Multi-view Hockey Tracking with Trajectory Smoothing and Camera Selection

by

Lan Wu

B.E., Zhejiang University, 2005

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

Master of Science

in

The Faculty of Graduate Studies

(Computer Science)

The University Of British Columbia

(Vancouver)

April, 2008

Abstract

We address the problem of multi-view multi-target tracking using multiple stationary cameras in the application of hockey tracking and test the approach with data from two cameras. The system is based on the previous work by Okuma et al. [50]. We replace AdaBoost detection with blob detection in both image coordinate systems after background subtraction. The sets of blob-detection results are then mapped to the rink coordinate system using a homography transformation. These observations are further merged into the final detection result which will be incorporated into the particle filter. In addition, we extend the particle filter to use multiple observation models, each corresponding to a view. An observation likelihood and a reference color model are also maintained for each player in each view and are updated only when the player is not occluded in that view. As a result of the expanded coverage range and multiple perspectives in the multi-view tracking, even when the target is occluded in one view, it still can be tracked as long as it is visible from another view. The multi-view tracking data are further processed by trajectory smoothing using the Maximum *a posteriori* smoother. Finally, automatic camera selection is performed using the Hidden Markov Model to create personalized video programs.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to acknowledge those who gave me invaluable help during the duration of this research. Without their support, I can never accomplish this work. First of all, my sincere gratitude goes to my supervisor, Dr. James Little, for his tremendous patience in providing me with extraordinary guidance and leading me to the right direction of research. I appreciate Dr. Sidney Fels for the opportunity of working on the MyView project and the exposure of many research topics. I would also like to thank my colleagues, Kenji Okuma and Wei-Lwun Lu for their beneficial suggestions and discussions. I also appreciate Jiang Hao for his help with the research data. My special thanks go to my family for their encouragement and support.

Chapter 1

Introduction

1.1 Motivation

With the increasing demand for managing and processing large volumes of data in video sequences, for example, in the context of sports games and surveillance, there is a great potential for the development of efficient and reliable approaches which are capable of tracking the visual features and analyzing the scenes in complex environments.

Our motivation arises in the challenge of inventing a computer system that detects and tracks multiple moving targets in a video sequence from multiple views, and automatically creates personalized video programs and diaries of people from different perspectives at the end of the event. To accomplish these tasks, the system, combining the information from several cameras at all stages, should be able to detect persons when they appear in the scene, track the locations and recognize the identities of the persons even when there is a lot of occlusion and interaction in the video sequence. In addition, the system should be able to automatically switch among the cameras and select the best view for any given person based on the occlusion analysis.

The system is set up in the context of an ice hockey game. In our experiment, the input to the system is two synchronized video sequences, videotaped using two high-definition video (HDV) camcorders from two views, the right view and the end view respectively. The layout of the two cameras is demonstrated as Figure 1.1. Both of these HDV camcorders are static without panning, tilting or zooming. The first process outputs the tracking results from both the right view and the end view, as shown in Figure 1.2. The locations and sizes of hockey players are represented by bounding boxes and the identification of the hockey players is represented by the color of the bounding boxes. The second process generates the personal video for each player and selects the best view for each frame through the video sequence.

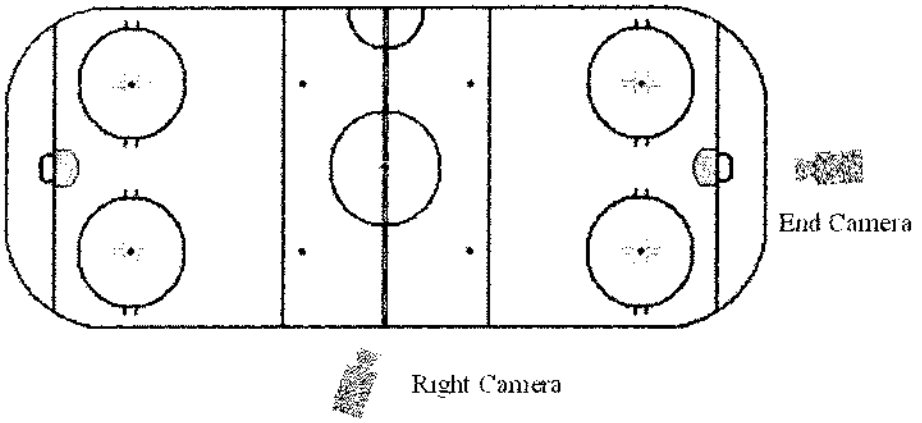
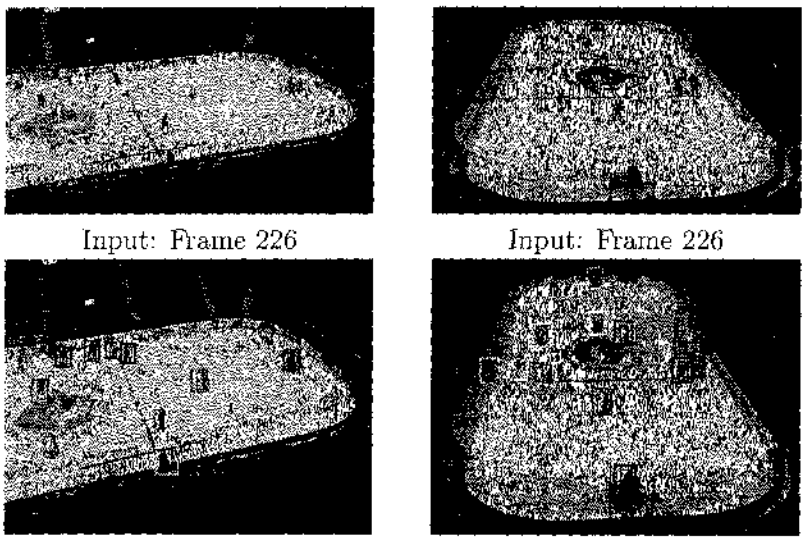


Figure 1.1: The layout of the two cameras.

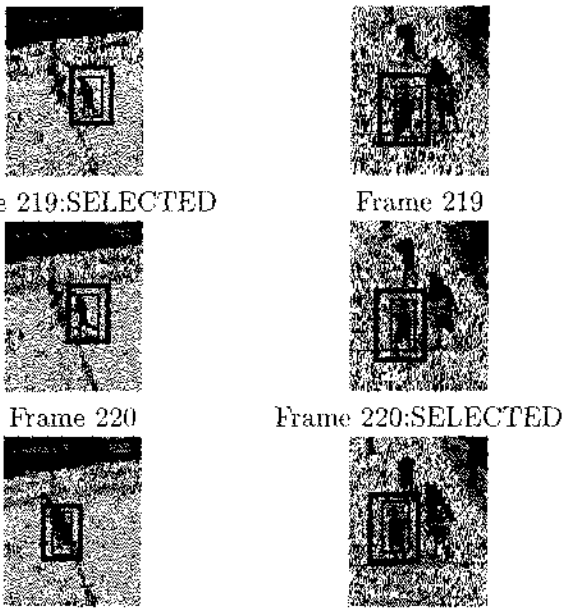


(a) Right View (b) End View

Figure 1.2: **The system input and output of the first process.**

1.2 The Problem

Unlike most existing tracking systems, this system is not trying to seek an on-line or real-time solution. Instead, the system produces off-line solutions



(a) Right View (b) End View

Figure 1.3: **The system output of the second process:** The player in the blue bounding box is our focus of interest. The views that are selected are the better view of the two for the given player at that frame.

in four dependent processes to tackle two major problems: detecting and tracking moving hockey players from two synchronized video sequences in two processes, followed by the other two processes which respectively select the best view and smooth the trajectories.

The video sequences are 1986×1117 high-resolution hockey games originally videotaped by two stationary HDV camcorders. The right-view camera can only cover the right part of the rink, while the end-view can cover the whole rink.

The task of detecting and tracking hockey players is to automatically estimate the locations and sizes of the hockey players on the ice rink coordinate system as well as keeping the identities of the players given the two synchronized video sequences. The static background simplifies the problem, so the first process starts with background subtraction followed by blob detection. The next step, the joint detection step, maps the blob detection

results from both views to the ice rink and obtains the positions of the players in the rink coordinate system. The two-view tracking system is built based on the work of Okuma et al. [50]. We replace the AdaBoost detection results used by Okuma et al. [50] with the blob detection results. We maintain two reference color models and two observation likelihoods, which both contribute to the updating of the posterior distribution. Compared with previous versions, the new two-view tracking system is more robust in the presence of occlusion and overlap.

Automatically switching among multiple cameras and selecting the best view is an interesting topic in sports broadcasting. To the best of our knowledge, currently in the hockey game broadcasting, most of the view selection has been processed manually. The camera selection is always puck-centered or play-centered rather than player-centered. However, our system will satisfy the users who are more interested in the performance or positions of the players from different perspectives than the game itself. Although there are many view quality measurements regarding what is the best view, the best view in our application, given a player, is the one where this player is involved in the least overlap and that minimizes switching between cameras. In addition, the motion and trajectories of the players will be smoothed to enhance tracking stabilization.

1.3 Challenges

The challenges mostly come from the ice hockey game itself. The players move very fast and there is a lot of occlusion and interaction involved. Players are sometimes occluded by other objects than players too. In addition, the players often cast shadows on the ice rink, which could confuse the foreground with the background.

The players do not have a rigid form and look different while they are performing different actions or are in different postures. Therefore, it could be difficult to construct a reliable appearance model of the players. Furthermore, the trajectories of the players are usually highly non-linear and non-Gaussian, making the prediction of the players' positions in the future frames more difficult.

1.4 Outline of Thesis

The remainder of the thesis will be organized as follows. The next chapter covers basic probabilistic models and related theory for tracking. We also

review some existing multi-view tracking systems with the emphasis on finding correspondence among cameras. A variety of methods has been listed in this area. Some related work on automatical camera selection has been presented and the basic criteria for a best view are covered. Each of the following chapters corresponds to a component of the tracking system.

In the third chapter, joint detection is presented in three steps. First, background subtraction is applied in both views, since the cameras are static. The pre-processing, such as deinterlacing, will be presented as well as the details of background subtraction. The second step is the single-view blob detection, which takes the foreground images from the first step as the input. We use the two functions provided by Matlab, `imclose` and `bwlabel`, to merge the scattered blobs and label them all. A linear interpolation system is implemented to calculate the scales of the players, and how many pixels a blob should have if the blob contains a single player. This can help us split the merged blobs and remove the noise resulting from illumination changes. The last but the most important step is the joint detection. We merge the detection results from the views on the global ice rink. Homography is used for the mapping between the image coordinate system and the rink coordinate system. Occlusion analysis and noise removal are also conducted at this stage.

The results from the third chapter, positions and occlusion information will be sent into multi-view tracking using particle filters. Details about particle filtering and how we extend a standard particle filter to a new one with one single hidden state and two observations will be covered. We reviewed the probabilistic models in the new hockey tracking system, and the similarities with and the improvement from the previous versions, followed by experiments and results.

The fifth chapter is trajectory smoothing, very important to create a personalized video for each player with smooth motion. Details about particle smoothing, especially the Maximum *a posteriori* (MAP) smoother as well as the Viterbi algorithm, are presented. Results are shown and compared with particle filtering and Kalman smoothing.

The sixth chapter is about automatic handoff among best-view cameras. Hidden Markov Models (HMM), the probabilistic models we fit the camera selection scenario in, have been presented. We review the criteria of the best views in our hockey tracking and the HMM in the context of camera selection.

In the last chapter, we summarize our tracking system and propose future work.

Chapter 2

Related Work

2.1 Filtering

In the field of object tracking, *Bayesian Sequential Estimation* [3], also called *Bayesian Filtering*, is the most widely accepted framework. In this section, we present the fundamental theory of Bayesian filtering as a preparation for its sequential Monte Carlo implementation, also known as *Particle Filters* [19]. This is also the basic scheme of our hockey tracking system. In addition, we introduce some previous work that used two major implementations of Bayesian filtering: *Kalman Filters* and *Particle Filters*.

2.1.1 Bayesian Sequential Estimation

During the tracking process, the evolution of the targets through time can be interpreted as a state transition process, where each state can be represented as a vector including the location, scale and velocity of the target, and any value required to represent the complete knowledge of the target. Here, we denote the state variable as $x_t \in \mathbb{R}^{n_x}$, where $t \in \mathbb{N}$ indicates the time and n_x is the dimension of the state variable.

In the real world, the state variables are always difficult to measure directly. For example, the task of a tracking system is to estimate the state variables or the ground truth of the targets through some given observation or measuring devices. Therefore, such state variables are called *hidden variables* or *latent variables*. The observation of the hidden variables are denoted as $y_t \in \mathbb{R}^{n_y}$, where n_y is the dimension of the observed variable. Usually $x_t \in \mathbb{R}^{n_x}$ and $y_t \in \mathbb{R}^{n_y}$ are in different spaces, therefore n_x and n_y are not necessarily the same. The relation between the two variables can be represented by a generative model and the corresponding graphical model in Figure 2.1. The directed graph structure shows that the observed variable only depends on its corresponding latent variable and the conditional probability lying on the edge can be represented as $p(y_t|x_t)$.

The *State Space Model*[3] is known as an appropriate model for sequential data, such as the tracking data. It links the hidden variable of each time

Figure 2.1: The graphical model of the relationship between an observed variable y_t and a latent variable x_t, $p(y_t|x_t)$

step as a chain to represent the dependence relationship between the latent states through the time. Assuming that we can represent sequential data using a Markov chain of hidden variables, with each observation conditioned on the state of the corresponding hidden state, the graphical model of the simplest State Space Model can be represented as in Figure 2.2. The hidden states of the State Space Model are continuous, which suits most tracking applications, where the locations of the targets are often represented in a continuous 2D or 3D space.

Figure 2.2 also forms the foundation for both the *Hidden Markov Model* (HMM) [3] and the *Linear Dynamical System* [9]. An HMM can be viewed as a specific instance of the State Space Model, in which the latent variables are discrete. So the transition probability on the arrows between latent variables in Figure 2.2 is an $M \times M$ matrix, given the number of possible values M. The observed variables in an HMM can be discrete or continuous, and a variety of different conditional distributions can be used to model them. If both the hidden and observed variables are Gaussian, we obtain the Linear Dynamical System. A specific instance of Linear Dynamical Systems is an *autoregressive* or AR model [9]. It uses linear-Gaussian conditional distributions in which each hidden state has a Gaussian distribution whose mean is a linear function of its parents. The HMM will be covered later in the thesis. For more details of the Linear Dynamical System, please refer to [9].

In most cases, the evolution of the hidden states are modeled as a first-order Markov chain, as shown in Figure 2.2. A first-order Markov chain has the following conditional independence property, also known as the Markov

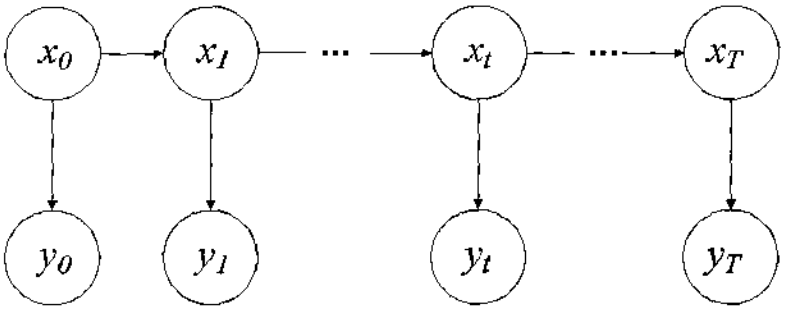


Figure 2.2: The graphical model representation of the State Space Model

assumptions or the Markov property [9],

$$x_t \perp y_{0:t-1} | x_{t-1},$$
$$y_t \perp y_{0:t-1} | x_t,$$

which means that the current hidden state is independent from the past observations given the previous hidden state, and the current observation is independent from the past observations given the current hidden state. Here, we use $x_{0:t} \triangleq (x_0, ..., x_t)$ and $y_{0:t} \triangleq (y_0, ..., y_t)$, respectively, to denote the hidden variables and observed variables up to time t.

To sum up, the three components in Bayesian sequential estimation are the prior distribution $p(x_0)$, the transition model between two consecutive hidden states $p(x_t|x_{t-1})$ and the observation model $p(y_t|x_t)$. The transition model and the observation model are also represented as Equation 2.1 and Equation 2.2 [9], respectively,

$$x_t = f_t(x_{t-1}, v_{t-1}), \tag{2.1}$$

$$y_t = h_t(x_t, u_t), \tag{2.2}$$

where $f_t : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_x}$ and $h_t : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ can either be linear or nonlinear, v_t and u_t are sequences of independent and identically distributed (i.i.d.) process noise and measurement noise respectively, and n_v and n_u are the dimensions of the noise vectors.

The goal of Bayesian sequential estimation is to estimate the posterior distribution $p(x_{0:t}|y_{1:t})$ or its marginal distribution $p(x_t|y_{1:t})$. The marginal

posterior distribution can be computed by the following two steps [19] recursively based on the Markov assumptions,

$$\begin{array}{ll} \textbf{Prediction step:} & p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \\ \textbf{Updating step:} & p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t} \end{array} \tag{2.3}$$

However, the evaluation of the complex integrals in 2.3 is intractable, so we need to make approximations for these distributions.

2.1.2 Kalman Filter

The Kalman filter [5] is one of the solutions to the Bayesian sequential estimation and has been widely used in visual tracking [18, 32] and robotics [33]. However, Kalman filters can perform filtering only on linear Gaussian state space models, more specifically, only when both function $f(\cdot)$ in Equation 2.1 and $h(\cdot)$ in Equation 2.2 are linear and Gaussian.

In order to handle the non-linear and non-Gaussian situations, extensions have been made to the standard Kalman filter. The *Extended Kalman Filter* (EKF) [3] is one of the extensions which uses a first order Taylor series expansion of the nonlinear functions in Equation 2.1 and Equation 2.2 for the approximation. The *Unscented Kalman Filter* (UKF) [59] has higher accuracy, because it approximates the posterior $p(x_t|y_{1:t-1})$ directly instead of approximating the nonlinear functions $f(\cdot)$ and $g(\cdot)$. In particular, it uses a deterministically selected sample set and passes it through the true nonlinear function to capture the true distribution. Because of its capability of handling nonlinear models and its efficiency in computation, it has been successfully used in tracking [16].

