Using Line and Ellipse Features for Rectification of Broadcast Hockey Video

by

Ankur Gupta

Bachelor of Technology, The Indian Institute of Technology, Kanpur, 2005

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

Master of Science

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

December 2010

Abstract

To use hockey broadcast videos for automatic game analysis, we need to compensate for camera viewpoint and motion. This can be done by using features on the rink to estimate the homography between the observed rink and a geometric model of the rink, as specified in the appropriate rule book (top down view of the rink). However, occlusion due to players, wide range of camera motion and frames with few reliable key-points pose a significant challenge for robustness and accuracy of the solution. In this work, we describe a new method to use line and ellipse features along with key-point based matches to estimate the homography. We combine domain knowledge (i.e., rink geometry) with an appearance model of the rink to detect these features accurately. This overdetermines the homography estimation to make the system more robust. We show this approach is applicable to real world data and demonstrate the ability to track long sequences on the order of 1,000 frames.

Table of Contents

List of Figures

Acknowledgments

First and foremost, I would like to thank my co-supervisors Prof. Little and Prof. Woodham for their constant support and encouragement. Our meetings were very insightful, stimulating and great fun. Thanks to Prof. David Lowe for reading the manuscript and providing invaluable suggestions.

I would like to thank Panu Turcot, David Meger, Scott Helmer, Eric Brochu, Bo Chen and Wei-Lwun Lu for many thought provoking discussions and interesting insights. In addition, thanks to Kenji Okuma for many interesting discussions and providing a GUI for manual annotation of homography. A special thanks to Rashmi Bhadauria for generating CAD model of the hockey rink.

I would also like to thank Dr. David Pearsall and Antoine Fortier from the Department of Kinesiology and Physical Education at McGill University for providing us with high quality HD data used to test the system. Thanks to Natural Sciences and Engineering Research Council of Canada (NSERC) for supporting this research.

Finally, I thank my parents for their unconditional love and support in all my efforts.

Chapter 1

Introduction

Automated sports video analysis is an active and challenging research area in computer vision. A large part of the motivation comes from the complexity and variety of activities occurring in a multi-player field game. There are some game-specific events (i.e., a pass in soccer or a shot in a hockey game) which are crucial to understand game play along with the rules and domain knowledge. These key events can be manually annotated by a human to be searched for future reference. Automation of these event detections can lead to saving a significant amount of time in manual annotation. Knowing player locations and velocities with respect to the field is one of the most basis requirements for such applications. This information can be used to analyze [35] or even predict [33] game play.

To motivate the problem further, consider the problem of finding player trajectories (with respect to the field) over time. The problem is simpler in the case of a stationary camera because of the stationary background. As players are constantly moving background subtraction combined with the motion model for players can be used to track players in the image. Recent advances in generic object category recognition [15](specifically human recognition) have opened doors for tracking by detection for players in the case of a moving camera. However, given player detections in each frame, to obtain the trajectories of players on the field (or rink), we need to transform the field's geometric model and map it to the the image frame. The inverse transformation can be used to map activities from image coordinates to the field (or rink) coordinates (see Figure 1.2). All the images of a plane are

related to each other by a homography transformation [25]. Hence the geometric model is also related to its image (image of the rink in a video frame) by a homography. Again for a stationary camera the problem is simpler as this homography transformation is not changing over time. For a moving camera, the problem is to calculate the homography for each frame, given a geometric model and features on the rink in the frame.

Homography estimation given point matches between two images is a well studied problem, but there are no direct point matches available between the geometric model and the image frame (some point matches can be obtained by using curve intersections). Even if we obtain an appearance model of the whole rink, many parts of the rink are relatively textureless and therefore cannot generate point matches. However, there are other geometric shapes like lines and circles on the rink surface which can be utilized to overcome this limitation.

In this work, we present a novel method to combine point, ellipse and line matches to get an accurate homography estimate using linear methods. We also propose an area-based geometric error measure, which can be minimized to fine-tune our linear estimate. We combine an appearance model (key frames) with the geometric model of the rink to estimate the homography robustly over time.

1.1 Problem definition

The problem we address in this work can be described as: finding the best possible homography estimate between the geometric model and each video frame over a long sequence of images (see Figure 1.2). Here are some properties of the desired solution:

- geometric accuracy
- robustness to error in feature detection
- robustness to image artifacts like lens distortion
- robustness to motion blur due to fast camera motion

This can also be seen as finding the transformation to map the sports field image in the video frame to the world coordinates (a process known as *rectification*). We

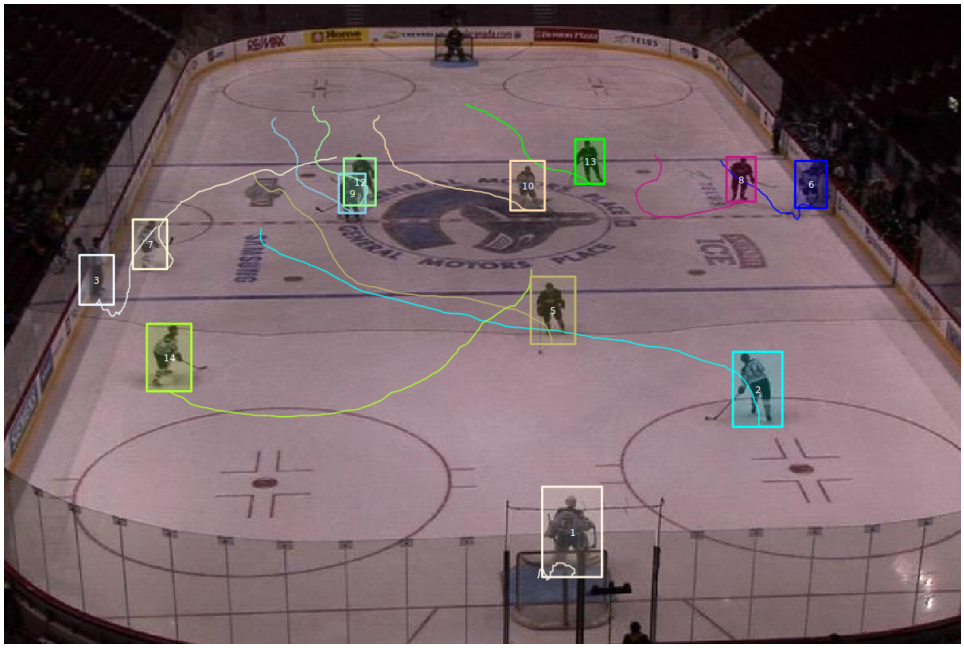


Figure 1.1: Player detections with trajectories on the rink from last few frames. This is one of the examples of data visualizations achieved by tracking player positions on the ground plane.

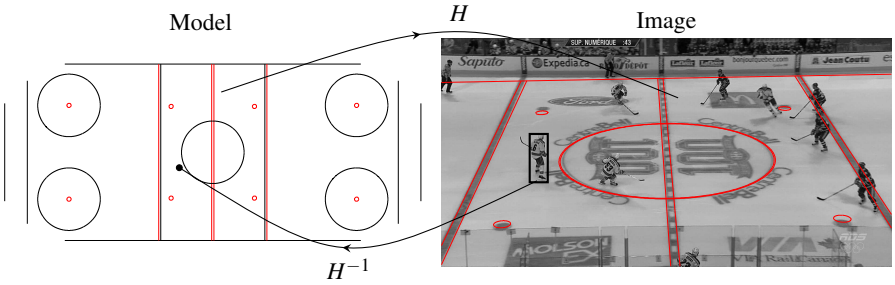


Figure 1.2: The problem definition: finding a best fitting transformation matrix H between geometric model and every frame in the sequence. The transformed geometric model superimposed on the image is shown in red.

are specifically dealing with rectification of hockey videos. We refer to the sports field as a rink in the following discussion.

