Multi-view Tracking of Soccer Players with Dynamic Cameras

by

Yixin Li

B.S. Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 2015

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 2016 [June 2016]

Signature of Author: Signature redacted
Department of Electrical Engineering and Computer Science
May 23, 2016

Certified by: Signature redacted
John W. Fisher III
Senior Research Scientist
Thesis Supervisor

Certified by: Signature redacted
Oren Freifeld
Post-Doctoral Associate
Thesis Co-supervisor

Accepted by: Signature redacted
Dr. Christopher Terman
Chairman, Master of Engineering Thesis Committee

Multi-view Tracking of Soccer Players with Dynamic Cameras

by

Yixin Li

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2016 in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Challenges such as player occlusion, fast player motion, small size of players relative to the background make it difficult to track soccer players accurately and consistently throughout a game. To solve these challenges, in this work we present a multi-view approach to tracking soccer players.

Here, we formulate tracking as the problem of assigning a label to each pixel in every frame of each camera view, where the label is either the background or one of the players. As a preprocessing step, we utilize the information from the soccer field for camera trajectory estimation and background modeling. Tracking is first carried out independently for each camera view with a layered tracker. Then we integrate the results of layered trackers from multiple views through MCMC inference over tracklet-to-player association.

We show that through camera calibration, common background and shared states of the players, inference across multiple camera views significantly alleviates the problem of player occlusion and loss of tracks in some view. As a result, we are able to produce accurate and long tracks for players, enabling further analysis of the game.

Thesis Supervisor: John W. Fisher III
Title: Senior Research Scientist
Thesis Co-supervisor: Oren Freifeld
Title: Post-Doctoral Associate

Acknowledgments

I would like to express my gratitude to my thesis supervisor, Dr. John W. Fisher, for his guidance and encouragement over the last two years.

I would also like to thank Oren Freifeld for our helpful discussions on all the relevant background materials and guidance over the past two years.

I would like to thank all of the SLI group members for providing me with an awesome experience in the group.

Lastly, I would also like to thank my family and friends, in particular, my parents and sister, whom I owe everything for their unconditional love.

Contents

List of Figures

Chapter 1

Introduction

Soccer is a popular sport around the world. To facilitate quantitative analysis of various aspects of the game, particularly player performance, automatic tracking of soccer players is needed. However, several challenges make tracking soccer players difficult. These include player-to-player occlusion, similar appearance of players from the same team, fast player motion, background illumination changes, and presence of shadows. Thus, a single camera is often inadequate to produce robust tracking of players. Instead, this thesis employs a multi-camera system for soccer tracking to address these challenges. We take as inputs manually synchronized videos of a soccer game recorded by three moving cameras overlooking the soccer field from different angles and aim to track players in these video recordings of real-world games.

Closely related to the task of multi-view soccer player tracking is the general problem of multi-view multi-target tracking, for which rich literature exists. Common approaches can be divided into two categories: "fuse-first" approaches and "track-first" approaches. In "fuse-first" approaches, observations from multiple views are fused first and then tracking is carried out. In contrast, "track-first" approaches perform tracking independently or collaboratively and then fuse the tracks that belong to the same target. No matter which approach is taken, some kind of data-association process, either on detections at frame level or on tracks, need to be performed. In this work, we take the "track-first" approach. Specifically, we integrate the results of individual layered trackers from multiple views to mitigate ambiguities.

We note that using multiple cameras to track soccer players is also not a completely new idea. People have utilized multiple cameras to take videos of the soccer field from different angels simultaneously, thus increasing the observability of the players. Tracking is then carried out in each camera view for non-occluded players and final player locations are determined by fusing the results from individual views [13]. But in our case, we only have a limited number of cameras, i.e., three, and each camera

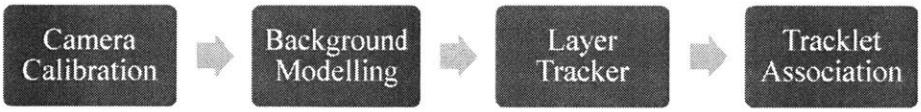


Figure 1.1: Overview of the soccer tracking pipeline

is moving in space and has a limited field of view. As a result, we sometimes do not have observations of some players as they step out of the camera views. We thus resort to inference for associating player identities to reappeared tracklets. In addition, our method is able to produce pixel-level results as opposed to bounding boxes of players.

■ 1.1 Our Approach

The main contribution we make is to adapt the existing single-view tracking algorithm to the specific problem of tracking soccer players, as well as show how multi-view enables us to associate consistent player identities to tracklets. Figure 1.1 gives an overview of the inference procedure we take.

First, we explicitly model the camera pose and background of the soccer field. The benefits are two-fold: at each frame of the video, we can obtain foreground moving objects through the calculation of background probability. Moreover, camera calibration allows us to map the player pixel locations to the world coordinate system, thus establishing correspondences among camera views.

Next, we apply the layered tracker from [4] for each camera view independently. The layered tracker assigns pixels to tracklets where each tracklet is defined as a layer, giving us pixel-to-layer assignment.

Then we combine independent tracking results across views by modelling the state of players via state-space models and establishing spatio-temporal correspondence of the same player across different views. We infer jointly across views and use the Markov-Chain Monte-Carlo method to sample from the posterior distribution of layer-to-player assignment. The layer-to-player assignment, combined with the pixel-to-layer assignment obtained from individual trackers, allows us to assign every pixel in the frame to either the background or one of the players, thus implicitly solving our tracking problem.

The remainder of this thesis is structured as follows. Chapter 2 introduces relevant background materials. Chapter 3 to 5 are devoted to each step of the tracking

procedure: Chapter 3 details how we perform camera calibration; Chapter 4 presents background modelling and single-view tracking; Chapter 5 discusses the problem of tracklet association. Finally, we conclude and point out future works in Chapter 6.

Chapter 2

Background Materials

In this chapter, we introduce the necessary background materials for the thesis. We start with a review of camera calibration. Then we introduce state-space models with Gaussian Hidden Markov Models. Finally, we discuss relevant works related to visual tracking, especially the so-called "tracking by detection" methods and Markov-Chain Monte-Carlo data association.

2.1 Camera Calibration

Pinhole Camera Model

Pinhole camera models the geometry of perspective projection [11] (Figure 2.1). It describes the relationship between the coordinates of a 3D point in the world and its projection onto the 2-D image plane. Images are formed by intersecting with the image plane all the rays that connect 3D points and the pinhole. In the case when distortion has already been accounted for, pinhole camera can be used to model real-

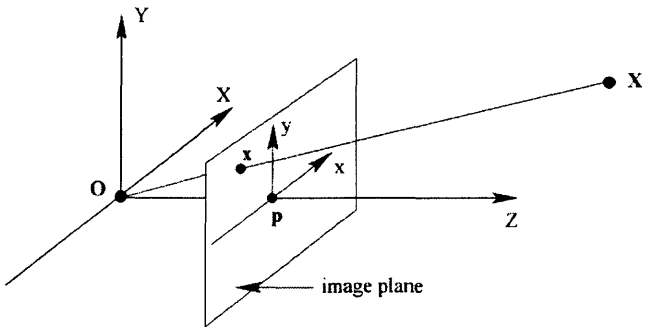


Figure 2.1: Perspective projection
[11]

world cameras.

With the pinhole camera model, we next review the concepts of camera *intrinsics* and *extrinsics*. First, we denote the *intrinsics* by matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$, where $\mathbf{K}$ is usually defined as

$$\mathbf{K} = \begin{bmatrix} f_x & sk & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

Here, f_x and f_y are the focal lengths in the horizontal and vertical direction in pixels; sk is the skew of the two image axes; c_x and c_y denote the principal point in pixels.

The *extrinsics* of a camera frame include $\mathbf{R} \in \mathbb{R}^{3\times3}$ and $\boldsymbol{T} \in \mathbb{R}^{1\times3}$, which are the rotation and the translation of the coordinate system of the camera frame relative to the world coordinate system. Together, $\mathbf{P} = \mathbf{K}[\mathbf{R}\ \mathbf{T}] \in \mathbb{R}^{3\times4}$ is called the camera projection matrix.

Since perspective projection is a non-linear transformation due to division by depth, homogeneous coordinates are introduced to make projection a matrix multiplication operation. Suppose we have a 3D point $\boldsymbol{M} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ in the world coordinate system, whose projection onto 2D image plane as pixel coordinates is $\boldsymbol{m} = \begin{bmatrix} x & y \end{bmatrix}^T$. Furthermore, let us denote the homogeneous-coordinate representations of $\boldsymbol{m}$ and $\boldsymbol{M}$ as $\widetilde{\boldsymbol{m}} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$ and $\widetilde{\mathbf{M}} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$, respectively. With the use of homogeneous coordinates, by pinhole camera model, the 3D point $\mathbf{M}$ is projected to the pixel coordinate $\boldsymbol{m}$ on the image by two matrix multiplications:

$$s\widetilde{\boldsymbol{m}} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{T} \end{bmatrix} \widetilde{\boldsymbol{M}}, \tag{2.1}$$

where s is a scalar.

The first multiplication $\begin{bmatrix} \mathbf{R} & \mathbf{T} \end{bmatrix}$ transforms $\widetilde{\mathbf{M}}$ to a 3D point in the camera coordinate system. Next, we have the multiplication by the camera intrinsic matrix $\mathbf{K}$, which further projects this 3D point in the camera coordinate system to $\widetilde{\boldsymbol{m}}$, the homogeneous-coordinate representation of the pixel location in the image plane.

Camera Calibration

If both camera intrinsics and extrinsics are known, we can project any 3D world point onto the image plane. But often it is the other way around: we do not know the intrinsic and extrinsic parameters and want to perform camera calibration instead.

Camera calibration refers to the extraction of both intrinsic and extrinsic parameters

from 2D images. One standard technique is to use a checkerboard pattern [21]. The corners of the checkerboard can be easily detected, yielding correspondences between pixel coordinates and world coordinates so that we have a number of $(\widetilde{m}, \widetilde{\mathbf{M}})$ pairs. Based on these correspondences, the camera projection matrix $\mathbf{P}$ can be calculated. Then we can obtain $\mathbf{K}$ and $[\mathbf{RT}]$ from $\mathbf{P}$ by QR decomposition.

In addition, camera calibration accounts for radial distortion. The latter occurs when light rays bend more near the edges of a lens than they do at the optical center. In addition, the distortion is more severe when the object is closer to the camera. We can model radial distortion with some distortion coefficient $\boldsymbol{k} = [k_1, k_2, k_3]$, such that a distorted pixel location $(x_{\text{distorted}}, y_{\text{distorted}})$ maps to an undistorted pixel (x, y) according to the equations

$$x_{\text{distorted}} = (1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)x \tag{2.2}$$

$$y_{\text{distorted}} = (1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)y \tag{2.3}$$

where $r^2 = x^2 + y^2$.

Multi-view Geometry

Now let us consider the case of two images. One common approach to recover camera poses and scene geometry is through the calculation of the fundamental matrix $\mathbf{F}$. The fundamental matrix $\mathbf{F}$ is a 3×3 matrix of rank 2. According to the epipolar constraint, if a point $\boldsymbol{M}$ in 3D is imaged as $\boldsymbol{m}$ in the first view, and $\boldsymbol{m'}$ in the second, then the image points pair satisfy the relationship $\boldsymbol{m'}^T\mathbf{F}\boldsymbol{m} = \mathbf{0}$. So from point correspondences we can solve for $\mathbf{F}$. Then from $\mathbf{F}$, the relative camera rotation and translation between two frames can be obtained. Moreover, we can perform *triangulation*, i.e. solve for the position of 3D point $\mathbf{M}$ from camera poses and image point correspondences.

However, there are two degenerate cases in which we are not able to recover the full $\mathbf{R}$ and $\mathbf{T}$. The first case is when the motion that the camera undergoes is a pure rotation. The second case is when the observed scene is a plane. Unfortunately, in our tracking scenario, the scene imaged is mostly planar (the ground of the soccer field). Furthermore, one of our dynamic cameras only rotates. We could use the other two more stable cameras for triangulation but these two cameras are placed at very far apart, making it difficult to automatically extract point correspondences.