2.1.3 Particle Filter

The particle filter gained its popularity because it can model non-linear and non-Gaussian components in real data. The key idea is to approximate the posterior distribution $p(x_t|y_{1:t})$ with a set of i.i.d. samples or particles $\{x_t^{(i)}; i = 1, ..., N\}$ with weights associated to them, where $x_t^{(i)}$ is the sample, $w_t^{(i)}$ is the weight, N_s is the number of samples. The posterior distribution is approximated by iteratively propagating the particles according to the proposal distribution and updating the weights. More details of particle filtering are presented in Chapter 4.

Particle filtering has been widely used in computer vision and tracking systems. It was first introduced to visual tracking by Isard and Blake [29],

where it was known as the CONDENSATION method. Pérez et al. [28, 62] extended the particle filter framework to track multiple targets. Okuma et al. [50] further extended it [62] by incorporating a boosting detector [63] into the particle filter initialization and the proposal distribution. Both [62] and [50] used a *Mixture of Particle Filters* (MPF) as the main framework to maintain multi-modality. The key contribution of MPF is to represent the overall posterior distribution with an M-component mixture model

$$p(x_t|y_{0:t}) = \sum_{m=1}^{M} \pi_{m,t} p_m(x_t|y_{0:t}) \tag{2.4}$$

where $\sum_{m=1}^{M} \pi_{m,t} = 1$, M is the number of targets or particle clouds. p_m represents the distribution of each target and evolves independently over time. The only interaction between targets is through $\pi_{m,t}$. Particle clouds are allowed to merge or split if different targets have significant overlap or overlapping targets split into two. Particles can be associated to different targets during merging or splitting. When new targets come into the scene, some of the particles from existing targets are allocated to the new targets because the total number of particles in the MPF framework is fixed during the initialization. As a result, as more and more new targets come into the scene, the particles of each target will decrease and not be able to properly approximate the distribution. In addition, it cannot track a variable number of targets and maintain their identities at the same time. Therefore, we track each target with a set of particles independently instead of the MPF framework in our system.

2.2 Multi-view Tracking

In recent years, multi-target tracking [27] using multiple cameras has attracted much attention in the computer vision community. This is because by using multiple cameras the area of surveillance is expanded and information from multiple views is extremely helpful to handle many issues such as occlusion. However, visual surveillance using multiple cameras also brings a number of problems such as correspondence across multiple cameras, calibration of multiple cameras, data fusion, and automated handoff among best-view cameras. Associating objects in different cameras and camera calibration (considered as one prerequisite for constructing the correspondences across cameras under some circumstances) will be shortly discussed. Automated camera switching will be presented in the next section.

2.2.1 Correspondence across Multiple Cameras

Only after correspondence among multiple cameras is well constructed can the information from multiple cameras be fused. So, associating objects in different cameras is one of the most important and underlying issues in tracking with multiple cameras. Approaches to the correspondence across views problem can roughly be divided into two parts [15]: recognition-based methods and geometry-based methods.

Recognition-based correspondence is trying to extract recognizable features, such as color and apparent height [12, 14, 17, 49], from multiple views to associate identities among multiple cameras. Previous attempts, such as [51, 56], apply the color information extracted from one camera as a model to match subjects in the other cameras without any transformation.

It is natural and simple to use color information to construct correspondence among multiple cameras. However, color-based correspondence across multiple cameras is not effectively reliable for all applications. On one hand, it greatly relies on the color of people's clothes. When different people wear clothes with similar color, this method may produce wrong correspondences. On the other hand, viewpoint difference (clothing is white from the front and black from the back) and lighting variations may cause the same person to be observed with different colors in different cameras. To address this problem, the feature-base correlation of visual information between different cameras in [15] is learnt using Support Vector Regression and Hierarchical PCA to estimate the appearance across cameras. The camera then uses the learnt correlation to transform the information among cameras.

Geometry-based method transforms the geometric features to the same spatial reference before matching is performed. Such methods always use 2D information from homographies among views [65] or 3D information from camera calibration [10, 42].

Yue et al.[65] describes a two-view tracking approach which could handle occlusions by using the homography transformation between two views. An adaptive appearance-based model with adaptive noise is incorporated in the particle filter to realize robust visual tracking. When there is occlusion in one view, the homography from this view to other views can be robustly estimated from previous tracking results and used to infer the correct transformation for the occluded view. Black et al.[10] presents a multi-view tracking method to resolve occlusions using a set of calibrated cameras. The homography relations between each camera view is recovered to associate objects in the multiple views. A combined 2D/3D Kalman filter has been employed for object tracking, where the 2D Kalman filter state is

used to estimate the observation uncertainty of the 3D Kalman filter.

As the correspondence of the players in different views is constructed using the geometry-based method, the thesis will focus on this method. In the next subsection, we will present the geometry-based method and some commonly used schemes associated with this method.

2.2.2 Planar Assumption

A more applicable scheme for multi-view tracking is to maintain a tracker for each camera to separately track the targets from every single view. By using correspondence information from either camera calibration or homographies relations, all the evidence collected from multiple cameras, such as detection and tracking results and occlusion information, will be mapped and further incorporated to a common *ground plane* shared by all the views. The tracking results from each single camera will be improved after information sharing and data fusion. The common ground plane either exists or is assumed to exist. A lot of recent work has been assuming the targets are within the common ground plane of the multiple views.

Otsuka and Mukawa [52] presented multi-view occlusion analysis for tracking densely populated objects moving separately on a 2D plane using a set of calibrated cameras. It first explicitly modeled the spatial structure of the occlusion process between objects and their uncertainty. The problem is then formulated as recursive Bayesian estimation consisting of hypothesis generation of the occlusion structure and the estimation of the posterior probability distribution.

The system by Mittal and Davis [42] set up multiple calibrated and synchronized surveillance cameras. From all the views, a region-based stereo algorithm was introduced to find 3D points inside an object and employ Bayesian pixel classification with occlusion analysis to segment occluded people. The evidence obtained from multiple views is combined for further tracking.

The work by Kim and Davis [35] is similar to [42]. However, unlike M2Tracker's requirement of having calibrated stereo pairs of cameras, it employs a ground plane homography instead of strong calibration. Human appearance models are used to segment foreground pixels obtained from background subtraction. A top view is reconstructed after integrating segmented blobs across views with the help of the ground plane homography.

Hu et al.[26] proposed a principal axis-based method to find correspondence across multiple cameras. Principal axes and ground-points of people in each single camera view can be well detected even when the people are

in a group or under occlusion. The correspondence likelihood reflecting the similarity of pairs of principal axes is constructed based on the geometric fact that the intersection of the principal axis of a person in a view and the transformed principal axis of this person from another view using homography relations correspond to the ground-point of this person in the first view.

Khan and Shah [34] resolved occlusions and determined locations of people in a crowd on the ground plane. They presented a planar homography constraint that combined foreground likelihoods from all views into a reference view, and segmented out the blobs that represents the feet of the people. Feet regions belonging to the same person form contiguous spatio-temporal regions that are clustered using a graph cuts segmentation approach. Therefore, each cluster is the track of a person and an instance in time of this cluster is the location. However, in real applications, the points of people's feet may not be robustly or accurately detected or even may be invisible due to occlusions.

Fleuret et al.[22] performed occlusion analysis on the ground plane using multiple calibrated cameras. The system takes synchronized video streams taken at eye level and from different angles as the input, obtains the binary images via background subtraction, and builds the probabilistic occupancy map (POM) as a very robust estimation of the occupancy on the ground plane in individual time frames. These probabilities of occupancy are combined with a color and a motion model to track individuals using the Viterbi algorithm.

The multi-camera tracking approach proposed in [21] provides collaborative particle filters in the ground plane as well as in each camera. The particle filter in each camera passes messages to those in the ground plane where the multi-camera information is integrated using the homography of each camera. These fusion results are then incorporated by the trackers in each camera as boosted proposal functions. The collaboration among all the cameras largely relaxes the dependence on precise foot positions when mapping targets from images to the ground plane using homographies.

Similarly, in our hockey tracking system, the hockey players are first detected separately in both fixed viewpoints using blob-detection after performing background subtraction. The detection results from the two cameras are fused on the ice rink coordinat system after mapping the foot positions from both views using homography transformation. The noise due to the shadow and motion will be removed as much as possible in the joint detection process.

However, if the moving objects violate the ground plane constraints, the

matching under the planar assumption will become less reliable. The work in [43] does not require the information from feature matching, camera calibration or planar assumptions. It exploits the spatial and temporal correlations presented in multiple camera views using nonlinear manifold learning and target dynamics. The coordinates of the observed objects can be extracted, which makes finding correspondences simpler. This is actually a nonlinear dimensionality reduction method to map the high dimensional images into low dimensional manifolds which could preserve neighborhood properties of the original data.

2.2.3 Multi-view Tracking in Soccer

A soccer game is a perfect context where the planar assumption is allowed. Usually, fixed cameras are set up from sides to cover the entire soccer field. The location of the soccer players from the top view can be generated using camera calibration or homography transformation.

Iwase and Saito [30] proposed a method for parallel tracking of all soccer players on the ground using fixed multiple-view cameras. Positions of players are detected in each camera, and then the detected positions are projected onto the virtual top-view image via homography of the ground between the camera image and the top-view image.

The input of [64] is eight video streams from static cameras, each processed to generate measurements for the multi-view tracker. The single-view processing includes foreground detection and generating a bounding box tracker for each player. In the multi-view process, the measurements, each from a different camera and matched to the same player, are fused into an overall measurement using Kalman Filtering.

The planar assumption will also apply in hockey games. However, their assumption of smooth and constant motion of players and the use of Kalman Filtering will not hold true for hockey games. Hockey is a much faster game than soccer, and motion of the hockey players is non-linear and non-Gaussian.

2.2.4 Other Models and Algorithms in Multi-view Tracking

Other models and algorithms, such as integer programming [44], linear programming [54, 57], dynamic programming [7] and belief propagation [58] have been used in tracking. Jiang et al.[31] presented a linear programming relaxation scheme to optimize multi-target tracking simultaneously through multiple cameras by specifying constraints in both spacial layout and mu-

tual occlusion. The multi-object tracking was formulated as a multi-path search problem, where each path consisted of a sequence of states (locations and appearances of an object through time) represented by nodes. By imposing constraints, such as the relative positions of the objects cannot change abruptly from frame to frame and different tracks could not occupy the same spatial region at the same time, the algorithm could add edges to the nodes to finalize the optimal path. The linear programming relaxation could solve the path searching problem even when the paths are overlapped and the objects are occluded. The relaxation solution would be rounded to an integer to obtain the object locations and occlusion states.

2.3 Best View Selection

Selecting the best-view camera is an intuitive extension of multi-view tracking and an interesting way of using the multi-view data. The best view for a specific object at a specific frame has to be selected among all the cameras by using some view quality measures. Different measurements and criteria have been used in previous work.

In Computer Graphics, the best view is sometimes defined as the one giving the most information of the object being inspected. Therefore, measurements such as the projection of polygons or the view entropy [61] can be used for the best view selection. Similarly, Vazquez et al.[60] took enough images to reconstruct the scene as the input of image-based rendering. The goal is to reduce the data redundancy as much as possible to minimize the number of images needed to represent a scene. Therefore, the criterion for the best view selection was also the amount of information seen from a viewpoint or view entropy.

In the context of 3D visualization of urban rescue simulations, the aim of [45] was to provide the most pertinent information ensuring a good understanding of the evolving environment. Therefore, it used measurements such as visibility, relevance, redundancy and eccentricity for the best view selection and an optimization step can be further applied to the whole sequence.

In video surveillance applications, facial appearance [49] is a natural feature for determining the view quality. The multi-camera setup can see different parts of the person's head from different prospectives at the same time. Camera hand-over is controlled on the basis of the tracking results by evaluating the Bhattacharyya coefficient of each tracker with respect to the target color histogram. The system will take the camera view which

provides the highest Bhattacharyya coefficient as the best view on the face of the person. 3D locations of objects and their relative poses to cameras are other features [48] that are often used in best view selection.

Even though view quality measurements have been extensively studied, simply selecting the best view at each frame based on the best-view measure usually works poorly and may contain annoying abrupt view changes. These naive methods of video synthesis will not guarantee smooth transitions between video sequences. Therefore, the view smoothness as well as the view quality measure has been taken into consideration.

Jiang et al.[31] combined these two criteria very well. He generated the best view video for a specific object by using dynamic programming as a natural extension of multi-target tracking as explained in the previous section. The choice among the multiple cameras always favored the view which had been selected in the previous frame and from which the size of the bounding box for this object could be satisfyingly stable.

Our hockey tracking system builds a Hidden Markov model to automatically select the best-view camera. The best view for a specific player is the one with the minimum overlap between this player and other objects in the scene. At the same time, by setting the proper parameters for the transition model, we can make it more probable to pick the same view which has been selected in the previous frame than to switch to the other views, which can ensure the smoothness of the video sequences.

Chapter 3

Multi-view Detection

3.1 Background Subtraction

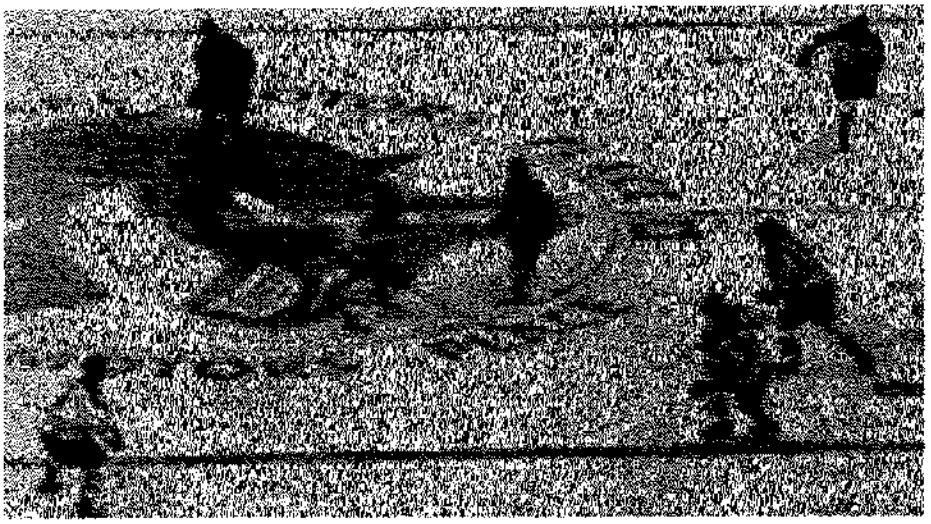
The input in our experiment is two synchronized sequences of 1986 $\times$ 1117 high-resolution digitalized images. Since the original videos are videotaped by two stationary HDV camcorders without panning, tilting or zooming, the single-view detection can be completed by foreground detection after background subtraction. However some pre-processing is needed for background subtraction.

3.1.1 Interlacing and Deinterlacing

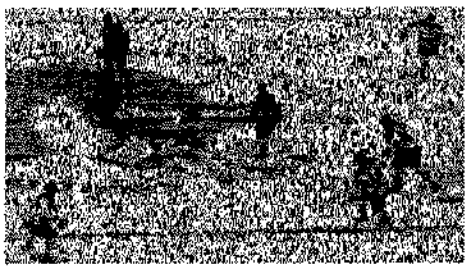
Interlacing and *deinterlacing* are terminologies in the area of digital videos. Field-based videos are recorded and displayed in different fashions. In *interlaced video* or *interlaced display*, the fields that make up a video frame are not recorded or displayed simultaneously, usually the odd field (odd lines of the video frame) first and even field (even lines of the video frame) next. The split of image across field due to the motion is an *interlacing artifact*. Interlacing artifacts may appear as fringes or horizontal lines near the border of the moving objects, as shown in Figure 3.1 (a). *Deinterlacing* is the operation to remove interlacing artifacts. In our application, we remove the even fields of the video and downsize the remaining rows by taking the pixel pair and obtaining the average to maintain the original aspect ratio. Therefore, the images shrink to 993 $\times$ 559. The result is shown as Figure 3.1 (b).

3.1.2 Background Subtraction and the Mask

The background image was taken before the hockey game began. After converting all the original images and the background image from true color images to gray scale intensity images, we simply take the absolute difference between them. If the difference exceeds some threshold (0.1 in our application), a median filtering is applied on that difference using the 3-by-



(a)

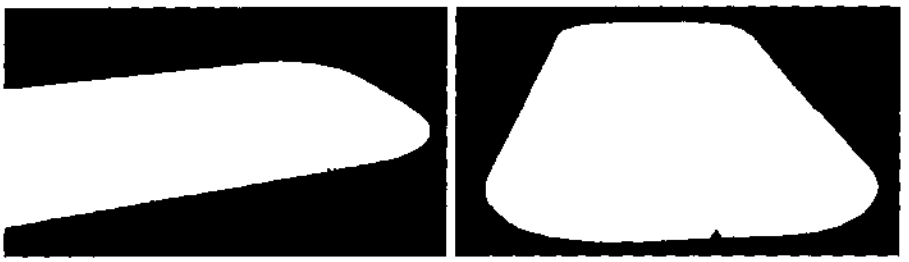


(b)

Figure 3.1: **The interlacing and deinterlacing:** (a) Interlacing artifacts: If you look closely at the both sides of the fast-moving hockey players, you will notice a series of horizontal lines. (b) Result after deinterlacing and removal of the horizontal lines.

3 neighborhood. Otherwise, the pixel is considered as noise. The filtered images are regarded as the foreground images.

This works well for the area inside the ice rink. However, there is quite a lot of noise outside the rink, because the spectators were not sitting there before the hockey game. To remove the noise, we maintained a binary map of the same size with the image after deinterlacing, as shown in Figure 3.2. 1 is set to the ice rink and 0 to the area outside the ice rink. The finalized foreground image will be the blurred image after filtering *logical and* the



(a) (b)

Figure 3.2: **Masks to tell the inside and outside of the ice rink.** (a) The mask for the right view. (b) The mask for the end view.