1.1.1 Scope

We limit the scope of our problem and make some simplifying assumptions. We assume that there are no scene cuts in the video sequence and all the video frames correspond to an actual game (i,e. the rink is visible all the time). Other important assumptions are:

Constrained camera motion

We obtain the image sequence from a single camera, mounted on a tripod. However, it is allowed to pan, tilt and zoom. We assume that the optical center of the camera coincides with its center of rotation. This assumption is useful for our current homography estimation technique, however it can be relaxed with some minor changes in the system (see Section 5.1.2 for details). Also, there is a significant radial distortion in the hockey video sequence. In this work, we choose not to estimate the radial distortion parameters. We design our system to operate even in presence of minor distortions.

Accurate geometrical model

For all the methods described in this work, we assume that the geometric model used for the hockey rink is an accurate representation of the rink in the image sequence. However, we know that it might not be the case for every rink. Hockey rinks do not have perfectly matching dimensions.

Manual input

The system as of now is not completely automated. The key-frames used are manually annotated for the homography and the homography estimate for the first frame is also provided by the user. Another set of frames are annotated to obtain a range of parameters for the optimization (discussed in Section 5.3.3). The manual input is used to simplify certain challenges which can be dealt with later. We clearly mark it in the system overview. We try to keep it minimal and discuss possible

ways to relax this assumption in the future work section.

1.1.2 Data

All the experiments are done on McGill high definition (HD) data from an NHL game. We choose a 1000 frame sequence for our experiments. This is expected to be a typical sequence, however we still need to test the system on larger sequences and variety of games to prove its robustness of to various camera angles, zoom levels and other variables in the input data. In this sequence, we crop out the graphical score bar at the top of the frame. Apart from this, we utilize full HD resolution (1920x) for all our calculations.

1.2 Thesis outline

In Figure 1.3 we present a flow chart to summarize the complete working of the system.

Manual initialization

We are not targeting the problem of detecting and localizing the rink in an isolated image; an initial estimate of the homography needs to be provided for the first frame. Another user input is choice of key-frames and their homography estimates. These are also manually obtained by choosing points on the key frame images and the model (see Section 3.1.2 for details).

Key-frame matching

We take a key-frame based approach. The key-frames are representative images from the sequence. We choose these as to allow some feature overlap. Also, the homography between each key-frame and the model is known. Each new frame is matched with the key-frames to find consistent point matches. A key-frame with the maximum number of matches and a set of well distributed features (discussed in Section 3.2.2) is selected as the matching key-frame.

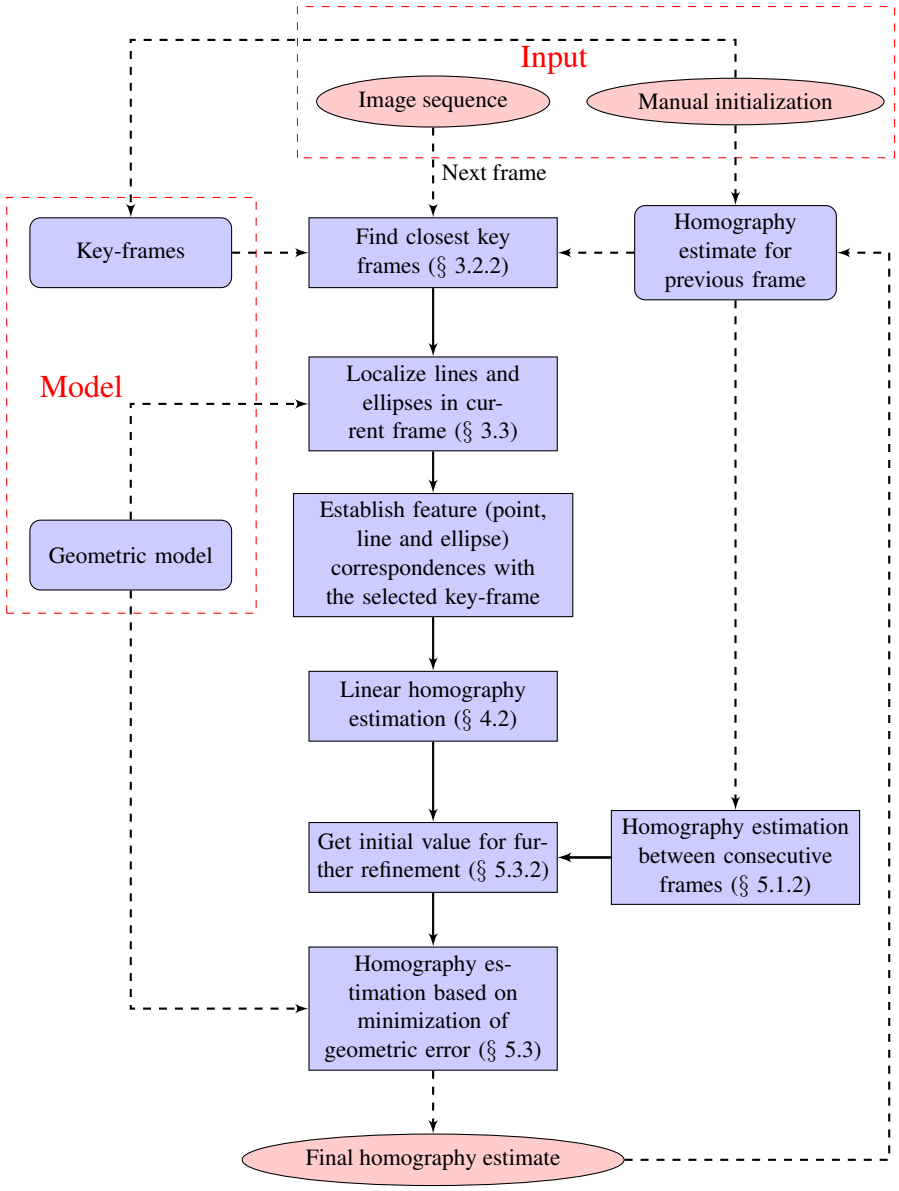


Figure 1.3: System overview. Rectangles denote important software modules. Rounded rectangles represent data. The program flow is shown with solid lines with arrows. Dashed lines with arrows reflect the data flow. The corresponding sections with the detailed discussions are shown in parentheses.

Feature localization

To establish point correspondences between the current frame and a selected key-frame we use key-point based matching. We already know the homography between key-frames and the geometric model. We can back-project these points from the matching key frame to the model to get point correspondences between the geometric model and the current frame. For locating line and ellipse features, we initialize the search for features using an estimate of the homography for the previous frame. We begin with edge and gradient information. Edges undergo model based filtering to limit false matches. Once we have filtered edges RANSAC based ellipse and line fitting is used to find the best fit for each line and circle in the model.

Linear homography estimation

Given point, line and ellipse correspondences between the matching key-frame and the current frame, we can estimate the homography between these two images. We describe a novel approach to include ellipses in the same DLT (Direct Linear Transform) algorithm used for the homography estimation using points (described in Chapter 4. However, this linear estimate is sensitive to the error in detections. We cannot always trust this as a good initialization for the non-linear refinement step.

Choosing initial value for non-linear estimation

We have one more estimate available from the homography estimate from previous frames, as we also calculate the frame to frame homography between all the consecutive frames based on key-point based matches and RANSAC. Multiplying this frame to frame homography with the last homography estimate gives us a close estimate for the homography of the current frame. We choose between this estimate (tracking) and the estimate from linear homography estimation (detection) depending upon the measure of residual error we use (defined in Section 5.2). This switching between initial estimates makes the system robust to errors in detections and drift caused due to tracking.

