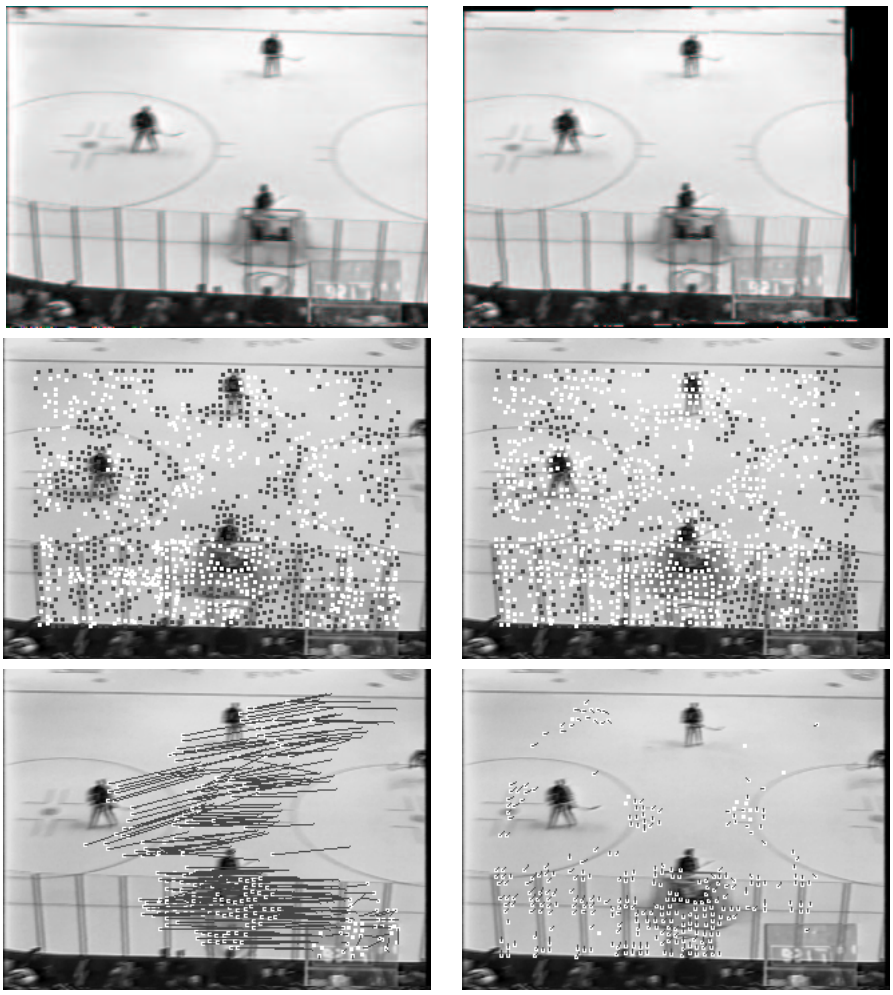


Figure 3.4: **KLT features successfully being tracked with and without prediction**

Figure 3.5 shows the effect of *prediction* and what happens in Frame 35 where there is a large camera motion. In the first row, the image in (b) is the same image as in (a) shifted toward left by *prediction*. This reduces the amount of motion to the next frame shown in the second row. (a) shows 537 KLT features (lighter dots) to be tracked. (b) has 805 KLT features to be tracked. Both (a) and (b) has 1200 KLT features in total to be selected by the tracker. By reducing the amount of motion between two frames, there are 268 more features to be successfully tracked with *prediction*. The images in the third row show clearly how much camera motions are reduced by displaying how long the trace of each feature is. The effect of *prediction* depends on the amount of camera motions between frames. Larger camera motions are, the more effective *prediction* is. We show that if there is a reasonable amount of camera motions, then *prediction* helps KLT tracker to track features in the image. The most important contribution of *prediction* is that it prevents large projection errors by reducing camera motions and helping KLT tracker to track features between frames so that a more reliable set of correspondences is obtained. Since we automate the pro-



(a) without *prediction* (b) with *prediction*

Figure 3.5: **KLT tracking result on Frame 35**

(a) is the result of KLT tracking without *prediction*. (b) is the result of KLT tracking with *prediction*. They are all the result of KLT tracking result on Frame 35. Each row shows the frames before KLT tracking, the frames with KLT features tracked or not tracked, and the final set of inliers with their trace to describe camera motions between frames. The images in the third row are the result obtained by Algorithm 5 explained later in Section 3.6. They are presented here only to show how large the difference of camera motions is between (a) and (b).

cess of gaining frame-to-frame homographies, it is crucial to prevent large projection errors by which the automatic process might fail. In general, by reducing camera motions between frames, *prediction* makes the tracker better or at least as good as the one without, and more importantly, prevents occasional large projection errors due to large camera motions.

3.6 Algorithm

This section explains our algorithm of automatically computing 2D homographies using KLT for tracking correspondences between images, RANSAC for eliminating unreliable features (i.e. outliers), and the normalized DLT algorithm for calculating homographies.

3.6.1 Acquisition of correspondences

For successful homography computation, it is crucial to have a reliable set of point correspondences that gives an accurate homography. KLT gives those correspondences automatically by extracting features and tracking them. That is, those features that are successfully tracked by KLT between images are ones that are corresponding each other. We can easily eliminate obvious outliers by discarding features that KLT loses a track of.

3.6.2 RANSAC: Elimination of outliers

Correspondences gained by KLT are yet imperfect to estimate a correct homography because they also include outliers. Though an initial set of correspondences selected by KLT contains a good proportion of correct matches, the RANSAC algorithm is used to eliminate even more of unreliable correspondences and obtain a better homography. In the RANSAC algorithm, we produce a putative set of correspondences by a homography based on a random set of four correspondences, and use the homography to eliminate outliers.

Sample Selection

In the process of selecting a random set of four correspondences, there are several issues to be dealt with, such as avoiding a degenerate homography and determining when to terminate sampling. In order to avoid choosing three collinear points to produce a degenerate homography, we implement distributed spatial sampling by ensuring that each sample is from a different region of the image. In our experiments, we divide a 320×240 image into four sub-regions of an equal size of 80×60, so that each point correspondence is sampled from a different sub-region. Once four point correspondences are sampled with a good spatial distribution, we calculate a homography based on those correspondences and use the homography to select an initial set of inliers. For inlier classification, we use the symmetric transfer error $d^2_{transfer}$, defined in [9]:

Let $\mathbf{x} \leftrightarrow \mathbf{x}'$ be the point correspondence and $\mathbf{H}$ be a homography such that $\mathbf{x}' = \mathbf{H}\mathbf{x}$, then

$$d^2_{transfer} = d(\mathbf{x}, \mathbf{H^{-1}}\mathbf{x}')^2 + d(\mathbf{x}', \mathbf{H}\mathbf{x})^2 \tag{3.18}$$

where $d(\mathbf{x}, \mathbf{H^{-1}}\mathbf{x}')$ represents the distance between $\mathbf{x}$ and $\mathbf{H^{-1}}\mathbf{x}'$. After the symmetric transfer error is estimated from each point correspondence, we then calculate the standard deviation of the sum of the symmetric errors from all correspondences, which is denoted by σ_{error} and defined as:

Suppose there are N point correspondences and each one of them has the symmetric transfer error $\{d^2_{transfer}\}_{i=1...N}$, then:

$$\sigma_{error} = \sqrt{\frac{\sum_{1 \le i \le N}(\{d^2_{transfer}\}_i - \mu)^2}{N-1}} \tag{3.19}$$

where μ is the mean represented as:

$$\mu = \frac{\sum_{1 \le i \le N}\{d^2_{transfer}\}_i}{N}$$

Now we can classify an outlier as any point $\mathbf{x}_i$ that satisfies the following condition:

$$\gamma(\mathbf{x}_i) = \begin{cases} 0 & \{d^2_{transfer}\}_i \ge \sqrt{5.99} * \sigma_{error} \quad \text{(outlier)} \\ 1 & \text{Otherwise} \quad \text{(inlier)} \end{cases} \tag{3.20}$$

where γ is a simple binary function that determines whether the point $\mathbf{x}_i$ is an outlier. The constant real number, $\sqrt{5.99}$, is user-defined, and chosen to make the criteria of

selecting outliers to be neither too tight nor too loose so that a reasonable number of inliers can be selected.

Adaptive algorithm to terminate sampling

Once we know how to sample four spatially distributed correspondences and classify inliers and outliers, we need to determine when to stop sampling so that we can avoid unnecessary computation. For that purpose, We implement an adaptive algorithm for determining the number of RANSAC samples, that is introduced in [9]:

Algorithm

(1) **Initialization:**
Set the number of selections $N = \infty$, the number of samples $S_n = 0$, the sample size $s = 4$ and a probability $p = .99$.

p is the probability that at least one of the random samples of s points is free from outliers.

(2) **Iteration:**
While $N > S_n$

- Select a sample of four point correspondences and count the number of inliers based on the criterion in Eq.(3.20)
- Set $\epsilon = 1 - (\text{number of inliers})/(\text{total number of points})$
- Set $N = \log(1-p)/\log(1-(1-\epsilon)^s)$
- Set $S_n = 1 + S_n$
- Repeat

Algorithm 4: **Algorithm for adaptively determining the number of RANSAC samples referred from Harltey and Zisserman [9]**

As it is shown in Algorithm 4, ϵ is the probability that any selected data point is an outlier. Therefore, the derivation of N with respect to ϵ, p and s is now straightforward from $(1-(1-\epsilon)^s) = 1-p$. This adaptive algorithm gives us a homography that produces the largest number of inliers by adaptively determining the termination of the algorithm and saving unnecessary computation. Please refer to [9] for more details, where Harley and Zisserman show that the algorithm works well in practice.

3.6.3 Further estimation: Selection of best inliers

Once the adaptive algorithm produces an initial set of putative correspondences determined by selecting four spatially distributed point correspondences and a homography based on those points, we can refine the set by eliminating points with a large amount of the symmetric transfer error in Eq.(3.18) and making a set of better inlying matches. The aim of this further estimation is, therefore, to obtain an improved estimate of a homography with better inliers. It is quite possible because the error is now based on a homography for randomly selected 100 point correspondences of the set, unlike the RANSAC algorithm where the error is based on a homography for only randomly selected four point correspondences.

The process of the further estimation is that at each iteration, we estimate a homography with a set of 100 randomly selected point correspondences that are considered to be inliers, classify a set of all correspondences based on our simple classifier in Eq.(3.20) and update a set of inliers. We repeat the process until the symmetric error of all the inlier becomes less than $\sqrt{5.99} * \sigma_{error}$. An important remark of this estimation process is that we always take an initial set of correspondences into account without eliminating any one of them so that we don't miss the opportunities of some outliers being re-designated as inlilers.

Our algorithm of automatically computing 2D homographies is summarized in Algorithm 5. Figure 3.6 shows the final set of inliers selected in the process of the further estimation. In (b), it is shown clearly that our algorithm eliminates outliers successfully by selecting features only from static objects (i.e., features on the rink) in the image. The white dots are features to be selected and a black lines represent the trace of the feature. In this case, a black line is not so visible because there is not much camera motions. Figure 3.7 shows how successfully our algorithm captures a variety of camera motions. As it is shown, a trace of features show how features move from one frame to the other.

Algorithm

(1) **Flash Detection:**
Apply our simple flash detector in Section 3.5.1 to avoid camera flashes. If a flash is detected in the image, skip that image and proceed to the next.

(2) **Prediction:**
Transform the current frame by using the previous homography and reduce camera motions.

(3) **KLT:**
Apply the KLT tracking system to gain an initial set of correspondences $\{C_i\}_{i=1...N}$ where N is the number of point correspondences.

(4) **RANSAC:**
Repeat T times, where T is determined by the adaptive algorithm, Algorithm 4.