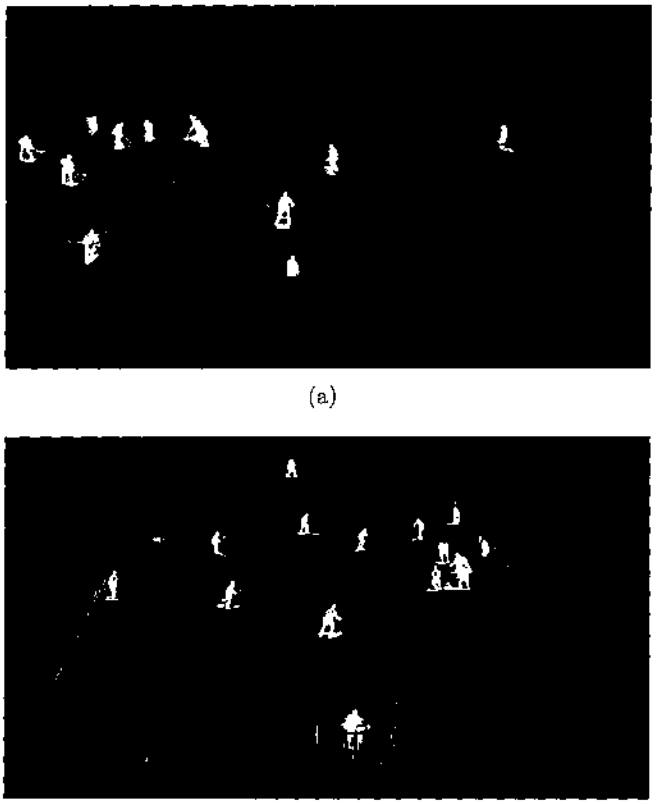
mask, as shown in Figure 3.3.

3.2 Single-view Blob Detection

The images we get after background subtraction have white blobs as the foreground. In the perfect situation, each blob corresponds to a player. However, the background subtraction step is not noise-proof. For example, if the illumination changes after the game starts, white blobs may appear in the background and be mistaken for players. If part of the players' clothes are of the same color as the ice rink, the system may take part of the player as the background, where a single blob can be scattered into different parts. In this case, we need to merge the blob. If the players are really close to each other with occlusion from one view, these players may all correspond to a single blob. In this case, we need to split the blob. In this section, we deal with the merging and splitting of the blobs and detecting the players from single views. The output of the single-view detection is a text file with the foot positions of all the players through the time.

3.2.1 Blob Detection

Our solution is to use a Matlab function, `imclose`, which performs morphological closing on the gray scale images or binary images, and returns the closed image. Essentially, the image closing performs dilation followed by erosion. Dilation adds pixels to the boundaries of objects in an image by convolving with a disc. Erosion removes the convolving pixels on the object boundaries: the pixel near the boundary is on if and only if the full disc is



(a)

(b)

Figure 3.3: **The Background subtraction results:** the finalized foreground image will be the blurred image after filtering *logical and* the mask. (a) The foreground blobs from the right view. (b) The foreground blobs from the end view.

on. Therefore, this function can be used to join the separated parts of the same player together by filling the gaps between them and smoothing their outer edges. The function also takes another parameter to specify the radius of the disc, which is also the largest gap that will be filled during closing. In our application, we used 3. In this way, the scattered blobs merge to one, reducing the confusion between a player and noise. The result is demonstrated in Figure 3.4. The morphological closing operation may have side effects. For example, the noise may still remain in the image since we do erosion after dilation. And it is also possible that the image closing operation may

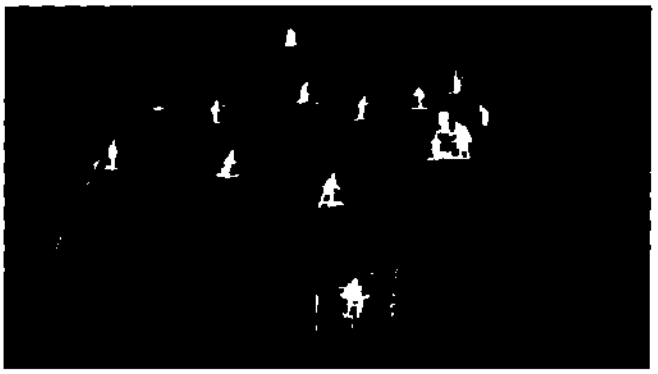


Figure 3.4: The foreground blobs after applying `imclose`. Comparing the nearer goalkeeper in this figure and the one in Figure 3.3 (b), we can see the effect of using `imclose`, which is to glue the scattered parts back to one integral blob.

merge two or more nearby players. To fix the problem, we implement a linear interpolation system and set thresholds on the size of the blob to tell whether the blob contains only noise, one player or more than one player. The details will be covered later.

Another interesting function provided by Matlab is `bwlabel`, which labels the connected components in an image. The function takes a matrix representing the image as the parameter, and returns a matrix of the same size as the input matrix, containing labels for the connected objects. For example, all the pixels whose entries in the returning matrix are equal to 0 correspond to the background in the input image, and all the pixels whose entries are equal to 1 correspond to a blob in the input image. The output matrix is visualized as Figure 3.5. Therefore, the labeling for the connected components provides information such as players' foot positions and scales. Bounding boxes are used to annotate the players in the images.

Foot positions

Thanks to the `bwlabel` function, it is very easy to get the foot position of a player. Assuming each blob or connected component corresponds to a player, we can obtain the foot position of the player by simply locating the pixel at the bottom of the blob, which is the one with the maximum value along the y-axis within that connected component. However, taking the bottom pixel of the blob as the player's foot position could be wrong

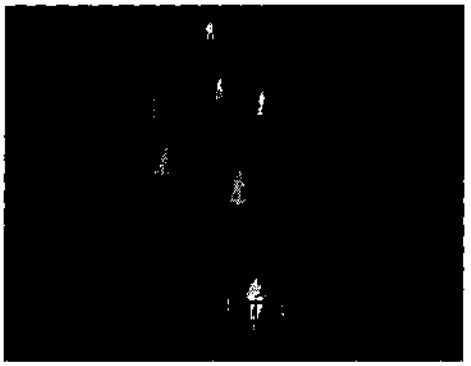


Figure 3.5: The visualization of the labeling of the foreground blob using `bwlabel`. Each connected component has a different color.

when there is shadow cast on the ice rink. The pixel we located is probably the shadow rather than the foot position of that player. We will tackle this problem during the joint detection stage.

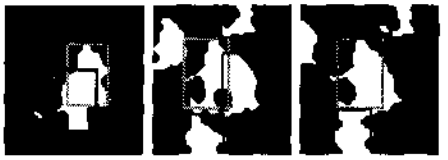
Scales and Aspect Ratios

Each player has two scales and two aspect ratios, respectively in the right view and the end view. The players may be of different sizes and display various shapes and poses during the game, so ideally the bounding boxes should be aware of the difference in the size and vary with the poses. However, for now we maintain only one unified aspect ratio for all the players in each view. From our experience with the system, it works fine.

We maintain an independent linear system for each view, where, given the size and the positions, especially the y component, of both the nearest and farthest persons in a view (a linesman and the referee for the right view, the two goalkeepers for the end view), we can get the rough sizes of all the other people according to their positions by using linear interpolation. The linear system and the scale are very important clues for splitting the merged blobs in the next step.

3.2.2 Splitting of the Merged Blobs

One side effect of performing the image closing is that, the originally separated blobs representing different players may merge. However, some blobs originally contain multi-players with complicated overlapping and interaction, and these players do not merge because of the image closing operation,



(a) (b) (c)

Figure 3.6: **The Splitting of the merged blobs in three different cases:** (a) The aspect ratio is smaller than the threshold, so the players are standing in front of each other. (b) The aspect ratio is larger than the threshold, so the players (the green bounding box and the red one on its right) are standing side by side. (c) Blobs (the green bounding box) may originally contain multi-players with complicated overlapping and interaction.

as in Figure 3.6 (c). At the single-view detection stage, we are not trying to split these blobs due to the lack of information. Therefore, we will deal with these blobs during the joint detection.

Before trying to split the merged blobs, we should first find a way to tell which blobs are suspected to contain more than one player. The number of the pixels within a blob is a good way to tell. By using the same idea of the linear interpolation system and how we get the sizes of the players, we can obtain the number of pixels a blob at a specific position containing a single player is supposed to have. If a blob at the same position contains a lot more pixels (parameter controlled) than that, this blob may contain more that one player. Since we ignore the poses of the players, this is only a simple approximation and not totally reliable. More information will be recovered when we combine the the two views.

For the blobs which are merged during the image closing operation, we are trying to split them only when the layout of the players are clear. For example, if the aspect ratio is drastically smaller (parameter controlled) than the unified one for this view, the players in the blob are standing in from of each other, shown as in Figure 3.6 (a). If it is a lot larger (parameter controlled) than the unified one, the players must be standing side by side, shown as in Figure 3.6 (b). To split these blobs, we need the information about the position, the scale, the aspect ratio and the pixel number. The splitting process is from outside to inside.

(a) (b)

Figure 3.7: To get rid of the hockey sticks inside the bounding boxes, split the blob and pick the part with more pixels. (a) Before the operation, the bounding box also includes the stick. (b) After getting rid of the stick.

3.2.3 Getting the Hockey Sticks Out of the bounding boxes

Since the images are high-resolution, the hockey sticks are clearly presented after the background subtraction, especially after the operation `imclose`. This will increase the sizes of the bounding boxes during blob detection and thus affect the calculation of color histogram in the tracking process. Therefore, we have to figure out a way of reducing the bounding box to exactly covering the hockey player.

Blobs with a regular pixel number but a bigger size are more likely to contain a hockey stick. Since the stick is either on the left or right side of the player, the main idea of removing the sticks is to split the blob into the left part and the right part according to the scale and aspect ratio. The part with the hockey stick contains fewer foreground pixels than the other part. The other part should exactly include the player. The result is demonstrated in Figure 3.7.

The results of this section are the foot positions of the players from both the right view and the end view, respectively. An example of the final result of the single-view blob detection is shown in Figure 3.8. The results, especially the merged blobs, will be refined during the joint detection step.

3.3 Homography and Joint Detection

In this section, the detection results from both views will be mapped to the global rink for information integration. The noise that are not removed yet and the merged blobs that are not split yet from both views will be resolved. The basic knowledge of homography will be presented, followed by the details of joint detection.

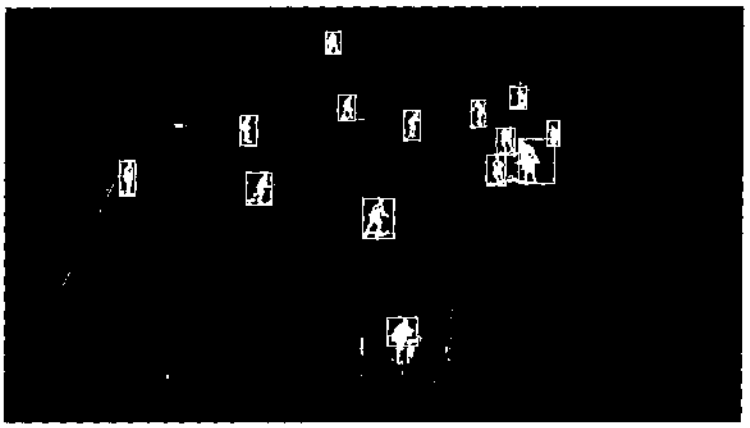


Figure 3.8: An example of the single-view blob-detection result from the end view. As you can notice, there is a unexceptionally big bounding box which is not split. The reason is we are not sure about the layout of two or more players merged in the blob. This will be refined in the joint detection step.

3.3.1 Homography

Homography, also known as *projectivity* or *collineation*, is defined by Hartley and Zisserman in [25] as follows:
Definition. A *projectivity* is an invertible mapping h from $\mathbb{P}^2$ to itself such that three points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ lie on the same line if and only if $h(\mathbf{x}_1)$, $h(\mathbf{x}_2)$ and $h(\mathbf{x}_3)$ also lie on the same line.

Images recorded by cameras are 2D projections of the real world through lenses. For any plane in the world, its images from a camera are exactly the projection described above because any line in the world is still a line in the images as long as there is no noticeable lens distortion. As the hockey players are always moving in the plane formed by the ice rink, they are in the same plane both in the rink coordinates and the image coordinates. Therefore, it is possible to project their locations between the two planes.

The mathematical way to accomplish such projection is possible because of the following theorem [25]:

Theorem. *A mapping h: $\mathbb{P}^2 \rightarrow \mathbb{P}^2$ is a projectivity if and only if there exists a non-singular 3×3 matrix $\mathbf{H}$ such that for any point in $\mathbb{P}^2$ represented by a vector $\mathbf{x}$ it is true that $h(\mathbf{x}) = \mathbf{H}\mathbf{x}$.*

Here, the vector $\mathbf{x}$ can be written as $(x, y, w)^\top$, which is a *homogenous*

representation of a point in the plane. The corresponding inhomogeneous representation of the point in Euclidean 2D space is written as $(x/w, y/w)^\top$, where x/w and y/w are the x-axis and y-axis of the point. According to the theorem, given the coordinates of a pair of matching points $\{(x, y), (x', y')\}$, in two planes, the transformation can be written as

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \tag{3.1}$$

where $\mathbf{x}'$, $\mathbf{x}$ and $\mathbf{H}$ are defined as:

$$\mathbf{x}' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \tag{3.2}$$

For all point pairs $\{\mathbf{x}'_i, \mathbf{x}_i\}$ from the two planes, Equation 3.1 holds for the same matrix $\mathbf{H}$. Since we are using a static camera, it holds for all the frames throughout the video sequence. In order to compute $\mathbf{H}$, at least four pairs of points are needed because there are 8 unknown entries.

An important property of the homography is that it is an invertible transformation, which makes it easy to map the points in the images and rink in both directions in the following way with the same transformation matrix H^{fr}.

$$\begin{aligned} &\textbf{from the image coordinates to the rink coordinates:} \quad \mathbf{x}^r = H^{fr}\mathbf{x}^f \\ &\textbf{from the rink coordinates to the image coordinates:} \quad \mathbf{x}^f = H^{fr^{-1}}\mathbf{x}^r \end{aligned} \tag{3.3}$$

where $\mathbf{x}^r$ and $\mathbf{x}^f$ are the points on the rink and image coordinates respectively.

In our application, H^{rr} and H^{er} are the homographies from the ice rink to the right view and from the ice rink to the end view, respectively. After the homography transformation, we have the foot positions on the ice rink mapped from both the right view and the end view.

3.3.2 Joint Detection

Detection results from the right view and the end view are incorporated on the ice rink. We will analyze the occlusion among players as well as handling the problems we cannot fix during the single-view detection stage.

Mapping the Single-view Detection to the Ice Rink

In the perfect case, the foot positions of the same player from the right view and the end view are mapped to the same position on the ice rink using

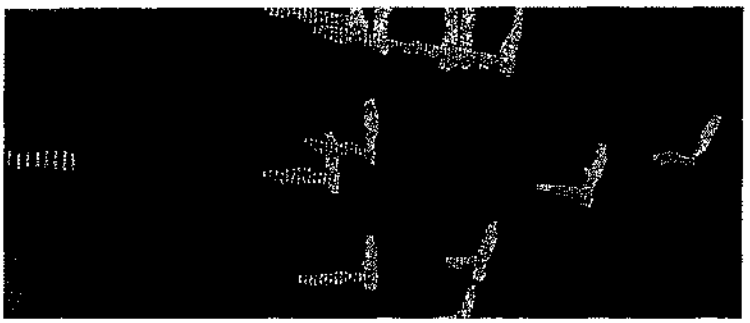


Figure 3.9: This is the result of mapping the blobs of the same frame from the two views onto the ice rink. The red figures are from the right view and the blue figures are from the end view. As you can notice, the leftmost blue figure does not have a match. This is because this player is not covered by the right camera. The foot positions of the figures matched as a pair cannot always map to exactly the same position on the ice rink due to the existing of shadows.

the homography transformation presented in the last section. However, as shown in Figure 3.9 and Figure 3.10, this is not always true. Due to occlusion, some players are not displayed from certain views. The shadow of the players on the rink also accounts for the fact that the results from two views cannot perfectly match.

Matching the Detection Results from Both Views

The players have been detected from both the right view and the end view, so for each player we have two detection results on the rink after the homography transformation. The coupling of the two detection results for the same player is the first step of information integration. It is important for occlusion analysis and noise removal.

The coupling is based on finding the nearest neighbor. For each detection result mapped from one view, we calculate the distance from this detection result to all the detection results mapped from the other view on the rink, and consider the one with the shortest distance as its match. However, it is possible that two detection results from the same view have the same detection result from the other view as the nearest neighbor. Therefore, finding the nearest neighbor has to be done in both directions, both right-to-end and end-to-right, to ensure it is a one-to-one mapping.

The player is not always visible from both views either because he may

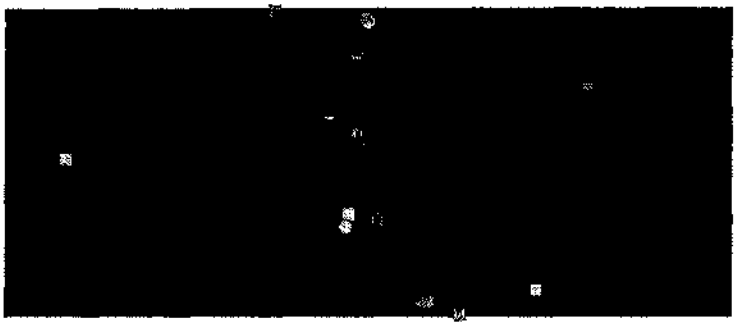


Figure 3.10: This illustration is similar with Figure 3.9 except that it only shows the foot positions. The dots are from the right view and the squares are from the end view. The dot and the square of the same pair are not always exactly mapped to the same position either.

not be in the camera coverage area or he is occluded by other players. To take this into consideration, we set a threshold, so that if the distance between the detection result and its nearest neighbor from the other view exceeds the threshold, they are not considered as a match. The two detection results from the two views are considered as a couple only if they claim each other as the nearest neighbor and the distance between the two detection results on the ice rink is lower than the threshold.