Residual area minimization

The last fine-tuning step aligns features as close as possible to the projected model. The aim is to calculate the homography such that the area between features in the model superimposed on images and features detected is minimized. We achieve this by minimizing a residual area-based cost function. We use a sampling based optimization technique called DIRECT to solve the optimization problem. The final homography estimate is the output of the system.

1.3 Organization

We organize this thesis as follows: first, we discuss related work in Chapter 2. Here, we summarize video rectification systems developed for hockey and other sports. The past approaches taken for solving different sub-problems are also discussed. In Chapter 3, we describe our feature detection and matching framework. We discuss how to find ellipse and line matches in the video frame corresponding to lines and circles in the geometric model. Including ellipse correspondences in the DLT (a linear homography estimation technique) framework is discussed in Chapter 4. We utilize simple geometric properties of ellipses to generate additional point and line correspondences. We then move on to describe our homography estimation technique based on minimization of geometric error in Chapter 5. In this chapter we also discuss the tracking approach combining detections in the current frame with information from previous frames to ensure robust tracking over long sequences. We analyze the system's performance in different scenarios in the results chapter (Chapter 6). Finally, the main contributions and possible future directions are described in Chapter 7.

Chapter 2

Related Work

We are looking at the problem of sports video rectification. There are similar systems developed for rectification for hockey [44], soccer [27], tennis [14], and American football [29] videos. However, these systems differ in goals and scope. They often comprise multiple modules each dealing with different functionality e.g., feature detection (and matching), homography estimation and tracking.

In this chapter, we discuss related work for each of these sub-problems. We also highlight cases where we borrow ideas from an approach or suggest an improvement over an existing method. We also justify our design choices, with these methods as basis for comparison.

2.1 Feature detection and matching

Feature matches are used to establish correspondence between model and the image (or between two images). These correspondences are used for homography estimation and tracking. Points and lines are two widely used features. Lines are useful as most sports fields have line markings, which also correspond to fixed geometry (as defined in the rule book). Some sports fields (e.g, hockey, soccer, basketball) also have circular (or partial circular) markings which transform to ellipse (or partial ellipses) in the image. Ellipses have also been used for homography estimation (see Section 2.2.1 for details) but, to best of our knowledge, not specifically applied to the problem of sports video rectification.

2.1.1 Points

Okuma et al. [44] and Hayet and Piater [27] use KLT [46] based point feature tracking. KLT tracking has an advantage of being fast. However, features tracked using KLT can only be used to match frames close in time. Hess and Fern [29] use Harris affine [42] key-points with SIFT [37] descriptors. They use these features to build a map of features registered to field coordinates. The main motivation behind using these key-points and descriptors is that they are robust to minor changes in viewpoint [29]. Also, two frames need not be adjacent in time to be matched, as long as there are some common features visible in the frames. We adopt a similar approach by using SFOP [23] key-points with SIFT descriptors (see Section 3.2 for details).

2.1.2 Lines

One of the common techniques used for detecting line segments ([3, 13]) in the images is to connect edges obtained using the Canny edge detector [7]. These connected edges are then split at points of high curvature. A line is fitted to each of these connected edge components using orthogonal regression. Orthogonal regression minimizes the perpendicular distance from data points to fitted line in contrast with linear regression (in 2D) which minimizes the sum of squared vertical distance between y data values and the corresponding y values on the line. One of the obvious limitation of this method is: we get one line fit corresponding to each connected component but there may be multiple components contributing to the same line. We may not be able to fit a line to data from multiple connected components.

The Hough transform [10] applied to the edge image is another popular method for line detection (used in [14] and [32]). Hayet and Piater [27] use RANSAC [18] for line fitting. RANSAC makes fitting robust to outliers. We combine the two by using Hough transform to generate initial line hypothesis and narrow down the search for edges. We fit a single line to all the selected edges in the narrow neighborhood. RANSAC helps us to avoid fitting a separate line to each connected edge, rather we want to fit a single line using all the corresponding edges (connected or separate) in the image.

Line matching between two views far apart is a challenging problem. A recent

work by Wang et al. [49] approaches the problem by defining features based on a cluster of line segments lying in proximity and generating matching based on this. This can be useful where images are able to produce fewer or no point matches dues to low texture. Fan et al. [13] successfully use noisy point matches for generating robust line matches. We simplify the problem of line matching by searching only in a narrow rectangular region and assigning the best fit lines (as per our score, see Section 3.3.4 for details) as a matched line. This simplification is justified because in a typical video sequence the overall movement of features in two consecutive frames is small and not drastic. Also, we do not attempt to solve the problem of initializing tracking by matching the first frame in the sequence automatically (see Section 2.3.5 for details on initialization).

2.1.3 Ellipses

The problem of ellipse detection deals with finding ellipse candidates in a noisy image where other shapes might be present. The Hough transform [10] has been one of the popular approaches for ellipse detection [53]. These methods are computationally and memory intensive. McLaughlin [41] suggested using a randomized Hough transform [52] to overcome these issues. There are multiple variants of Hough transform and other approaches (e.g., fuzzy logic, genetic algorithms) employed to solve the problem (see [38, 55] for a summary of previous work in the field). However, as the number of ellipses in the image increases the accuracy in detections goes down. Zhang and Liu [55] suggest a rule based system and show a significant improvement in speed (almost real-time) and accuracy. A recent work by Mai et al. [38] suggests ways to overcome the problem of false matches by using a set of arc segments for each ellipse candidate and then using RANSAC [18] to find a robust fit. Their method works even in the case of occluded and overlapping ellipses.

Our problem also requires finding multiple ellipses in a noisy image. However, we simplify the problem by using the matches from the last frame to guide the search. Our focus is more on finding an accurate fit given a set of noisy points (i.e., robust ellipse fitting). There are a series of geometric and algebraic methods described for ellipse fitting given 2D points by Gander et al. [24]. Geometric fits are

much slower compared to algebraic fits (10-100 times) [24]. Various constraints and weighting schemes are suggested to make algebraic solutions stable and close to the geometric fit. Fitzgibbon et al. describe such a method in their highly cited work [19]. They suggest a constraint for algebraic fitting to make the solution stable even for noisy data. This algorithm is fast and easy to implement. We choose to use this approach for ellipse fitting inside a RANSAC [18] loop (see Section 3.3.3 for details).

2.2 Homography estimation

A homography transformation can be estimated given a set of feature matches. These algorithms can be again divided in two main categories: geometric and algebraic. Geometric methods generally involve iterative minimization of an error function based on geometric properties of features. Geometric fits are often used as a refinement to an algebraic fit. In this section we summarize various homography estimation techniques and error measures.

2.2.1 Algebraic fit

The homography transformation between two images can be calculated by four point correspondences by solving a linear system of equations. In case of an overdetermined system (more than four point matches), the homography solution is approximate. There are various cost functions (or error measures) that can be used to define the "best fit". The DLT algorithm (see Section 4.1.1 for details) is based on minimization of algebraic error. Due to the linear nature of this problem, the DLT algorithm is computationally inexpensive and easy to implement but algebraic error does not correspond to any physically or statistically meaningful quantity. However, given appropriate normalization of data (see Section 4.1.2 for details) the results obtained are good [25]. Lines being the dual of points in homogeneous coordinates can be similarly used for homography estimation [54]. Dubrofsky and Woodham [9] show how to combine line and point matches in the same image to solve for homography using the DLT approach. Conic correspondences have also been used in a similar linear framework. However conics, to the best of our knowledge, have not been combined with line and point matches for

homography estimation. We describe a method to combine these heterogeneous features in a single framework in Chapter 4.