For three or more views, it is possible to determine the geometry in a similar way as the case of two views. One popular method for optimizing the extrinsics is called

bundle adjustment. It works by minimizing the reprojection error between the image locations of observed and predicted image points. The correspondences among image points are required, as well as some initialization of the camera pose. Unfortunately, bundle adjustment can not be guaranteed to converge to the optimal solution from an arbitrary starting point.

Special Euclidean Group SE(3)

In this section we review the concepts of SE(3), SO(3) and so(3).

First, the special euclidean group SE(3) is defined as the set $\{(\mathbf{R}, \mathbf{T}) \in \mathbb{R}^{3\times3} \times \mathbb{R}^3 : \mathbf{R^T R} = \mathbf{I}_{3\times3}, \det(\mathbf{R}) = 1\}$. It contains all the rotations and translations in 3D.

Next, we can represent a 3D rotation in many ways. The first way is via a rotation matrix $\mathbf{R}$: $\mathbf{R} \in$ SO(3) iff $\mathbf{R}^T\mathbf{R} = \mathbf{I}_{3\times3}$ and $\det(\mathbf{R}) = 1$. Here, SO(3) is called the Special Orthogonal group and it contains all 3D Rotation matrices. The second way to represent a rotation is by a rotation vector $\boldsymbol{r}$ consisting of an axis of rotation (a line through the origin) and an angle of rotation. In this way, a rotation can be parametrized by using only 3 parameters.

The Lie algebra of SO(3), so(3), contains all 3×3 skew-symmetric matrices, which are elements of the tangent space of the manifold SO(3) at $\mathbf{I}_{3\times3}$. Matrix logarithm maps from SO(3) to so(3) and matrix exponential maps from so(3) to SO(3). In addition, the Rodrigues' rotation formula gives us the relationship between a rotation matrix $\mathbf{R}$ and a rotation vector $\boldsymbol{r}$ with the matrix logarithm/exponential given in closed forms:

Rodrigues' Rotation Formula. *If* $[u_x, u_y, u_z]$ *is a unit vector on the rotation axis and* θ *is the rotation angle (anticlockwise) from that axis, i.e.,* $\boldsymbol{r} = \theta * [u_x, u_y, u_z]$*, then the corresponding rotation matrix* $\mathbf{R}$ *is*

$$\mathbf{R} = \exp(\theta\mathbf{B}) = \mathbf{I}_{3\times3} + \sin(\theta)\mathbf{B} + (1 - \cos(\theta))\mathbf{B}^2 \tag{2.4}$$

$$\textit{where } \mathbf{B} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \in so(3).$$

■ 2.2 Message Passing in Gaussian Hidden Markov Model

Next, we switch our focus to state-space models for modelling object dynamics. Consider the Hidden Markov Model (HMM) shown in Figure 2.2. In this model, the states are $\boldsymbol{x}_i$ and the observations are $\boldsymbol{y}_i$. The Gaussian HMM can be expressed as a Linear

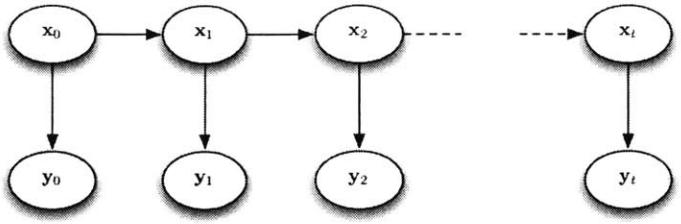
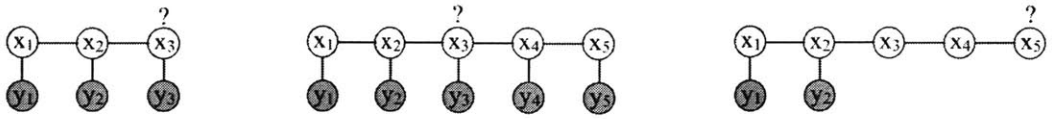


Figure 2.2: Hidden Markov Model



(a) filtering (b) smoothing (c) prediction

Figure 2.3: Different inference tasks on HMM. Shaded nodes represent observations.

Dynamical System. In particular, states evolve according to the linear dynamics

$$\boldsymbol{x}_{i+1} = \mathbf{A}\boldsymbol{x}_i + \boldsymbol{v}_i \tag{2.5}$$

where $\boldsymbol{v}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is a white Gaussian noise process and $\mathbf{A}$ and $\mathbf{Q}$ are known. Furthermore, the observation $\boldsymbol{y}_i$ at step i is generated as

$$\boldsymbol{y}_i = \mathbf{C}\boldsymbol{x}_i + \boldsymbol{w}_i \tag{2.6}$$

where $\boldsymbol{w}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is a white Gaussian noise process and $\mathbf{C}$ and $\mathbf{R}$ are also known.

The message passing or belief propagation algorithm aims to calculate the marginal of an unobserved node given all the observed nodes in the graph. Moreover, it does so for every unobserved node in an efficient manner by reusing the messages. In the special case of Hidden Markov models, there are different ways to rearrange forward-backward computation to efficiently produce marginals. One variant corresponds to the Rauch-Tung-Striebel (RTS) algorithm. In this algorithm, the forward and backward passes are referred to as Kalman filtering and smoothing. Here, filtering (use the past and current observations to predict the current state), smoothing (use all observations to estimate the current state) and prediction (use all observations to estimate some future state) refer to different inference tasks for a Hidden Markov Model (Figure 2.3).

Kalman Filtering

Kalman filter generates the marginals $p(\boldsymbol{x}_i \mid \boldsymbol{y}_{0:i})$ for $i = 0, 1, ..., t$, i.e., a marginal at node i based on the data only through step i. First, we note that marginals $p(\boldsymbol{x}_i \mid \boldsymbol{y}_{0:j})$ are Gaussian distributions. So let us define the mean and the covariance for the Gaussian distribution $p(\boldsymbol{x}_i \mid \boldsymbol{y}_{0:j})$ to be the state estimate $\hat{\boldsymbol{x}}_{i \mid j}$ and the error covariance matrix $\mathbf{P}_{i \mid j}$, respectively.

Suppose the initial state of the system is $\boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{x}_0; \hat{\boldsymbol{x}}_{0 \mid -1}, \mathbf{P}_{0 \mid -1})$. At each time step i for $i = 0, 1, ..., t$, the Kalman filter operates in two steps: a prediction substep followed by an update substep. In the prediction substep, we start with the current belief $p(\boldsymbol{x}_i \mid \boldsymbol{y}_{0:i})$, or equivalently, $\mathcal{N}(\boldsymbol{x}_i; \hat{\boldsymbol{x}}_{i \mid i}, \mathbf{P}_{i \mid i})$, and generate $p(\boldsymbol{x}_{i+1} \mid \boldsymbol{y}_{0:i}) = \mathcal{N}(\boldsymbol{x}_{i+1}; \hat{\boldsymbol{x}}_{i+1 \mid i}, \mathbf{P}_{i+1 \mid i})$ where

$$\hat{\boldsymbol{x}}_{i+1 \mid i} = \mathbf{A}\hat{\boldsymbol{x}}_{i \mid i} \tag{2.7}$$

$$\mathbf{P}_{i+1 \mid i} = \mathbf{A}\mathbf{P}_{i \mid i}\mathbf{A}^{\top} + \mathbf{Q}. \tag{2.8}$$

In the update substep, Kalman filter incorporates the observation $\boldsymbol{y}_{i+1}$, arriving at the posterior belief $p(\boldsymbol{x}_{i+1} \mid \boldsymbol{y}_{0:i+1})$. We compute the Kalman gain $\mathbf{G}_{i+1}$ and update the state estimate and error covariance by

$$\hat{\boldsymbol{x}}_{i+1 \mid i+1} = \hat{\boldsymbol{x}}_{i+1 \mid i} + \mathbf{G}_{i+1}(\boldsymbol{y}_{i+1} - \mathbf{C}\hat{\boldsymbol{x}}_{i+1 \mid i}) \tag{2.9}$$

$$\mathbf{P}_{i+1 \mid i+1} = \mathbf{P}_{i+1 \mid i} - \mathbf{G}_{i+1}\mathbf{C}\mathbf{P}_{i+1 \mid i} \tag{2.10}$$

where the Kalman gain can be pre-computed as

$$\mathbf{G}_{i+1} = \mathbf{P}_{i+1 \mid i}\mathbf{C}^{\top}(\mathbf{C}\mathbf{P}_{i+1 \mid i}\mathbf{C}^{\top} + \mathbf{R})^{-1}. \tag{2.11}$$

Kalman Smoothing

The backward pass of the RTS algorithm, which implements Kalman smoothing, directly generates the desired marginals $p(\boldsymbol{x}_i \mid \boldsymbol{y}_{0:t})$, i.e., a marginal at node i based on all the data from time 0 to t. It calculates $p(\boldsymbol{x}_i \mid \boldsymbol{y}_{0:t})$ from $p(\boldsymbol{x}_{i+1} \mid \boldsymbol{y}_{0:t})$ for $i = t, t-1, ..., 1, 0$, recursively by the following equations:

$$\hat{\boldsymbol{x}}_{i\,|\,t} = \hat{\boldsymbol{x}}_{i\,|\,i} + \mathbf{F}_i(\hat{\boldsymbol{x}}_{i+1\,|\,t} - \hat{\boldsymbol{x}}_{i+1\,|\,i}) \tag{2.12}$$

$$\mathbf{P}_{i\,|\,t} = \mathbf{F}_i(\mathbf{P}_{i+1\,|\,t} - \mathbf{P}_{i+1\,|\,i})\mathbf{F}_i^{\top} + \mathbf{P}_{i\,|\,i} \tag{2.13}$$

where

$$\mathbf{F}_i = \mathbf{P}_{i\,|\,i}\mathbf{A}^{\top}\mathbf{P}_{i+1\,|\,i}^{-1}. \tag{2.14}$$

Note that these marginals in the smoothing pass are generated in a way that does not require access to the observations any more, but only the marginals obtained in the forward filtering pass.

■ 2.3 Tracking by Detection

Object detectors such as the HOG human detector [5] have become increasingly accurate and powerful such that they are used as a key step in many vision applications. In the so called "tracking-by-detection" paradigm, some kind of object detector suitable for the task at hand is first used to produce detections, i.e. bounding boxes. Following the detection stage, the detections across frames are connected together to produce final tracks. The task of tracking thus becomes correctly linking the detections across successive frames, i.e. given a set of detections at the current frame, how to associate them to the past detections.

The main challenge faced by "tracking-by-detection' framework is that the number of plausible associations of detections across frames is huge. In addition, false detections as well as missed detections exist, complicating the association task. Many heuristic, statistical and graph-based algorithms have been developed to effectively explore the large search space.

Probabilistic approaches aim to find the best assignment of detections to tracks in order to maximize a certain likelihood function. For example, the joint probabilistic data association filter (JPDAF) [7] performs a time step by time step greedy measurement association with a fixed number of targets based on computing the posterior probability of each candidate measurement found. The multiple hypothesis tracker (MHT) [2] enumerates all possible tracks at every time step and over time, the track branches into many possible directions. To deal with this exponentially growing hypothesis space, heuristics are applied to restrict the search space. Another work proposes a three-level hierarchical association approach [12]. At the first level the method generate reliable

tracklets by linking detection responses based on conservative affinity constraints. At the middle level, these tracklets are further associated to form longer tracklets by formulating the association as a MAP problem and solving by the Hungarian algorithm. At the high level, entries, exits and scene occluders are estimated using the already computed tracklets, which are used to refine the final trajectories.

The data association method most closely related to our work is the Markov-Chain Monte-Carlo (MCMC) method. So we give a detailed review of it in the next section and explain how it is used in the context of data association for tracking.