Occlusion Analysis

The occlusion analysis is very naive and based on the matching of detection results from two views. Any detection result from one view without a match from the other view implies that this player is only visible from this view. He is either occluded by other players or not covered by the camera in the occluded view. The occlusion analysis is based on this clue.

Jiang et al.[31] demonstrates the matching of detection results from multiple views and occlusion analysis as an optimization problem over the whole video sequence.

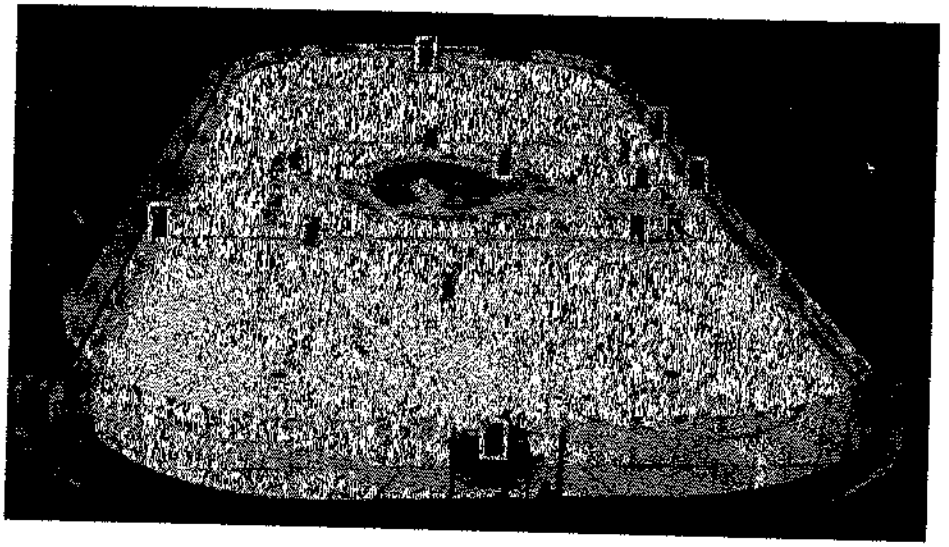
Noise Removal

Noise is mainly introduced by the shadows of the players cast on the ice rink. It is also the major reason why the detection results from the two views cannot be mapped to exactly the same position on the ice rink. Since the shadows are usually located below players' feet, there are more errors in

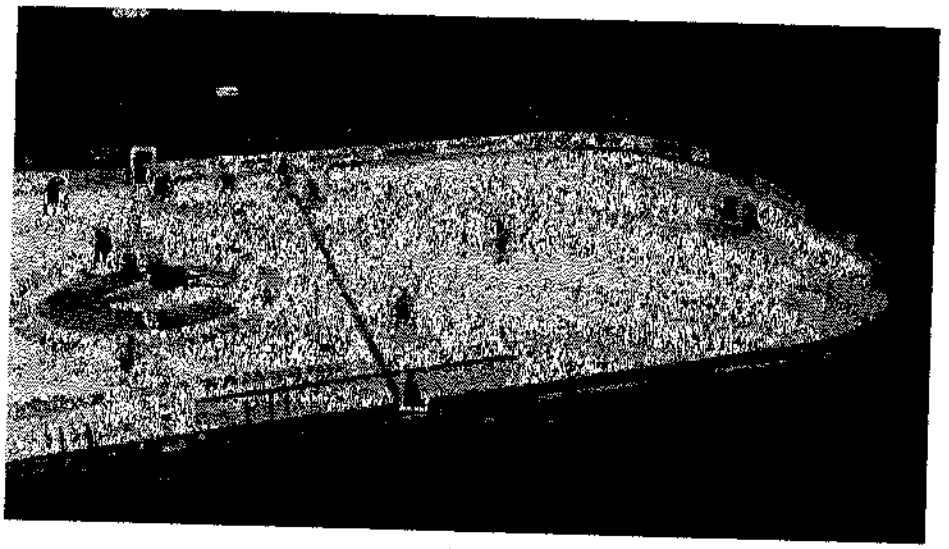
the y-axis than in the x-axis in the image coordinates. After the detection results from both views are mapped to the rink coordinates, due to the camera setup, the right-view detection results contain more errors in the y-axis and the end-view results contain in the x-axis. In our application, we take the x coordinate from the right view and the y coordinate from the end view as the final detection results. However, this method is not very robust and the results are not totally noise-proof. A better choice is to get more ground truth of the shadows and model the priors for them.

3.3.3 Results

The information is stored in two matrices after the joint detection. One is the combined positions on the ice rink obtained from two-view joint detection, for all players in all frames, as visualized in Figure 3.11. The other stores the occlusion information for all players from both views. It is like a binary map, with 0 meaning unoccluded and 1 meaning occluded or not covered by the camera. Both matrices will be sent to the two-view tracking in the next chapter.



(a)



(b)

Figure 3.11: **The visualization of the joint detection results:** To compare with the single-view detection, we used the same frame as we used for plotting Figure 3.8. The most obvious improvement as compared (a) with Figure 3.8 is that, now we solve the merged blob that we could not solve in the single-view detection. (a) The end view. (b) The right view. The two players in the right view clearly present themselves. We use the position information from the right view to solve the merged blob.

Chapter 4

Multi-view Tracking

As mentioned before, particle filtering performs well in Bayesian sequential estimation with non-linear, non-Gaussian transition models and observation models. In our system, the motion of hockey players is highly non-linear and non-Gaussian. In addition, because the system uses the observation model in [50, 53, 62], which is an exponential function based on the Bhattacharyya coefficient between two color histograms, the likelihood function is non-linear and non-Gaussian as well. Moreover, particle filtering has been widely applied to multi-target tracking [28, 62]. Therefore, the particle filter model is the ideal choice for the hockey tracking system. This chapter covers the basic theory of generic particle filters, the extended particle filter with one hidden state and two observation models, and the details of the two-view multi-target tracking system.

4.1 Particle Filter

The Kalman Filter is a well-known filter often used in the field of computer vision and object tracking. However, the Kalman Filter cannot handle the non-linear or non-Gaussian components in the real data models. The details of Kalman Filtering are beyond the scope of the thesis. More details can be found in [5]. The most popular numerical approximation technique for this problem is the Sequential Monte Carlo (SMC) method [19], which has the advantage of not being subject to the non-linear or non-Gaussian components in the real data models. It is also known as Particle Filtering [4] or CONDENSATION method [29].

4.1.1 Monte Carlo Simulation

The basic idea of Monte Carlo simulation is to represent the posterior distribution by a set of random samples with associated weights. Assuming that we are able to simulate N independent and identically distributed (i.i.d.) random samples or particles $\{x_t^{(i)}; i = 1, ..., N\}$ according to $p(x_t|y_{1:t})$, an

empirical estimate of this distribution is given by

$$P_N(dx_t|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta_{x_t^{(i)}}(dx_t), \tag{4.1}$$

where $\delta_{x_t^{(i)}}(dx_t)$ denotes the Dirac delta mass located in $x_t^{(i)}$[20].

$$\delta_{x_t^{(i)}}(dx_t) = \begin{cases} 1 & \text{if } x^{(i)} \in dx \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Therefore, with the Monte Carlo simulation, any integral can be approximated by a discrete form

$$\mathbb{E}(f(x_t)) = \int f(x_t)p(x_t|y_{1:t})dx_t \approx \frac{1}{N}\sum_{i=1}^{N} f(x_t^{(i)}). \tag{4.3}$$

According to the law of large numbers, when the number of samples approaches infinity, the approximations of the posterior distribution and the integral are equivalent to the true distribution and integral.

4.1.2 Importance sampling

It is often difficult and infeasible to sample directly from $p(x_t|y_{1:t})$. A possible solution is to use *importance sampling* instead to sample from a *proposal distribution* $q(x_t|y_{1:t})$, from which it is easy to draw samples. The *importance weight* is therefore defined as

$$w(x_{0:t}) = \frac{p(x_{0:t}|y_{1:t})}{q(x_{0:t}|y_{1:t})}. \tag{4.4}$$

Then, Equation 4.1 can be rewritten as

$$P_N(dx_t|y_{1:t}) = \sum_{i=1}^{N} w_t^{(i)}\delta_{x_t^{(i)}}(dx_t), \tag{4.5}$$

If the proposal distribution is the same as the posterior, the weights reduce to the same as in Equation 4.1.

4.1.3 Sequential importance sampling

Importance sampling is a general Monte Carlo integration method. However, it is not adequate for recursive estimation. *Sequential Importance Sampling* is a modification of importance sampling to satisfy the need for recursive estimation.

To make it recursive, the importance distribution $q(x_{0:t}|y_{1:t})$ at time t admits the importance distribution $q(x_{0:t-1}|y_{1:t-1})$ as the marginal distribution at time $t-1$. That is,

$$q(x_{0:t}|y_{1:t}) \propto q(x_{0:t-1}|y_{1:t-1})q(x_t|x_{0:t-1}, y_{1:t}) \tag{4.6}$$

Similarly for the posterior distribution $p(x_{0:t}|y_{1:t})$,

$$\begin{aligned} p(x_{0:t}|y_{1:t}) &\propto p(x_{0:t}|y_t, y_{1:t-1}) \\ &\propto p(y_t|x_{0:t}, y_{1:t-1})p(x_{0:t}|y_{1:t-1}) \\ &\propto p(y_t|x_t)p(x_t|x_{0:t-1}, y_{1:t-1})p(x_{0:t-1}|y_{1:t-1}) \\ &\propto p(y_t|x_t)p(x_t|x_{t-1})p(x_{0:t-1}|y_{1:t-1}) \end{aligned} \tag{4.7}$$

Dividing Equation 4.7 by Equation 4.6, one will get

$$\widetilde{w}_t^{(i)} \propto \widetilde{w}_{t-1}^{(i)} \frac{p(y_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|x_{0:t-1}^{(i)}, y_{1:t})}, \tag{4.8}$$

where

$$\widetilde{w}_t^{(t)} = \frac{w(x_{0:t}^{(i)})}{\sum_{j=1}^{N} w(x_{0:t}^{(j)})}, \tag{4.9}$$

is called normalized importance weights[20].

The most attractive part of the strategy is that the updating of the belief about the posterior filter distribution is performed only through the re-computation of the importance weights without changing the past simulated trajectories. The re-computation is exactly in the form of Equation 4.8.

4.1.4 Resampling

There is one problem in the sequential importance sampling method called the degeneracy phenomenon. The discussion below follows Bolic and Hong's work [11]. After several iterations, the variance of the particle weights quickly increases. Very few normalized weights are substantial and the majority of the computing is wasted on those particles with low weights. The

inference is degraded because it is made by using only a very small number of particles. Therefore, resampling is required to ignore particles with very low weights and concentrate on the more promising ones, which is critical in every implementation of particle filtering.

Resampling was proposed for use in particle filtering in various works including [8, 11, 13, 36, 38–40]. One of the most widely used resampling schemes, which is also used in our system, is the *Sampling-importance resampling* (SIR) technique. The basic idea is to remove the particle trajectories with small weights and replicate the trajectories with large weights. It is implemented by mapping a set of weighted particles $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^{N_s}$ to a set of equally weighted samples $\{\tilde{x}_t^{(i)}, N_s^{-1}\}_{i=1}^{N_s}$ through a discrete cumulative distribution of the original sample set. More details can be found in [24]. Please refer to Figure 4.1 [59] for the whole process of SIR particle filtering.

4.2 Boosted Particle Filter

There are two major criteria for well-designed particle filters, the automatic initialization of the particle filter and the proper design of the importance proposal distribution. The Boosted Particle Filter (BPF), first introduced by Okuma et al. [50], has successfully satisfied both criteria by incorporating the AdaBoost detection algorithm by Viola et al. [63] into the construction of proposal distribution.

As a result of combining the AdaBoost detection, the most recent observations can be included in the construction of proposal distribution. Therefore, the particles are ensured to propagate near the regions with high likelihood even if the mode of the transition prior distribution is far from the mode of the likelihood distribution. The expression of the proposal distribution is in the form of mixtures of Gaussian as following:

$$q_B^*(x_t|x_{t-1}, y_{0:t}) = \alpha q_{ada}(x_t|y_t) + (1 - \alpha)p(x_t|x_{t-1}), \qquad (4.10)$$

where the parameter α can be dynamically tuned without affecting the convergence of the particle filter. When α is set to zero, the algorithm is reduced to a mixture particle filter. By increasing α, we place more importance on the AdaBoost detection. The value of α can be tuned through experiments. In our system, it is set to 0.9. The mixture of Gaussian proposal can be best described by Figure 4.2 [50].

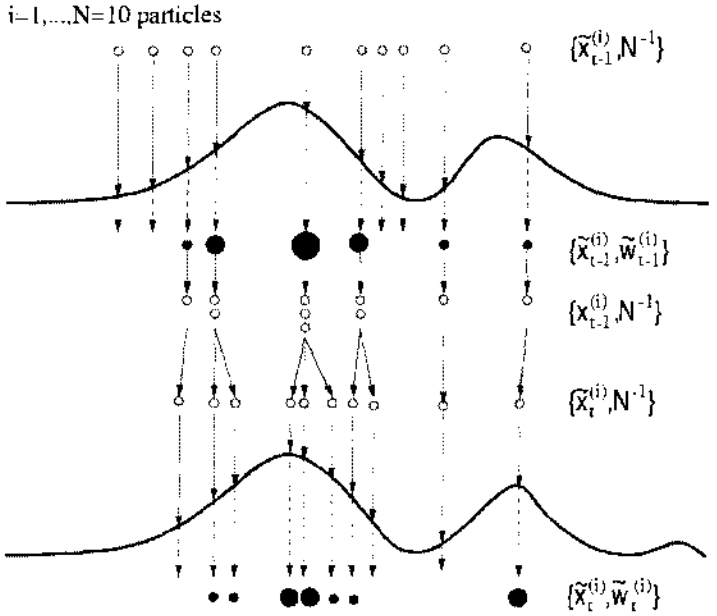


Figure 4.1: The SIR particle filtering process starts at time $t-1$ with a set of unweighted particles $\{\tilde{x}_{t-1}^{(i)}, N^{-1}\}_{i=1}^{N}$, which approximates the distribution $p(x_{t-1}|y_{1:t-2})$. For each particle we update the importance weights with information at time $t-1$, yielding $\{\tilde{x}_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\}_{i=1}^{N}$ as an approximation of $p(x_{t-1}|y_{1:t-1})$. Resampling selects only the promising particles to obtain the unweighted measure measure $\{x_{t-1}^{(i)}, N^{-1}\}_{i=1}^{N}$, which is still an approximation of $p(x_{t-1}|y_{1:t-1})$. Finally, the prediction step is performed, resulting in the measure $\{\tilde{x}_{t}^{(i)}, N^{-1}\}_{i=1}^{N}$ as an approximation of $p(x_t|y_{1:t-1})$. This figure is taken from [59].

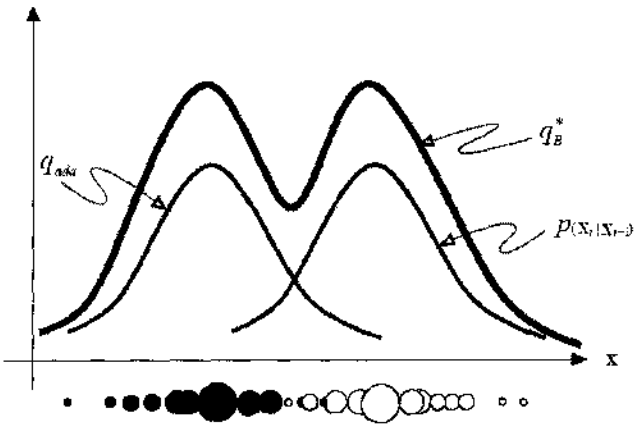


Figure 4.2: Mixture of Gaussians for the proposal distribution which combines the transition prior and the boosting detection. This figure is taken from [50].

4.3 Extended Particle Filter in Two-view Tracking

A visual tracking system must be able to track objects which are partially or even fully occluded. Multi-view tracking has the obvious advantage over single view tracking because of its wide coverage range. When a scene is viewed from different viewpoints there are often regions which are occluded in some views but visible in other views.

The Probabilistic Model in Two-view Tracking

The input of our system is two synchronized video sequences from the right view and the end view. It is very intuitive to extend the standard particle filter to a new one with one hidden state $p(x_t)$ and two observations, $p(y_t^1|x_t)$ and $p(y_t^2|x_t)$, which are totally independent from each other, as shown in Figure 4.3. For the recursive version of proposal distribution, one can replace Equation 4.6 with new observations as following,

$$q(x_{0:t}|y_{1:t}^1, y_{1:t}^2) \propto q(x_{0:t-1}|y_{1:t-1}^1, y_{1:t-1}^2) q(x_t|x_{0:t-1}, y_{1:t}^1, y_{1:t}^2) \qquad (4.11)$$

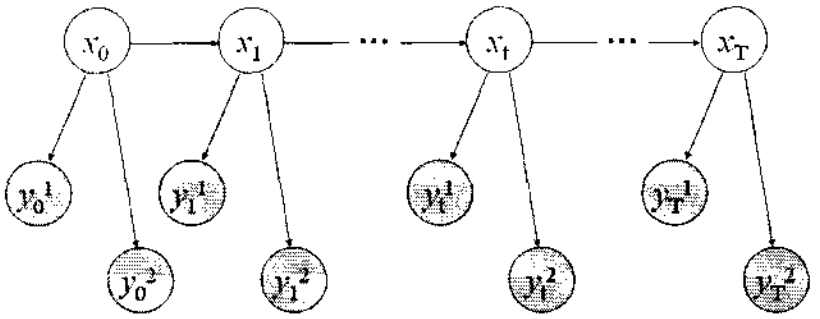


Figure 4.3: The graphical model representation of the particle filter with one hidden state and two observation models.