Conic based homography estimation

An early attempt at conic based homography estimation by Sugimoto [48] proves that conic coefficients transform linearly under homography. Using this approach the homography can be estimated using a linear method. However, this requires a minimum of seven conic correspondences. It can be shown that two conic correspondences are enough to solve for a homography Kumar et al. [34]. However, the work assumes that both the conics are images of circles transformed by the same homography and obtains four different possible solutions, one of which is geometrically accurate. A more generic approach is suggested by Kannala et al. [31], which provides a linear solution based on three of more conic correspondences. Conomis [8] shows that a new set of invariant points can be obtained using conics. This approach uses the pole-polar relationship to generate additional points of intersection which are also invariant under homography. These point correspondences are then used to estimate the homography using the DLT.

2.2.2 Geometric fit

Generally the homography estimate obtained by an algebraic method is used as an initial value for non-linear minimization of geometric error. Methods are dependent on the choice of error measure. We describe some of the common geometric error measures based on geometric distance between points. We also suggest a geometric error measure for homography estimation based on line and ellipse features (see Section 5.2 for details).

Geometric error

Geometric error function, unlike algebraic error, is defined in terms of measurement of geometric distances in the image [25]. Figure 2.1 summarizes some of the geometric error measures. Let us assume point matches in two images are given by $\{x_i \longleftrightarrow x_i'\}$, where x_i and x_i' are measured point coordinates. If error is only present in one of the images (let us say in second image in this case) the geometric

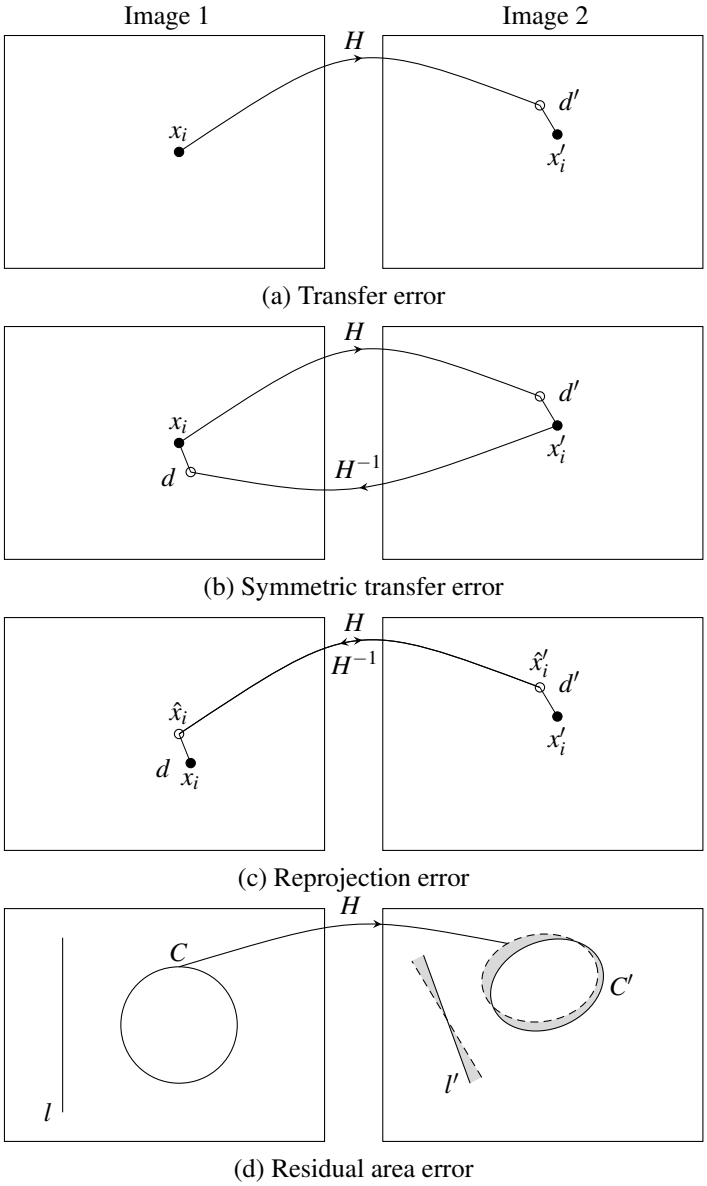


Figure 2.1: Summary of geometric error measures for homography estimation. (a) transfer error, (b) symmetric transfer error, and (c) reprojection error are defined for point correspondences. We extend this to lines and ellipses by (d) using residual area (area of the shaded region) as our error measure. Dotted lines show features from Image 1 transformed under the homography H.

error function can be defined as

$$\sum_i d(x_i', Hx_i)^2 \tag{2.1}$$

where H is the homography we need to estimate. This error is also known as *transfer error* (see Figure 2.1(a)). Now, let us consider the case when measurement in both the images can be noisy. We can sum the geometric errors corresponding to two transformations to define *symmetric transfer error* (see Figure 2.1(b))

$$\sum_i d(x_i, H^{-1}x_i')^2 + d(x_i', Hx_i)^2 \tag{2.2}$$

A third error measure is *reprojection error* (see Figure 2.1(c)) which defines a corrected location for each of the measured points and minimizes the sum of squared distances between measured and corrected points in both the images. In optimization based on this cost function, we not only need to estimate the homography H but also pairs of perfectly matched points $\{\hat{x}_i \longleftrightarrow \hat{x}_i'\}$ such that $\hat{x}_i' = H\hat{x}_i \ \forall i$. The error function is give by

$$\sum_i d(x_i, \hat{x}_i)^2 + d(x_i', \hat{x}_i')^2 \tag{2.3}$$

This cost function is hard to minimize because of the large number of parameters. For n point correspondences the total number of parameters in the optimization is $2n+9$ (9 parameters correspond to the homography estimate). There is an approximate method called Sampson approximation, initially proposed to fit conics [45], which only requires 9 parameters. For more details on Sampson error see Section 4.2.6 of Hartley and Zisserman [25].

Hartley and Zisserman analyze geometric error measures in a maximum likelihood framework under the assumption of independent Gaussian noise. When measurements in the first image are noise free, minimization of *transfer error* gives the maximum likelihood estimate for the homography. When measurements in both the images are noisy, *reprojection error* provides the maximum likelihood estimate [25].

These error measures are defined for point correspondences and are not useful in our case. We intend to estimate the homography between the geometric model

and the video frame using line and ellipse features, in addition to points. Instead of using a distance measure for points, we use an area measure between projected features from the geometric model and the features detected in a video frame (see Section 5.2 for details). Under the assumption that there is no error in the measurement of geometric model features, the area based error measure we define (as shown in Figure 2.1(d)) is analogous to the *transfer error* (Equation 2.1) for point correspondences.

2.3 Tracking

Detecting and tracking lines is one of the popular methods for homography estimation over a sequence of frames ([26, 32]). On a textureless field like a soccer pitch, lines prove to be a useful features. However, usually there are not enough lines visible in each frame to uniquely determine the homography. Methods based on line tracking are often combined with frame to frame point tracking. In the following sub-sections we summarize a few different approaches described in the literature.