■ 2.4 Markov-Chain Monte-Carlo Method

Markov-Chain Monte-Carlo (MCMC) method is a class of algorithms used to sample from a probability distribution $p(x)$ that is hard to sample from directly. For example, sometimes we do not know the partition function Z (i.e. $p(x) = \frac{p^\star(x)}{Z}$) but we can calculate $p^\star(x)$. When the distribution is hard to approximate, we can characterize the distribution with these samples. We construct a Markov chain $\mathcal{P}$ whose stationary distribution π on space Ω is the desired probability $p(x)$ that we want to sample from. Let us denote the state transition matrix for this desire Markov chain $\mathcal{P}$ as $\mathbf{P}$, where the (i,j) entry of $\mathbf{P}$ is the probability of transitioning from state $i \in \Omega$ to state $j \in \Omega$. Next, we review one popular MCMC approach, the Metropolis-Hastings algorithm.

Metropolis-Hastings Algorithm

We start with a "proposed" Markov chain $\mathcal{K}$ from which we can sample and we will modify it to create the desired Markov chain $\mathcal{P}$. We proceed as follows to combine this proposal distribution with some additional acceptance probability. At each iteration with the current state $\omega \in \Omega$, we propose a new state $\omega' \in \Omega$ according to our proposal distribution $K(\omega,\omega')$ (i.e., the state transition probability for the Markov chain $\mathcal{K}$). This proposal to move from ω to ω' is accepted with acceptance probability $R(\omega,\omega')$ where

$$R(\omega,\omega') = \min\left(1, \frac{\pi(\omega')K(\omega',\omega)}{\pi(\omega)K(\omega,\omega')}\right) \tag{2.15}$$

With this construction, we can show that the detailed balance condition is satisfied, i.e., for all $\omega,\omega' \in \Omega$, $\omega \neq \omega'$,

$$\pi(\omega)P(\omega,\omega') = \pi(\omega')P(\omega',\omega), \tag{2.16}$$

where $P(\omega, \omega') = R(\omega, \omega')K(\omega, \omega')$ is our desired transition probability. In other words, we have implicitly construct $\mathcal{P}$ from $\mathcal{K}$. Furthermore, it has been shown that if the Markov chain $\mathcal{P}$ is irreducible and aperiodic (i.e., ergodic), then it in fact converges to its stationary distribution π [17].

■ 2.5 Markov-Chain Monte-Carlo Data Association

Markov-Chain Monte-Carlo methods have been widely applied for association problem in the context of tracking ([8], [9], [20], [10], [15], [1]). Associations on different levels such as detections to tracks association or association among tracks have been considered.

Detection to Track Association

[15] shows how to use Markov-Chain Monte-Carlo data association (MCMCDA) method to sample the detection to track assignment. The authors show that unlike MHT and JPDA, MCMCDA is a true approximation scheme for the optimal Bayesian filter; i.e., when run with unlimited resources, it converges to the Bayesian solution. [8] considers targets whose dynamics are individually described by state space models and places a Dirichlet Process (DP) prior on the number of targets to be tracked. The Gibbs Sampling approach can be used to sample the associations in this case, due to the efficient way in which the Kalman smoother updates all the state marginals.

[20] removes the assumption that one detection can be assigned to at most one track. This assumption is incorrect in the case when detected foreground region does not correspond to one target faithfully, such as when occlusion happened and segmentation of foreground region is thus not "pure". [20] tries to find the best spatial and temporal association of the observations to maximize the consistency of both motion and appearance of trajectories and uses the Metropolis-Hastings method for sampling the associations.

Tracklet Association

Small fragments of tracks, or tracklets, offer a good mid-level representation that preserves spatio-temporal context for efficient tracking. Some works consider the association among tracklets. For example, [10] approaches multi-target tracking via tracklet association. First, tracklets are extracted by some point tracker. The authors then define the posterior of racklet association based on some measurement of track sim-

ilarity. They also use a general exponential model to define the prior probability of assignments, which incorporates factors such as the length of a track and the number of players. The search for the best association among tracklets, together with the search for model parameters, are carried out with the Metropolis-Hastings sampling procedure. The proposed approach is able to infer the optimal model parameters for different tracking scenarios in an unsupervised manner.

Similar to [10], [1] associates HOG detections with simultaneous KLT tracking. Where previous approaches such as [10] use ad-hoc models for data association, the authors use a more principled approach based on Minimal Description Length (MDL) to model the affinity between observations. Two kinds of MCMC moves are considered: add or remove a frame detection to a track, as well as switch parts of two tracks. By performing data association over a sliding window, the authors are able to correct many data association errors and fill in gaps for stable head location estimates in real-time.

Both [10] and [1] regard the problem of estimating object state as a separate task from inference over track association; they obtain the state estimate as a post-processing step after the MAP estimate of the track association is obtained.

Chapter 3

Camera Calibration

In this chapter, we discuss how we carry out camera calibration for all frames and all cameras. We explicitly model the camera pose and perform camera calibration because it enables us to determine the location of each player in our world coordinate system from input 2D images. From this projection we can establish correspondence for the same player in the overlapping views. Here, to handle the ambiguity caused by the fact that each 2D point corresponds to a ray in 3D, we work with only the feet position of players and assuming that the corresponding 3D points lie on the ground plane.

In the next three sections, we describe the graphical model for the problem and how we obtain the camera pose at each individual frame. We initialize the camera pose through manual labeling of points and running Kanade-Lucas-Tomasi (KLT) point trackers so that we can establish the correspondence between pixel locations in the camera frames and 3D points in the world coordinate system. By utilizing the orthogonality of lines on the field, we are able to optimize over the rotation matrix R. Finally, we apply sampling method Particle Swarm Optimization (PSO) to optimize the camera pose.

Note that in the data sets we work with, the camera is panning. As a result, changes in extrinsic parameters are predominantly due to rotation and translation is mostly negligible. However, we do not exploit this fact in our inference procedure and our method is general for the case of estimating both rotation and translation.

■ 3.1 Generative Background Model

Recall that camera calibration refers to the extraction of both intrinsic and extrinsic parameters of a camera from 2D images. Here, we assume that the intrinsics for each of our cameras are fixed across the entire duration of the video. In addition, the intrinsic matrix K, as well as the radial distortion coefficients k of each camera, are calculated

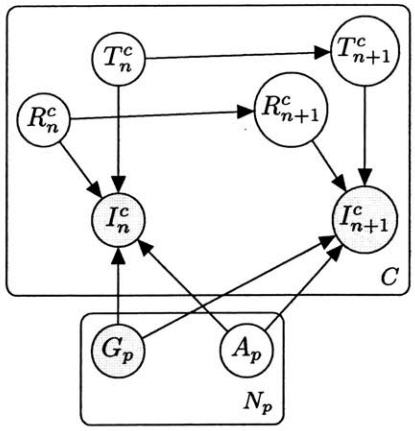


Figure 3.1: Graphical model of the camera calibration problem. The variables consist of background geometry G, background appearance A, camera frame I, camera rotation R and translation T, in which shaded nodes I and G are considered as known. C is the number of cameras and N_p is the number of primitives in the background model. Subscript n refers to the frame index. Note that we only show two time steps and the actual model contains all frames.

beforehand with the MATLAB calibration toolbox through different views of a planar grid pattern [21]. As the camera is either panning to follow to players or slightly moving due to the wind, what remains for our camera calibration problem is to find out the rotation R and translation T of all frames from all cameras.

Figure 3.1 shows the graphical model for our camera calibration problem. The key is that given the frame, the background and the camera pose become dependent. Let $T \in N^+$ denote the duration of tracking. Suppose we have a total of C cameras overlooking the soccer field. Let I_n^c represent the frame generated by camera c at frame n, for $n = 1, 2, ..., T$. Furthermore, I_n^c has already been rectified, i.e. we have account for radial distortion. Denote the rotation and the translation of the camera c at frame n as R_n^c and T_n^c, respectively. Finally, the background model consists of a set of 3D points P. Each point $p \in P$ has a 3D position G_p and color intensity A_p. For now, we assume G is known by fixing the points to lie on the world ground plane. In addition, we restrict P to those points that are inside the soccer field.

Finally, we define the world coordinate system by placing the origin at the middle of the lower side of the field. The x-y plane ($z = 0$) is the ground plane; the z direction is pointing upward from the ground plane. Figure 3.2 gives an illustration of the world coordinate system.

Figure 3.2: World coordinate system

■ 3.2 Camera Pose Initialization

In this section, we describe how we obtain an initialization of the camera pose, specialized for the case of soccer tracking. Camera pose at each frame needs to be initialized for a variety of reasons. The first necessity stems from the lack of background appearance at the start of the calibration process, as well as the need to initialize background appearance through some known camera pose. Furthermore, optimization requires a good initialization of the camera pose.

■ 3.2.1 Manual Labelling of Points

Similar to the case in which we calibrate a camera beforehand with a checkerboard pattern [21], in order to calculate the rotation matrix R and the translation vector T, we need to know the correspondences between some set of points in the image frame and their 3D world coordinates. To establish such correspondences, we manually label in frames a number of points on the ground for which we know their positions in the world coordinate system. Typically, these points are those on the markings of the field. Then we solve for R and T by minimizing the distance between pixels and the projection of the corresponding 3D points onto the image. This nonlinear minimization problem is solved by the Levenberg-Marquardt algorithm.

As it is impossible to manually label all frames, we track already labelled points across frames with a KLT point tracker, so that each frame has at least four points whose world coordinates are known. However, with a moving scene, the point tracker loses some of the points over time (for example, when points are out of sight, or occluded by players). To make tracking more robust, we hand-label a few frames that correspond to views of different part of the soccer field. When the camera comes closer to one of the labeled views, we re-initialize the point tracker. To decide which view the current

camera is closest to, we calculate the distance between the current camera pose and that of a particular labeled view. Since the camera is panning, only rotation is taken into account in this distance calculation. In other words, the distance is calculated as the distance between two camera rotation matrices R_1 and R_2: $\|logm(R_1^T R_2)\|_F$, where *logm* is the matrix logarithm and we take the Frobenius norm. This distance measures the angle (3D) between two rotations.

■ 3.2.2 Line Orthogonality

In addition to the KLT point tracker, we use the fact that many lines marked on the ground are parallel to the x axis or the y axis that defines our world coordinate system. This enables us to recover the rotation of the camera relative to the world.

We first use Hough transform to extract a number of straight lines from the image. For each line extracted, we back-project it onto the world ground plane and calculate the direction of the projected line l, $\boldsymbol{d}_l$, as a unit vector by

$$\boldsymbol{d}_l = \frac{(X_1, Y_1, 0) - (X_2, Y_2, 0)}{\|(X_1, Y_1, 0) - (X_2, Y_2, 0)\|} \tag{3.1}$$

with

$$(X_1, Y_1) = Proj(x_1, y_1; R, T, K), \tag{3.2}$$

$$(X_2, Y_2) = Proj(x_2, y_2; R, T, K), \tag{3.3}$$

where (x_1, y_1) and (x_2, y_2) are two pixels on the image that define the line l, $Proj(x, y; R, T, K) : \mathbb{R}^2 \mapsto \mathbb{R}^2$ is a function that back-projects a pixel (x, y) on the image to a point $(X, Y, 0)$ in the world coordinate system by equation 2.1.

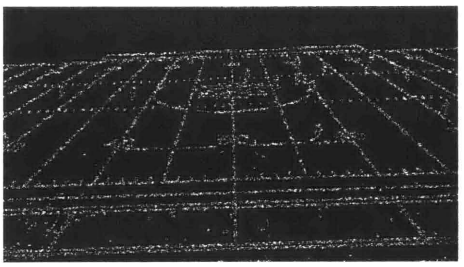
We model the direction of the field line, $\boldsymbol{d}_l$, as drawn from a mixture of four von Mises distributions with means equal to the positive and negative x and y axis of the world coordinate system. Each direction is equal likely to belong to any one of the von Mises distributions. In other words, we have

$$\begin{aligned} \mathbb{P}(\boldsymbol{d}_l; \kappa) =& \frac{1}{4} f(\boldsymbol{d}_l; \boldsymbol{e}_1, \kappa) + \frac{1}{4} f(\boldsymbol{d}_l; -\boldsymbol{e}_1, \kappa) + \frac{1}{4} f(\boldsymbol{d}_l; \boldsymbol{e}_2, \kappa) + \frac{1}{4} f(\boldsymbol{d}_l; -\boldsymbol{e}_2, \kappa) \\ &\propto \exp(\boldsymbol{d}_l \boldsymbol{e}_1) + \exp(-\boldsymbol{d}_l \boldsymbol{e}_1) + \exp(\boldsymbol{d}_l \boldsymbol{e}_2) + \exp(-\boldsymbol{d}_l \boldsymbol{e}_2) \end{aligned} \tag{3.4}$$

where $f(\boldsymbol{d}_l | \boldsymbol{\mu}, \kappa)$ is a von Mises distribution with mean $\boldsymbol{\mu}$ and concentration parameter κ, $\boldsymbol{e}_1 = [1, 0, 0]^T, \boldsymbol{e}_2 = [0, 1, 0]^T$.