Similarly for the posterior distribution,

$$\begin{aligned} p(x_{0:t}|y^1_{1:t}, y^2_{1:t}) &\propto p(x_{0:t}|y^1_t, y^2_t, y^1_{1:t-1}, y^2_{1:t-1}) \\ &\propto p(y^1_t, y^2_t|x_{0:t}, y^1_{1:t-1}, y^2_{1:t-1}) p(x_{0:t}|y^1_{1:t-1}, y^2_{1:t-1}) \\ &\propto p(y^1_t, y^2_t|x_t) p(x_t|x_{0:t-1}, y^1_{1:t-1}, y^2_{1:t-1}) p(x_{0:t-1}|y^1_{1:t-1}, y^2_{1:t-1}) \\ &\propto p(y^1_t|x_t) p(y^2_t|x_t) p(x_t|x_{t-1}) p(x_{0:t-1}|y^1_{1:t-1}, y^2_{1:t-1}) \end{aligned} \quad (4.12)$$

Dividing Equation 4.12 by Equation 4.11, one will get

$$\widetilde{w}^{(i)}_t \propto \widetilde{w}^{(i)}_{t-1} \frac{p(y^1_t|x^{(i)}_t) p(y^2_t|x^{(i)}_l) p(x^{(i)}_t|x^{(i)}_{t-1})}{q(x^{(i)}_t|x^{(i)}_{0:t-1}, y^1_{1:t-1}, y^2_{1:t-1})} \quad (4.13)$$

As in the case with one observation, the update of the belief about the posterior filter distribution is performed only through the re-computation of the importance weights as shown in Equation 4.13.

4.4 Current Hockey Tracking System

The new tracking system is built based on the work of Okuma et al. [50]. The major difference is that the new system is a two-view tracking system instead of single-view.

Correspondingly, we make many other changes to adapt to this change. We borrow Okuma's idea of incorporating the detection results into the construction of proposal distribution, though the input is the detection results in Section 3.3.3 rather than AdaBoost detection. This section will cover the details of the new system.

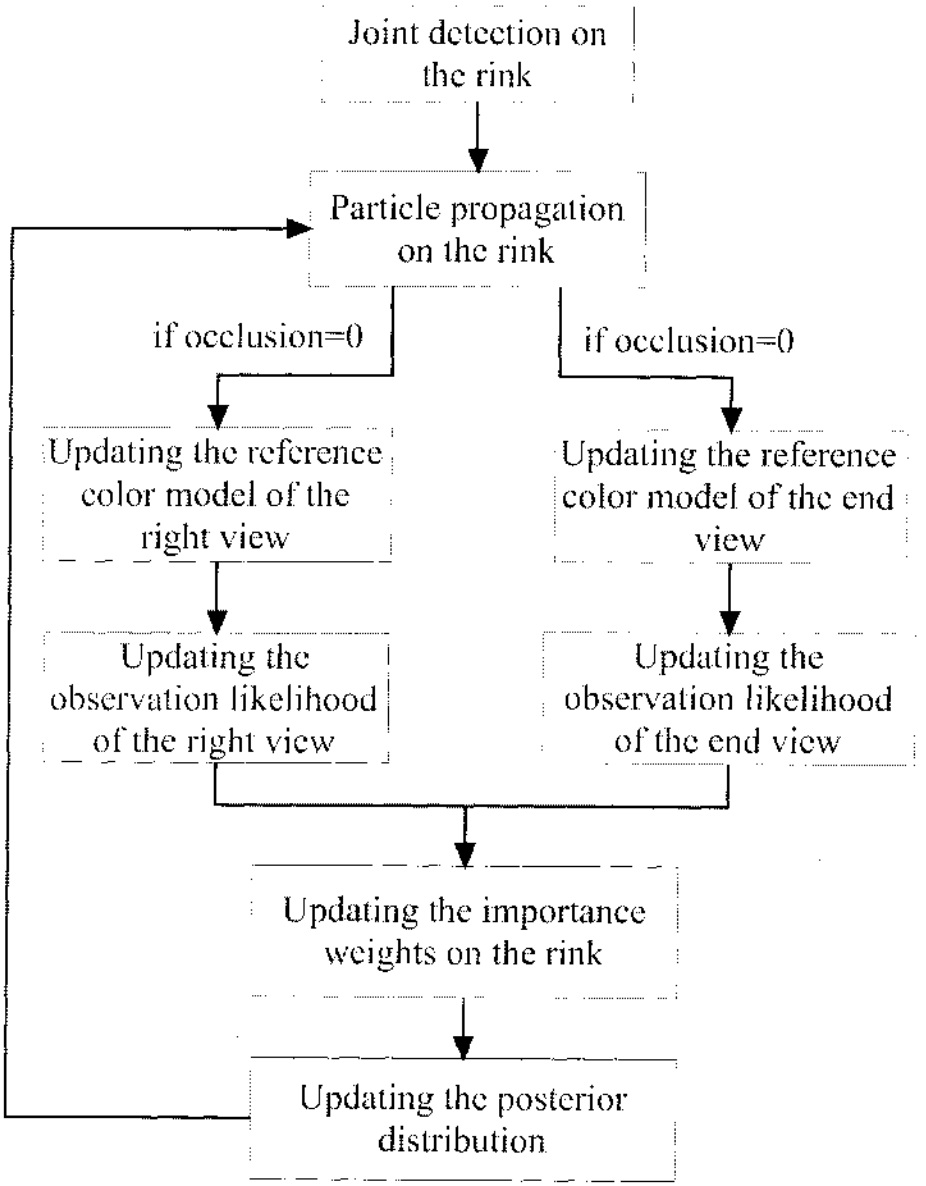


Figure 4.4: Framework of the multi-view tracking.

4.4.1 Statistical Models

Our tracking system is a non-Gaussian and non-linear tracking system. The 2D hidden state includes the x and y coordinates of the the hockey players' foot positions on the ice rink coordinate system. The transition model is built using either Gaussian noise or a second-order autoregressive dynamic model assuming that players move at a constant velocity over all the frames.

The observation model is built based on Hue-Saturation-Value (HSV) color histogram. Taking advantage of the different color components of the different team uniforms, the tracking is basically color-based[53]. From the final location in the previous frame, we search the current frame for such a region, whose color distribution best matches the reference color model with a minimum distance measured based on the HSV color histogram. We choose the Bhattacharyya coefficient to measure data likelihood or similarity

$$\xi[K^*, K(x_t)] = [1 - \sum_{n=1}^{N} \sqrt{k^*(n; x_0)k(n; x_t)}]^{\frac{1}{2}},$$

where $K^* = \{k^*(n; x_0)\}_{n=1,...,N}$ is the reference color model and $K(x_t)$ is the candidate color model. The likelihood distribution is then given by

$$p(y_t|x_t) \propto exp\{-\lambda\xi^2[K^*, K(x_t)]\}.$$

For more details, please refer to the work by Okuma et al. [50].

4.4.2 Difference from the Previous Versions

The inherent difference between the new system and previous versions is that the new system has two views and two coordinate systems, while all the previous versions have only one. We will present the differences as well as the framework of the new system. The framework of the multi-view tracking is shown in Figure 4.4.

Hidden States

Previous versions have a 3D hidden state, including the x and y coordinates of the top-left corner of the player's bounding box and the scale in the image coordinate system. The filtering operates on all the three dimensions for all the frames. The current system has a hidden state with two dimensions, the players' foot positions in the rink coordinate system. As the players are represented as points in the rink coordinates which do not need such description as scale, the scale is removed from the hidden state.

However, we need the scale to calculate the color region for updating the color histogram. Since the cameras in our experiment are static without panning, tilting or zooming, there is no sudden changes in scales. The scale can be obtained using the linear interpolation system in the image coordinates. Therefore, the concept of scale has been shifted from the rink coordinates to the image coordinates. There are two scales, for the right view and the end view, respectively.

Okuma et al. [50] imported the scales from the AdaBoost detection results. Since the cameras in our system are static, we use the linear interpolation system to obtain the scales based on the players' positions. Please refer to Section 3.2.1 for the details.

Two Views and Two Coordinate Systems

In previous systems, the components, such as the propagation of the particles, the updating of transition priors, the calculation of observation likelihood and the updating of the importance weights, were all performed on the image coordinate system. In the current system, these components have been deployed to either the image or rink coordinate system.

The propagation of the particles takes place in the rink coordinate and operates on only the positions. So does the updating of transition priors, an essential step to incorporate the detection results with the transition dynamics. As mentioned, the calculation of the HSV color histogram is performed respectively in the right and end image coordinates. In fact, the system maintains two reference color models to update observation likelihood. Finally, the updating of the importance weights will be operated on the rink coordinates.

Updating the Reference Color Model Based on the Occlusion Analysis

Most color-based tracking systems maintain a reference color model for each player by initializing it when the target first appears and updating it in each of the following frames. We assume the target is highly likely to locate at the region, where the color distribution best matches the reference color model.

Single-view tracking systems may sometimes fail when dealing with full occlusion, such as cross-over. The main reason is that the particles are simulated at the region belonging to both the occluded player and the one in the front. The system may update the reference color model and the observation likelihood of the occluded player based on the color histogram

information of the player in the front. As a result, it is highly like that bounding box of the occluded player will follow the player in the front in the rest of the sequences, even if the two targets separate in future. Therefore, it is crucial to provide some mechanics to determine when to update the correct information regarding the reference color model and observation likelihood.

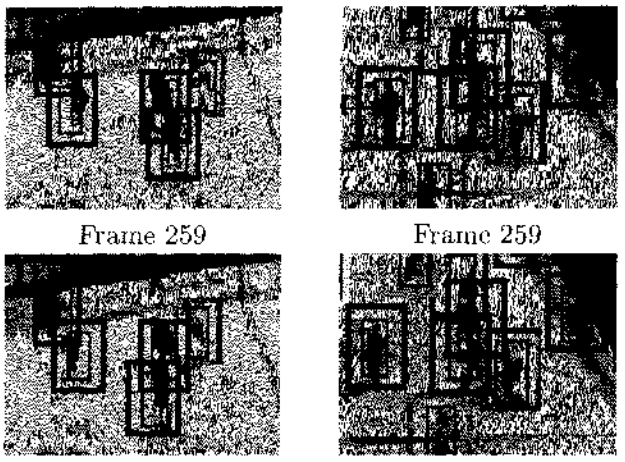
Our system provides occlusion analysis, as explained in detail in the previous chapter. Once the player is suspected to be occluded in one view, the reference color model or the observation likelihood will not be updated in this view. This is to make sure we will not feed the system with wrong information. However, this occluded view can always obtain the position information from the other view by performing the homography transformation and the scale information from linear interpolation in this image coordinates. Therefore, the player is still being tracked in the occluded view. This is the key point of the information integration and information sharing.

4.4.3 Experiments and Results

The most noticeable improvement of our two-view tracking system is that, even when the player is partially or fully occluded for quite long sequences, the two-view tracking system is still able to locate and track the target as long as it is visible from the other view. It is possible that neither of the two views contains the entire information to track all the players, but the information from the the two views together is integral for the tracking. As shown in Figure 4.5, there is occlusion among the 5 players in both views. However, every player can be accurately located using the information from both views. The two views are supportive to each other to keep the track of the players through the frames. With more cameras available, we will have wider coverage ranges to track and identify the targets.

Another example can be found in Figure 4.6. The first column shows the original images from the end view in these 5 frames and the two players near the corner of the snapshots are our focus of interest. From Frame 163, the two players are getting closer and closer and at Frame 172, half of one player is hidden by the other one in the front. From Frame 193, the two players start to separate from each other. At Frame 209, the two players are clearly presented.

The second column is the results when we use single-view tracking (only from the end view). The system starts to get confused about the players' identification from Frame 172, as seen from the color of the bounding boxes. Then the two bounding boxes are reduced to one at Frame 193. Even if the



(a) Right View (b) End View

Figure 4.5: **Though the information from either view is not enough for tracking, the two views are supportive to each other to keep track of the players through the frames.**

two players are separated at Frame 209, there is still only one bounding box.

The third column and the fourth column are snapshots of our two-view tracking system, from the end view and the right view, respectively. In spite of the complicated occlusion and interaction of the two players from the end view, the two players are clearly presented from the right view. So by locating the players in the right view and performing joint detection using homography transformation, we can still track the players in the end view. As shown, not only does the two-view tracking system correctly locate the two players, it also keeps the identities of the two players. However, the bounding boxes sometimes drift due to the noise and lack of robustness of the detection methods.

However, the system sometimes fails to keep the identities of the players when the players are occluded in both views. As shown in Figure 4.7, the system still keep the identities of the two players at Frame 184. But from Frame 186 to Frame 190, these two players are very close to each other in both views. The ambiguity in both views fails the system. The misidentification is going on even when the players separate in future frames.

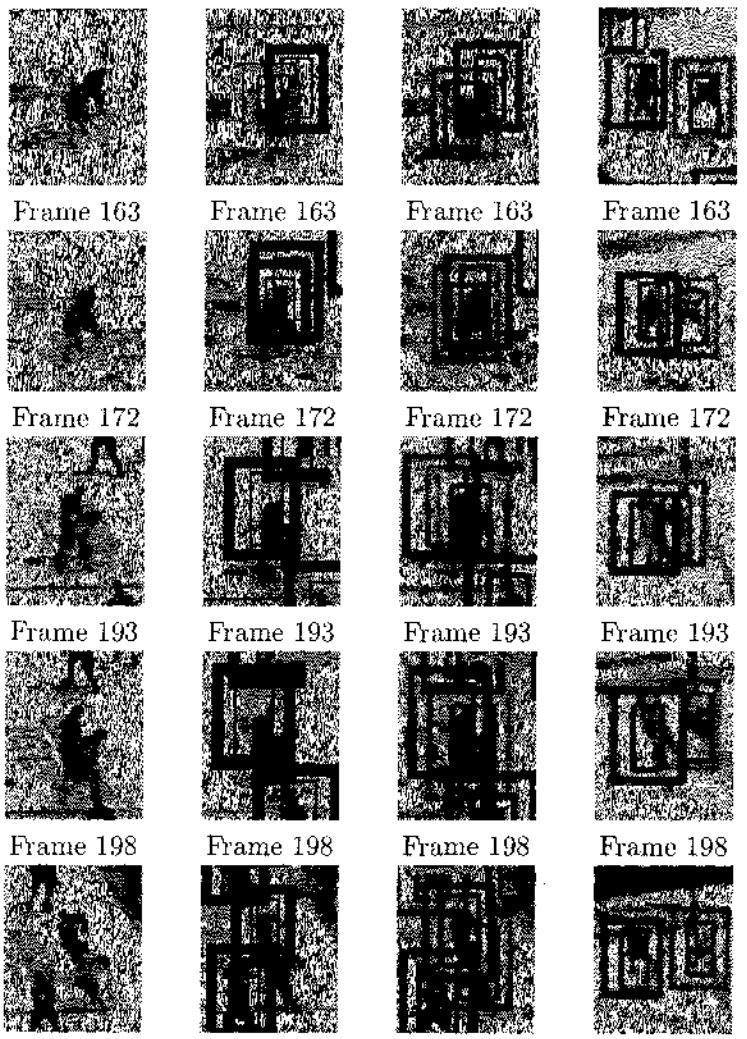


Figure 4.6: **Comparison between previous single-view tracking and two-view tracking:** Even when the player is partially or fully occluded for quite long sequences in one view, the two-view tracking system is still able to locate and track the target as long as it is visible from the other view.

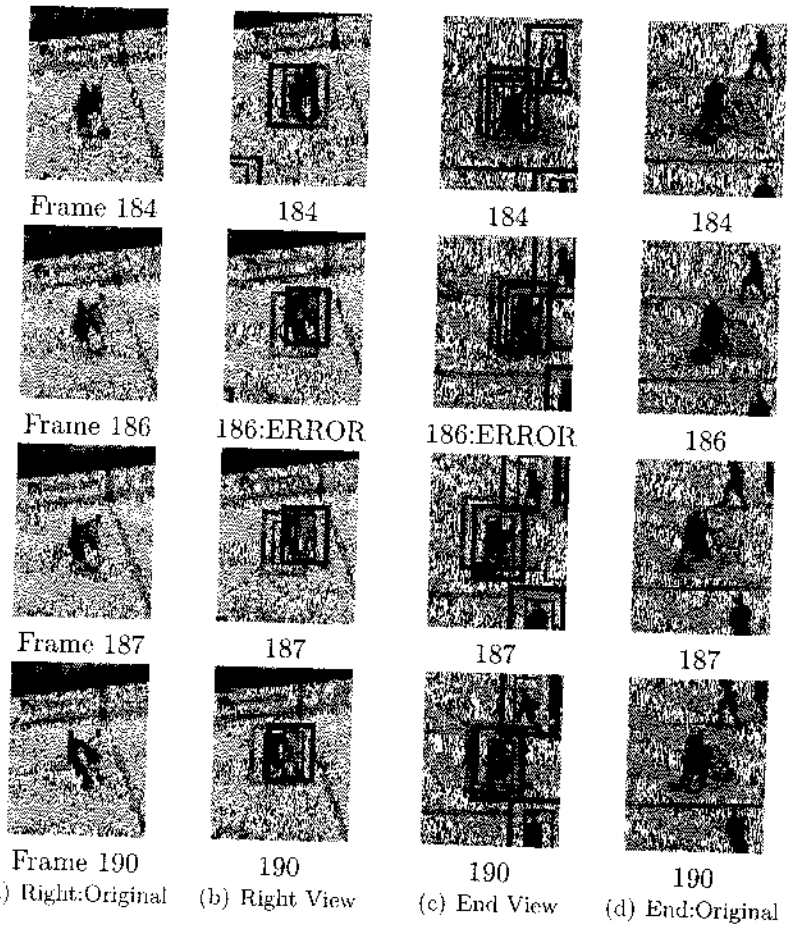


Figure 4.7: **The two-view tracking may fail when there is overlap and occlusion in both views.**

Chapter 5

Smoothing

Now we have the players' trajectories and homography information from the rink coordinates to the image coordinates. Before creating the personalized diaries for the players, we have to first remove the noises and artifacts of the tracking data by smoothing the trajectories. Like the filtering process, smoothing takes place on the rink coordinate and only operates on the 2D, the x-axis and y-axis of the players' positions. A Maximum *a posteriori* (MAP) smoother is adopted for trajectory smoothing. In this chapter, MAP smoother and the Viterbi Algorithm are covered. We also present the experiment results and compare MAP smoother with Kalman smoother.