- Choose four point correspondences with a good spatial distribution
- Estimate a homography $\mathbf{H}_{4pts}$
- Using $\mathbf{H}_{4pts}$, estimate the symmetric transfer error $\{d^2_{transfer}\}_{i=1...N}$ from Eq.(3.18) and the standard deviation of the error σ_{error} from Eq.(3.19)
- For classifying inliers, apply a binary function γ from Eq.(3.20) and count the number of inliers
- Repeat

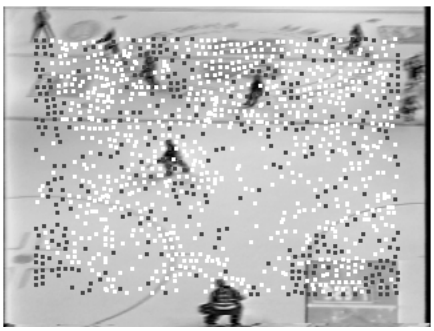
Choose the homography $\mathbf{H}_{4pts}$ with the largest number of inliers and update $\{C_i\}_{i=1...N}$.

(5) **Further estimation:**
Let $\{I_i\}_{i=1...I_N}$ be a set of inliers and set $I = C$ initially.
Repeat until $\{d^2_{transfer}\}_{i=1...N} < \sqrt{5.99} * \sigma_{error}$

- Perform the normalized DLT algorithm to estimate a homography $\mathbf{H}$ from a set of 100 randomly selected point correspondences considered to be inliers $\{I_i\}_{i=1...I_N}$
- Using $\mathbf{H}$, estimate $\{d^2_{transfer}\}_{i=1...N}$ and σ_{error} for all points in $\{C_i\}_{i=1...N}$
- Apply γ and update a set of inliers $\{I_i\}_{i=1...I_N}$
- Repeat

The final estimation is the homography $\mathbf{H}$ at the last iteration.

Algorithm 5: **Algorithm for the automatic homography computation**



(a) KLT features

(b) the best set of inliers selected in Algorithm 5

Figure 3.6: **Best inliers selected in Algorithm 5**

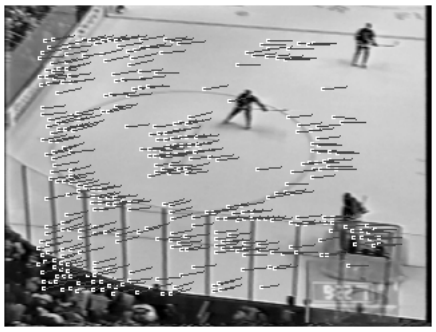
(a) is the image with KLT features both successfully tracked (light dots) and ones that are not successfully tracked (black dots). (b) is the result of Algorithm 5 after its successful outlier elimination. It is successful because all of the best inliers selected are on the rink and none of the features on moving objects is selected.

3.7 Model fitting

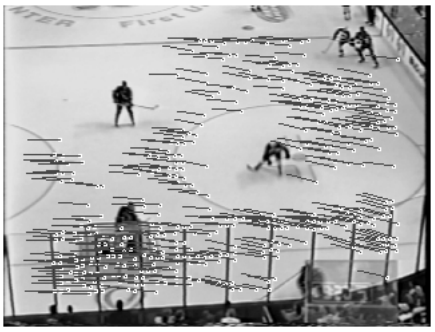
Although Algorithm 5 captures camera motions between frames reasonably well, there is still a difficult issue remained to be dealt with: we need to reduce the error accumulation of frame-to-frame homographies over time. As Intille [11] points out, it is still true that lens distortion and image processing limitations lead to imperfect image rectification process. However, even if we cannot get the perfect rectification result, we can still minimize the amount of error accumulated over time. That is, we make a use of the domain knowledge by using the rink features as the model.

3.7.1 Model

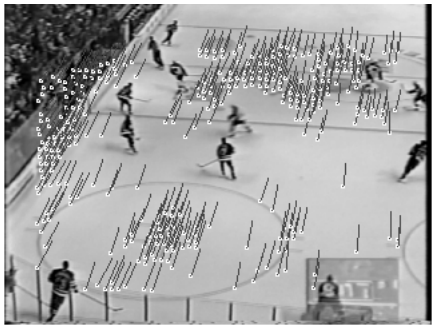
We implement model fitting to the result of homography transformation gained by Algorithm 5. Figure 3.8 shows our model of the rink. The rink dimensions and our model are strictly based on the official measurement presented in [10]. We define features on lines and circles of the rink and use them as our model. According to the official NHL rulebook [10], the measurement between two blue lines in the center of the rink varies by different stadiums. Since a difference of even 2 feet might cause a failure of our vision system, we need a careful selection of data for the implementation. In our experiment, we select data that is taped from stadiums that have the fixed measurement



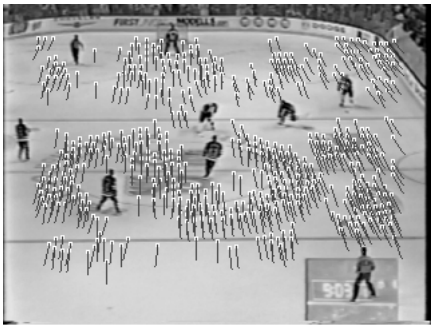
(a) camera panning toward right



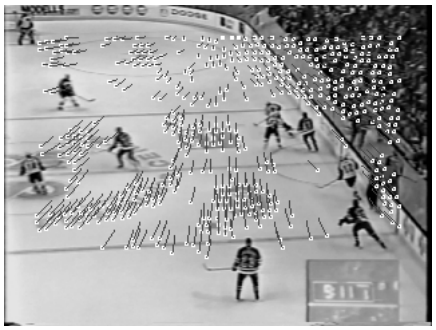
(b) camera panning toward left



(c) camera tilting up



(d) camera tilting down



(c) camera zooming in

(d) camera zooming out

Figure 3.7: **Different camera motions**

These images are the result of our algorithm that shows the extraction of different kinds of camera motions. The trace of features shows how each feature moves from the previous frame to the current frame. For zooming in or out, features move toward or away from the center of focus.

of 54 feet in-between blue lines.

With the model in Figure 3.8, we can now correct our projected image to fit the real dimensions of the rink and reduce projective distortions.

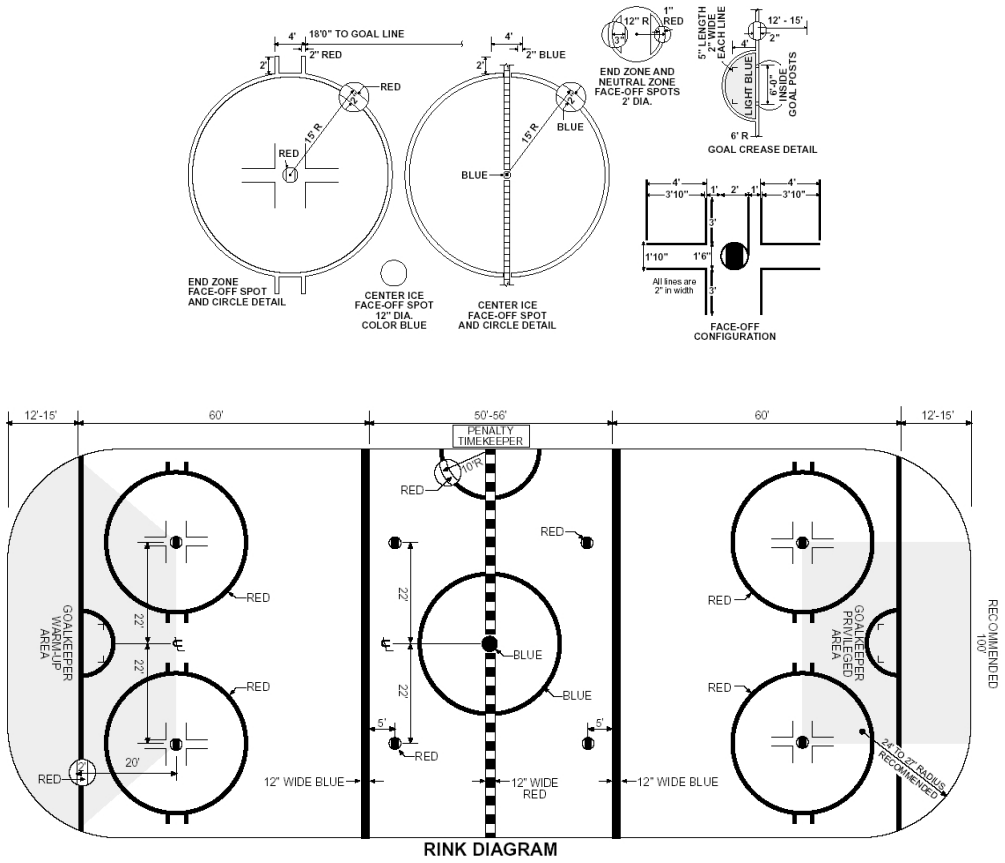
3.7.2 Edge search

This section describes how we fit projected images produced by Algorithm 5 to our model of the rink. In order to fit the projected images to the model, we perform a local search on each model point appeared within the region of each projected image. The local search is conducted to find the nearest edge pixel in the image. Figure 3.9 shows how to fit the projected image to our rink model.

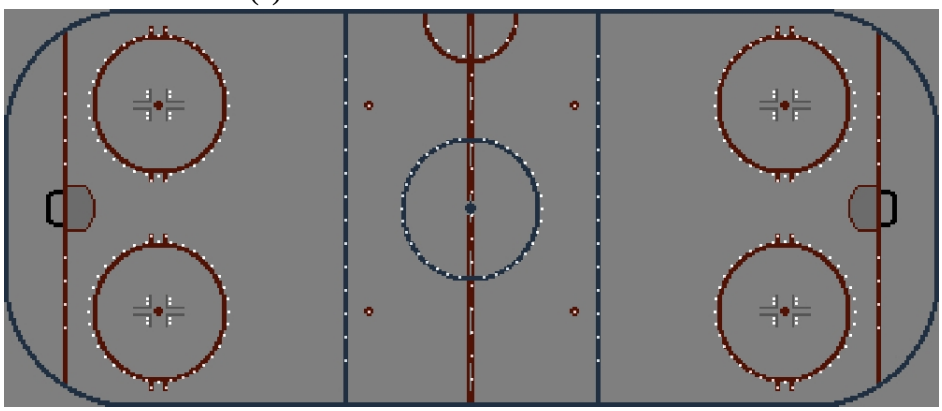
It is true that the projected point is the nearest edge pixel found from the local edge search performed on each model point and thus there is a possibility that the projected point may not be the true correspondence of the model point. This concern is true if the projected images are too distorted to match the model. However, in our case, it still works because Algorithm 5 gives us an approximate solution that is good enough to project the original image sequence close to the model of the rink. For edge detection, the search is performed locally only on high gradient regions in the original sequence where there are most likely edges. We do not perform the edge detection for the entire image region in order to save on the computational time. After the edge search is focused on high gradient regions, then edge orientation is considered to find a most likely edge pixel. Given an image, I, the image gradient vector $\mathbf{g}$ is represented as:

$$\mathbf{g}(\mathbf{x}) = \begin{pmatrix} \frac{\partial}{\partial x}(I) \\ \frac{\partial}{\partial y}(I) \end{pmatrix}$$

The gradient vector represents the orientation of the edge. The orientation varies on whether edges compose lines or circles. For circles, the orientation is tangent of the circle. For lines, it is perpendicular to the direction of the line. Figure 3.10 shows the orientation of two edges that form a thick line in the image. Since lines and circles of the hockey rink are not single edges but thick lines, they give two peaks of gradients. We compute the image gradient vector $\mathbf{g}$ of the original image because the projected image may not give accurate gradients due to resampling effects. Figure 3.11 shows how the edge search is conducted.



(a) Official dimensions of rink surface



(b) Model of the rink

Figure 3.8: **Model of the rink based on the official NHL measurement**

(a) has details of the official measurement of the rink. These images are from [10]. (b) is the model of the rink based on the NHL official measurement. The model of the rink is basically the set of points on circles and lines on the rink surface. There are 296 features in total: 178 features on four End-Zone circles, 4 features on center ice face-off spots around the center circle, and 114 features on lines.