We need to maximize $\prod_l \mathbb{P}(\boldsymbol{d}_l; \kappa)$ or equivalently, $\sum_l \log(\mathbb{P}(\boldsymbol{d}_l; \kappa))$, which has an

(a) Edges (shown as white) are extracted from the frame and dilated in width by factor of 2.



(b) A subset of 100,000 points are randomly sampled from dilated edges. Pixels that do not map onto the background are further excluded. The resulting pixels constitute the set S.

Figure 3.3: Set S of pixels over which the posterior probability of a camera pose is calculated.

analytical gradient. (Please refer to the appendix in Chapter 7, which gives a detailed derivation). We thus apply steepest gradient descent on R while using the T from the previous initialization step. The resulting pose estimate serves as an initialization to the optimization step below.

■ 3.3 Pose Refinement

After we obtain an initial estimate of the camera pose at each frame, we optimize this pose estimate further by maximizing its posterior probability based on current background appearance. We explain this posterior probability in 3.3.1 and then describe the optimization method taken in 3.3.2.

■ 3.3.1 Posterior Probability

According to our graphical model, the posterior probability of a camera pose given the background geometry G and appearance A, current frame I_n, intrinsic matrix K and previous frame's camera pose R_{n-1}, T_{n-1} is:

$$\begin{aligned}
&\mathbb{P}(R_n, T_n | I_n, A, R_{n-1}, T_{n-1}; G, K) \\
\propto &\mathbb{P}(R_n, T_n | R_{n-1}, T_{n-1}) \mathbb{P}(I_n, A | R_n, T_n; G, K) \\
\propto &\mathbb{P}(R_n, T_n | R_{n-1}, T_{n-1}) \mathbb{P}(I_n | A, R_n, T_n; G, K) \\
\propto &\mathbb{P}(R_n, T_n | R_{n-1}, T_{n-1}) \prod_{(x,y) \in S} \mathbb{P}(I_n(x,y) | A_p) + \mathcal{N}(0, \Sigma_c)
\end{aligned} \tag{3.5}$$

where p is the point on the background for which $G_p = Proj(x, y; R, T, K)$, Σ_c models the camera noise. Ideally, S should be the set of all pixels in the image that correspond to the background. However, we don't know which pixel corresponds to the background yet. In addition, multiplication over every pixel in the frame yields high computation cost. As a result, S should only contain pixels that are most informative with regard to the camera pose. For example, in [6], S includes only pixels that yield the most information about the parameters to be estimated. Following this idea, we let S to be pixels on the line markings of the field. To be more specific, we detect edges from the image by Canny edge detector, followed by a dilation to make edges thicker (Fig 3.3). A subset of 100,000 points are randomly sampled from the dilated edges and pixels that do not map onto the background are further excluded, finally resulting in a set of pixels S over which we calculate the posterior probability of a camera pose.

The term $\mathbb{P}(R_n, T_n | R_{n-1}, T_{n-1})$ models the camera motion and encodes our prior belief about the camera movement. In the current implementation, this prior is specified as a Gaussian distribution with a diagonal covariance, with the magnitude of standard deviation set according to the type of camera motion, i.e., whether the camera is nearly static or is panning.

■ 3.3.2 Approximating the MAP Estimate via PSO

Given the posterior probability of a camera pose, we want to obtain a MAP estimate. The parameters consist of a 3D translation vector T and a 3D rotation vector r. There are several ways to explore this 6-dimensional search space. For example, we can either optimize each of the six parameters in turn, or use MCMC sampling and keep the sample that yields the maximum posterior probability. We notice that our optimal solution resides in a very narrow place in the search space (Figure 3.4 shows how negative log likelihood changes when only one parameter is changing for a sample reference image). In addition, MCMC sampling is slow due to its sequential nature. As a result, we consider another sampling-based optimization approach, Particle Swarm Optimization (PSO) [14], which has a good balance of speed and accuracy. Although the method is very heuristic-based and does not even guarantee to find any local optimum, [16] and [18] use it for fitting hand model to depth data to achieve accurate real-time hand tracking.

PSO maintains a swarm of candidate solutions as particles. In each iteration, each particle moves in the search space and remembers its best position. PSO improves over random sampling in that a particle moves in a way that is drawn toward its best

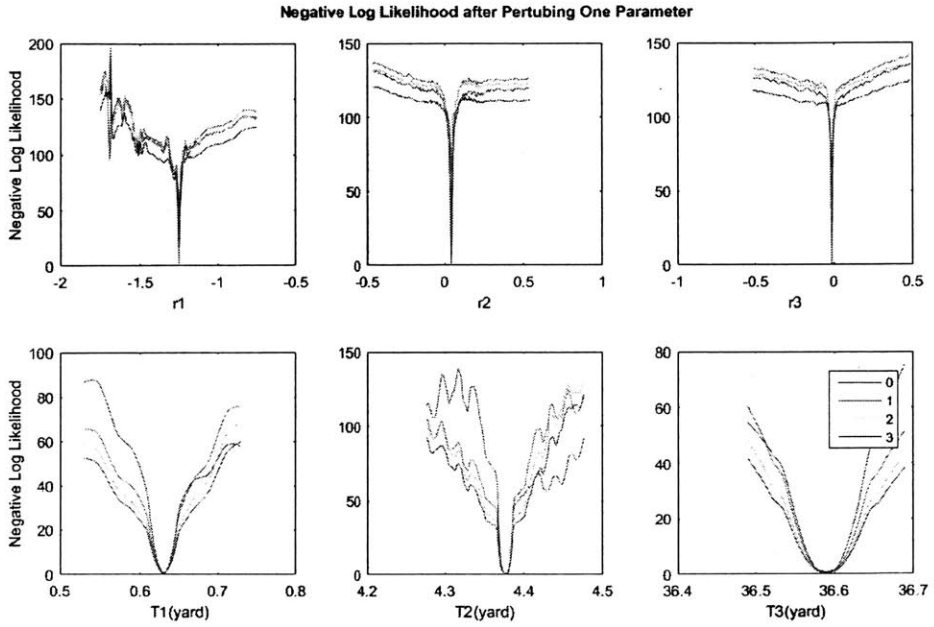


Figure 3.4: Change in the negative log likelihood when only one parameter is changing for a sample reference image. The colors of the lines encode the width of dilation.

position so far as well as toward the swarm's best position. Define v_k and x_k to be the velocity and position of the particle at iteration k. B_k is the particle's best position up to iteration k and G_k is the swarm's best position up to iteration k. The particle re-estimates its velocity and position in every iteration k according to the update equations

$$v_{k+1} = wv_k + c_1 r_1 (B_k - x_k) + c_2 r_2 (G_k - x_k) \tag{3.6}$$

and

$$x_{k+1} = x_k + v_{k+1}, \tag{3.7}$$

where w, c_1, c_2 are some constants; r_1 and r_2 are random numbers in the uniform range over [0 1].

■ 3.4 Implementation and Results

For PSO optimization, the initial particles are set with zero initial velocity and drawn from a Gaussian distribution whose mean is the initial estimate of the camera pose. The variance of this Gaussian is set empirically as follows: for a ground-truth frame,

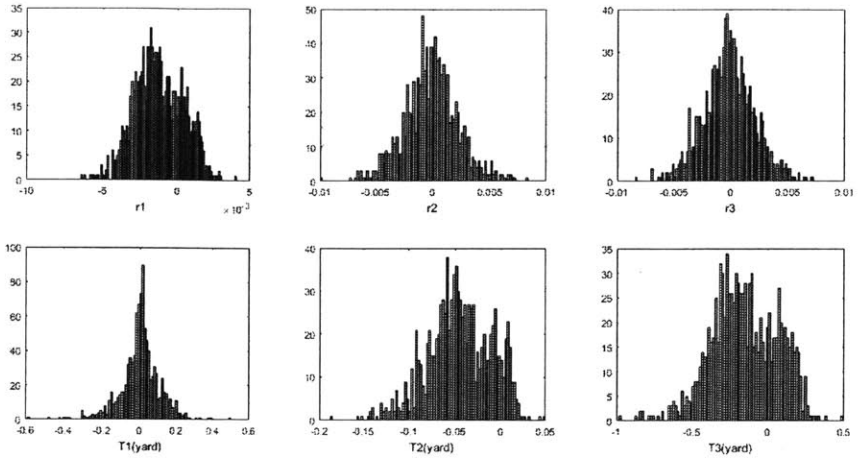


Figure 3.5: Histogram of Rs and Ts obtained based on 1000 draws of 4 randomly chosen points. Each plot corresponds to variability in one parameter of the search space (Top row is for the rotation vector r and the bottom row is for the translation vector T).

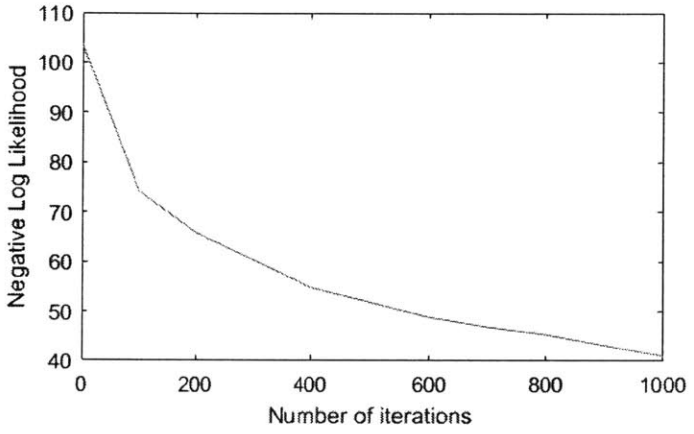


Figure 3.6: Negative log likelihood vs. iteration number. Note that our negative log likelihood does not go to zero. One reason is that we calculate the posterior probability on set S that does include pixels on the frame that correspond to players.

each time we randomly sample 4 labeled points in the image and calculate R and T based on them. Figure 3.5 shows the histogram of camera poses obtained from 1000 random draws. Based on the histogram, we set the standard deviation to [0.005, 0.005, 0.005, 0.1, 0.1, 0.5]. The number of particles is set to 100. The current implementation uses the MANOPT MATLAB library [3] for optimization on SE(3).

Figure 3.6 plots the change in log likelihood during optimization and the negative log likelihood decreases as the iteration number increases for a sample frame. From our experiments, we conclude that we can both initialize the camera pose based on soccer field properties, as well as further refine the current camera parameters using current background appearance.

Chapter 4

Layered Tracking

In this chapter, we discuss about independent tracking for each camera view. We use standard layered trackers, but specialize them for the case of soccer tracking. We start with an explanation of the background model and the calculation of probability that a pixel belongs to the background. Next, we proceed to the details of layered tracker that gives us pixel to layer assignment. We end the chapter with a discussion of player model, shadow detection as well as how the tracked layers are used as observations for the next tracklet association step.

■ 4.1 Background Probability with Gaussian Mixture Models

We use the Gaussian Mixture Model (GMM) detailed in [19, 22] to model the background appearance and calculate the probability that a pixel belongs to the background. The benefit of having multiple Gaussians as opposed to a single Gaussian is to deal more robustly with lighting changes and repetitive motions of scene elements.