5.1 Particle Smoothing

The statistical models for smoothing are almost the same as those for filtering except that our goal is to estimate the *smoothing distribution* $p(x_t|y_{1:T})$ rather than the *filter distribution* $p(x_t|y_{1:t})$ or the *posterior distribution* $p(x_{0:t}|y_{1:t})$. Most smoothing algorithms perform a forward filtering pass as the first step and store all the particles with their associated importance weights $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ over the time space. In this section, we will cover the two most well-known smoothers, the Maximum *a posteriori* (MAP) smoother and the Two-Filter smoother.

5.1.1 Maximum *a posteriori* (MAP) Smoother

The MAP estimation of a state sequence [23] is defined as

$$x_{1:T}^{MAP} \triangleq \arg\max_{x_{1:T}} p(x_{1:T}|y_{1:T}) \tag{5.1}$$

First, it performs a standard forward filtering. The forward particle pass generates and stores particles $\{x_t^{(i)}\}, i = 1, ..., N, t = 1, ..., T$. This can be viewed as a finite discrete approximation of the continuous state space [37]. Although this grid is not optimal for MAP estimation, these particles are more likely to concentrate at the most prominent states. The goal is to

select a particle at each time t to find the most probable sequence through all the possible particle combinations. That is,

$$\tilde{x}_{1:T}^{MAP} \triangleq \underset{x_{1:T} \in \bigotimes_{k=1}^{T} \{x_k^{(i)}\}_{i=1}^{N}}{\arg\max} \; p(x_{1:T}|y_{1:T}) \tag{5.2}$$

A brute-force evaluation of this MAP state sequence estimation requires consideration of a number of paths that grows exponentially with the time. However, using the following probability chain rule for the joint distribution of states and observations shown as Equation 5.3,

$$p(x_{1:T}|y_{1:T}) \propto \prod_{t=1}^{T} p(x_t|x_{t-1}) p(y_t|x_t) \tag{5.3}$$

and replacing multiplication with addition in log space, one can get,

$$x_{1:T}^{MAP} \triangleq \underset{x_{1:T}}{\arg\max} \sum_{t=1}^{T} [\log p(y_t|x_t) + \log p(x_t|x_{t-1})] \tag{5.4}$$

MAP sequence estimation can be solved efficiently with the Viterbi algorithm using dynamic programming. Consider time step t and a particular particle i at that time step. Although there are many possible paths converging at the corresponding node in Figure 5.1, the algorithm only needs to retain the path which so far has the highest probability. Since there are altogether N particles at time step t, we need to keep track of N such paths. At time step $t+1$ there will be possible N^2 paths to consider, comprising N possible paths leading out of each of the N current particles, but again we only need to remember N of these corresponding to the best path for each particle at time $t+1$. When reaching the final time step T, the algorithm will discover which state corresponds to the overall most probable path. Since there is a unique path coming in to that state we can trace back to pick up that path. Therefore, the computational expense of this Viterbi algorithm is $O(N^2T)$.

$\delta_t(i)$ is used to score the highest probability along a single path, at time t, which accounts for the first t frames and ends at the i-th particle. By induction we obtain,

$$\delta_{t+1}(j) = \log \; p(y_{t+1}|x_{t+1}^{(j)}) + \max_i \; [\delta_t(i) + \log \; p(x_{t+1}^{(j)}|x_t^{(i)})] \tag{5.5}$$

To retrieve the path of the particles, we need to keep track of the argument which maximizes Equation 5.5. We do this by updating the array $\psi_t(j)$. Please refer to Table 5.1 for the details of the algorithm.

Viterbi Algorithm

1. *Initialization.* For $1 \leq i \leq N$

$$\delta_1(i) = \log\ p(x_1^{(i)}) + \log\ p(y_1|x_1^{(i)})$$

2. *Recursion.* For $2 \leq k \leq t$ and $1 \leq j \leq N$

$$\begin{aligned} \delta_t(j) &= \log\ p(y_t|x_t^{(j)}) + \max_i\ [\delta_{t-1}(i) + \log\ p(x_t^{(j)}|x_{t-1}^{(i)})] \\ \psi_t(j) &= \arg\max_i\ [\delta_{t-1}(i) + \log\ p(x_t^{(j)}|x_{t-1}^{(i)})] \end{aligned}$$

3. *Termination.*

$$\begin{aligned} i_T &= \arg\max_i \delta_T(i) \\ \tilde{x}_T^{\text{MAP}} &= x_T^{(i_T)} \end{aligned}$$

4. *Backtracking.* For $t = T-1, T-2, \ldots, 1$

$$\begin{aligned} i_t &= \psi_{t+1}(i_{t+1}) \\ \tilde{x}_t^{MAP} &= x_t^{i_t} \end{aligned}$$

5. *Aggregation.*

$$\tilde{x}_{1:T}^{MAP} \triangleq \{\tilde{x}_1^{MAP}, \tilde{x}_2^{MAP}, \ldots, \tilde{x}_T^{MAP}\}$$

Table 5.1: Viterbi Algorithm to solve the MAP smoother

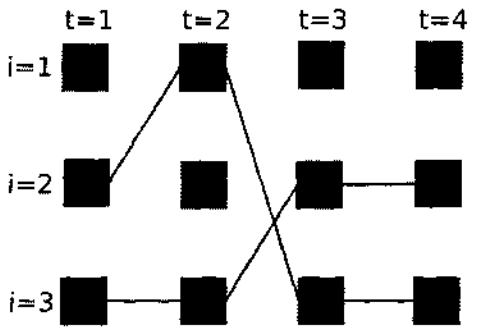


Figure 5.1: Illustration of the MAP sequence estimation with Viterbi algorithm.

5.1.2 Two-Filter Smoother

A smoothed marginal distribution can also be obtained by performing backward filtering as well as forward filtering and then combining the results.

$$\begin{aligned} p(x_t|y_{1:T}) &= p(x_t|y_{1:t-1}, y_{t:T}) \\ &= \frac{p(x_t|y_{1:t-1})p(y_{t:T}|y_{1:t-1}, x_t)}{p(y_{t:T}|y_{1:t-1})} \\ &\propto p(x_t|y_{1:t-1})p(y_{t:T}|x_t) \\ &\propto p(x_t|y_{1:t})p(y_{t+1:T}|x_t) \end{aligned}$$

The forward filter $p(x_t|y_{1:t})$ is the normal particle filter. $p(y_{t+1:T}|x_t)$ is the backward filter and can be computed using an inverse function $p(y_{t+1:T}|x_t) = \frac{p(x_t|y_{t+1:T})}{\gamma_t(x_t)}$, where $\gamma_t(x_t)$ is an artificial distribution[37]. The backward filter works almost the same as the forward one, except that the recursive process is towards the opposite direction. For more details on the evaluation of $\gamma_t(x_t)$, please refer to [37] and [36].

The two-filter smoother is a successful algorithm because of the integration of the backward filtering process, which could find locations where particles are likely to concentrate but the forward pass may miss.

5.2 Difference between Smoothing and Filtering

Filtering performs an on-line updating of the posterior distribution for the current frame and takes it as the prior distribution for the next frame, though at each time step it can only wrap the information from the past.

Smoothing algorithms can improve the performance of filtering by exploring the future information through the entire video sequence as well as previous history. The more information the system can integrate, the more accuracy it can achieve.

Filtering always outputs the mean values of all the particles as the prediction of the hidden states. It is quite possible that one really erroneous and noisy particle deteriorates the overall performance even though all the other samples are correctly propagated according to the latent variables. In contract, the MAP smoother used in our tracking application picks up the most prominent particle at each time step using Viterbi algorithm and generates the path as the most possible sequence of states.

However, the access to the future information reduces smoothing to an off-line algorithm, which is a lot less attractive than the on-line methods. This drawback can be mitigated by using a moving window which can go through the entire video sequence. The size of the window is a trade-off between how much information the system can integrate and how fast it can be. This method will only introduce a few frames' delay depending on the window size rather than downgrading the algorithm to off-line. We keep the off-line method because it can fulfill the purpose of setting up this tracking system, which is to create personalized video programs for players in sports games at the end of the events.

5.3 Experiments and Analysis

5.3.1 Comparison of Filtering and Smoothing in a Simple Example

Experiments

Consider the following non-Gaussian, non-linear model,

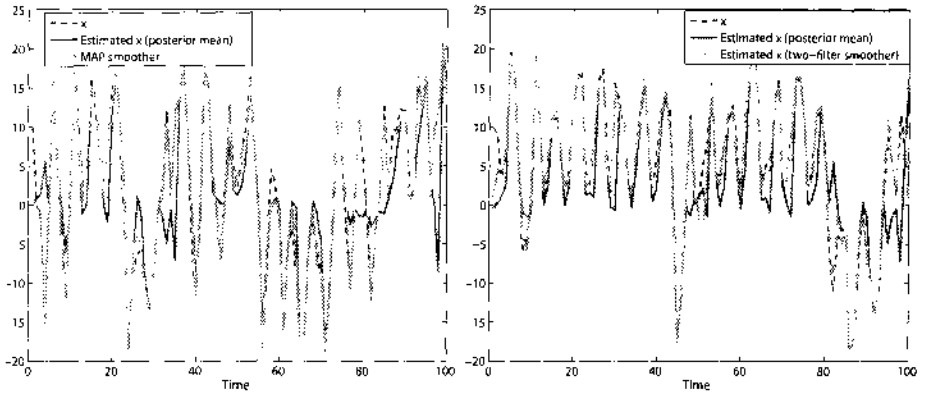
$$x_t = \frac{1}{2}x_{t-1} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8cos(1.2t) + v_t,$$

$$y_t = \frac{x_t^2}{20} + w_t$$

where $x_1 \sim N(0, \sigma_1^2)$, $v_k \sim N(0, \sigma_v^2)$ and $w_k \sim N(0, \sigma_w^2)$ with $\sigma_1^2 = 1$, $\sigma_v^2 = 10$ and $\sigma_w^2 = 10$.

Results and Analysis

After running a standard particle filtering, we use the MAP sequence estimation and the two-filter smoother as described above. The accuracy that smoothing algorithms can achieve for the prediction of hidden states compared with filtering methods is demonstrated in Figure 5.2. In addition, there can sometimes be a big variance among all the particles simulated for hidden states. Although using the mean value of all the particles can moderate the prediction error, a few poorly sampled and evolving particles can greatly affect the overall performance.



(a) (b)

Figure 5.2: **The Comparison between particle smoothing and particle filtering:** (a) Particle filter and MAP estimates of the hidden states in the 1-D time series experiment. Mean error for the particle filter is 3.4551, while the MAP smoother has a mean error of 1.2910. (b) Particle filter and two-filter smoother estimates of the hidden states in the 1-D time series experiment. Mean error for the particle filter is 3.2948, while the two-filter smoother has a mean error of 1.3306.

5.3.2 Comparison of MAP Smoother and Kalman Smoother in the Context of Hockey Tracking

Experiments

We smooth the players' trajectories after storing the particles (2D) with their associated weights. Only the players' positions are smoothed. All the noise is Gaussian noise. Two covariance matrices are for initialization noise

and transition noise, respectively and both equal to,

$$\begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}$$

The particle weights function as the observation likelihood in the algorithm. At the time step t, we compare the i-th particle at time t with all the particles at time $t-1$ and save the particle which generates the path with the highest probability.

In addition, the particles and their associated weights are also passed into Kalman smoother as a comparison with our MAP smoother. We directly used the Kalman filter toolbox for Matlab developed by Murphy [47].

Results and Analysis

We conducted two sets of experiments in the two-view tracking step, with the particle number as 30 and 300, respectively. Two sets of particles and their associated weights were stored. We picked a hockey player and plotted his trajectory of 200 frames, shown in Figure 5.4. Figure 5.4 (a) and (b) show the trajectories using the mean value of all 30 and 300 particles, respectively. Figure 5.4 (c) and (d) show the trajectories after applying MAP smoothing to 30 particles with their associated weights, and 300 particles with their associated weights, respectively. Figure 5.4 (e) shows the trajectory after applying Kalman smoothing to 30 particles with their associated weights. Figure 5.4 (f) shows the trajectory after applying Kalman smoothing to the optimal path selected by MAP smoothing.

It is confirmed again that the MAP smoothing (Figure 5.4 (c) and Figure 5.4 (d)) can significantly smooth the trajectories generated by particle filtering (Figure 5.4 (a) and Figure 5.4 (b)). In addition, the results also show that the performance of particle smoothing unavoidably relies on the filtering results, while the Kalman smoothing displays less dependence, as shown from Figure 5.4 (c) and Figure 5.4 (e). The reliance of the particle smoothing on the filtering is reasonable. The best particle chosen by the MAP smoothing may still be very noisy if the whole particle set contains serious errors.

If we enlarge the sampling space by increasing the covariance from 40 to 400 and the particle number for each player from 30 to 300, we get a larger particle pool to choose the most significant state from. Comparing Figure 5.4 (d) with Figure 5.4 (c), we can see the resulting trajectory after increasing the pool becomes smoother. The best smoothing result is obtained when we

pass the optimal path from the particle smoothing to Kalman smoothing. The trajectory after performing smoothing twice is shown in Figure 5.4 (f).

Finally we plot the smoothed trajectories of all the people on the ice rink (including hockey players, goalkeepers, linesmen and the referee) in one image, shown as Figure 5.3.

Figure 5.3: The smoothed trajectories of all the people on the ice rink plotted using the method in Figure 5.4 (f).

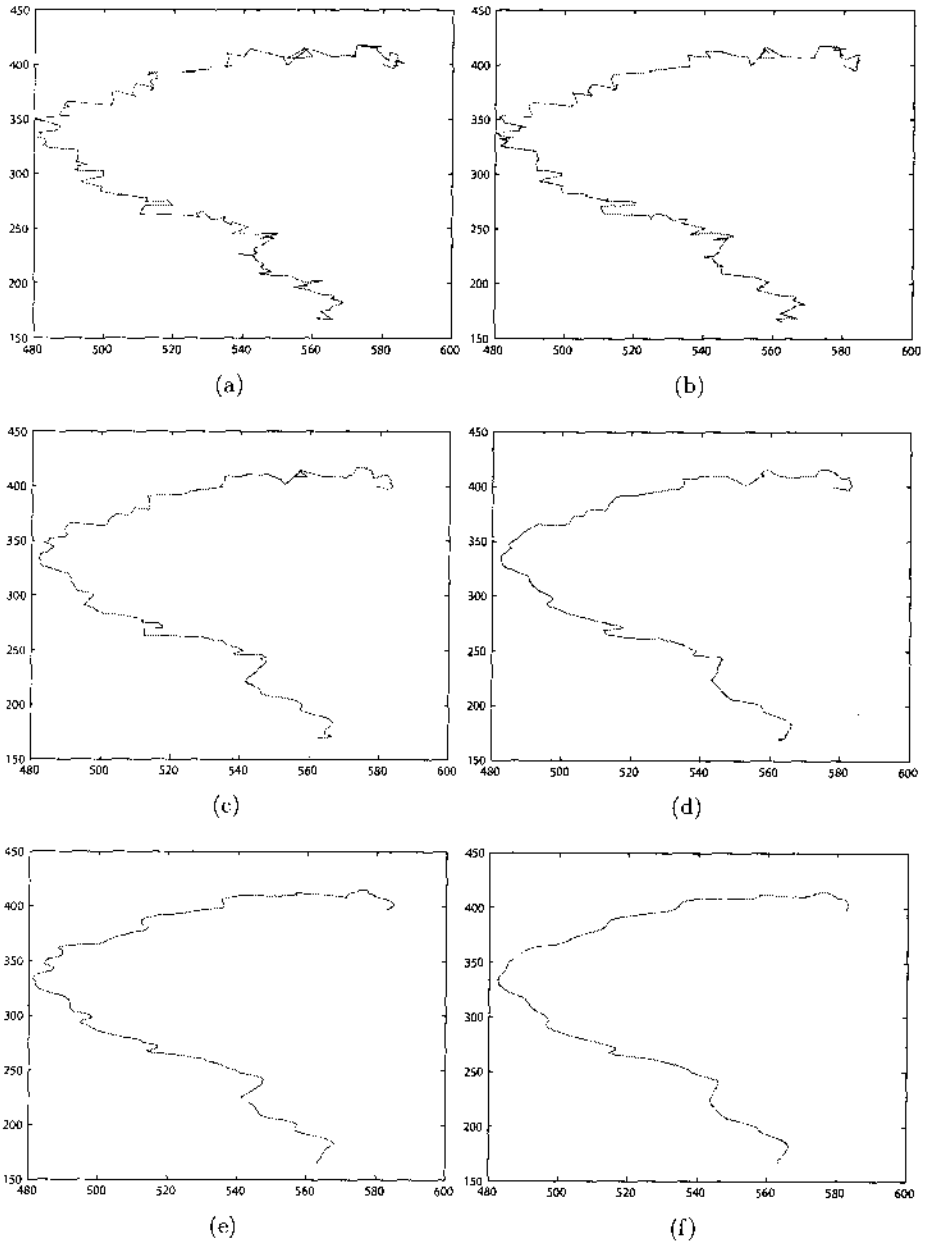


Figure 5.4: **The comparison of particle smoothing, particle filtering and Kalman smoothing in the context of hockey tracking:** The horizontal axis and vertical axis in all the figures represent the x-axis and y-axis of the rink coordinate system, respectively (see text for description.)