2.3.1 Generating point correspondences

Okuma et al. [44] tackle the problem of rink rectification for hockey videos. Their approach is based on tracking point correspondences (using KLT [46]) to estimate the homography between consecutive frames (using RANSAC [18] for robustness). As this can lead to significant drift over time, they correct their estimate based on a geometric model. This approach works as follows: after finding a close estimate based on a frame to frame homography, a set of points are found in the image corresponding to sampled points on lines and circles in the model. These point matches are searched along the normals of lines and ellipses. This gives an additional set of points for a better homography estimation and also registers the current frame to the model, canceling drift. They initialize their system by manually specifying the point matches for the first frame. Since this work is closest to ours, we describe the key differences in the approach:

Feature detections vs. sampling points

Okuma et al. generate additional point correspondences by searching for edges along the normals for lines and ellipse models projected onto the image. These points correspond to 296 features pre-sampled in the model. We, on the other hand, detect corresponding lines and ellipses in the image. Once a line or ellipse is fit to a set of edge points the rest of the inconsistent and noisy edges can be discarded and do not interfere with the error measure. We utilize the structure inherent in geometric shapes present in the rink image.

Combining lines and ellipses in homography estimation

Finding point correspondences along the normals can only approximate the actual point correspondences. We do not sample points on our lines or ellipses to find additional point correspondences but use parameters of line and ellipse detections to estimate the homography.

Using geometric error measure

In [44] final drift correction is based on the DLT (a linear estimate based on algebraic minimization of error). We use a new measure to combine different geometric errors in line, point and ellipse matches (see Section 5.2 for details).

2.3.2 Using lines

The idea of using line features (boundary lines) to avoid drift while tracking planar surfaces is explored by Xu et al. [51]. They consider planar surfaces with clearly defined boundaries and track them robustly over time. They show that line features make tracking more accurate. However, when they do correction based on lines the point match information is discarded. We overcome this limitation by combining points in the same framework (see Chapter 4 for details).

Hayet et al.[26] use direct line to line matching between the model and the image to estimate the homography. However this approach works only for cases when there are enough line correspondences (four or more) available in the image. For other cases the image to image homography is calculated based on point and line correspondences. However, they do not clearly describe how they combine

these two. They also simplify the algebraic fitting of the homography under the assumption of a camera with fixed center (it can only only pan, tilt or zoom). This reduces the minimum number of points required to estimate the homography to 3 (compared to 4).

Farin et al. [14] use lines to calculate real and virtual points of intersection. These points are used to establish the homography between image and the model. They also define a geometric error measure which they minimize for estimating the homography based on lines. They project the white pixels (court lines in the case of tennis) onto the model. The error measure is defined as the sum of the geometric distance between model lines and these projected points.

2.3.3 Key-frames

Our use of key-frames (described in Section 3.1.2) is inspired by the work of Hess and Fern [29]. They also use a set of manually annotated frames as reference images (or key-frames). They use these images to assemble a set of model features registered to the field. They argue that many of these features are not distinctive at a global level (i.e., if we try to match each new frame with the whole model feature set). However, these can be distinctive if matched with the features spatially located in a smaller neighborhood in the model. We use this idea by defining a set of key-frames covering only a smaller portion of the complete model. For every new image we match only one key-frame at a time. Also, for every new frame, we use the last matched key-frame to choose the key-frames (in the neighborhood) we should match it with. We do not explicitly construct a set of model features but our method takes advantage of the local distinctiveness.

2.3.4 Modeling motion

Kim and Sang Hong [32] use camera calibration as the first step for homography estimation for a pan-tilt and zoom camera. After initial line matching between image and model they estimate the camera parameters by an exhaustive search over a wide range of discrete parameters. Once the camera intrinsics are known tracking lines in subsequent frames is assisted by camera parameters (CP based tracking). This system also tracks camera zoom and rotation for every new frame.

Hayet and Piater [27] describe a method to estimate the covariance of the estimated homographies to measure uncertainty. They switch modes between detection and tracking when uncertainty becomes too large. We do not specifically build a motion model for the homography but, similar to [14], we also use the previously computed homography to limit the search space for features in the current frame. In this work, we also do not deal with the problem of camera calibration as the homography can be estimated without knowing camera parameters.

2.3.5 System initialization

Hayet et al. [28] propose a system for field-to-image registration based on geometric reasoning. They start with noisy line matches to generate multiple hypotheses for matches with the field (or geometric model). Farin et al. [14] also prune and search possibilities for line matches to fit a model to images from tennis videos for automatic initialization. Okuma et al. [44] use hand-picked points in the image and the model to initialize the first frame. Since we use key-frames with a known homography, for initialization we match the first frame with all the key-frames to get an initial estimate of the homography. Handpicking and annotating key-frames is part of our initialization process.

Chapter 3

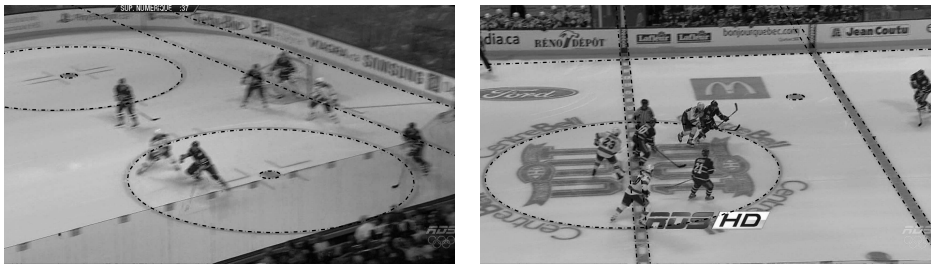
Feature Detection

Along with local point features, we use lines and ellipses as additional features for tracking. Lines and ellipses in the video images are the projection of lines and circles (in the model/world) respectively. There are multiple reasons we focus on detecting actual lines and ellipses on the rink rather than just relying on key-point based matches. First, these features can be localized accurately under motion blur (as shown in Figure 3.1(a)) and are robust to minor occlusions (as shown in Figure 3.1(b)) due to players. Second, these features correspond to markings on the rink which are important for game play (i.e., a player crossing a certain line has a meaning in the game). On the other hand key-point detections have no physical significance from a rules perspective.

In this chapter, we discuss the rink model used for detection and tracking. We also summarize the methods for establishing key-point based matches between a video frame and a set of key-frames (defined in Section 3.1.2). We choose the best matching key-frame as the closest key-frame. Then, we move onto localizing line and ellipse features given a close initialization (i.e., approximate parameter estimates from previous frames).

3.1 Modeling sports rink

A geometric model (CAD model) is extremely useful as it gives us accurate geometry. However, it does not have any appearance information (i.e., logos, advertise-



(a) (b)

Figure 3.1: Figure shows actual line and ellipse detection results returned by the detection module under motion blur and occlusion. A fast panning camera causes (a) motion blur in the image. It is hard to localize specific edges (e.g., some characters on the boards are unreadable). However, lines and ellipses are localized well in the image. Also, (b) minor occlusion due to players (in this image there is additional occlusion due to the broadcaster's logo), lines and ellipse detections remain stable and accurate.

ments and writings on the rink on that particular day). Appearance information is useful to obtain point correspondences between model and image (using key-point detection and matching). To augment the geometric model with appearance information, we select a few frames from the sequence and use them as key-frames. This has two advantages: first, we need not build an appearance model of the whole rink, we just use a set of images. Second, we need to provide a ground truth homography only for these frames. Additionally, these key-frames have a point of view specific to the camera which makes point matches more accurate and repeatable. A synthesized top down view will not have the same property. Manually choosing point correspondences to calculate the homography between the key-frames and the model is part of the initialization procedure for the system which needs to be done only once for each camera in the game.