In particular, the intensity value of each pixel on the ground plane is modelled with a mixture of Gaussian distributions. Each Gaussian in the mixture has a weight, intensity mean and intensity covariance. More specifically, let us define the intensity value for the pixel p in frame t to be $c_{p,t}$. Then the probability of observing the current pixel value is

$$\mathbb{P}(c_{p,t}) = \sum_{i=1}^{K} w_{i,p,t}\mathcal{N}(c_{p,t}; \mu_{i,p,t}, \Sigma_{i,p,t}) \tag{4.1}$$

where K is the number of Gaussians used to model the pixel and $w_{i,p,t}, \mu_{i,p,t}, \Sigma_{i,p,t}$ are the weight, intensity mean and intensity covariance for the ith Gaussian for pixel p at time t.

When a new frame comes in, we first project it onto the ground world plane using estimated camera parameters. Then the parameters (weights, means and covariances) of

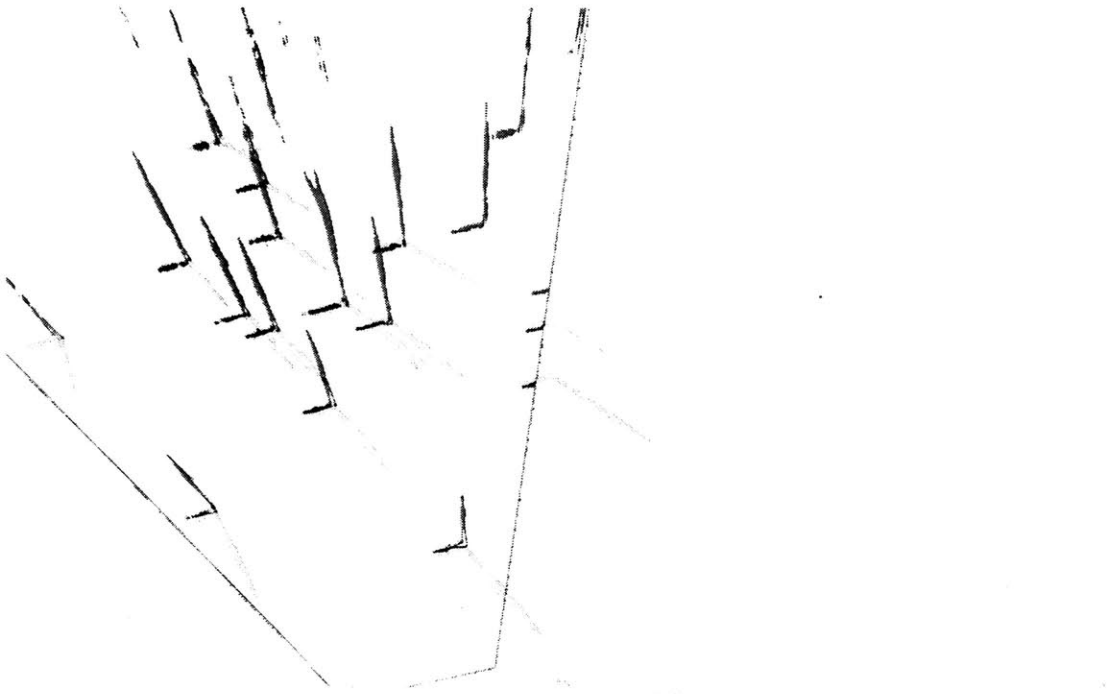


Figure 4.1: Top-down view of the probability that a pixel belongs to foreground. We superimpose the projections for three camera views. The choice of color (magenta, yellow, blue) encodes the camera view while the intensity encodes the probability value. We see that the pixels correspond to the players are successfully associated with high foreground probability.
Notice that: 1. The shadows of players (in dark blue) overlap on the ground plane. 2. We are missing observations for some players at the center of the field and we only observe their shadows. This poses one challenge for multi-view tracking. 3. Players at far away have been segmented into multiple parts and are hard to detect because some part of them have similar appearance with the shadow on the ground. This represents another challenge for our multi-view tracking problem.

the Gaussians are updated differently depending on the intensity value of the projected pixel. If the pixel has a high probability under any of the existing Gaussians, then the parameters of that Gaussian are updated to incorporate the new evidence using a simple moving average. Otherwise, a new Gaussian is created for this single pixel with very low weight.

Among the mixtures of Gaussians for the pixel p at time t, the ones with the largest weights corresponds to the background, while the rest of the Gaussians correspond to the foreground. In other words, the first B distributions are chosen as the background model with

$$B = \arg\min_{b} \sum_{k=1}^{b} w_{k,p,t} > T, \tag{4.2}$$

where T is the minimum portion of the Gaussians that are regarded as background. The probability that pixel p belongs to the background is calculated as the probability under the mixture of first B Gaussians:

$$\mathbb{P}(p \text{ is background}) = \frac{\sum_{i=1}^{B} w_{i,p,t}\mathcal{N}(c_{p,t}; \mu_{i,p,t}, \Sigma_{i,p,t})}{\sum_{i=1}^{K} w_{i,p,t}\mathcal{N}(c_{p,t}; \mu_{i,p,t}, \Sigma_{i,p,t})}. \tag{4.3}$$

Since this GMM model for background assumes independence among all the pixels, we additionally run a belief propagation on the Markov Random Field representation of the image. In Figure 4.1, we show the superimposition of three foreground probability images projected onto the top-down views.

Background gradually changes for reasons such as illumination and shadows. As a result, instead of having a fixed background appearance, we update the Gaussian mixture models as new frames come in. To decrease the running time, this update is only carried out every few frames. Note that we still need to calculate the background probability detailed above on every frame. As the Gaussian mixture model does not generate a single background appearance, we keep a background appearance image by the moving average of observed pixels that have background probability larger than one half. Figure 4.2 shows an example background mean appearance for visualization.

■ 4.2 Layered Tracker

Next, we apply a simplified version of the layered tracker [4] in each view independently. [4] approaches multi-target tracking problem on pixel-level and does not have an explicit object detection stage. Instead of solving the general detection association problem, it

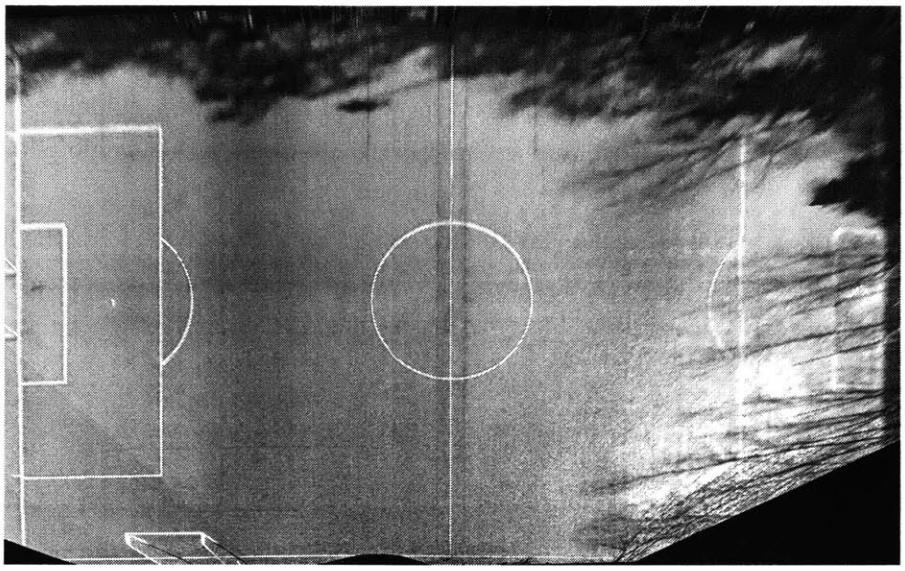


Figure 4.2: An example of background mean appearance

works on boundary-accurate assignment and considers each object as one layer. The support of a layer consists of a number of pixels which form a single connected component and layer ordering determines the visible part of the support. [4] models the evolution of appearances and layer shapes, while inferring about explicit ordering of the layers and a Gaussian process flow. Note that background is also considered as one layer and the probability of a pixel belongs to the background layer is given by us as an input. The overall inference procedure is summarized in Algorithm 1. Figure 4.3 shows the snapshots of layered trackers in progress.

for $t = 1$ *to* T **do**
1. Initialize new layers if needed
2. Calculate pixel observation probability under each layer
3. Infer layer supports
4. Optimize layer ordering
5. Update layer parameters

end

Algorithm 1: Overivew of Layered Tracker

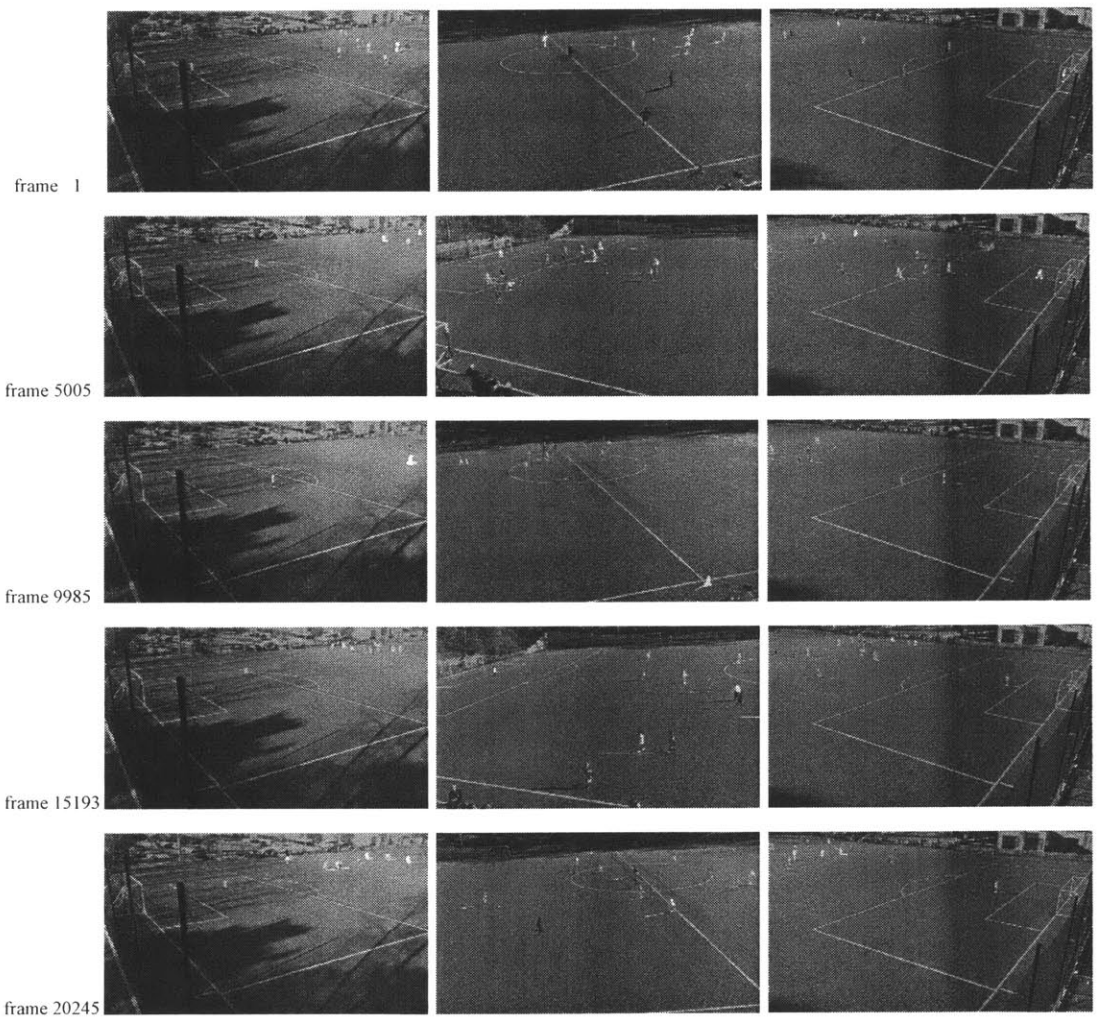
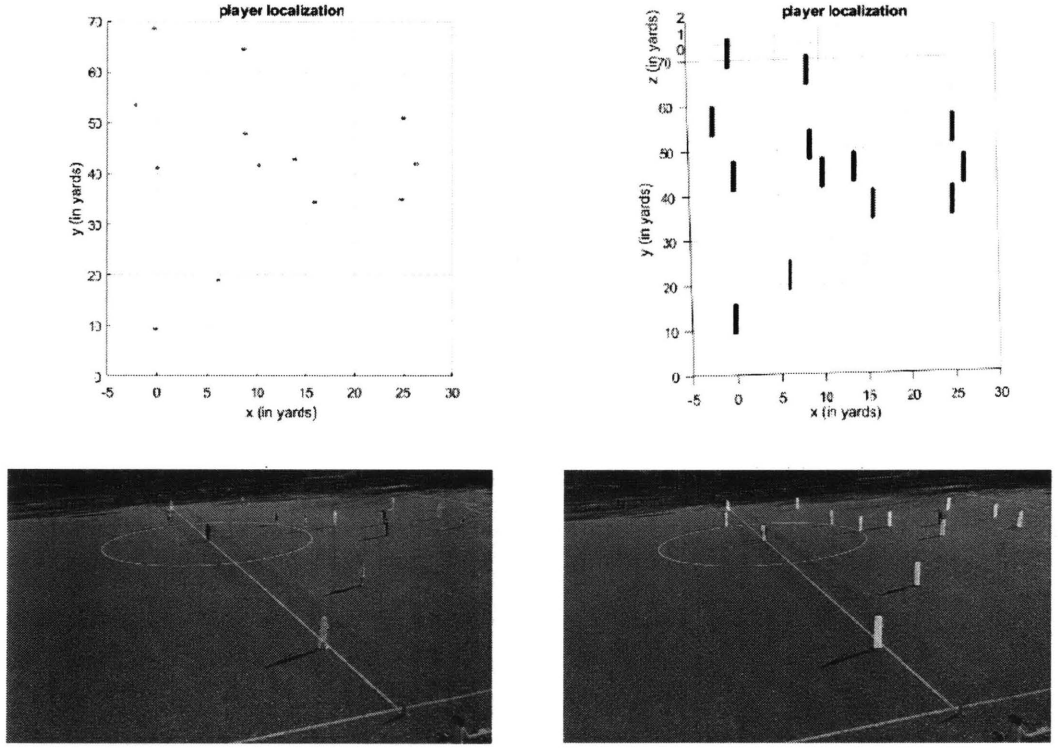