Chapter 6

Automatic Camera Selection

Automatic handoff among best-view cameras is an intuitive extension of multi-view tracking and an interesting way of using the multi-view data. The two criteria as we discussed previously are that the best views should be selected according to the view quality measurements, and the switching among cameras should be smooth and frequent view changes should be avoided. This scenario has been built in *Hidden Markov Models*(HMMs) [3] in our system. In this chapter, we will cover the basic knowledge of HMMs and present the inference and learning in HMMs in regard of this scenario.

6.1 Hidden Markov Model

HMMs are statistical models of sequential data and can provide the theoretical basis for a wide range of applications. In this section, we will review the theory of *Markov Models* and the concepts of hidden states. This will give rise to HMMs, where the observation is a probabilistic function of the hidden states. We use Bishop's definition and formulation [9] below. Due to the limitation of our thesis, we can only cover a bit of the application of HMMs, which is related to our work. Bishop's book [9] and Rabiner's tutorial [55] are good references for HMMs.

6.1.1 Markov Models

Markov models are one of the simplest probabilistic models to simulate sequential data. It is assumed that the sequential data have such property that the joint probability distribution of a sequence of observations can be factored as [6]

$$p\left(y_1, y_2, \ldots, y_N\right) = \prod_{n=1}^{N} p\left(y_n | y_1, \ldots, y_{n-1}\right), \tag{6.1}$$

Assuming that each of the conditional distributions on the right-hand side of Equation 6.1 is independent of all previous observations except the most

recent, we obtain a *first-order Markov chain* [9],

$$p\ (y_1, y_2, \ldots, y_N) = p\ (y_1) \prod_{n=2}^{N} p\ (y_n|y_{n-1}), \tag{6.2}$$

where $p(y_1)$ is the initial probability and $p(y_t|y_{t-1})$ is the transition probability. In a discrete, first-order Markov chain, the transition probability among observations can be denoted as a_{ij}, where,

$$a_{ij} = p\ (y_t = j|y_{t-1} = i), \quad 1 \leq i, j \leq N \tag{6.3}$$

where N is the number of the possible observations. If we allow the predictions to depend also on the previous-but-one observation, we get a *second-order Markov chain* [9],

$$p\ (y_1, y_2, \ldots, y_N) = p\ (y_1)\ p\ (y_2|y_1) \prod_{n=3}^{N} p\ (y_n|y_{n-1}, y_{n-2}), \tag{6.4}$$

So far, all the models have been assuming that we have the access to the observation space and are able to directly evaluate the probabilities of the observations. These models are too restrictive to be applicable in most problems. Constraining ourselves within the observation space can restrict us to using a very low order Markov model due to the computational efficiency [6]. For example, a Markov model of order k with each possible observation as a multinomial variable $y_t \in \{1, \ldots, n\}$ will quickly become intractable for large k's, because the number of required parameters for representing the transition probabilities is $O(n^{k+1})$ [6].

6.1.2 Hidden Markov Models

HMMs can handle both issues just mentioned. In HMMs, we do not assume that the observed data sequence has low order. Instead, another sequence of unobserved but related variables is assumed to exist with low order. These variables are called *hidden states*. In most cases, hidden states evolve according to a first-order Markov model and obey the Markov assumptions in a different space from the observation space. An HMM of order one can emulate an HMM of any higher order by increasing the number of values that the state variable can take [6]. The observation thus becomes a probabilistic function of the corresponding hidden state. The resulting HMM therefore has an unobservable stochastic process defined by the evolution of the hidden states and another set of stochastic processes that generates the sequence of observations.

Components in Hidden Markov Models

According to the Markov assumptions of order one, the joint distribution of the hidden and observed variables can be factored as

$$p\left(y_{1:T}, x_{1:T}\right) = p\left(x_1\right) \prod_{t=1}^{T-1} p(x_{t+1}|x_t) \prod_{t=1}^{T} p(y_t|x_t), \tag{6.5}$$

The joint distribution is therefore completely specified in terms of the *initial state probabilities* $p(x_1)$, *the transition probabilities* $p(x_t|x_{t-1})$ and the *emission probabilities* $p(y_t|x_t)$. For convenience, we use the compact notation λ to indicate the complete parameter set of the model.

While the states variables of HMMs have to be discrete, the observed variables can be discrete or continuous, and a variety of different conditional distributions can be used to model them.

6.1.3 The Most Probable State Sequence

An interesting problem in HMMs is, given the observation sequence $\mathbf{Y} = y_1, y_2, \ldots, y_T$, and a model λ, to determine the best state sequence $\mathbf{X} = x_1, x_2, \ldots, x_T$, which is optimal in explaining the observation.

One possible solution is to choose the states x_t which are individually most likely, in other words, to maximize $p\left(x_t|y_t\right)$, where $t = 1, 2, \ldots, T$, independently. This optimality criterion maximized the expected number of correct individual states. However, there could be problems with the resulting state sequence due to the disregard of the transition probabilities. For example, when the HMM has state transitions with zero probability ($a_{ij} = 0$ for some i and j), the "optimal" state sequence may not even be a valid state sequence [55].

A formal solution to this problem is *Viterbi algorithm* to take both stochastic processes into consideration. To find such an optimal state sequence, which is to maximize $p\left(\mathbf{X}|\mathbf{Y}, \lambda\right)$, is equivalent to maximizing $p\left(\mathbf{X}, \mathbf{Y}|\lambda\right)$. We need to define the quantity,

$$\delta_t(i) = \max_{x_1, x_2, \ldots, x_{t-1}} p\left(x_1, x_2, \ldots, x_{t-1}, x_t = i, y_1, y_2, \ldots, y_{t-1}, y_t|\lambda\right) \tag{6.6}$$

$\delta_t(i)$ is used to score the highest probability along a single path, at time t, which accounts for the first t observations and ends at the i-th state. By induction we obtain,

$$\delta_{t+1}(j) = p\left(y_{t+1}|x_{t+1} = j\right) + \max_i \left[\delta_t(i)\, p\left(x_{t+1} = j|x_t = i\right)\right] \tag{6.7}$$

Viterbi Algorithm

1. *Initialization.* For $1 \leq i \leq N$

$$\begin{aligned} \delta_1(i) &= p\ (x_1 = 1)\ p(y_1|x_1 = 1) \\ \psi_1(i) &= 0 \end{aligned}$$

2. *Recursion.* For $2 \leq t \leq T$ and $1 \leq j \leq N$

$$\begin{aligned} \delta_t(j) &= p\ (y_t|x_t = j) + \max_i\ [\delta_{t-1}(i)\ p\ (x_t = j|x_{t-1} = i)] \\ \psi_t(j) &= \arg\max_i\ [\delta_{t-1}(i)\ p\ (x_t = j|x_{t-1} = i)] \end{aligned}$$

3. *Termination.*

$$\begin{aligned} s &= \max_i\ \delta_T(i) \\ q_T^* &= \arg\max_i \delta_T(i) \end{aligned}$$

4. *Backtracking.* For $t = T-1, T-2, \ldots, 1$

$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$

5. *Aggregation.*

$$q_{1:T}^* \triangleq \{q_1^*, q_2^*, \ldots, q_T^*\}$$

Table 6.1: Viterbi algorithm to determine the most probable state sequence.

To retrieve the state path, we need to keep track of the argument which maximizes Equation 6.7. We do this by updating the array $\psi_t(j)$. The complete procedure is stated in Table 6.1:

6.2 Automatic Best-view Camera Selection

This piece of work is motivated by the project MyView[1] led by Professor Sidney Fels. In this project, it is assumed that the users of the system have decided to focus on one particular player during the game. For exam-

[1] http://hct.ece.ubc.ca/research/myview/index.html

ple, the player can be someone they know in person or someone from the same country. Therefore, after performing the two-view tracking and the trajectory smoothing, the next step is to present the data to users in the form of personalized video programs and diaries of the players from different perspectives. We interpret this task as camera selection, which is to automatically select a view at each time so that the resulting video can best demonstrate the players in the hockey game.

6.2.1 Criteria for the Best Views

The camera selection process is player-centered rather than puck-centered or play-centered. The users of the system are more interested in the performance or positions of the players from different perspectives than the game itself. Therefore, the viewpoint which can better represent the players will be selected, while the puck could be lost track of.

To get a high-quality personalized video for a single player, the viewpoints where this player is involved in less overlapping and interaction (either occluding other players or being occluded by other objects) will be selected. The changes of this player's poses and positions over frames should be as continuous and smooth as possible.

To get a smooth video sequence, the system will switch to the other camera only when it has to and the change of viewpoints cannot be too frequent. For example, if for some frame, both cameras can represent the players in an equally good or bad way, the system will keep using the camera which was selected in the previous frame rather than switching to the other camera.

6.3 Experiments and Results

6.3.1 Calculating the Overlap Fraction For Each Player

Given a player's trajectory on the rink and his scales in the two views, we can derive the foot positions and size of this player in the view coordinates and then draw bounding boxes. The bounding box is considered as the region belonging to this player.

A 2D binary map of the same size with the bounding box is maintained for each player and initialized to all zeros. The system stepped through all the bounding boxes belonging to the other 14 players. If a pixel in this binary map is also a pixel belonging to other players' regions, the pixel in

the binary map will be set to 1.

$$\texttt{Overlap Fraction} = \frac{\sharp \text{ pixels in the binary map that are set to 1}}{\sharp \text{ pixels in the binary map}}$$

For each player has `Overlap Fraction(right)` and `Overlap Fraction(end)`.

6.3.2 Fitting the Scenario into an HMM

Camera selection can be viewed as determining the camera sequence which could best explain the observation and avoid overlap. Since there are two cameras in our application, the hidden state is a discrete two-state model, where each state corresponds to a camera. At the first frame, the two cameras are selected with the same probability. The frequency of switching between cameras is controlled by the transition matrix. The observation model is continuous, and build based on the overlap fraction. After filling in the parameters for the three components in the HMM, we can use the HMM toolbox for Matlab developed by Murphy [46] by simply calling the function `viterbi_path`.

Hidden States and Initial State Probability

The discrete two-state model has the right-view camera(with value 1) and the end-view camera(with value 2) as the two states. At the first frame the two cameras are chosen with the same probability. Therefore, the initial state probability (or *prior*) is

$$\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

Transition Matrix

We favor selecting the camera which was chosen in the previous frame to switching to the other camera. Therefore, the probability of switching to the other camera is set to be lower than the that of keeping the same camera. The lower the probability of switching cameras, the smoother the resulting video will be. The sum of the two probability should be 1. The transition matrix (or *transmat*) is

$$\begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$$

Emission Probability

The emission probability is built based on the player's overlap fraction. However, they have an inverse relation in that, the lower the overlap fraction is, the higher the emission probability is. Therefore, the emission matrix (or *lkhd*) is

$$\begin{pmatrix} \frac{\texttt{Overlap Fraction(end)}}{\texttt{Overlap Fraction(right)+Overlap Fraction(end)}} \\ \frac{\texttt{Overlap Fraction(right)}}{\texttt{Overlap Fraction(right)+Overlap Fraction(end)}} \end{pmatrix}$$

where the first row is the likelihood of switching to the right-view camera and the second row end-view camera. There are two special cases. First, when there is no overlap in either view, the likelihood of switching to the right view and that of switching to end are the same and both set to 0.5. Second, if the player is not even covered by the camera, the likelihood of switching to this view is 0 and that of switching to the other is 1.

The Optimal Path

After filling in all the necessary parameters for the HMM, we can obtain the best combination of the two cameras by calling the function

```
path = viterbi_path(prior, transmat, lkhd);
```

from Kevin Murphy's HMM toolbox.

6.3.3 Results and Analysis

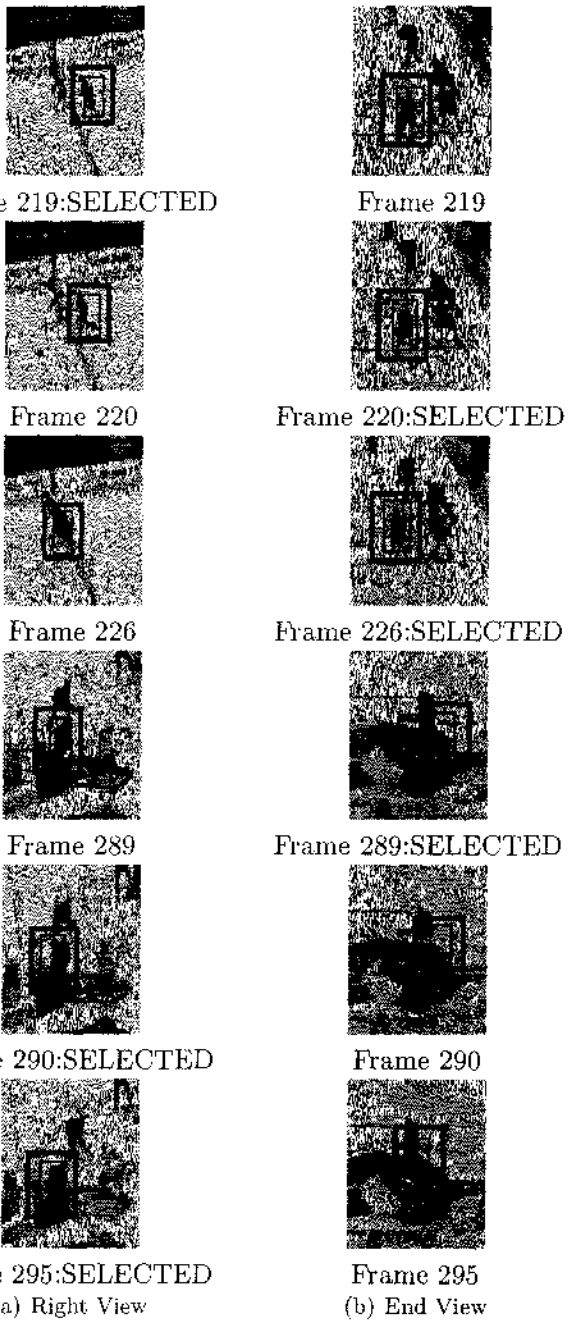
As expected, if from one view, the player is partially or fully overlapping with other players, or surrounded in a crowded scene, or not fully covered by the camera, the system will be automatically switched to another view. The overall performance is smooth and the frequency of the switching is controlled by the transition matrix. Sometimes, to ensure the smooth transition, the system may switch to the other view 5 to 10 frames before the overlap actually happens in the current view.

Snapshots of the camera selection system are shown in Figure 6.1. The player in the blue bounding box is our focus of interest. The left column shows the right view of the 6 frames and the right column shows the end view. There are two view switches involved in this example. The first happened at Frame 220, from the right view to the end view. However, as you can notice, the player in the right view is as separated from other players as in

the end view at this frame. But if we follow the video for another 6 frames, we can see at Frame 226 that, obvious overlaps happen in the right view. The system sometimes moves the view switching to a few frames earlier to ensure transition smoothness. The end view is selected to be the best view until Frame 290. Similarly, no obvious overlap happens in the end view until Frame 295.

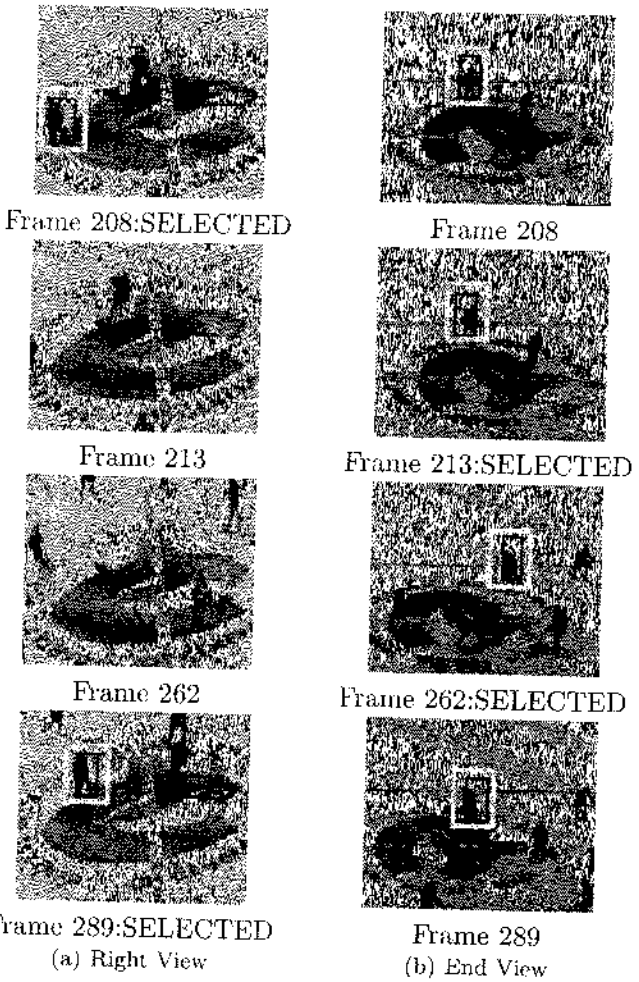
The snapshot in Figure 6.2 show examples of camera switching due to the player's not fully covered in the current view. The player in the yellow bounding box is our focus of interest. For Frame 208, the system switches to the right view, although the two views are selected at equal possibility. In Frame 213, the end view was selected because the player is not fully visible from the right camera. Same as in Frame 262 and afterwards, until when overlapping happens in the end view in Frame 289, the right view is selected again.

We also tune the parameters in the transition matrix and inspect how that will affect the switching frequency. We find that as long as the probability of staying at the same camera is set higher than that of switching to the other camera, the transition between the two cameras is smooth and switching frequency is almost the same. It does not make a big difference if we set the parameters to 0.9 with 0.1 or 0.6 with 0.4. However, if we set both two parameters to 0.5, the switching will be really frequent as shown in Figure 6.3. As you can see, the system constantly selects the end view if we set to 0.9 and 0.1, as shown in the third column. However, the switching between the views is very frequent if we set to 0.5, as shown in the last column.