3.1.1 Geometric model

The geometric model of the rink is defined by the rule book for the game. These models slightly differ for different sports authorities (e.g., International Ice Hockey Federation (IIHF), National Collegiate Athletic Association (NCAA), National

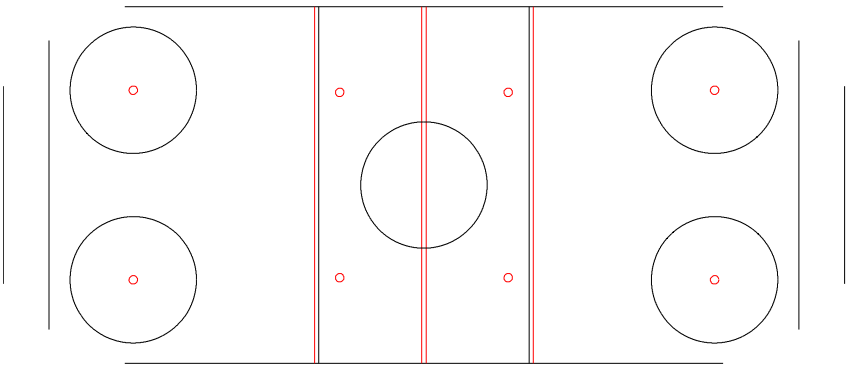
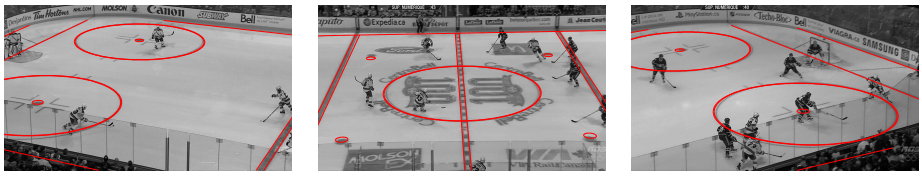


Figure 3.2: The simplified geometric model of hockey rink. This model also contains the normal directions for lines and circles in the images. Black circles (represent inner boundary of the circular shape) have normals pointing towards the circle center. Red circles are filled which makes their normals point away from the center. Similarly, red lines represents lines with normal pointing away from from the model center (i.e., center of the central circle). Black lines have opposite normal direction.

Hockey League (NHL)). We obtained our rink model from the official website of the NHL [43], as our dataset is taken from an NHL game. Our model would only need minor metric modifications to make it compatible with other rule book rink specifications. The model gives us information about absolute sizes and positions of lines and circles on the rink. We have simplified the model considerably (i.e., not all lines are used), as shown in Figure 3.2. Please note that we leave out corners in the model as we do not intend to track the projected curves (partial ellipses) corresponding to partial circles at the corners. Models also keep track of the normal direction for each line and circle. These normals correspond to gradient directions at the corresponding feature in the image (we use it as described in Section 3.3.2). This geometric model serves as the backbone of the complete tracking and detection cycle. It is also used for initializing the system.

3.1.2 Key-frames

The key-frames are a set of images with overlapping features to cover the whole range of camera motion. For our dataset, we choose five frames as the key-frames



Key-frame 1 Key-frame 3 Key-frame 5

Figure 3.3: The key-frames used in appearance model of the rink. Figure shows three key frames with the transformed geometric model superimposed. The homography between these frames and the geometric model is obtained by manually selecting point correspondences.

to cover the complete rink. The choice of number of key-frames is mainly guided to cover the whole rink with frames having significant overlap (around 20-30 % of the area). Three of these key-frames are show in Figure 3.3. During tracking, we calculate the closest key-frame for every new frame (as described in Section 3.2.2).

3.2 Key-point based feature detection

Once we have chosen our model, we need to detect corresponding features between the model and images. Key-point based detection relies on two main steps: first, detection of salient points (key-points) in the image; second, calculating a feature vector (descriptor) corresponding to each key-point (e.g., SIFT [37], SURF [4] descriptors), which is used for matching it to key-points in another image. There are various key-point detection algorithms suggested in the recent past (e.g., MSER [40], Harris Affine[42], SIFT [37])). These methods use different definitions for the "saliency" of a point.

3.2.1 SFOP key-point detection

SFOP [22], suggested by Förstner et al., is a key-point detection algorithm designed around *interpretability* of features (i.e., feature points detected correspond to a physical shape). In their paper Förstner et al. show that their method is more effective (generates more matches) for indoor, man-made and low-texture environments and performs at par with other state-of-the-art detectors on other scenes.

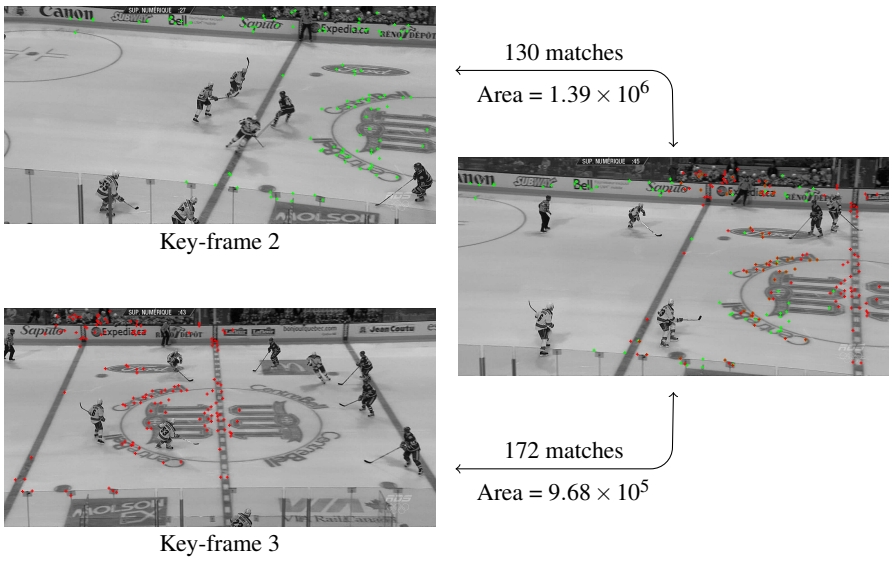


Figure 3.4: Choosing between two competing key-frames for a given input frame. Consistent matches with key-frame2 and key-frame3 are represented by green and red markers respectively. We also show the number of matches with each key-frame. Area value denotes the area of the smallest rectangle containing all the matches in the input frame. Following Algorithm 3.1, we prefer key-frame2 over key-frame3 as the closest key frame.

This property makes it a good choice for us as a large part of a hockey rink contains geometric shapes and no significant texture. We choose to use SFOP key-point detection along with SIFT descriptor. For a given pair of images we first find SFOP key-points in both the images and then calculate SIFT descriptors for each of these key-points. To match these key-points, we use the demo software provided here [36], with the default threshold for matching (a distance ratio of 0.6). For SFOP, we use the Matlab implementation from the authors [21]. To extract the SIFT descriptor, we use the code available here [1].

Algorithm 3.1: findClosestKeyFrame

Input: H_{est}, current frame f, a set of key-frames K, dTh, W
Output: $keyInd$, $ptMatches$
$maxArea \longleftarrow 0$
$maxMatches \longleftarrow 0$
$keyInd \longleftarrow (-1)$
foreach $k_i \in K$ **do**
  $M \longleftarrow \phi$
  $[X_k, X_f] \longleftarrow$ Get all point matches between k_i and f
  foreach $x_{fj} \in X_f$ **do**
    if $dist(x_{fj}, H_{est} \times x_{kj}) \leq dTh$ **then**
      $M \longleftarrow \{M, j\}$
    end
  end
  $a_i \longleftarrow$ Area of the smallest rectangle enclosing points $X_k\{M\}$
  $n_i \longleftarrow$ Size of the set M
  if $a_i > maxArea$ **and** $n_i > maxMatches/W$ **then**
    $keyInd \longleftarrow i$
    $maxArea \longleftarrow a_i$
    $maxMatches \longleftarrow n_i$
    $ptMatches \longleftarrow [X_k, X_f]$
  end
end

3.2.2 Finding closest key-frame

For every new frame in the video we need to find the closest key-frame. First, we perform SFOP-SIFT (as described in the last section) matching between the current frame and a set of key-frames. We choose this set to be the last key-frame and the two nearest neighbor key-frames. However, when initializing the system, we match the first frame with all the key-frames.