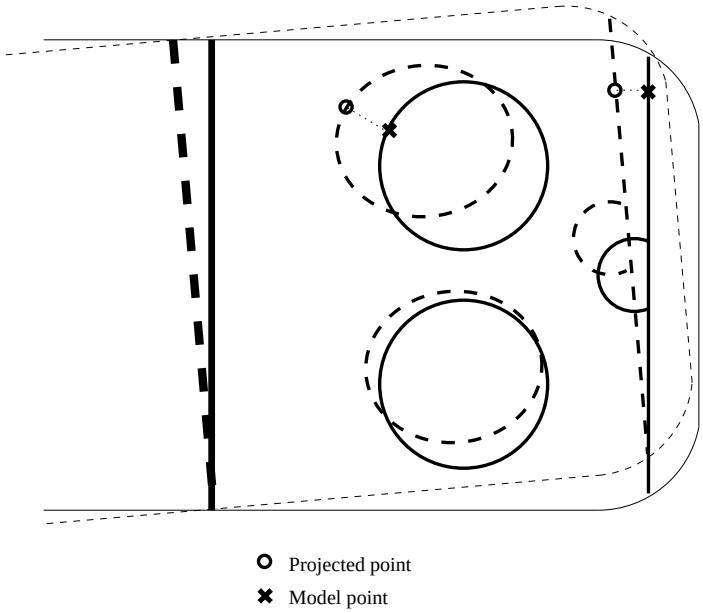


Figure 3.9: **Fitting a projected image to our model of the rink**

Dotted lines represent the projected image and solid lines represent the model. Although only two examples of matching a projected point to a model point are presented in this image, we perform a local search for finding the nearest edge pixel (i.e., a projected point) from all model points appearing within the projected image.

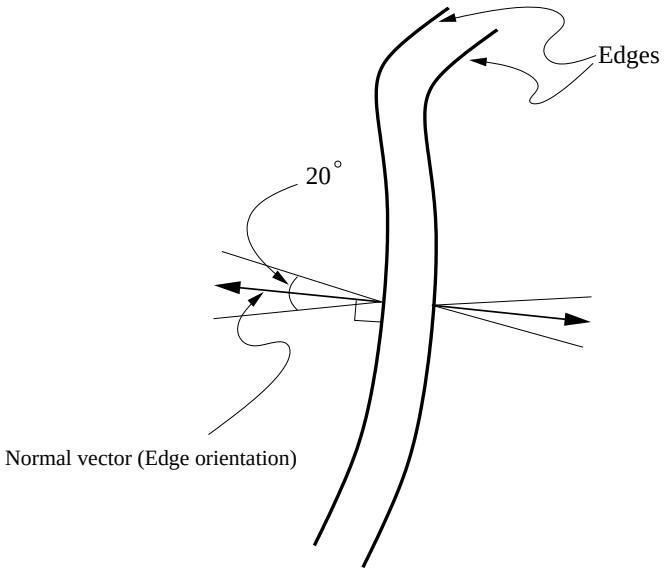
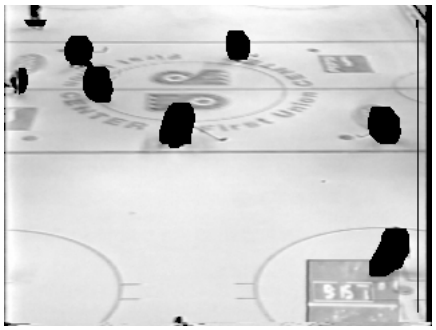
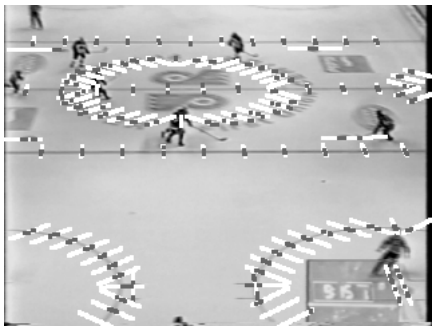


Figure 3.10: **Edge orientation**

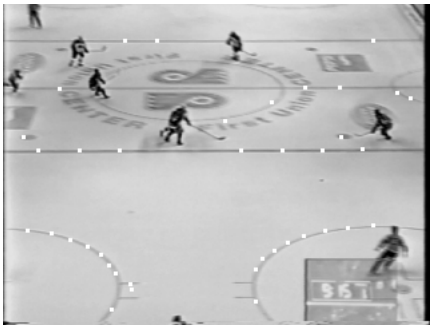
The orientation of the edge is represented as the normal vector (i.e., gradient vector) that is perpendicular to the edge. We set the threshold as 20° to match the orientation.



(a) player blobs



(b) searching edges on high gradient regions



(c) Edge points found by the search

Figure 3.11: **Searching edges**

(a) is the image of detecting player's blobs by simply finding dark regions of the image. Those dark regions are avoided for the edge search because the edges of the player give erroneous information. (b) shows the search regions (lighter points) and high gradient regions (darker points). It is shown that high gradient regions lie on edges we want to find. (c) is the result of the edge search. It shows how successfully our search detects edge points for each model points.

As it is shown in the above figure, the edge search does not perfectly detect all the edge pixels on the rink surface. For instance, in (c) of Figure 3.11, there is one edge pixel that does not belong to any lines in the left bottom face-off circle. Furthermore, there are not many edge points detected on the center circle since there are many gradient peaks detected on the line of the circle, the edges of the logo, and the edges of the letters. In order to avoid finding edge points that are not on the edge of the circles or lines on the rink, our edge search ignores ambiguous regions with many edges by detecting multiple gradient peaks in the search region. Given n edge points found by our edge search, these points can be used to compute a transformation, $\mathbf{H}_{corr}$, to rectify a projected image to the model. We use the normalized DLT algorithm to compute $\mathbf{H}_{corr}$ based on 2D to 2D point correspondences $\{\mathbf{x}_i^{Edge} \leftrightarrow \mathbf{x}_i^{Model}\}_{i=1...n}$ where $\{\mathbf{x}_i^{Edge}\}_{i=1...n}$ denote n edge points detected by our edge search and $\{\mathbf{x}_i^{Model}\}_{i=1...n}$ are n corresponding model points. Overall, our edge search gives us reliable performance and can prove that our model fitting mechanism works well. The next section shows the effect of the model fitting.

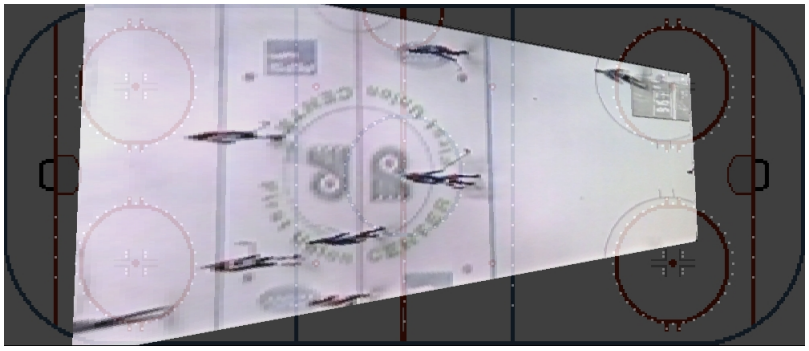
3.7.3 Result

This section shows how much our model fitting mechanism can reduce error accumulation. In order to justify the effectiveness of the mechanism, we show the result with and without the model fitting. Figure 3.12 shows two different transformation results.

As it is shown in the above figure, our algorithm is applied to process over 300 frames. Without the model fitting, the result shows a large projective distortion accumulated over time. By correcting errors at every frame, the model fitting successfully reduces projective distortions and prevent the accumulation of errors. The result therefore shows the effectiveness of our model fitting algorithm.

3.8 Rectification system

In this section, we summarize our rectification algorithm and introduce how to incorporate the algorithm into our vision system of acquiring trajectories of hockey players. Figure 3.13 shows the summary of our algorithm to rectify a sequence of hockey video.



(a) The result without fitting the model

(b) The result with fitting the model

Figure 3.12: **The result of our model fitting**

(a) is the result after 323 frames without using the model fitting mechanism. (b) is the result after 323 frames with the model fitting mechanism. It clearly shows how much projective distortion is reduced over 300 frames in (b).

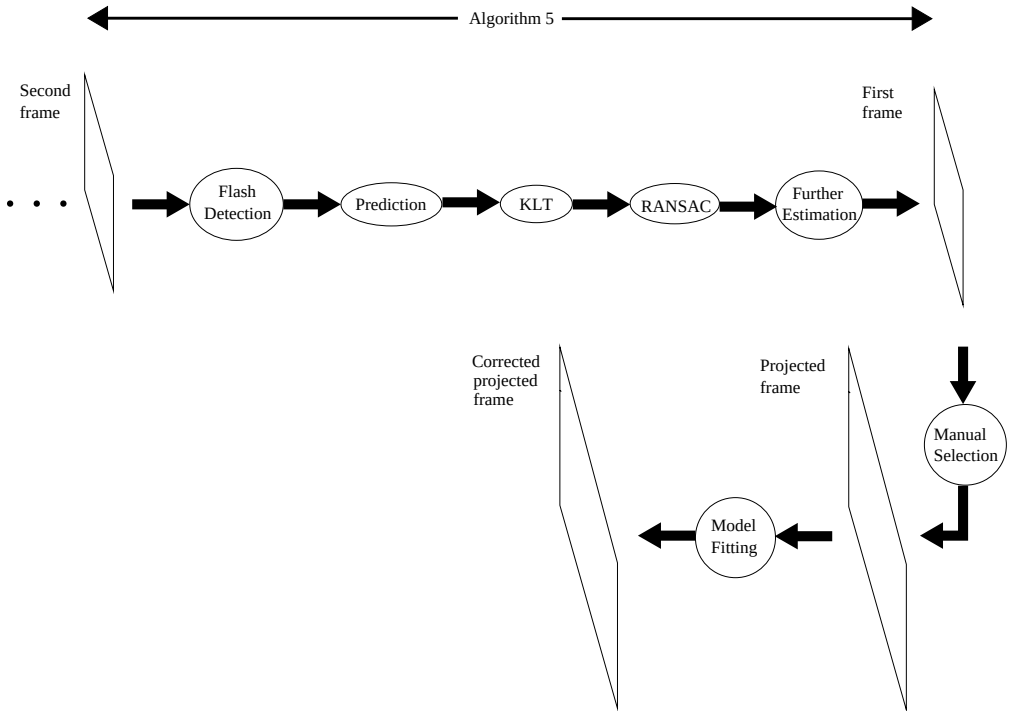


Figure 3.13: **The process of automatic homography calculation**

This is the picture that summarizes how our algorithm is implemented to transform a sequence. First we manually select the correspondences to gain the transformation to the rink map and then use Algorithm 5 to gain frame-to-frame homographies while detecting camera flashes and predicting camera motions.

We use manually selected points to gain the mapping between the very first frame of the sequence and the rink map. Then Algorithm 5 is applied to compute a frame-to-frame homography. Then we use the model fitting to correct and reduce the accumulation of errors over time.

Here, we denote the homography with manually selected correspondences between the projected image and the very first frame of the sequence as $\mathbf{H}_{ToRink}$ and the frame-to-frame homography that maps Frame n to Frame n-1 as $\mathbf{H}_{n,n-1}$. The homography gained by the model fitting is denoted as $\mathbf{H}^n_{corr}$ where n is the number of the current frame to be processed since $\mathbf{H}^n_{corr}$ is updated every frame we process. Given that, we can express the process of our algorithm of automatically computing homography as the chained homography below:

$$\mathbf{H}_{n,Rink} = \mathbf{H}^{n}_{corr}\mathbf{H}^{n-1}_{corr}\ldots\mathbf{H}^{2}_{corr}\mathbf{H}^{1}_{corr}\mathbf{H}_{ToRink}\mathbf{H}_{2,1}\mathbf{H}_{3,2}\ldots\mathbf{H}_{n,n-1}$$

where $\mathbf{H}_{n,Rink}$ means the transformation from Frame n to the rink map (i.e., the final projected image). As the chained homography expands in both directions, the set of correction homographies (i.e., $\mathbf{H}^{n}_{corr}\ldots\mathbf{H}^{1}_{corr}$) reduces the accumulation of errors produced from the set of frame-to-frame homographies (i.e., $\mathbf{H}_{2,1}\ldots\mathbf{H}_{n,n-1}$).

This rectification process plays a major role of the whole system with the following two reasons. Firstly, the result of the rectification determines the relationship between the original hockey video sequence and the rink map. Secondly, the automation of this process leads to the great savings of time because the rectification is a tedious and time consuming task to be done manually. Figure 3.14 describes the whole system and shows how the rectification process is incorporated into the system.