Figure 4.3: Outputs of three layered trackers. Each column corresponds to one camera.



(a) Detected player masks by the layered tracker

(b) We overlay the projection of cylinders to the frame on detected player masks.

Figure 4.4: Player localization. We can infer the position of player via the overlap between layer support mask and the projection of the player cylinder model onto the camera frame.

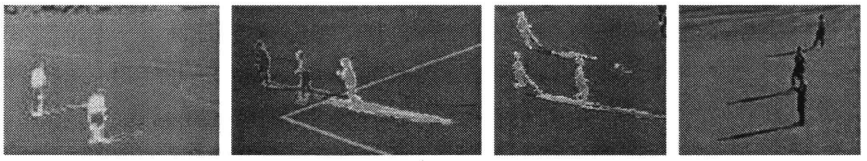


Figure 4.5: Presence of shadows connect two originally un-occluded players into one blob.

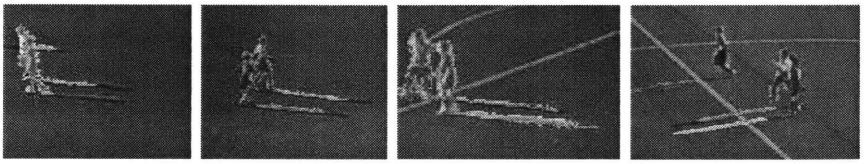


Figure 4.6: Presence of shadows helps us differentiate between two occluded players.

■ 4.3 Player Model

The player model should be powerful enough to differentiate between players in one camera view, while recognizing the same player across camera views. Each player is modelled as a cylinder standing on the ground plane, with a radius of 0.6 foot and a height of 5.4 feet. Given the camera parameters and the player location in the world coordinate system, we can project the players cylinders onto the camera frame, as well as obtain the depth ordering of players for a given camera view. In this way, given a detected layer support, we can infer the location of players in the world coordinate system by a generative model that takes into account the overlap between the detected player binary support mask and the binary mask of the cylinder projection onto image frame (Figure 4.4).

■ 4.4 Shadow

One difficulty that complicates tracking is the presence of shadows of players. Due to player occlusion, one detected foreground region does not always faithfully correspond to one target. This occlusion problem is further complicated by the presence of shadows, which connect originally unoccluded players into one region (Figure 4.5).

Shadows can be subtracted out based on the difference from the hue value of the background. In our specific videos, we have strong shadows that no longer have the color of the background, i.e. we do not have green-black shadow but black shadow. We find it easier to differentiate shadows from players based on the orientation of the

shape: in our camera views, players appear to be vertical while shadows are mainly horizontal. As a result, we model the spatial distribution of the pixels on the binary support mask as a single Gaussian. From the learned spatial covariance value, we can know the relative spread of the pixels in the 2D x-y space. Then we can classify a binary mask into either shadow or non-shadow based on the spatial covariance.

Although shadows merge detected blobs into one region, they are helpful in the soccer tracking scenario as they provide additional information about player locations. Shadows can be thought of as created by an accidental camera, in effect, the sun. When two players are occluded in one camera view, their shadows are unlikely to completely overlap with each other. As a result, shadows can be used to mitigate the problem of player occlusion (Figure 4.6). So we track the shadows throughout instead of subtracting out the shadow pixels.

■ 4.5 Observation Preparation

Finally, we discuss how we extract useful information from the layers for use in the tracklet association step.

We call the track fragments generated by layer trackers as tracklets. First, let $\text{Tracklet}_{i,c}$ denote the ith tracklet obtained by camera c during the interval $[1, T]$. We define $\text{Tracklet}_{i,c}$ as a set that contains the binary mask of the support pixels and the visible pixels for every frame index during the length of the tracklet. In other words, $\text{Tracklet}_{i,c} = \{S_t, V_t, \forall\, t \in [t_1, t_2],\ 1 \leq t_1 \leq t_2 \leq T\}$, where S_t is a binary mask of support pixels and V_t is an image of visible pixels, t_1 and t_2 are the starting time and the ending time of the tracklet, respectively. Thus, the output of layered trackers can be thought of a set of tracklets $\{\text{Tracklet}_{i,c}, \forall i = 1, 2, ..., n_c, c = 1, 2, ..C\}$, where C is the number of cameras and n_c is the number of tracklets generated by camera c.

From the support mask of $\text{Tracklet}_{i,c}$, we extract the position and velocity of player in the world coordinate system, denoted as $y_{i,c}$. The extraction of player location is based on whether the layer is classified shadow or not. We first classify each time instance of the layer into shadow or non-shadow. If a layer is classified as a non-shadow layer, we extract the median lowest position of the support mask, corresponding to the feet of players. Then we project this position onto the ground world plane. If a layer is classified as a shadow layer, then we extract either the leftmost or the rightmost point on the support mask, depending on the viewing angle of the camera. Then as with the non-shadow case, we project this location to the world coordinate system using the

obtained camera parameters. During the tracklet association step in the next chapter, we treat $y_{i,c}$ as known observations.

In conclusion, in this chapter we describe and demonstrate the utility of multiple layered trackers in producing tracklets for individual camera views. In addition, we propose a cylinder player model that, combined with the estimated camera parameters, enables us to infer about player locations in the world coordinate system from layer support. We also discuss about shadow detection and how the tracker output is further processed for use in the tracklet association step detailed in the next chapter.

Chapter 5

Tracklet Association

In this chapter, we discuss the problem of tracklet association. Recall that we take the "track-first" approach in which each camera processes its own frame and keeps tracklets separately. An important question is to decide whether two tracklets generated by different cameras represent the same player. If so, the next problem is how to combine the tracklets assigned to the same player and produce the final track. To solve the tracklet-to-player assignment problem, we first introduce the necessary probability model. Given the huge number of possible associations, the Markov-Chain Monte-Carlo sampling method is applied. We model the dynamics of the players with independent state-space models as a natural way to merge tracklet estimates. This also allows for efficient estimation of players' states via Kalman smoothing. Finally, we discuss the results of our tracklet association algorithm.

Note that our formulation works with a batch setting in which we consider all available tracklets at once, but it is easily extensible to an online setting. In addition, we assume that the input layered tracklets are "pure", i.e., different sections of the tracklet correspond to the same player. Extending the algorithm to handle tracklets which are not "pure" is straightforward, but is beyond the scope of this thesis.

■ 5.1 Probability Model

Recall that the layer trackers for different camera views output a set of tracklets for the time interval $[1, T]$. We have defined $\text{Tracklet}_{i,c}$ in Chapter 4.5 as the ith tracklet generated by camera c. A tracklet contains the binary mask of support pixels and an image of visible pixels for all the frames during the start and end of the tracklet. In addition, recall that we have defined the observation for $\text{Tracklet}_{i,c}$ as $y_{i,c}$. Here, $y_{i,c}^t$ is the location of player at time t in terms of the world coordinate system, for $t = t_1, ..., t_2$ where t_1 and t_2 are the starting and ending frames of $\text{Tracklet}_{i,c}$. Finally, let us define

the neighbors of $y_{i,c}$ as $\text{Neigh}(y_{i,c})$, which is a set of indices of those tracklets that are generated by the same camera c and overlap with $\text{Tracklet}_{i,c}$ in time.

Next, we define the assignment Z from tracklets to players. Suppose there are K unknown moving targets that appear within the entire time interval $[1, T]$. We define the label of $\text{Tracklet}_{i,c}$ to be $z_{i,c}$, where $z_{i,c} \in \{0, 1, 2, ..., K\}$ is the id of the player assigned and $z = 0$ represents a false-alarm detection. The assignment Z can also be thought of as a partition of all of the tracklets into K disjoint sets, with each set corresponding to exactly one player.

The only constraint that we put on Z is

$$z_{m,c} \neq z_{i,c} \text{ for } z_{i,c} \neq 0, \forall\, i = 1, 2, .., n_c, c = 1, 2, ..., C, m \in \text{Neigh}(y_{i,c}), \tag{5.1}$$

where n_c is the number of tracklets generated by camera c and C is the number of cameras. This constraint states that for every camera view at every time instance, a player can not appear more than once. In case we also track shadows, we add a similar constraint for the players' shadows. The constraint limits the range of valid assignments and we will discuss implementation details in section 5.2.

Next, we specify the probability model for our tracklet association problem. First of all, we assume that the players are independent of each other. The prior probability of an association value $\mathbb{P}(Z)$ incorporates our prior belief about the specifics of the tracking task. Here, we define it to encourage longer tracks and fix the number of targets K to be the total number of players plus one referee, which is 23. So we have

$$\mathbb{P}(Z) \propto \prod_{k=1}^{23} \frac{l_k}{T} \tag{5.2}$$

where l_K is the length of kth player's track.

The likelihood of tracklet observations given the tracklets' associations to players, $\mathbb{P}(Y|Z)$, should incorporate features of the players such as motion, color, height and encourage temporal consistency of the players. We define this likelihood by describing the dynamics of players using state-space models. More specifically, let us denote the latent state of player m at frame t as x_m^t. The latter includes the position and velocity of player m in terms of the world coordinate system: $x_m^t = \begin{bmatrix} \text{pos}_x & \text{pos}_y & v_x & v_y \end{bmatrix}^T$. We could also easily add the player appearance into the likelihood term. For example, by a color histogram or a Gaussian Mixture Model.

We model the motion of players to be Markovian and linear with constant velocity,

i.e. the state update equation for player m is

$$\boldsymbol{x}_m^{t+1} = \mathbf{A}\boldsymbol{x}_m^t + \boldsymbol{w}_t \tag{5.3}$$

where $\boldsymbol{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is a white Gaussian noise process and $\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

The observation model is also linear:

$$\boldsymbol{y}_{i,c}^t = \mathbf{C}\boldsymbol{x}_m^t + \boldsymbol{v}_t \text{ for } m = z_{i,c} \tag{5.4}$$

where $\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$, and $\boldsymbol{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ a white Gaussian noise process.