Consistent matches

Not all the matches we obtain from key-point matching are correct. To eliminate the spurious matches, we use the homography estimate from the last frame. We project the matched key-points from the key-frame to the current frame. All the points lying within a loose threshold of corresponding points in the frame are re-

tained as consistent matches.

Defining closest

We keep track of the number of consistent point matches for all the candidate key-frames. As key-frames have visible overlap, it may not be appropriate to choose the closest key-frame just based on the maximum number of consistent matches. Another important factor is the spatial distribution of points in the frame. Well distributed points are crucial for an accurate homography estimate [5]. The area covered by these consistent matches along with the number of matches proves to be a simple but robust criteria for choosing closest key-frame. Figure 3.4 illustrates this point, where we prefer a key-frame with more spatially distributed key-point matches over a key frame with a larger number of key-points matches.

Algorithm 3.1 summarizes the routine we use to find the closest matching key-frame and corresponding point matches. It requires an approximate homography estimate between each key-frame and the current frame H_{est} (we obtain it from the last frame homography estimate). There are two more parameters in this algorithm: dTh is the distance threshold used to filter out inconsistent matches and W is the weight we assign to area covered by matches with respect to number of matches. In our experiments, we choose $dTh = 0.05 \times imageDimension$ and $W = 3$.

3.2.3 Correspondence with geometric model

We have now obtained the point correspondences between the current video frame and its closest key-frame. This is still an image to image correspondence. We would like to find out the point correspondence between the geometric model and the current frame. We already have the homography between key-frames and the model. We can back-project these matching points onto the model to get the correspondence. We use it later for the homography estimation (described in Chapter 5).

3.3 Line and ellipse detection

Under perspective projection lines transform to lines [25] and circles transform to ellipses [38]. We use this as the basis of our feature detection process. We also assume that we have a close initialization available from the previous frames. We

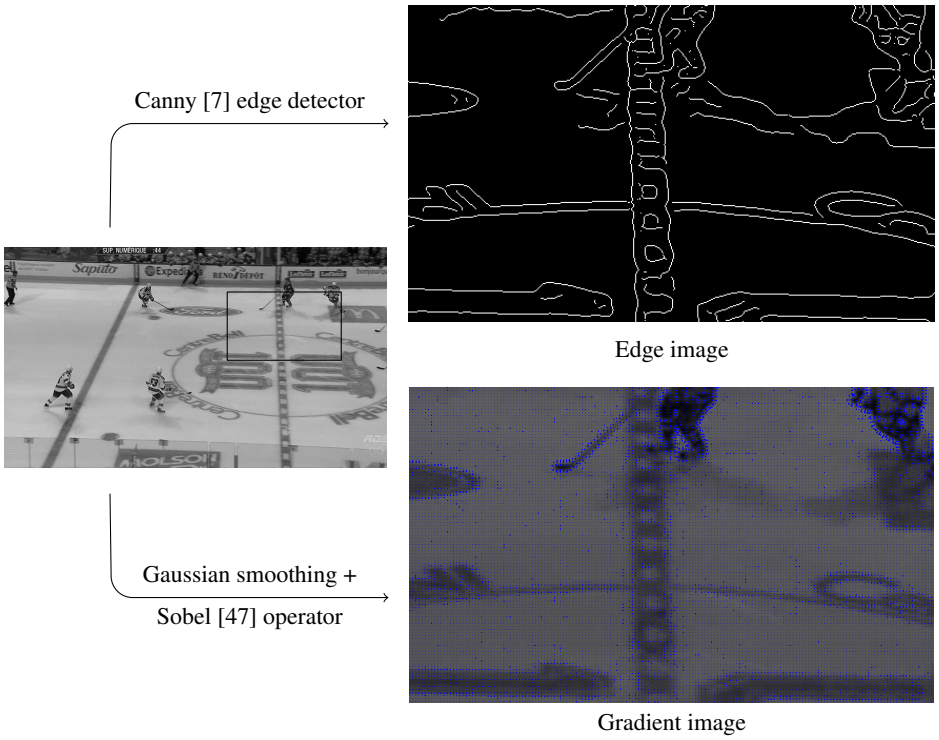


Figure 3.5: Preprocessing for each new frame involves finding edges and calculating gradients. We show results only for a selected area (the black rectangle) in the image. Gradients are depicted by blue arrows pointing along the gradient direction. The length of each arrow is proportional to the gradient magnitude. We have plotted the gradient for every third pixel to avoid clutter.

use edge detection as the first step in our detection module. However, it also returns edges corresponding to the foreground objects and other neighboring features. The close initialization available from previous frames helps us to filter edges for a particular feature based on the image gradient and model parameters. We use these filtered edges for robust line and ellipse fitting.

3.3.1 Preprocessing

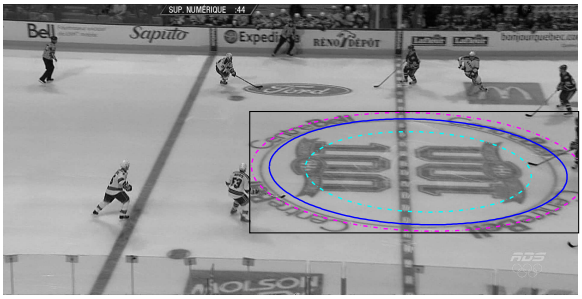
We combine edges with gradient information to use as building blocks for detecting and matching features. First, we calculate an edge image for each frame using Canny edge detection [7]. For gradient calculation, we smooth the image using the Gaussian filter to suppress noise. We then convolve it with (3×3) Sobel masks [47] to obtain the gradient at each pixel. Please note that gradient calculation is also an intermediary step in Canny edge detection algorithm, but since we are using the Matlab implementation for edge detection, we do not have access to the intermediate results. We redo the calculation to obtain gradients. Figure 3.5 summarizes the preprocessing step.

3.3.2 Model based edge filtering

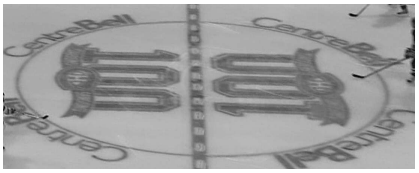
We assume that we have approximate feature locations to begin the detection process. First, we only consider edges in the vicinity of the initial estimate. However not all edges in the vicinity belong to the the feature we are looking for. To filter out these noisy edges further, we use additional information (i.e., gradients, continuity of edges).

Ellipse

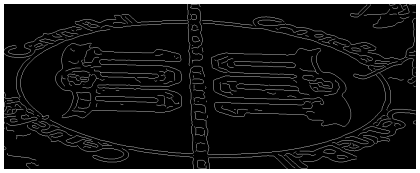
For an ellipse, we choose an annular region around the expected ellipse boundary (see Figure 3.6(a)). We only consider edges in this region. The rink model also keeps track of the normal for each ellipse to help us choose between the outer and inner edges of a thick line in the image. This also helps to filter out edges which are inconsistent with the expected gradient direction (i.e., edges due to foreground objects or neighboring features). To achieve this, we divide the angular space for the ellipse boundary into small regions. We get a unit vector along the expected normal direction for each of these regions. We similarly calculate the unit vector for edges lying in this region and calculate the dot product with the expected unit vector. We choose only the edges for which the dot product is higher than a certain threshold th (We choose $th = 0.85$ in our experiments). Now, we have edges in the vicinity of the expected ellipse boundary which are consistent with the normal at the model ellipse boundary (see Figure 3.6(e)).