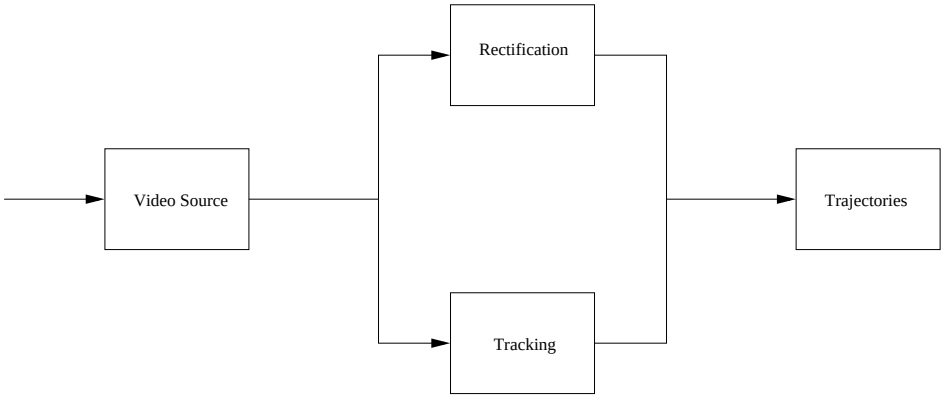


Figure 3.14: **System for acquiring trajectories of hockey players**

This is a simple presentation of our system. As it is shown, our system has two major components to create the final result.

3.9 Result

This section presents the result of our rectification process. Figure 3.15 shows the successful transformation automatically achieved over 500 frames. Although the result is quite satisfactory, the only bottleneck of the system is the computational time. This is due to the computation of singular value decomposition in the process of the further

estimation after RANSAC. Since we set the number of initial KLT features as 1200, there is an iterative process of estimating the best set of inliers from over 800 features. That means to perform the least square minimization on over 800×800 dimensions multiple times. Although we consider reducing the initial set of inliers, our experiments convince us to use 1200 features for the best result. The computational time is about 1 CPU hour to process 250 frames on PCs with 2.66 GHz Pentium 4 CPUs, 1GB RAM.

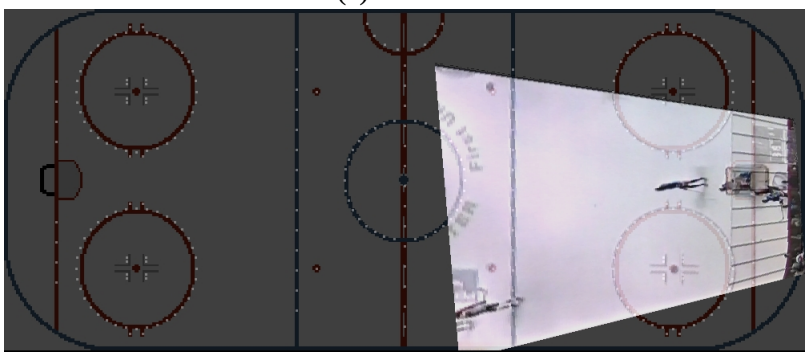
3.10 Limitations

Although our rectification system has the mechanism of reducing image processing errors at running time, there is a limitation on the quality of its performance. In this section, we analyze the accumulation of errors caused by frame-to-frame homographies and cases of which our rectification algorithm fails.

Our rectification system most likely fail in the following circumstances: when some of hockey data have feature-less regions where there are no features to be detected, and when camera motions are fast enough to blur visual features in the image. In the former case, features tend to be concentrated on a few texture rich regions when feature-less regions occupy the majority of the image. As a result, homographies, computed based on features that are not well spatially distributed, do not capture camera motions between frames well. If the amount of errors from frame-to-frame homographies is large enough for our model fitting method to be failed, then our system would not work. In the latter case, when rapid camera motions blur the image, many KLT features are easily lost by KLT tracker and we do not gain a reliable set of inliers for computing homographies. Furthermore, it becomes extremely difficult to detect true edges in blurred images. Consequently, it happens that not only frame-to-frame homographies are erroneous but also our model fitting method fails to detect edge points due to low resolution images.

In order to resolve the limitation of our system, we suggest to define the stopper of our program, i.e., the method to terminate our program when a large error occurs in the process. We could detect the large errors by monitoring the errors between the

(a) Frame 1



(b) Frame 200

(b) Frame 545

Figure 3.15: **Result of our rectification system**

Our system is applied to over 500 frames and successfully transform the sequence. Each of the image contains the rink mode in the background to show a high accuracy of the result. The original data is sampled at 30 frames per second, being digitized and deinterlaced.

projected image and the model of the rink map. The error per each model point is computed by averaging the total error distance between the projected image and the model based on the number of edge points found. This error measurement, however, has a problem: it could catch the large projection errors of over 10 pixels, but tend to ignore smaller projection errors. That is, the program does not terminate itself until a large error accumulation is detected. For the faster and more accurate detection of error accumulations, we currently investigate further to find alternatives for a better error detection.

3.11 Conclusion

Our rectification system captures camera motions by the automatic computation of homograhies and computes transformations between the original sequence in broadcast video and the globally consistent map of the hockey rink. We demonstrate the robustness of our system by showing the rectified sequence of over 500 frames in Figure 3.15. Our system accomplishes a highly accurate registration of the images regardless of many vision problems. We believe that our system is also applicable for other domains of sports, such as soccer and football. In the next chapter, we explain how to track hockey players and show their trajectories by combining transformations computed by our rectification system and tracking results obtained by our tracker.

Chapter 4

Tracking

4.1 Introduction

This chapter presents the other major component of our hockey annotation system: how to track hockey players and estimate their position on the hockey rink. In Chapter 3, we show how to remove rapid and non-smooth motions of broadcast cameras and register the original broadcast video sequence to the globally consistent map of the hockey field. In order to estimate real world positions of hockey players and generate their trajectories, we now need to track them. Our tracking method is an improved and extended version of the state of art color-based sequential Monte Carlo tracking method proposed in [25]. We use an adaptive multi-color model to improve the existing state of art color-based tracking and extend it to the multiple object tracking.

4.2 Initialization

In order to perform the tracking of hockey players, it is necessary to know where they are initially. We have three possibilities to consider: either manual initialization, semi-automatic initialization, or automatic initialization. For complicated models or objects with a constantly moving background, manual initialization [2, 11, 25, 28] is often preferred. However, even with a moving background, if the background is a uniformly colored region, then semi-automatic initialization can be implemented [20]. Misu *et. al* performs a semi-automatic initialization by automatically detecting isolated objects based on the difference between the color information of the object appearance (i.e.,

the uniform of soccer players) and the background (i.e., uniformly green soccer turf) and manually initializing overlapping objects. Automatic initialization is possible in many circumstances, such as smart environments [12, 37], surveillance [25, 26], sports tracking with a fixed background [3, 21, 38] or other situations with a fixed camera and a fixed background [25]. Wren *et. al* [37] uses the background model and detect their targets based on a large intensity change in the scene and the distribution of color on the appearance of an object. Perez *et. al* [25] demonstrates the use of background model to detect a person who enters in a room with a surveillance camera. Needham and Boyle [21] uses the foreground model (i.e., soccer players) and the background model (i.e., fixed background view of an indoor gymnasium) to extract the silhouette of moving objects.

In our case of small blob-like objects with a constantly moving background, we choose to initialize objects manually. If we have a reliable object recognition system, then it is still possible to have the automatic initialization. However, it is not an easy task to recognize small blob-like hockey players on the rink without distinguishing specific object features such as the number, logo, or their face. Object recognition is beyond the scope of this thesis and therefore considered to be a future development. Since our experiments show that the selection of a tightly bounding box on the targets is necessary for the successful tracking performance, manual initialization is at present the most appropriate method for the tracking of hockey players without implementing object recognition.

4.3 Sequential Monte Carlo tracking

Sequential Monte Carlo methods have become popular and already been applied to numerous problems in time series analysis [4], econometrics [18], and object tracking [18, 23, 25]. In non-Gaussian, hidden Markov (or state-space) models (HMM), the state sequence $\{\mathbf{x}_t; t \in \mathbb{N}\}, \mathbf{x}_t \in \mathbb{R}^{n_\mathbf{x}}$, is assumed to be an unobserved (hidden) Markov process with initial distribution $p(\mathbf{x}_0)$[1] and transition distribution $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, where $n_\mathbf{x}$ is the dimension of the state vector. The observations $\{\mathbf{y}_t; t \in \mathbb{N}^*\}, \mathbf{y}_t \in \mathbb{R}^{n_\mathbf{y}}$, are conditionally independent given the process $\{\mathbf{x}_t; t \in \mathbb{N}\}$ with marginal dis-

[1] we denote $p(\mathbf{x}_0)$ as $p(\mathbf{x}_0|\mathbf{x}_{-1})$ for the notational convenience

tribution $p(\mathbf{y}_t|\mathbf{x}_t)$, where $n_{\mathbf{y}}$ is the dimension of the observation vector. We denote the state vectors and observation vectors up to time t by $\mathbf{x}_{0:t} \triangleq \{\mathbf{x}_0 \dots \mathbf{x}_t\}$ and similarly for $\mathbf{y}_{0:t}$. Given the model, we can obtain the sequence of filtering density to be tracked by the recursive Bayesian estimation:

$$p(\mathbf{x}_t|\mathbf{y}_{0:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{0:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{0:t-1})} \tag{4.1}$$

where $p(\mathbf{y}_t|\mathbf{x}_t)$ is likelihood in terms of the observation model, $p(\mathbf{x}_t|\mathbf{y}_{0:t-1})$ is a prior that propagates past state to future and $p(\mathbf{y}_t|\mathbf{y}_{0:t-1})$ is evidence. The Kalman filter can handle Eq. (4.1) analytically if the model is based on the linear Gaussian state space. However, in the case of visual tracking, the likelihood is non-linear and often multi-modal with respect to the hidden states. As a result, the Kalman filter and its approximation work poorly for our case.

With sequential Monte Carlo techniques, we can approximate the posterior $p(\mathbf{x}_t|\mathbf{y}_{0:t})$ by a finite set of M particles (or samples), $\{\mathbf{x}_t^i\}_{i=1\cdots M}$. In order to generate samples from $p(\mathbf{x}_t|\mathbf{y}_{0:t})$, we propagate samples based on an appropriate proposal transition function $f(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t)$. We set $f(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$, which is the bootstrap filter [5]. We denote $\{\mathbf{x}_t^i\}_{i=1\cdots M}$ as fair samples from the filtering distribution at time t, then the new particles, denoted by $\tilde{\mathbf{x}}_{t+1}^i$, have the following association with the importance weights:

$$\pi_{t+1}^i \propto \frac{p(\mathbf{y}_{t+1}|\tilde{\mathbf{x}}_{t+1}^i)p(\tilde{\mathbf{x}}_{t+1}^i|\mathbf{x}_t^i)}{f(\tilde{\mathbf{x}}_{t+1}^i|\mathbf{x}_t^i, \mathbf{y}_{t+1})} \tag{4.2}$$

where $\sum_{i=1}^M \pi_{t+1}^i = 1$. We resample these particles with their corresponding importance weights to generate a set of fair samples $\{\mathbf{x}_{t+1}^i\}_{i=1\cdots M}$ from the posterior $p(\mathbf{x}_t|\mathbf{y}_{0:t})$. With the discrete approximation of $p(\mathbf{x}_t|\mathbf{y}_{0:t})$, the tracker output is obtained by the Monte Carlo approximation of the expectation:

$$\hat{\mathbf{x}}_t \triangleq \mathbb{E}(\mathbf{x}_t|\mathbf{y}_t) \tag{4.3}$$

where $\mathbb{E}(\mathbf{x}_t|\mathbf{y}_t) = \frac{1}{M}\sum_{i=1}^M \mathbf{x}_t^i$. The following two sections will explain our state dynamic model and the likelihood estimation by our adaptive multi-part color model.

The algorithm of our tracker is summarized in Algorithm 6.