So finally we have the likelihood of tracklet observation as

$$\mathbb{P}(Y|Z) \propto \int_X \mathbb{P}(Y|X,Z)\mathbb{P}(X|Z) \tag{5.5}$$

$$= \prod_{c=1}^{C}\prod_{i=1}^{n_c} \int_{x_{z_{i,c}}} \mathbb{P}(y_{i,c}|x_{z_{i,c}})\mathbb{P}(x_{z_{i,c}}) \tag{5.6}$$

$$= \prod_{c=1}^{C}\prod_{i=1}^{n_c} \mathcal{N}(y_{i,c}; \hat{x}_{z_{i,c}}, \mathbf{P}_{z_{i,c}}) \tag{5.7}$$

$$= \prod_{c=1}^{C}\prod_{i=1}^{n_c}\prod_{t=t_1}^{t_2} \begin{cases} \mathcal{N}(y_{i,c}^t; \hat{x}_{z_{i,c}}^t, \mathbf{P}_{z_{i,c}}^t), & \text{if } z_{i,c} \neq 0 \\ p_0, & \text{if } z_{i,c} = 0 \end{cases} \tag{5.8}$$

where $\hat{x}_m^t$ and $\mathbf{P}_m^t$ are the Kalman smoothed estimate of the state and the error covariance at time t for target m given all the observations assigned to target m and p_0 is the background noise probability for false alarms (Please refer to section 2.2 for the details of Kalman filtering and smoothing). First, we note that the assumption of independence among players allows us to factorize $\mathbb{P}(Y|Z)$ into multiplication over observation probability for each tracklet. Moreover, the Kalman smoothed state estimates summarize our belief about player states. The independence between observations at different time steps given the player states further enables us to factorize the tracklet observation probability $\mathbb{P}(y_{i,c}|z_{i,c})$ into multiplication over observation probability at every time step in the duration of the tracklet.

5.2 MCMC Inference

Inputs : $y, n_{mc}, z^\star = z^{(0)}, \hat{x}, \mathbf{P}$
Output: $z^\star$
for $n = 1$ *to* n_{mc} **do**
  Propose z' according to $q(z, z')$
  Sample U from Unif[0, 1]
  if $U < A(z, z')$ **then**
    $z_n = z'$
    update $\hat{x}$, $\mathbf{P}$ via Kalman smoothing
  else
    $z_n = z$
  end
  if $\frac{p(z_n \mid y)}{p(z^\star \mid y)} > 1$ **then**
    $z^\star = z_n$
  end
end

Algorithm 2: MCMC Tracklet Association

Given the huge number of possible assignments, similar to [8], [20], [10], [15] and [1], we resort to MCMC method for sampling from the assignment posterior $\mathbb{P}(Z|Y)$. For our specific problem of tracking soccer players, we output the maximum a posterior (MAP) estimator as the optimal partition of tracklets:

$$z^\star = \underset{z}{\text{argmax}}\mathbb{P}(z|Y) = \underset{z}{\text{argmax}}\mathbb{P}(Y|z)\mathbb{P}(z). \tag{5.9}$$

An overview of our MCMC tracklet association algorithm is shown in Algorithm 2. The inputs to the algorithm are the tracklet observations y, an initial assignment $z^{(0)}$, Kalman smoothed state estimate $\hat{x}$ and $\mathbf{P}$ based on $z^{(0)}$ and y and the total number of MCMC iterations n_{mc}. We construct a Markov Chain whose stationary distribution is $\mathbb{P}(Z \mid Y)$. The acceptance probability $A(z, z')$ is defined as

$$A(z, z') = \min\left(1, \frac{\pi(z')q(z', z)}{\pi(z)q(z, z')}\right). \tag{5.10}$$

In the next subsection, we specify the proposal distribution $q(z, z')$.

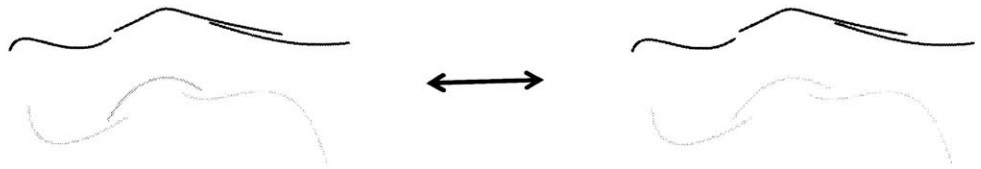


Figure 5.1: Graphical illustration of the MCMCDA move. Each curve represents one tracklet. Associations are indicated by curves with the same color and gray represents unassigned.

Proposal Distribution and MCMC Moves

We implement the following pair of moves. Each type of move is selected randomly at each iteration. Figure 5.1 shows a visualization.

(1) Assign a player id to an unassigned tracklet. We select a tracklet uniformly at random from the pool of unassigned tracklets. Assign it to the player with which the tracklet observation has maximum observation probability based on the current estimate of the player state. In addition, we consider only those players that are still available for this tracklet to ensure that a player does not appear more than once. If no player is available, the move is rejected.

(2) Move an assigned tracklet to the unassigned pool. We select an assigned tracklet based on its observation probability and move it to the unassigned pool. If the tracklet's original assigned player only has this tracklet, then the move is rejected since the move is not reversible.

Notice that at each MCMC iteration, we do not need to evaluate the observation probabilities over all the tracklets. Instead, we only need to calculate the change of observation probability for the tracklet proposed, as well as the change of prior assignment probability for the players involved. The update of the player state estimation via Kalman smoothing is also efficient, since we do a forward filtering pass followed by a backward smoothing pass only over the interval of the new tracklet observations.

Implementation Details

Recall that we constrain valid assignments to those in which a player does not appear more than once in a camera view for the entire duration of tracking. To efficiently look up possible moves, i.e., available player ids for an unassigned tracklet, for each camera we maintain a neighborhood graph. The nodes in the graph are all the tracklets generated by that camera. If two tracks overlap in time, then there is an edge connecting

them. In this way, we can decide available player ids for an unassigned tracklet by subtracting out the player ids of neighboring nodes.

Next, we describe how we obtain the initial assignment of tracklets to players. The initial assignment $z^{(0)}$ is obtained by a "greedy" scheme as follows. First, we manually assign correct player ids to all the tracklets that appear at frame 1. Then we sort the remaining tracklets by the starting time and work on assigning players id to these tracklets in the ascending order. We calculate the distance between two tracklets as the average of the distances during those frames when the two tracklets overlap. Then for each unassigned tracklet, we assign it to the player identity of the closest already-assigned tracklet. If this unassigned tracklet does not overlap with any assigned tracklet in time, we leave it unassigned. We run the assignment procedure backward one more time to have more tracklets assigned.

■ 5.3 Results and Discussion

We run our algorithm on the layers extracted by three layer trackers. Figure 5.2 shows the tracklet association results as well as the estimation of player states for a sample 450-frame sequence.

We see that some players are successfully tracked throughout the sequence, such as the goalkeepers on the left and right of the field (player 11 and 12), who are well separated from the rest of the players. There are some players, however, that get lost, such as player 9. This could be due to several reasons. The first one is that it is hard to distinguish between a false alarm and a single detection. With a bad initialization, we have a very inaccurate estimation of the player trajectory, resulting in that good unassigned tracks are unlikely to be picked up and associated with the right players. Furthermore, many of the tracks are not "pure", i.e., tracks in which different sections correspond to different players. As a result, this ambiguity leads to wrong associations as well as inaccurate estimations of the player states.

To conclude, in this chapter we explore Markov-Chain Monte-Carlo method for tracklet association. Specifically, we develop the probability formulation for the tracklet-to-player association problem and describe in details the use of MCMCDA to sample from the posterior distribution of tracklet-to-player assignments. In addition, we incorporate the dynamics of players and utilize Kalman smoothing for the estimation of player state. Finally, we show the results of our algorithm on a sample video sequence

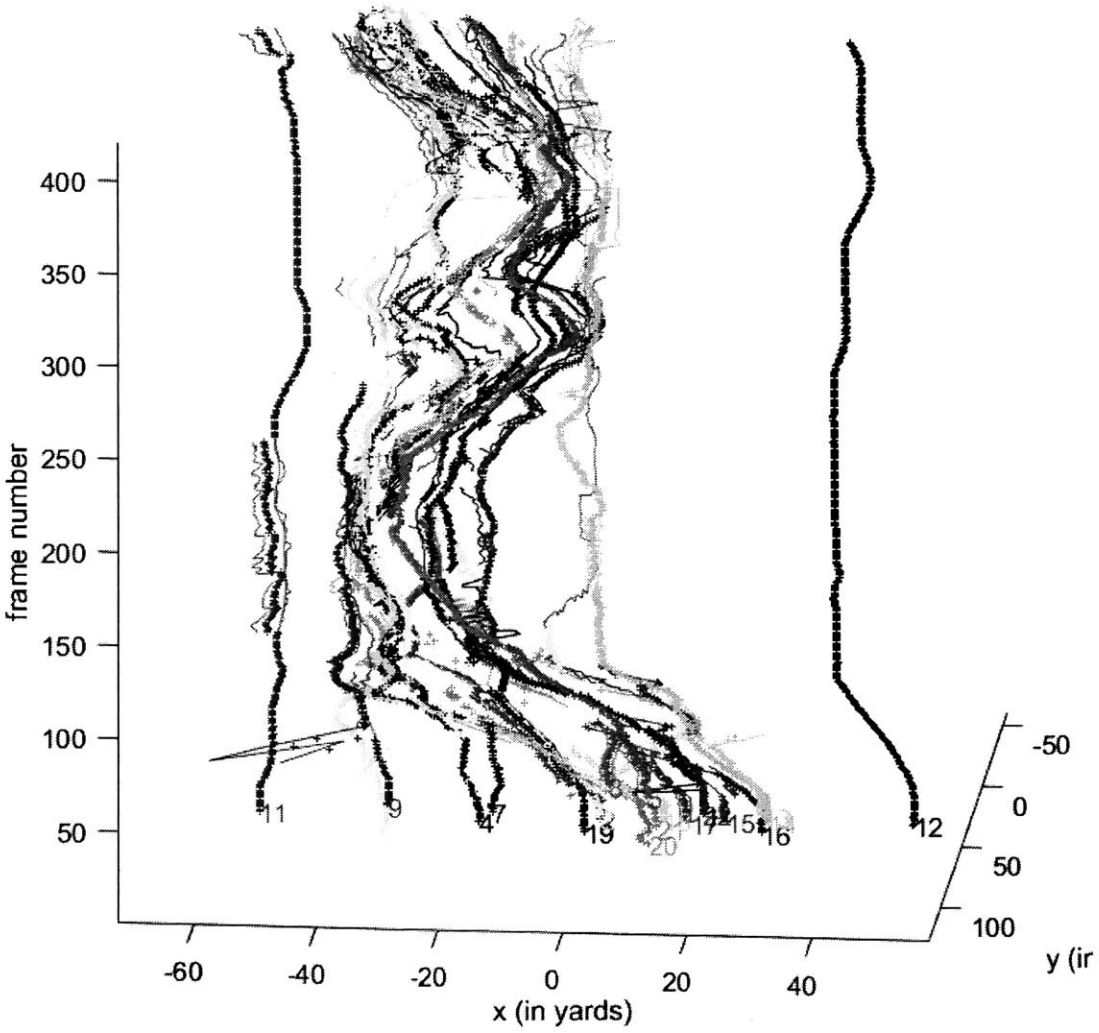


Figure 5.2: Estimated player trajectory and associated tracklet observations for all players. Each color represents one player. Estimated player positions are shown by "+", giving a thick line, while the associated tracklets are shown as thinner lines.

and discuss the limitations of the current approach.

Chapter 6

Conclusion

In this thesis we consider the problem of tracking soccer players using multiple dynamic cameras. We formulate the problem as assigning player identities to pixels and achieve it via twp-step pixel-to-layer assignment and layer-to-player assignment.

We begin our inference procedure with the estimation of camera poses for all cameras and all frames. We explore how we can use the properties of the soccer field to initialize all camera poses and how to further optimize the poses based on background appearance.

From the inferred camera parameters, we utilize the Gaussian Mixture Model to calculate the background probabilities for input frames. Then we demonstrate the use of layered trackers in individual views to produce track fragments called tracklets.