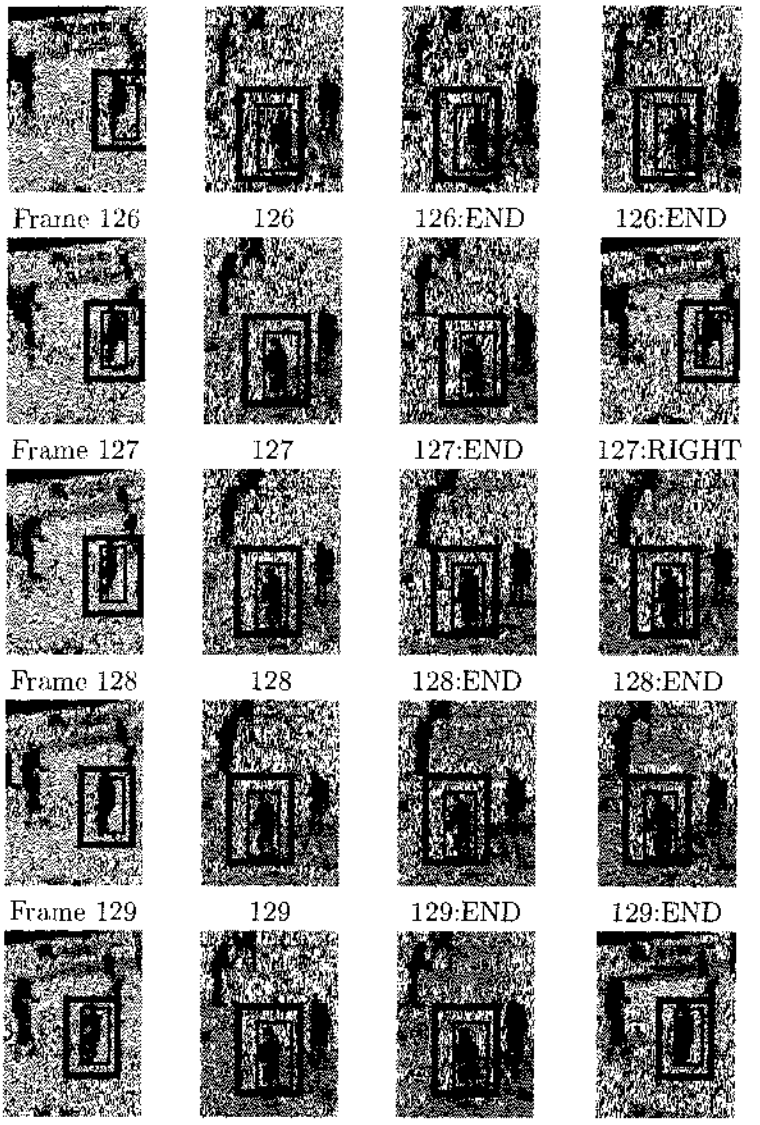
(a) Right View (b) End View

Figure 6.1: **An example of the scenarios when the system switches to other view due to overlapping happening in the current view.**



(a) Right View (b) End View

Figure 6.2: **An example of the scenarios when the system switches to the other view because the player is not fully covered in the current view.**



(a) Right View (b) End View (c) Res(0.9, 0.1) (d) Res(0.5, 0.5)

Figure 6.3: **An experiment to see how the parameters in the transition matrix will affect the switching frequency.**

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis presents a multi-view tracking system which can track multiple hockey players and create personalized video programs using trajectory smoothing and automatic view switching. The multi-view tracking system is based on Okuma's Boosted Particle Filter [50], which incorporates the detection results into the construction of the proposal distribution. The online AdaBoost detection is replaced by the off-line detection, including the single-view blob detection in the image coordinate systems and the multi-view detection in the rink coordinate system. Homography transformation maps the data between the image space and the rink space. We reduce the hidden state to two dimensions, representing the players' positions in the rink coordinates. We also extend the particle filter with multiple observations, each corresponding to a view. We maintain an observation likelihood and a reference color model for each player in each view and update them only when the player is not occluded. The multi-view tracking has the advantage over single view tracking because of its wide coverage range and multiple perspectives. Even when the player is occluded in one view, the multi-view tracking system is still able to track the target as long as it is visible from another view. The multi-view data, especially the trajectories, are first smoothed using Maximum *a posteriori* (MAP) smoother and the Viterbi algorithm. We interpret the camera selection problem as finding the most probable state sequence given the observation sequence and resolve the problem using HMM combined with the Viterbi algorithm. The system can automatically switch to the other view when the player has more overlap with other players or cannot be fully covered in the current view, as well as maintaining the smoothness of the transition.

7.2 Future Work

The detection method can be improved. For example, during background subtraction, we simply subtract the background image from the image. In this way, we have only considered the relation in space, so the noise due to the shadows cannot be totally removed. Many background subtraction algorithms [1, 2] try to model the background based on a series of images and take into consideration their relation in both time and space. They can deal with the shadow removal in a better way. It is always a good idea to obtain the ground truth of the noise, so that we can better model the noise and evaluate the shadow removal methods. In addition, the background images in our application are grey images. However, more information to help subtract the foreground from background could be revealed if the images are retained as colored images.

Although this off-line work can fulfil our purpose of creating a personalized video program for a given player at the end of the event, building an on-line application is more attractive, especially for sports tracking. The output video of the tracking system are low resolution compared with the resolution of the input video. The resolution is sacrificed, because we try to fit in our work as much as possible in Okuma's system. The resolution of the output should be increased for better visualization. The bounding boxes drift especially in the presence of occlusion and overlapping. This is mainly because of the shadow and will be improved as we improve background subtraction algorithms. There are some other ways to provide better localization, such as Lu and Little [41]. The appearance of the bounding boxes can be improved so that they are sensitive to the different sizes of the players and vary with the players' poses. In addition, the two cameras are static in our system and are specially set up to simplify the tracking. It will be interesting if we add more cameras into the system, to extend two-view tracking to multi-view tracking. Although the situation with multi-cameras is more complicated, the same approach in our application will be applicable. We can also try to remove the camera which can cover the whole rink, such as the end-view camera, to see how the system can handle the situation when players enter or leave the scene. Non-overlapping cameras can be installed to check data fusion and information sharing. To install an overhead camera will also be helpful to maintain the identification of the players.

As we can see from the comparison between the MAP smoother and the Kalman smoother, the MAP smoother relies mainly on the performance of the particle filtering. Its time complexity is $O(N^2T)$. A new smoothing

algorithm with less dependence on performance of the tracking stage and less time complexity is more desirable. For camera selection, the emission probability can be calculated based on the occlusion information instead of the overlap fraction. We can also provide the users with more choices rather than define only one criterion. User studies can be set up to decide which view quality measurement is the most acceptable.

Bibliography

[1] AGRAWAL, A., RASKAR, R., AND CHELLAPPA, R. Edge Suppression by Gradient Field Transformation Using Cross-Projection Tensors. *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 2* (2006), 2301–2308.

[2] AGRAWAL, A., RASKAR, R., NAYAR, S., AND LI, Y. Removing photography artifacts using gradient projection and flash-exposure sampling. *Proceedings of ACM SIGGRAPH 2005 24*, 3 (2005), 828–835.

[3] ANDERSON, B., AND MOORE, J. *Optimal Filtering.* Prentice-Hall, Englewood Cliffs, New Jersey, 1979.

[4] ARULAMPALAM, S., MASKELL, S., GORDON, N., AND CLAPP, T. A Tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing 50*, 2 (2001), 174–189.

[5] BAR-SHALOM, Y., AND FORTMANN, T. *Tracking and Data Association.* Academic-Press, Boston, 1988.

[6] BENGIO, Y. Markovian models for sequential data. *Neural Computing Surveys 2* (1999), 129–162.

[7] BERCLAZ, J., FLEURET, F., AND FUA, P. Robust People Tracking with Global Trajectory Optimization. *Conference on Computer Vision and Pattern Recognition 3* (2006).

[8] BERZUINI, C., BEST, N., GILKS, W., AND LARIZZA, C. Dynamical Conditional Independence Models and Markov Chain Monte Carlo Methods. *Journal of the American Statistical Association 92*, 440 (1997).

[9] BISHOP, C. *Pattern recognition and machine learning.* Springer, 2006.

[10] BLACK, J., ELLIS, T., AND ROSIN, P. Multi view image surveillance and tracking. *Motion and Video Computing, 2002. Proceedings. Workshop on* (2002), 169 -174.

[11] BOLIC, M., AND HONG, P. Resampling algorithms and architectures for distributed particle filters. *IEEE Transactions on Signal Processing 53*, 7 (2005), 2442–2450.

[12] CAI, Q., AND AGGARWAL, J. Tracking human motion in structured environments using a distributed-camera system. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 11 (1999), 1241–1247.

[13] CARPENTER, J., CLIFFORD, P., AND FEARNHEAD, P. Improved particle filter for nonlinear problems. *Radar, Sonar and Navigation, IEE Proceedings- 146*, 1 (1999), 2–7.

[14] CHANG, T., AND GONG, S. Tracking Multiple People with a Multi-Camera System. *IEEE Workshop on Multi-Object Tracking* (2001), 19–26.

[15] CHANG, T., GONG, S., AND ONG, E. Tracking multiple people under occlusion using multiple cameras. *Proc. 11th British Machine Vision Conference* (2000).

[16] CHEN, Y., HUANG, T., AND RAI, Y. Parametric Contour Tracking Using Unscented Kalman Filter. In *International Conference on Image Processing* (2002), vol. III, pp. 613 -616.

[17] COMANICIU, D., BERTON, F., AND RAMESH, V. Adaptive Resolution System for Distributed Surveillance. *Real-Time Imaging 8*, 5 (2002), 427–437.

[18] COX, I., AND HINGORANI, S. An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 2 (1996), 138 150.

[19] DOUCET, A., DE FREITAS, J., AND GORDON, N., Eds. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[20] DOUCET, A., DE FREITAS, N., AND GORDON, N. An introduction to sequential Monte Carlo methods. *Sequential Monte Carlo Methods in Practice* (2001), 4-11.

[21] Du, W., and Piater, J. Multi-camera People Tracking by Collaborative Particle Filters and Principal Axis-Based Integration. *LECTURE NOTES IN COMPUTER SCIENCE 4843* (2007), 365.

[22] Fleuret, F., Berclaz, J., Lengagne, R., and Fua, P. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence 30*, 2 (February 2008), 267–282.

[23] Godsill, S., Doucet, A., and West, M. Maximum a Posteriori Sequence Estimation Using Monte Carlo Particle Filters. *Annals of the Institute of Statistical Mathematics 53*, 1 (2001), 82–96.

[24] Gordon, N. *Bayesian Methods for Tracking.* PhD thesis, Imperial College, University of London, 1994.

[25] Hartley, R., and Zisserman, A. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[26] Hu, W., Hu, M., Zhou, X., Tan, T., and Lou, J. Principal axis-based correspondence between multiple cameras for people tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 4 (2006), 663–671.

[27] Hu, W., Tan, T., Wang, L., and Maybank, S. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 34*, 3 (2004), 334 352.

[28] Hue, C., Le Cadre, J., and Pèrez, P. Tracking Multiple Objects with Particle Filtering. In *IEEE Transactions on Aerospace and Electronic Systems* (2003), vol. 38, pp. 313–318.

[29] Isard, M., and Blake, A. CONDENSATION-Conditional Density Propagation for Visual Tracking. *International Journal on Computer Vision 29*, 1 (1998), 5–28.

[30] Iwase, S., and Saito, H. Parallel tracking of all soccer players by integrating detected positions in multiple view images. *Proceedings of the 17th International Conference on Pattern Recognition, 2004 4* (2004).

[31] JIANG, H., FELS, S., AND LITTLE, J. Optimizing Multiple Object Tracking and Best View Video Synthesis. *IEEE Transaction on Multimedia*. accepted with minor revision.

[32] KANG, J., COHEN, I., AND MEDIONI, G. Soccer Player Tracking across Uncalibrated Camera Streams. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS) In Conjunction with ICCV* (2003), pp. 172–179.

[33] KELLY, A. A 3D Space Formulation of a Navigation Kalman Filter for Autonomous Vehicles. Tech. Rep. CMU-RI-TR-94-19, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1994.

[34] KHAN, S., AND SHAH, M. A multiview approach to tracking people in crowded scenes using a planar homography constraint. *European Conference on Computer Vision* (2006).

[35] KIM, K., AND DAVIS, L. Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering. *European Conference on Computer Vision, May* (2006).

[36] KITAGAWA, G. Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics 5*, 1 (1996), 1–25.

[37] KLAAS, M., BRIERS, M., DE FREITAS, N., DOUCET, A., MASKELL, S., AND LANG, D. Fast particle smoothing: if I had a million particles. *Proceedings of the 23rd International Conference on Machine learning* (2006), 481 -488.

[38] KONG, A., LIU, J., AND WONG, W. Sequential Imputations and Bayesian Missing Data Problems. *Journal of the American Statistical Association 89*, 425 (1994), 278–288.

[39] LIU, J., AND CHEN, R. Blind Deconvolution Via Sequential Imputations. *Journal of the American Statistical Association 90*, 430 (1995).

[40] LIU, J., AND CHEN, R. Sequential Monte Carlo Methods for Dynamic Systems. *Journal of the American Statistical Association 93*, 443 (1998), 1032 -1044.

[41] Lu, W., and Little, J. Simultaneous Tracking and Action Recognition using the PCA-HOG Descriptor. *Proceedings of the The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)-Volume 00* (2006).

[42] Mittal, A., and Davis, L. M 2 Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene. *International Journal of Computer Vision 51*, 3 (2003), 189–203.

[43] Morariu, V. I., and Camps, O. I. Modeling correspondences for multi-camera tracking using nonlinear manifold learning and target dynamics. In *Computer Vision and Pattern Recognition (CVPR)* (2006), pp. 545–552.

[44] Morefield, C. Application of 0-1 integer programming to multitarget tracking problems. *Automatic Control, IEEE Transactions on 22*, 3 (1977), 302–312.

[45] MOREIRA, P., REIS, L., and de Sousa, A. BEST MULTIPLE-VIEW SELECTION FOR THE VISUALIZATION OF URBAN RESCUE SIMULATIONS. *International journal of simulation modelling 5*, 4 (2006), 167–173.

[46] Murphy, K. Hidden Markov Model (HMM) Toolbox for Matlab. *online at http://www. ai. mit. edu/murphyk/Software/HMM/hmm. html.*

[47] Murphy, K. Kalman filter toolbox for Matlab, 1999.

[48] Ng, K., Ishiguro, H., Trivedi, M., and Sogo, T. An integrated surveillance systemhuman tracking and view synthesis using multiple omni-directional vision sensors. *Image and Vision Computing 22*, 7 (2004), 551–561.

[49] Nummiaro, K., Koller-Meier, E., Svoboda, T., Roth, D., and Van Gool, L. Color-based object tracking in multi-camera environments. *25th Pattern Recognition Symposium, DAGM 2003* (2003), 591–599.

[50] Okuma, K., Taleghani, A., de Freitas, J., Little, J., and Lowe, D. A Boosted Particle Filter: Multitarget Detection and Tracking. In *European Conference on Computer Vision* (2004), vol. I, pp. 28–39.

[51] ORWELL, J., REMAGNINO, P., AND JONES, G. Multi-camera colour tracking. *Proceedings of the 2nd IEEE Workshop on Visual Surveillance (VS99)*, 14–21.

[52] OTSUKA, K., AND MUKAWA, N. Multiview occlusion analysis for tracking densely populated objects based on 2-D visual angles. *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on 1*.

[53] PÈREZ, P., HUE, C., VERMAAK, J., AND GANGNET, M. Color-Based Probabilistic Tracking. In *European Conference on Computer Vision* (2002), vol. I, pp. 661–675.

[54] POORE, A. Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking. *Computational Optimization and Applications 3*, 1 (1994), 27–57.

[55] RABINER, L. A tutorial on hidden Markov models and selected applications inspeech recognition. *Proceedings of the IEEE 77*, 2 (1989), 257–286.

[56] STILLMAN, S., TANAWONGSUWAN, R., AND ESSA, I. A system for tracking and recognizing multiple people with multiple cameras. Tech. Rep. GIT-GVU-98-25, Georgia Institute of Technology, Graphics, Visualization, and Usability Center, 1998.

[57] STORMS, P., AND SPIEKSMA, F. An LP-based algorithm for the data association problem in multitarget tracking. *Computers and Operations Research 30*, 7 (2003), 1067- 1085.

[58] SUDDERTH, E., MANDEL, M., FREEMAN, W., AND WILLSKY, A. Visual Hand Tracking Using Nonparametric Belief Propagation. *Computer Vision and Pattern Recognition Workshop, 2004 Conference on* (2004), 189 189.

[59] VAN DER MERWE, R., A, D., DE FREITAS, J., AND WAN, E. The Unscented Particle Filter. Tech. Rep. CUED/F-INFENG/TR 380, Cambridge University Engineering Department, Cambridge, England, 2000.

[60] VAZQUEZ, P., FEIXAS, M., SBERT, M., AND HEIDRICH, W. Automatic View Selection Using Viewpoint Entropy and its Application to Image-Based Modelling. *Computer Graphics Forum 22*, 4 (2003), 689–700.

[61] VÁZQUEZ, P., AND SBERT, M. Fast adaptive selection of best views. *Lecture Notes in Computer Science* (2003).

[62] VERMAAK, J., DOUCET, A., AND PÈREZ, P. Maintaining Multimodality through Mixture Tracking. In *International Conference on Computer Vision* (2003), vol. II, pp. 1110-1116.

[63] VIOLA, P., AND JONES, M. Robust Real-Time Face Detection. *International Journal on Computer Vision 57*, 2 (2004), 137–154.

[64] XU, M., ORWELL, J., AND JONES, G. Tracking football players with multiple cameras. *Image Processing, 2004. ICIP'04. 2004 International Conference on 5* (2004).

[65] YUE, Z., ZHOU, S., AND CHELLAPPA, R. Robust two-camera tracking using homography. *Proceedings.(ICASSP'04). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. 3* (2004).