4.4 State Dynamics

We formulate the state to describe a region of interest to be tracked. We assume that the shape, size, and position of the region are known *a priori* and define a rectangular window W. The shape of the region could also be an ellipse [2, 23] or any other appropriate shapes to be described, which depends mostly on what kind of object to track. In our case, the objects are hockey players that are small, non-rigid, and deformable. Therefore both a rectangle and ellipse are suitable. As in [2, 23, 25], the state consists of the location and the scale of the window W.

The state dynamics varies and depends on the type of motion to deal with. Due to the constant, yet often random, nature of hockey players' motion, we choose the second-order auto-regressive dynamics to be the best approximating their motion as in [25]. If we define the state at time t as a vector $\mathbf{x}_t = (l_t^\top, l_{t-1}^\top, s_t, s_{t-1})^\top$, where $^\top$ denotes the transpose, $l_t = (x, y)^\top$ is the position of the window W at time t in the image coordinate, and s_t is the scale of W at time t. We apply the following state dynamics:

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{x}_{t-1} + C\mathbf{v}_t, \mathbf{v}_t \sim \mathcal{N}(0, \Sigma). \tag{4.4}$$

Matrices A, B, C and Σ control the effect of the dynamics. As it is mentioned in [25], those matrices could be learned based on the perfect tracking result obtained in some way. However, we do not cover this issue since it is beyond the scope of this thesis. In our experiments, we define those matrices in ad-hoc way by assuming the constant velocity on hockey players' motion.

4.5 Multi-part Color Adaptation Model

This section explains how we incorporate the global nature of color in visual perception into our sequential Monte Carlo framework. We follow the implementation of HSV color histograms used in [25], and extend it to our adaptive color model.

4.5.1 Color distribution Model

We use histograming techniques in the Hue-Saturation-Value (HSV) color space for our color model. Since HSV decouples the intensity (i.e., Value) from color (i.e., Hue and Saturation), it becomes reasonably insensitive to illumination effects. An HSV histogram is composed of $N = N_h N_s + N_v$ bins and we denote $b_t(\mathbf{k}) \in \{1, \ldots N\}$ as the bin index associated with the color vector $\mathbf{y}_t(\mathbf{k})$ at a pixel location $\mathbf{k}$ at time t. As it is pointed out in [25] that the pixels with low saturation and value are not useful to be included in HSV histogram, we populate the HSV histogram without those pixels with low saturation and value. Figure 4.1 shows two instances of the color histogram.

If we define the candidate region in which we formulate the HSV histogram as $R(\mathbf{x}_t) \triangleq \mathbf{l}_t + s_t W$, then a kernel density estimate $\mathbf{Q}(\mathbf{x}_t) \triangleq \{q(n; \mathbf{x}_t)\}_{n=1,\ldots,N}$ of the color distribution at time t is given by [2, 25]:

$$q(n; \mathbf{x}_t) = \eta \sum_{\mathbf{k} \in R(\mathbf{x}_t)} \delta[b_t(\mathbf{k}) - n] \tag{4.5}$$

where δ is the Kronecker delta function, η is a normalizing constant which ensures $\sum_{n=1}^{N} q(n; \mathbf{x}_t) = 1$, and a location $\mathbf{k}$ could be any pixel location within $R(\mathbf{x}_t)$. Eq. (4.5) defines $q(n; \mathbf{x}_t)$ as a probability of a color bin n at time t.

If we denote $\mathbf{Q}^* = \{q^*(n; \mathbf{x}_0)\}_{n=1,\ldots,N}$ as the reference color model and $\mathbf{Q}(\mathbf{x}_t)$ as a candidate color model, then we need to measure the data likelihood (i.e., similarity) between $\mathbf{Q}^*$ and $\mathbf{Q}(\mathbf{x}_t)$. As in [2, 23, 25], we apply the Bhattacharyya similarity coefficient to define a distance ξ on HSV histograms and its formulation given by [2]:

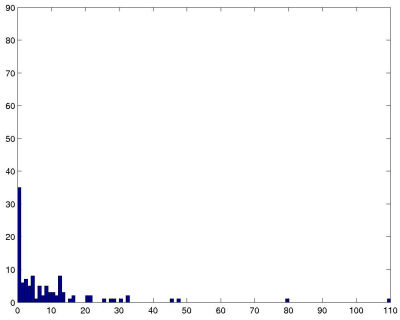
$$\xi[\mathbf{Q}^*, \mathbf{Q}(\mathbf{x}_t)] = \left[1 - \sum_{n=1}^{N} \sqrt{q^*(n; \mathbf{x}_0) q(n; \mathbf{x}_t)}\right]^{\frac{1}{2}} \tag{4.6}$$

Statistical properties of near optimality and scale invariance presented in [2] ensure that the Bhattacharyya coefficient is an appropriate choice of measuring similarity of color histograms. Once we obtain a distance ξ on the HSV color histograms, we use

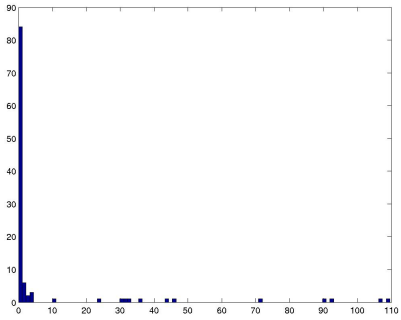
(a) Color histogram of two different players

(b) Close-up of the player on left

(c) Close-up of the player on right



(d) Color histogram of the player on left



(e) Color histogram of the player on right

Figure 4.1: **Color histograms**

This figure shows two color histograms, each of which is from a different region of the image. The player on left has uniform whose color is the combination of dark blue and white and the player on right has red uniform. One can clearly see concentrations of color bins due to limited number of colors. In (d) and (e), we set the number of bins, $N = 110$, where N_h, N_s, and N_v are set to 10.

the following likelihood distribution given by [25]:

$$p(\mathbf{y}_t|\mathbf{x}_t) \propto e^{-\lambda\xi^2[\mathbf{Q}^*,\mathbf{Q}(\mathbf{x}_t)]} \tag{4.7}$$

where $\lambda = 20$. λ is determined based on our experiments. Also, we set the size of bins N_h, N_s, and N_v as 10.

4.5.2 Multi-part Color Likelihood Model

The HSV color histogram is a reliable approximation of the color density on the tracked region. However, a better approximation could be achieved if we consider the spatial layout of the color because histograms ignore that. If we define the tracked region as the sum of r sub-regions $R(\mathbf{x}_t) = \sum_{j=1}^{r} R_j(\mathbf{x}_t)$, then we apply the likelihood as the sum of the reference histograms $\{q_j^*\}_{j=1,\ldots,r}$ associated with each sum-region by [25]:

$$p(\mathbf{y}_t|\mathbf{x}_t) \propto e^{\sum_{j=1}^{r} -\lambda\xi^2\left[\mathbf{Q}_j^*,\mathbf{Q}_j(\mathbf{x}_t)\right]} \tag{4.8}$$

Eq. (4.8) shows how the spatial layout of the color is incorporated into the data likelihood. In Figure 4.2, we divide up the tracked regions into two sub-regions in order to use spatial information of the color in the appearance of a hockey player.

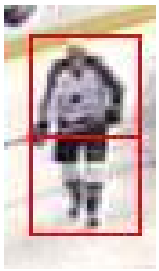


Figure 4.2: **Multi-part color likelihood model**

This figure shows our multi-part color likelihood model. We divide our model into two sub-regions and take a color histogram from each sub-region so that we take into account the spatial layout of colors of two sub-regions.

The effect of using the multi-part color likelihood model depends on the type of objects. For instance, if tracking targets have only a single color on their appearance, then the spatial layout of color does not help. As for hockey players, their uniforms

usually have a different color on their jacket and their pants and the spatial relationship of different colors becomes important to be considered. In fact, Figure 4.3 shows the benefit of using the multi-part color likelihood model. All results presented in this figure already use the model adaptation mechanism explained in the proceeding section. At this point, the objective is solely to show the benefit of the multi-part color model and thus the adaptation mechanism does not affect the content of the result. The result obtained by tracking a rectangular area without considering the spatial layout of the color information demonstrates that the tracking box tends to drift off the target and eventually loses the target when the target makes a sudden turn to left. On the other hand, the result with the multi-part color likelihood model shows not only that the box does not drift off the target but also that the tracker never loses the target. It is now evident that the multi-part color model increases the robustness of our color-based tracker.

In our experiments, we also consider using more than two subregions to track hockey players since three regions capture more precise spatial information of colors within the bounding box. However, our experiments show that using more than two subregions does not improve the tracking results on hockey players. Firstly, since the most of hockey players wear a different color jersey for the upper body and lower body, two subregions are enough to capture the spacial information of the dominant colors of their jersey. Secondly, using more than two subregions decreases the effective particle size and increases complexity of the likelihood distribution. Therefore, more particles are required for estimating the likelihood, which makes it more difficult to realize the real-time computation. With a reasonably small number of particles for the real-time computation, we decide to use the two-part color model for tracking hockey players.

4.5.3 Adaptation

Tracking dynamic objects requires the model of the object to be adaptable since they are constantly moving and change their shape. In [25], the reference color histogram is fixed to be $\mathbf{Q}^*$ initialized at the beginning of the sequence. The fixed reference model, however, becomes unreliable when the target undergoes a drastic change in the shape or color. Therefore, we extend our reference color histogram $\mathbf{Q}_t^*$ at time t to be

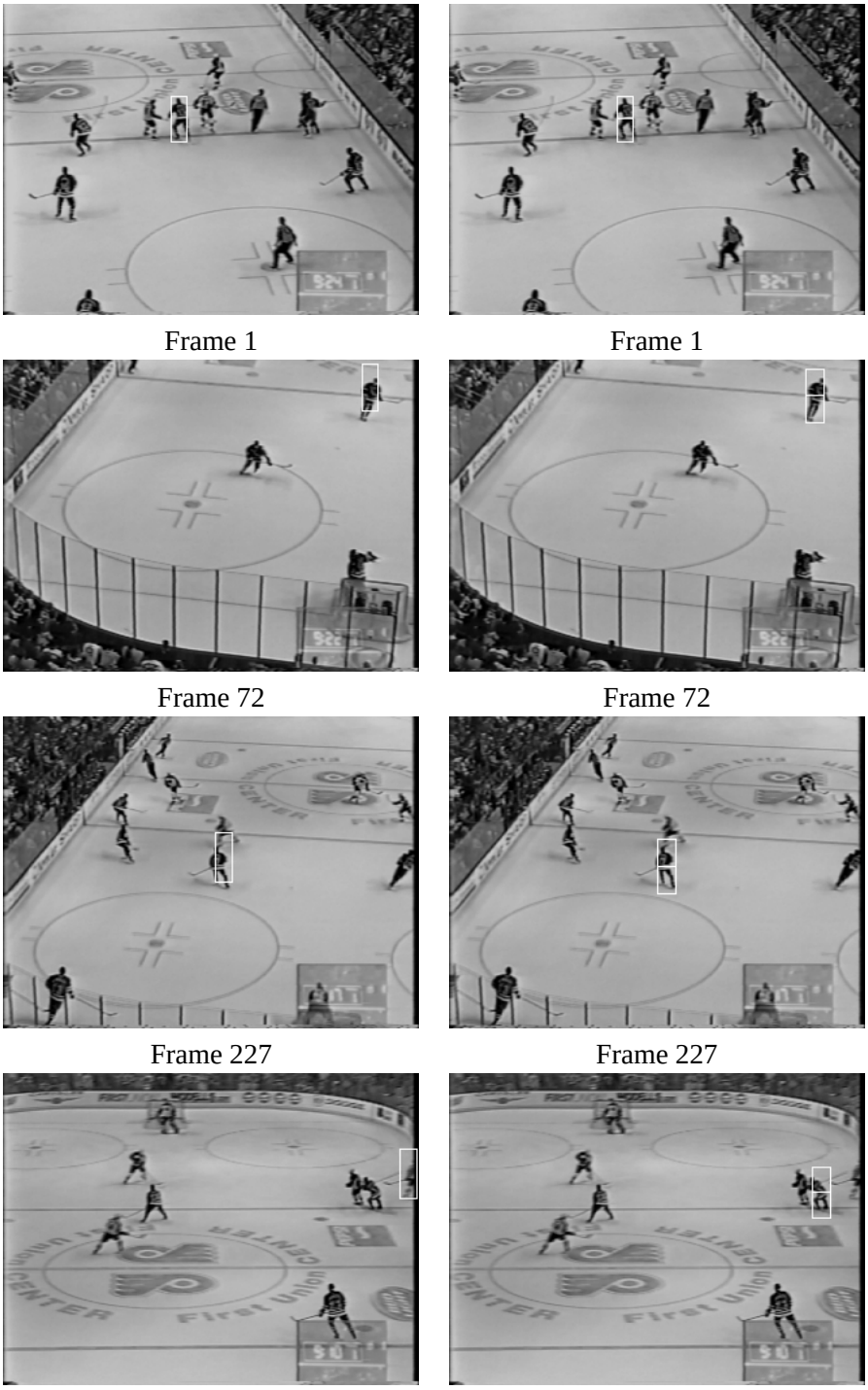


Figure 4.3: **Tracking result with and without multi-part color likelihood model** The images in the right column are the correct result, which the tracker never loses the target. On the other hand, the results with single part color model show that the tracker loses the target at Frame 436.