Next, we examine the problem of assigning player identities to tracklets. The inference-based approach allows us to incorporate specific probabilistic model into the tracking system. In particular, we describe the dynamics of players via state-space models and thus state estimation can be efficiently achieved by Kalman smoothing. Then we apply the Markov-Chain Monte-Carlo data association algorithm to sample from the posterior distribution of layer-to-player assignment.

Finally, we present preliminary track association results on real-world game recordings and show that we can successfully track several players throughout the sequence.

Future Works

In this work we adopt a "track-first" approach, which relies on the quality of tracklets produced by independent layered trackers. During our experiments, we find out that the layered tracker does not always produce a tracklet that is "pure", i.e. different sections of the tracklet actually correspond to different players. This typically happens when two players get together and then depart from each other. In addition, the presence of shadows leads to many layers that switch between tracking shadows and tracking

players. A method of producing pure tracklets or proposing a way to split current tracklets into ones that correspond to exactly one player identity and one player part (body or shadow) is needed. We also observe the necessity to take player cylinder model into account during the single-view tracking process to generate good observations and deal more robustly with player occlusion. Moreover, we can model and reason about the exits and entries of players based on the field of views of cameras.

The current tracklet-to-player association implementation does not take appearance-based features of players into account. In most of the occlusion cases, colors should help differentiate between players: although players from the same team wear uniforms of the same color, players from different teams wear uniforms of different colors. In addition, other appearance-based cues such as player height can help us better re-assign player identities to newly appeared tracklets and are thus worth exploring.

Another future direction is to make the number of targets part of the inference procedure and incorporate other MCMC move types such as creating new players and deleting players. It allows for more flexible scenarios in which the number of targets is unknown or is changing.

Last but not least, an interesting direction is to consider the "fuse-first" approach in which we fuse the detections across views first. During the detection fusion stage, we can use the cylinder player model to infer jointly across views the player positions. Then we can formulate data association problem on the detection to track level and use MCMC data association approach to solve it. It would be interesting to compare the current results with the outcome given by this "fuse-first" approach.

Bibliography

[1] Ben Benfold and Ian Reid. Stable multi-target tracking in real-time surveillance video. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3457–3464. IEEE, 2011.

[2] Samuel S Blackman. Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1):5–18, 2004.

[3] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459, 2014.

[4] Jason Chang and John Fisher. Topology-constrained layered tracking with latent flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 161–168, 2013.

[5] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[6] Frank Dellaert and Robert Collins. Fast image-based tracking by selective pixel integration. In *Proceedings of the ICCV Workshop on Frame-Rate Vision*, pages 1–22, 1999.

[7] Thomas E Fortmann, Yaakov Bar-Shalom, and Molly Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *Oceanic Engineering, IEEE Journal of*, 8(3):173–184, 1983.

[8] Emily B Fox, David S Choi, and Alan S Willsky. Nonparametric bayesian methods for large scale multi-target tracking. In *Signals, Systems and Computers, 2006. ACSSC'06. Fortieth Asilomar Conference on*, pages 2009–2013. IEEE, 2006.

[9] Emily B Fox, Erik B Sudderth, and Alan S Willsky. Hierarchical dirichlet processes for tracking maneuvering targets. In *Information Fusion, 2007 10th International Conference on*, pages 1–8. IEEE, 2007.

[10] Weina Ge and Robert T Collins. Multi-target data association by tracklets with unsupervised parameter estimation. In *BMVC*, volume 2, page 5. Citeseer, 2008.

[11] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[12] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by hierarchical association of detection responses. In *Computer Vision–ECCV 2008*, pages 788–801. Springer, 2008.

[13] Sachiko Iwase and Hideo Saito. Tracking soccer players based on homography among multiple views. In *VCIP*, pages 283–292, 2003.

[14] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

[15] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain monte carlo data association for multi-target tracking. *Automatic Control, IEEE Transactions on*, 54(3):481–497, 2009.

[16] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1106–1113. IEEE, 2014.

[17] Gareth O Roberts. Markov chain concepts related to sampling algorithms. *Markov chain Monte Carlo in practice*, 57, 1996.

[18] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim Christoph Rhemann Ido Leichter, Alon Vinnikov Yichen Wei, Daniel Freedman Pushmeet Kohli Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proc. CHI*, volume 8, 2015.

[19] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[20] Qian Yu, Gérard Medioni, and Isaac Cohen. Multiple target tracking using spatio-temporal markov chain monte carlo data association. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

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

[22] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.

Chapter 7

Appendix A

Here we show the derivation for the gradient of the objective function $log(\mathbb{P}(\boldsymbol{d}_l)$ in equation 3.4 with respective to R, i.e., $\frac{\partial log(\mathbb{P}(\boldsymbol{d}_l))}{\partial R}$.

For a point $(X, Y, 0)$ on the ground, we first have

$$s\begin{bmatrix} x & y & 1 \end{bmatrix}^T = K\begin{bmatrix} R & T \end{bmatrix}\begin{bmatrix} X & Y & 0 & 1 \end{bmatrix}^T.$$

Denote $H = K^{-1}$, we get

$$s\begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix},$$

i.e., $\begin{cases} s(H_{11}x + H_{12}y + H_{13}) = R_{11}X + R_{12}Y + T_1 \\ s(H_{21}x + H_{22}y + H_{23}) = R_{21}X + R_{22}Y + T_2 \\ s(H_{31}x + H_{32}y + H_{33}) = R_{31}X + R_{32}Y + T_3. \end{cases}$

Since K is an upper triangular matrix, $H_{21} = H_{31} = H_{32} = 0$. So $s = \frac{R_{31}X + R_{32}Y + T_3}{H_{33}}$.

We have

$$(R_{31}X + R_{32}Y + T_3)\underbrace{\frac{H_{11}x + H_{12}y + H_{13}}{H_{33}}}_{c_1} = R_{11}X + R_{12}Y + T_1$$

and

$$(R_{31}X + R_{32}Y + T_3)\underbrace{\frac{H_{21}x + H_{22}y + H_{23}}{H_{33}}}_{c_2} = R_{21}X + R_{22}Y + T_2.$$

We then have

$$c_1R_{31}X + c_1R_{32}Y + c_1T_3 = R_{11}X + R_{12}Y + T_1$$

and

$$c_2R_{31}X + c_2R_{32}Y + c_2T_3 = R_{21}X + R_{22}Y + T_2,$$

from which we obtain

$$\begin{aligned}
\begin{bmatrix} X \\ Y \end{bmatrix} &= \underbrace{\begin{bmatrix} c_1R_{31} - R_{11} & c_1R_{32} - R_{12} \\ c_2R_{31} - R_{21} & c_2R_{32} - R_{22} \end{bmatrix}^{-1}}_{A} \underbrace{\begin{bmatrix} T_1 - c_1T_3 \\ T_2 - c_2T_3 \end{bmatrix}}_{B} \\
&= \frac{1}{det} \begin{bmatrix} c_2R_{32} - R_{22} & R_{12} - c_1R_{32} \\ R_{21} - c_2R_{31} & c_1R_{31} - R_{11} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \\
&= \frac{1}{det} \underbrace{\begin{bmatrix} c_2R_{32}B_1 - R_{22}B_1 + R_{12}B_2 - c_1R_{32}B_2 \\ R_{21}B_1 - c_2R_{31}B_1 + c_1R_{31}B_2 - R_{11}B_2 \end{bmatrix}}_{D} \\
&= \frac{1}{det} \begin{bmatrix} D_1 \\ D_2 \end{bmatrix},
\end{aligned} \tag{7.1}$$

where $det = (c_1R_{31} - R_{11})(c_2R_{32} - R_{22}) - (c_1R_{32} - R_{12})(c_2R_{31} - R_{21})$.
Next, we take the derivative of X and Y respect to T and R:

$$\frac{\partial X}{\partial T} = \begin{bmatrix} A_{11} \\ A_{12} \\ -c_1A_{11} - c_2A_{12} \end{bmatrix}, \quad \frac{\partial Y}{\partial T} = \begin{bmatrix} A_{21} \\ A_{22} \\ -c_1A_{21} - c_2A_{22} \end{bmatrix},$$

$$\frac{\partial X}{\partial R} = \frac{1}{det^2} \begin{bmatrix} D_1(c_2R_{32} - R_{22}) & B_2 * det + D_1(R_{21} - R_{31}c_2) & 0 \\ D_1(R_{12} - c_1R_{32}) & -B_1 * det + D_1(R_{31}c_1 - R_{11}) & 0 \\ D_1(R_{22}c_1 - R_{12}c_2) & (c_2B_1 - c_1B_2) * det + D_1(R_{11}c_2 - R_{21}c_1) & 0 \end{bmatrix},$$

$$\frac{\partial Y}{\partial R} = \frac{1}{det^2} \begin{bmatrix} D_2(c_2R_{32} - R_{22}) - B_2 * det & D_2(R_{21} - R_{31}c_2) & 0 \\ B_1 * det - D_2(R_{12} - c_1R_{32}) & D_2(R_{31}c_1 - R_{11}) & 0 \\ D_2(R_{22}c_1 - R_{12}c_2) - (c_2B_1 - c_1B_2) * det & D_2(R_{11}c_2 - R_{21}c_1) & 0 \end{bmatrix}.$$

Let $d = [a, b, c]$ denote the distance between two points $(X_1, Y_1, 0)$ and $(X_2, Y_2, 0)$, where $a = \frac{X_1 - X_2}{\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2)}}$, $b = \frac{Y_1 - Y_2}{\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}}$, $c = 0$.

Then

$$\frac{\partial log(\mathbb{P}(\boldsymbol{d}_l))}{\partial R} = \frac{\partial log(e^a + e^{-a} + e^b + e^{-b})}{\partial R} \tag{7.2}$$

$$= \frac{\frac{\partial a}{\partial R}e^a - \frac{\partial a}{\partial R}e^{-a} + \frac{\partial b}{\partial R}e^b - \frac{\partial b}{\partial R}e^{-b}}{e^a + e^{-a} + e^b + e^{-b}} \tag{7.3}$$

$$= \frac{\frac{\partial a}{\partial R}(e^a - e^{-a}) + \frac{\partial b}{\partial R}(e^b - e^{-b})}{e^a + e^{-a} + e^b + e^{-b}}, \tag{7.4}$$

where

$$\frac{\partial a}{\partial R} = \frac{\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}(\frac{\partial X_1}{\partial R} - \frac{\partial X_2}{\partial R}) - (X_1 - X_2)\frac{\partial(\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2})}{\partial R}}{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$$

and

$$\frac{\partial(\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2})}{\partial R} = \frac{2(X_1 - X_2)(\frac{\partial X_1}{\partial R} - \frac{\partial X_2}{\partial R}) + 2(Y_1 - Y_2)(\frac{\partial Y_1}{\partial R} - \frac{\partial Y_2}{\partial R})}{2\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}}.$$

So finally we obtain

$$\frac{\partial a}{\partial R} = \frac{(\frac{\partial X_1}{\partial R} - \frac{\partial X_2}{\partial R})}{\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}} - \tag{7.5}$$

$$\frac{(X_1 - X_2)[(X_1 - X_2)(\frac{\partial X_1}{\partial R} - \frac{\partial X_2}{\partial R}) + (Y_1 - Y_2)(\frac{\partial Y_1}{\partial R} - \frac{\partial Y_2}{\partial R})]}{((X_1 - X_2)^2 + (Y_1 - Y_2)^2)^{\frac{3}{2}}}. \tag{7.6}$$

Similarly,

$$\frac{\partial b}{\partial R} = \frac{(\frac{\partial Y_1}{\partial R} - \frac{\partial Y_2}{\partial R})}{\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}} - \tag{7.7}$$

$$\frac{(Y_1 - Y_2)[(X_1 - X_2)(\frac{\partial X_1}{\partial R} - \frac{\partial X_2}{\partial R}) + (Y_1 - Y_2)(\frac{\partial Y_1}{\partial R} - \frac{\partial Y_2}{\partial R})]}{((X_1 - X_2)^2 + (Y_1 - Y_2)^2)^{\frac{3}{2}}}. \tag{7.8}$$