adaptable by combining the initial reference color histogram $\mathbf{Q}_0^*$ and the most recent color histogram $\mathbf{Q}_{t-1}^*$ estimated at time $t-1$. We define the following association:

$$\mathbf{Q}_t^* = \alpha \mathbf{Q}_0^* + (1-\alpha)\mathbf{Q}_{t-1}^* \tag{4.9}$$

where α is a constant that weights how much information of $\mathbf{Q}_0^*$ and $\mathbf{Q}_{t-1}^*$ should be included in the most recent reference histogram $\mathbf{Q}_t^*$. We set $\alpha = 0.7$, which is determined by the trial-and-error method in our experiments. The value of α is sensitive to the performance of our tracker especially in cluttered scenes or fast moving scenes. The tracker tends to lose the target in cluttered scenes if $\alpha = 0.6$ or 0.8. We conclude that there is no universally correct value of α that works in any situation.

While the model is updated and adapted overtime, the value of α is also updated based on the variance of the importance weights for all particles. Especially, when the variance of importance weights is increased to be too large, it indicates that the importance weights of the particles are all close to be a particular value and particles are distributed over a confusing area where there are many possible matches. Under such a circumstance, we increase the value of α to be 1. That is, we use only the original reference color model with no adaptation to avoid adapting our model with a false information. The threshold for the variance of importance weights is determined to be 0.004 from our experiments. This adaptation scheme is quite simple and yet makes our tracker better or at least as good as the one with the fixed reference model. We are currently investigating more on a better adaptive mechanism to determine the value of α based on the variance of importance weights. Our results show the benefit of the adaptation in Figure 4.4.

4.6 Robustness

This section demonstrates the robustness of our color-based tracking method. For all the result presented in this section, we use 200 particles with the model adaptation. As for the computational cost, our non-optimized implementation in C^{++} allows the tracking of the region with the size of 20×20 pixels at a sample rate of 30 frames per second on a PCs workstation with 2.66 GHz Pentium 4 CPU, 1GB RAM.

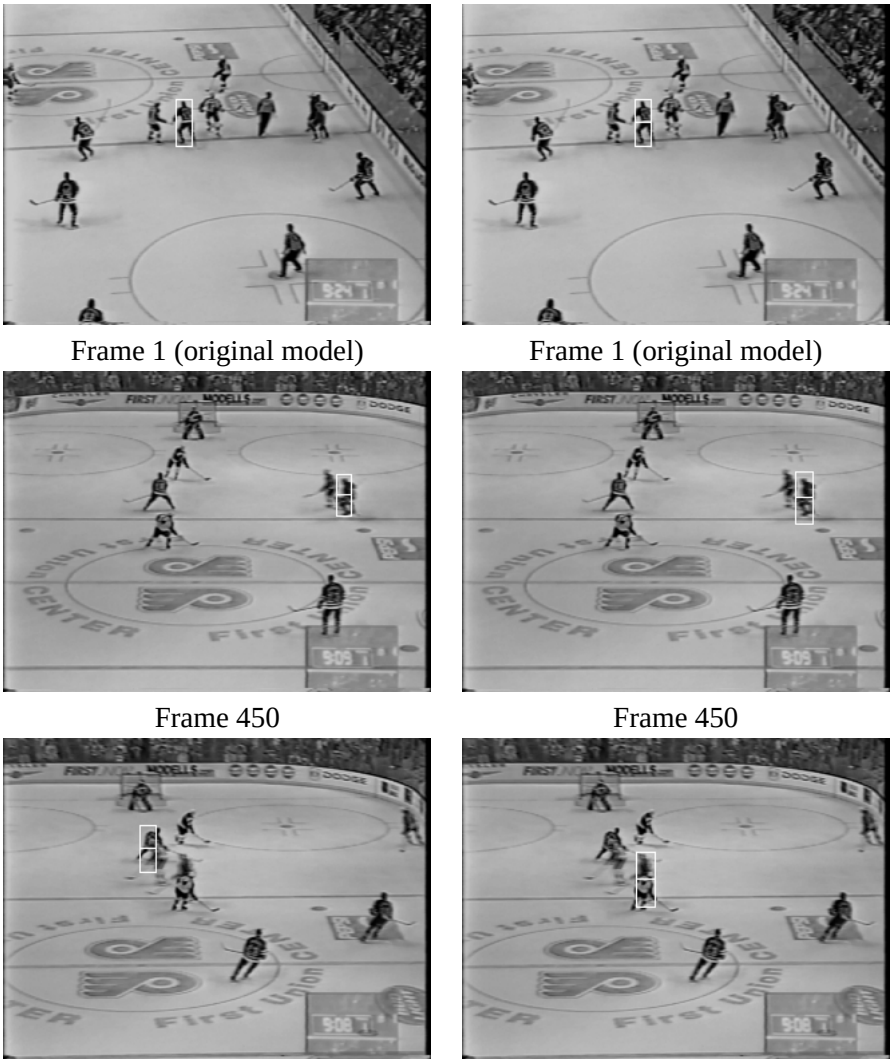


Figure 4.4: **Tracking result with and without adaptation**

The images in the left column are the result of tracking a single hockey player without our adaptation mechanism. The images in the right column are the result with the adaptation. As it is shown, the tracker with no adaptation loses the object when the target undergoes a drastic change of its appearance and a similar object is closeby. Apparently, on Frame 474, the box drifted off to a player in the same team whose appearance more closely resembles the original model of the target.

Algorithm

Given the initial reference color model $\mathbf{Q}_0^*$, track the window W.

(1) **Initialization:**
– Set $t = 0$.
– If $t > 0$, then update the reference color model $\mathbf{Q}_t^*$ by Eq. (4.9).
– For $i = 1, \ldots, M$, sample $\mathbf{x}_t^i \sim p(\mathbf{x}_0)$.

(2) **Propagation of particles:**
– For $i = 1, \ldots, M$, propagate $\mathbf{x}_t^i$ by second-order AR dynamics in Eq. (4.4).

(3) **Importance sampling:**
– For $i = 1, \ldots, M$, compute a candidate histogram $\mathbf{Q}_{it}^*$ by Eq. (4.5) and compute the importance weights

$$\pi_t^i = \kappa \, e^{\sum_{n=1}^{N} \lambda \sqrt{q^*(n;\mathbf{x}_t) q_i(n;\mathbf{x}_t)}}$$

where κ ensures $\sum_{i=1}^{M} \pi_t^i = 1$.

(4) **Selection:**
– If the effective particle size $S = (\sum_{i=1}^{M} \pi_t^i)^{-1} < \frac{M}{2}$, then resample M particles from the set $\{\tilde{\mathbf{x}}_t^i\}_{i=1,\ldots,M}$ which is generated according to the importance weights
– Estimate $\hat{\mathbf{x}}_t$ according to Eq. (4.3).
– Set $t = t + 1$ and go to (1).

Algorithm 6: **Color-based sequential Monte Carlo tracker**

4.6.1 Illumination change

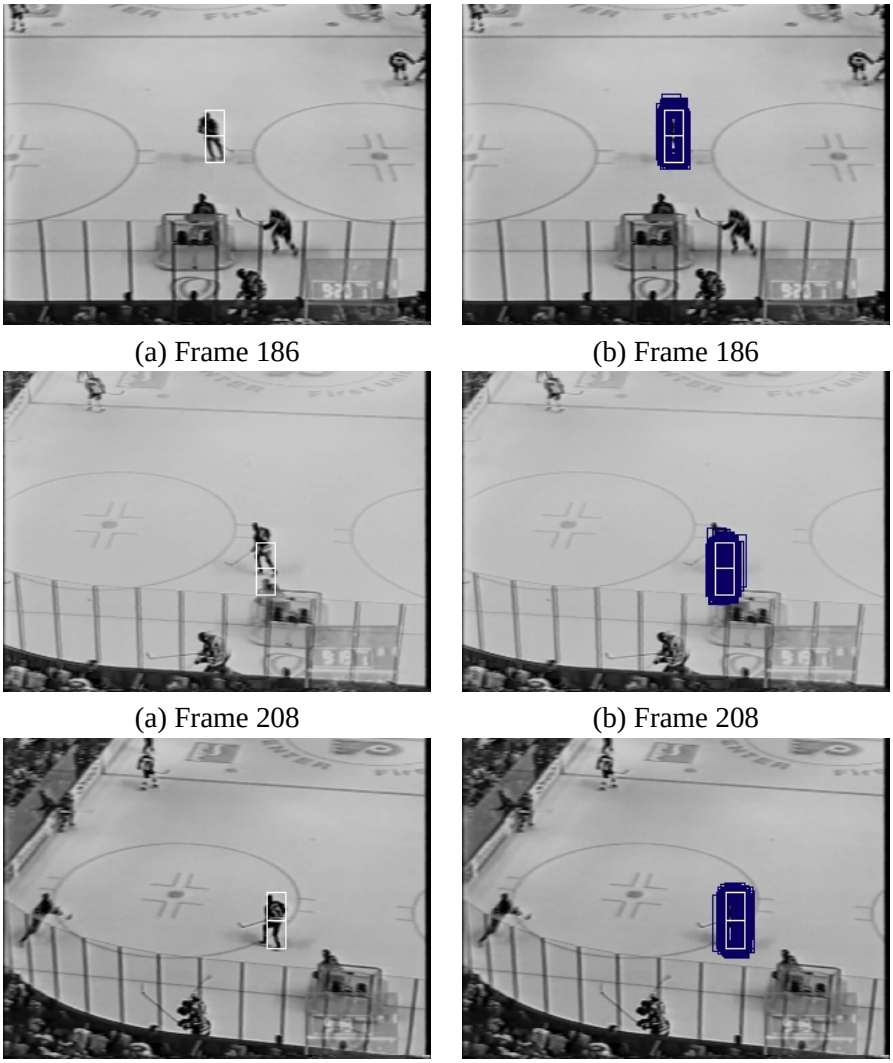
Figure 4.5 shows illumination insensitive performance of our tracker which does not lose the target even if there is a drastic illumination change due to camera flashes. The same figure also shows that the second-order AR dynamics makes our tracker robust in a scene even when targets make a sudden direction change. Furthermore, when the object moves by a small amount, the dynamics tends to fail to approximate the target's velocity and our tracker may not locate the object exactly (i.e., the bounding box tends to drift off by a little amount).

4.6.2 Cluttered scene

Figure 4.6 shows the successful performance of our tracker in a cluttered scene with similar objects nearby. It is evident that particles spread over the region which covers two similar objects, which makes it extremely hard for the tracker to locate the true target. However, the dynamics of the object and importance sampling successfully locate the majority of particles on the matched region and helps our tracker to locate the true target. Our extensive experiments also reveal the limitation of a single object color-based tracking method. Our tracker fails when there are identical objects such as players in the same team nearby the target, for the tracker does not know the association between two identical objects without any *prior* information. This is one open problem that we need to deal with in the future. Although Section 4.7 introduces the extended version of our tracker for multiple object tracking, the problem is still not solved.

4.7 Result

This section presents the tracking results by our tracker for both a single and multiple objects and the trajectories of these players by combining the tracking results obtained by our tracker and transformations computed by our rectification system. For a single object tracking, our tracker tracks a target for 500 frames. Figure 4.7 shows the result.



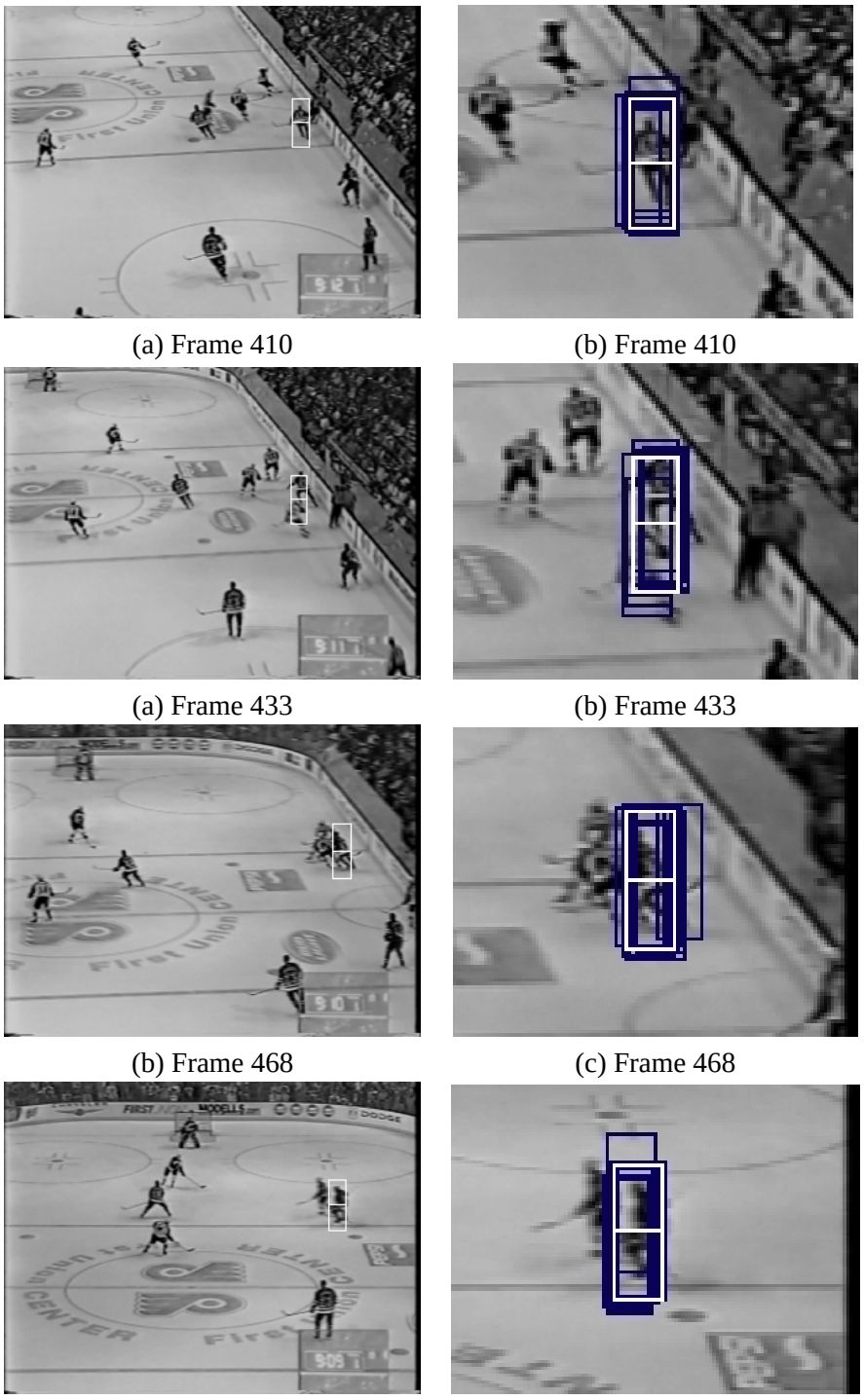
(a) Frame 186 (b) Frame 186

(a) Frame 208 (b) Frame 208

(b) Frame 220 (c) Frame 220

Figure 4.5: **Tracking results under a severe illumination change**

This figure shows the robustness of our tracker under a severe illumination change with a similar object closeby. The images in the right column show particles. On Frame 208, a camera flash makes a drastic intensity change on the image. Also the bounding box is drifting off due to a sudden direction change of the player. However, the tracker does not lose the target.



(a) Frame 410 (b) Frame 410

(a) Frame 433 (b) Frame 433

(b) Frame 468 (c) Frame 468

(b) Frame 500 (c) Frame 500

Figure 4.6: **Tracking results with a cluttered scene with similar objects**

This figure shows the robustness of our tracker in a cluttered scene with similar objects closeby. The enlarged images in the right column show 10 out of 200 particles to show how spread the particles are.

During a sequence of 500 frames, regardless of cluttered scenes and partial occlusions, the tracker never loses the target. We present every 100th frame to show the accuracy of our tracker. We consider the bottom of the bounding box as the real world position of the player on the rink. Therefore, the estimated positions of the object might be slightly off by from 30 cm to 1 m from the real world positions of the object. However, the gap is small enough to be ignored for displaying the trajectory of the object. For the multiple object tracking, we simply implement a separate color-based sequential Monte Carlo tracker for each individual object. Figure 4.8 shows the tracking results for three hockey players. The sequence is 250 frames long. Since our tracker does not deal with disappearance and reappearance of the object in the scenes, among our data, 250 frames are selected to be the longest sequence for which three players are always at present in the scene. As is shown in the figure, our tracker successfully tracks three small objects even in cluttered scenes. As for the computational cost of our tracker for both the single and multiple objects, our non-optimized implementation in C^{++} allows the tracking of the region with the size of 20×20 pixels at a rate of 30 frames per second on a PCs workstation with 2.66 GHz Pentium 4 CPU, 1GB RAM.

4.8 Conclusion

In this chapter, we use color-based sequential Monte Carlo tracking to track hockey players. We show that our adaptive color model improves the performance of the tracking method proposed in [25]. We also demonstrate several advantages for tracking a non-rigid object as our probabilistic approach is robust to partial occlusion, invariant to scale and illumination change, and efficient in computation time. The results presented in this chapter show the successful integration of our rectification system and tracking system. There are, however, many issues still remaining to be dealt with for better performance of the system. The next chapter summarizes our thesis and show several directions and extensions to be considered for future development of our automatic hockey annotation system.

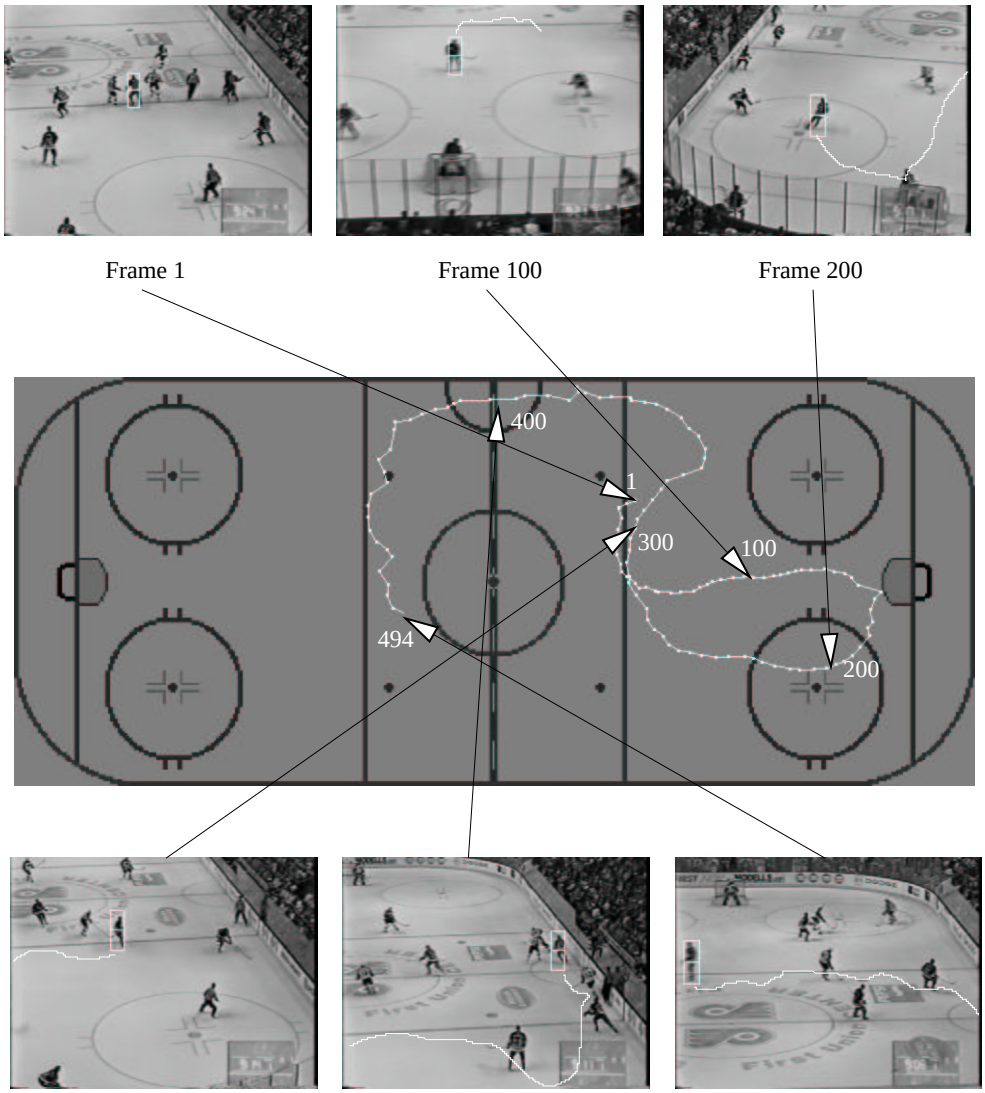


Figure 4.7: **Trajectory of a player being tracked**

This figure shows the tracking result on a single object and its trajectory. Our tracker successfully tracks the target for 494 frames. Our tracker works properly when the target is partially occluded or is cluttered with similar objects such as players from the same team.The trajectory is computed automatically based on transformations computed by our rectification system and the tracking result. We use arrows to direct where the object is on the trajectory in that particular frame. We also put hand drawn trajectory in each frame to give an idea how the object is moving around.

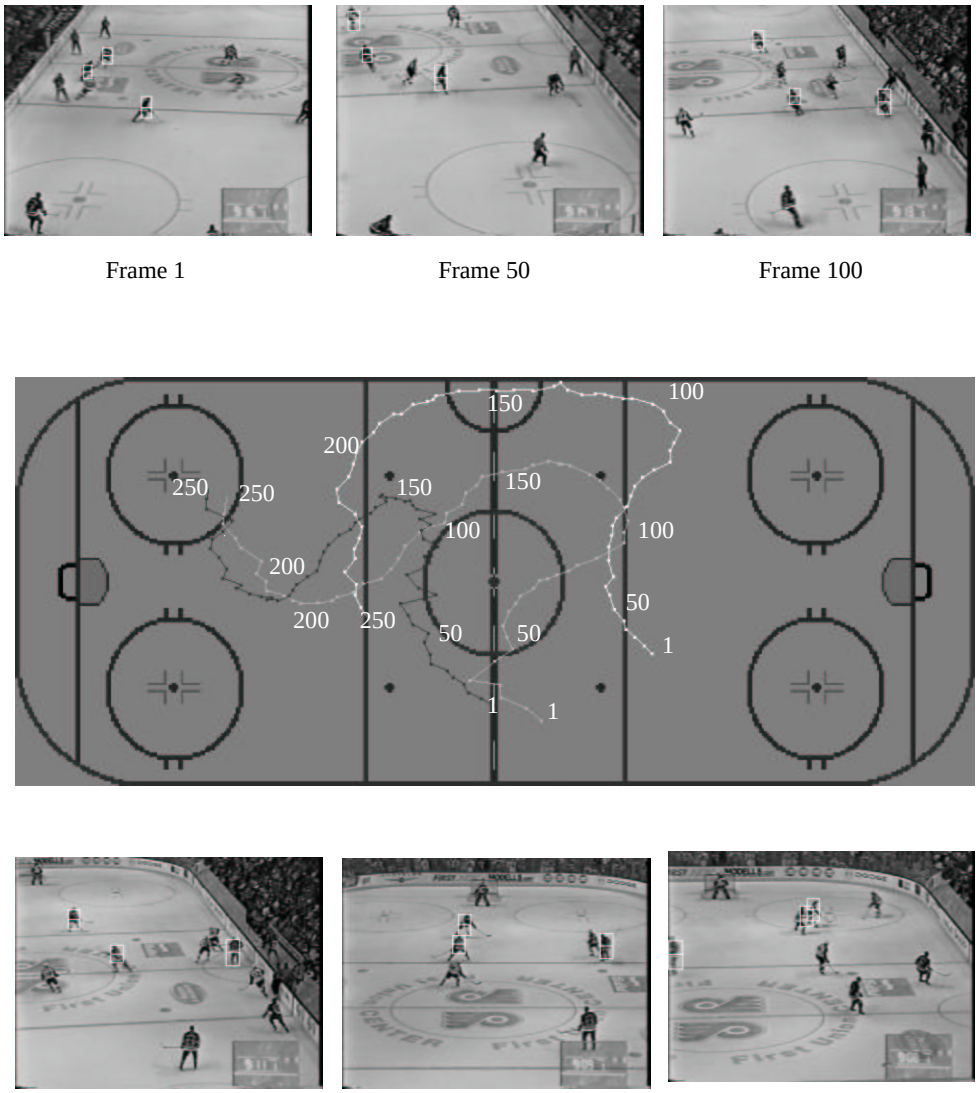


Figure 4.8: **Trajectories of three players being tracked**

This figure shows the tracking result on multiple objects and their trajectories. Our tracker successfully tracks three targets for 250 frames. The trajectory is computed automatically based on transformations computed by our rectification system and the tracking result. Unlike Figure 4.7, we do not use arrows or hand drawn trajectories of objects in the image for the clarity of the presentation. The numbers on the rink map represents the number of the frame in which each object is located at that particular location on a trajectory.

Chapter 5

Summary and extensions

5.1 Summary

In this thesis, we have addressed the problem of developing a descriptive system that analyzes what is happening in hockey scenes. Our contribution is twofold. Primarily, we have developed a method that removes camera motions of a broadcast camera and automatically computes the mapping between the original broadcast video sequence and the globally consistent rink map. Secondly, we have demonstrated excellent tracking results of hockey players that are all small blob-like, non-rigid, deformable, and low resolution objects.

A variety of vision problems make these contributions quite difficult. Due to a limited access only to the video that was taken with a panning, tilting, and zooming broadcast camera, we need to remove camera motions and create the globally consistent map to display trajectories of hockey players. Solving parameters of a broadcast camera in hockey has been quite challenging since camera motions are fast and non-smooth in order to capture highly complex and fast-moving dynamic scenes of hockey. Tracking hockey players is also difficult because they are quite fast, small, non-rigid, and deformable.

As we have overcome most of these vision problems and a literature survey has shown that there have not been any automatic hockey annotation systems in the past, we now hope that our efforts and accomplishments in this work would become a milestone for research in automatic video annotation in hockey. The methods pre-

sented here perform quite well on the test data and establish the infrastructure of our automatic video annotation system in hockey. In future work, we will improve our tracking algorithm for even more complex scenes with multiple player interactions and speed up the process of our system for real-time analysis of hockey scenes.

5.2 Future directions

For the successful progression toward our automatic video annotation system in future, we suggest several issues remained to be dealt with and possible improvements on our current system.

The major issue for our rectification system is an improvement on the registration accuracy. We suggest two ways for the further improvement. First, we develop an interactive user interface to correct projection errors accumulated over time from the automatic computation of frame-to-frame homographies. A better mechanism for detecting error accumulation is necessary for real-time interactions. Second, we improve our model fitting algorithm by making it more robust to large projection errors. Currently we are investigating a better edge search technique to improve our model fitting mechanism.

As for tracking hockey players, there are still many issues to be considered. First of all, we plan to automate the initialization of objects being tracked by computing frame-difference measurements [36]. The key for the initialization is to have a fixed background model to distinguish it from foreground objects. Therefore, the initialization should be done on the globally consistent map where the background is fixed. Secondly, we need to develop a method to detect objects that disappear and reappear in the scenes. Due to a limited field of view of a broadcast camera, players are constantly moving into or out of the scenes. There are some successful implementations for detecting objects that enter or disappear from the scenes with a fixed background by [36, 25, 23]. Their methods are most likely to fail in environments with a dynamic background. Thirdly, we plan to implement an unscented particle filter for better proposal distributions [28, 19] in order to place particles efficiently in the high likelihood area and realize a better tracking performance. Lastly, we need to extend

our sequential Monte Carlo tracker with multiple targets in the most complex scenes. Instead of using a separate tracker for each target, we plan to use only one tracker that could associate a given measurement to a target model automatically and estimate the position of multiple targets simultaneously.

Although our system is still open for many directions and improvements, it is now capable of providing visual input for a system that performs automatic video annotation of dynamic scenes. Since the methods presented in this work are applicable to video annotation in other domains of sports, surveillance, or many other situations that require object tracking on a planar surface, we hope to test our system in a different domain. We also hope that this thesis may become a foundation for automatic video annotation in hockey.

Bibliography

[1] S. Birchfield. *Depth and motion discontinuities*. PhD thesis, Stanford University, 1999. 18, 24, 28

[2] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conf. on Computer Vision and Pattern*, pages 142–151, 2000. 16, 57, 60, 61

[3] M. Y. D. Yow, Boon-Lock Yeo and B. Liu. Analysis and presentation of soccer highlights from digital video. In *ACVPR95*, Singapore, 1995. 2, 11, 58

[4] A. Doucet. On sequential monte carlo sampling methods for bayesian filtering, 1998. 15, 58

[5] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag New York, Inc., 2001. 59

[6] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981. 18

[7] Y. Gong, L. T. Sin, C. H. Chuan, H. Zhang, and M. Sakauchi. Automatic parsing of TV soccer programs. In *International Conference on Multimedia Computing and Systems*. IEEE Computer Society, May 1995. 2, 12

[8] R. Hartley. In defense of the eight-point algorithm. *PAMI*, 19(6):580–593, June 1997. 24

[9] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, June 2000. 18, 19, 21, 23, 24, 25, 40, 41

[10] U. H. Inc. The official rules of ice hockey, 2001. 44, 47

[11] S. Intille and A. Bobick. Visual tracking using closed-worlds. MIT Media Lab Perceptual Computing Technical Report 294, Massachusetts Institute of Technology, 20 Ames St. Cambridge, MA 02139, Nov. 1994. 2, 7, 13, 14, 17, 44, 57

[12] S. Intille, J. Davis, and A. Bobick. Real-time closed-world tracking. In *International Conference on Computer Vision and Pattern Recognition*, pages 697–703, Puerto Rico, June 1997. IEEE Computer Society Press. 14, 58

[13] M. Isard and A. Blake. The CONDENSATION algorithm — conditional density propagation and applications to visual tracking. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 361–? The MIT Press, 1997. 15

[14] A. Jacquin and A. Eleftheriadis. Automatic location tracking of faces and facial features in video sequences, 1995. 14

[15] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. In *2001 Conference On Computer Vision and Pattern Recognition*, volume Vol.I, pages 415–422, Kauai, 2001. IEEE Computer Society. 14, 16

[16] K. Kanatani and N. Ohta. Accuracy bounds and optimal computation of homography for image mosaicing applications. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV-99)*, volume I, pages 73–79, Los Alamitos, CA, Sept. 20–27 1999. IEEE. 18

[17] H. Kim and K. Hong. Soccer video mosaicing using Self-Calibration and line tracking. In *ICPR01 VOL I*, pages 592–595. IEEE, 2000. 2, 14

[18] J. S. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998. 15, 58

[19] R. Merwe, A. Doucet, N. Freitas, and E. Wan. The unscented particle filter. Techinical report cued/f-infeng/tr 380, Cambridge University Engineering Department, 20 Ames St. Cambridge, MA 02139, 2000. 15, 76

[20] T. Misu, M. Naemura, W. Zheng, Y. Izumi, and K. Fukui. Robust tracking of soccer players based on data fusion. In *16th International Conference on Pattern Recognition*, volume vol.1, pages 556–561. IEEE, 2002. 14, 57

[21] C. J. Needham and R. D. Boyle. Tracking multiple sport players through occlusion, congestion and scale. In *British Machine Vision Conference*, pages 1:93–102, 2000. 58

[22] C. J. Needham and R. D. Boyle. Tracking multiple sports players through occlusion, congestion and scale. In *British Machine Vision Conference*, volume vol.I, pages 93–102. BMVA, 2001. 14

[23] K. Nummiaro, E. Koller-Meier, and L. V. Gool. A color-based particle filter. In A. Pece, editor, *First International Workshop on Generative-Model-Based Vision*, volume 2002/01, pages 53–60. Datalogistik Institut, Kobenhavns Universitet, 2002. 15, 16, 17, 58, 60, 61, 76

[24] V. Pavlovic, J. M. Rehg, T.-J. Cham, and K. P. Murphy. A dynamic bayesian network approach to figure tracking using learned dynamic models. In *Proceedings of IEEE international Conference On Computer Vision (ICCV 99)*, pages 94–101, Corfu, Greece, 1999. IEEE Computer Society. 14, 16

[25] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *Eur. Conf. on Computer Vision, ECCV'2002*, volume 2350, pages 661–675, Copenhaguen, Denmar, 2002. 15, 16, 17, 57, 58, 60, 61, 63, 64, 72, 76

[26] J. H. Piater and J. L. Crowley. Multi-modal tracking of interacting targets using gaussian approximations. In *Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2001. 14, 58

[27] L. M. Rehg and T. Kanade. Visual tracking of high dof articulated structures: an application to human hand tracking. In *Proc. 3rd European Conference on Computer Vision*, pages 35–46, Stockholm, Sweden, 1994. Springer-Verlag. 15

[28] Y. Rui and Y. Chen. Better proposal distributions: Object tracking using unscented particle filter. In *2001 Conference on Computer Vision and Pattern Recognition (CVPR 01)*, pages 11–13, Kauai, Hawaii, 2001. IEEE. 14, 57, 76

[29] D. D. Saur, Y.-P. Tan, S. R. Kulkarni, and P. J. Ramadge. Automated analysis and annotation of basketball video. In *Storage and retrieval for image and video databases*, volume vol.3022, pages 176–187. SPIE, 1997. 2, 12

[30] J. Shi and C. Tomasi. Good features to track. Technical Report TR93-1399, Cornell University, Computer Science Department, Nov. 1993. 18, 24, 27, 28, 29, 31

[31] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 593–600, Los Alamitos, CA, USA, June 1994. IEEE Computer Society Press. 18, 24, 27, 28, 29, 31

[32] H. Sidenbladh, M. J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *European Conf. on Computer Vision (ECCV 02)*, volume Vol.I, pages 784–800, Copenhagen, Denmark, 2002. IEEE Computer Society. 14, 16

[33] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, Computer Science Department, 1991. 18, 24, 27, 28, 30, 31

[34] P. H. S. Torr and A. Zisserman. Feature based methods for structure and motion estimation. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *International Workshop on Vision Algorithms*, pages 278–295, 1999. 18

[35] K. Toyama and A. Blake. Probabilistic tracking in a metric space. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)*, pages 50–59, Los Alamitos, CA, July 9–12 2001. IEEE Computer Society. 14, 16

[36] J. Vermaak, P. Pérez, M. Gangnet, and A. Blake. Towards improved observation models for visual tracking: Selective adaptation. In *Proc. Europ. Conf. Computer Vision*, Copenhagen, Denmark, 2002. 15, 76

[37] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997. 14, 58

[38] A. Yamada, Y. Shirai, and J. Miura. Tracking players and a ball in video image sequence and estimating camera parameters for 3D interpretation of soccer games. In *ICPR02 VOL I*, pages 303–306. IEEE, 2002. 2, 10, 14, 58

[39] I. Zoghiami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2d mosaic from a set of images. In *CVPR97*, pages 420–425, 1997. 18, 20