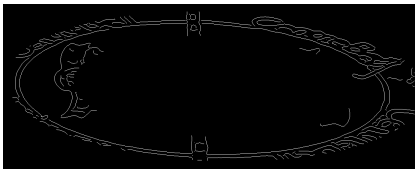
(a) Frame with initial estimate of ellipse



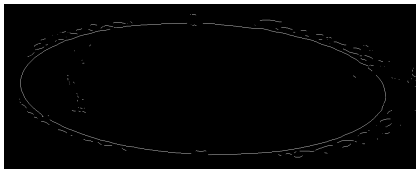
(b) Selected region



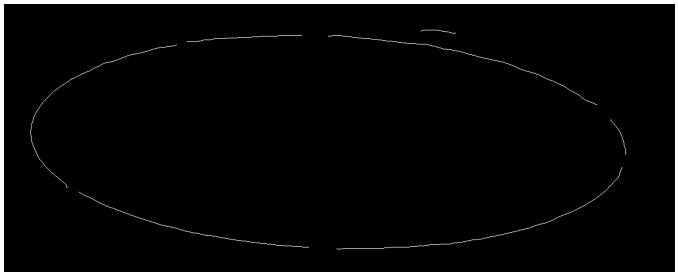
(c) All edges in the selected region



(d) Edges in annular region



(e) Filtering based on gradient information



(f) Final clean up based on size of the connected edge pixels

Figure 3.6: Edge filtering based on initial estimate and model. In (a) the initial ellipse estimate is shown in solid blue. The dashed ellipses around it mark the area we consider for searching for an ellipse. Only (d) edges in this annular region are considered. These edges are then (e) filtered based on gradient information. Only edges with gradient direction consistent with the geometric model remain. Please note that the outer edge of the thick ellipse line in the image is filtered out in (e). Finally, the small edges are cleaned up to give (f) the edge points used for ellipse fitting.

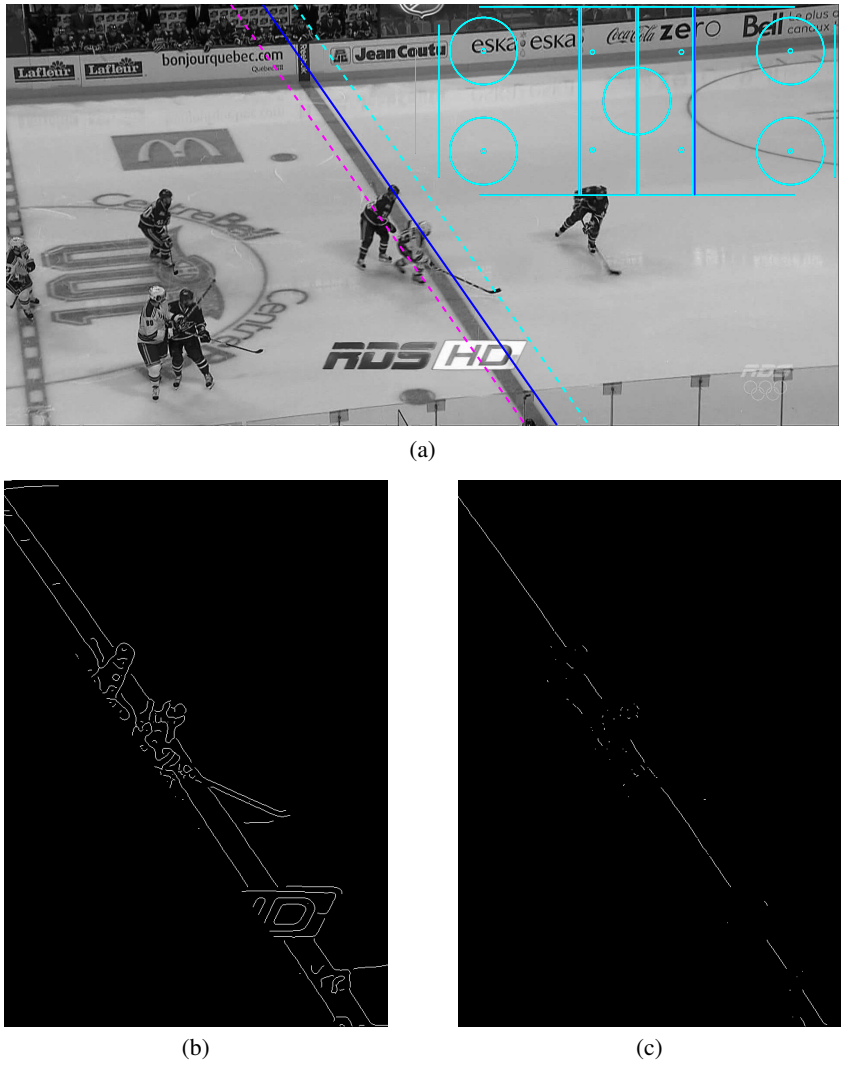
Continuity is another important property of the ellipses in the image. As the final step to filter out smaller edges, we calculate connected components for edges and filter out edges smaller than certain length threshold lth. This gives us cleaned up edges, even for ellipses surrounded with texture or occlusion due to moving foreground objects. Figure 3.6 outlines the steps involved in obtaining the final cleaned up edges in the image corresponding to a particular ellipse in the geometric model.

Line

A similar approach is taken to filter edges corresponding to lines. For each approximate estimate of the model line projected onto the image we choose a rectangular region around it (see Figure 3.7(a)). We only consider edges lying in the region. Similar to the ellipse case these edges can also be filtered on the basis of gradient directions. This approach is useful to distinguish between two lines close to each other, having a different gradient direction (see Figure 3.7(c)). Figure 3.7 summarizes the steps involved in filtering edges based on the line model and an initial estimate. We obtain a set of cleaned up edges which are used as input for line detection.

3.3.3 Ellipse and line fitting

After performing edge filtering, we fit a line or an ellipse to these set of edge points. In case of an ellipse, we require a minimum of five points to fit. Typically the number of edge points we have for ellipse fitting is much larger than that. Since there are still some noisy edge points, we also use RANSAC to remove outliers. An algebraic fit suggested by Fitzgibbon et al. [19] is used to fit the model in RANSAC. This work describes an ellipse specific fit (not a general conic) which is fast and robust. For a brief review of various ellipse fitting approaches, please see Section 2.1.3. Line fitting, because of fewer parameters, is relatively less sensitive to noise and outliers. However, there could be multiple lines in the neighborhood we are searching. We first use a Hough transform to generate hypotheses for possible lines. We also fit lines using RANSAC for each of these hypotheses.



(a)

(b)

(c)

Figure 3.7: Edge filtering based on the initial estimate and model for line detection. (a) Solid blue line shows the initial estimate for line detection. The corresponding line is highlighted in the geometric model shown in top-right corner. Dashed lines around the solid blue line mark the search region. (b) All the edges in the region. (c) Edges filtered based on direction of the gradient at each edge point. Please note that the edges corresponding to a parallel line in the neighborhood are filtered out because they have opposite gradient direction.

Ellipse

Cai et al. [6] suggested using RANSAC for ellipse fitting to give a robust fit with noisy data. Algorithm 3.2 describes the routine we use to fit an ellipse to noisy data. X denotes a set of candidate points in 2D. Other parameters of this algorithm are M and t. M is the maximum number of random samples allowed. This limit is set for the algorithm to finish within a reasonable time. t is the threshold to choose the inliers for a model generated using a set of random points. We use geometric distance between the model and the points, the points lying within the distance t are chosen as inliers. Sub-function $FitEllipse$ [20] is the authors' own implementation of the direct least square algebraic fit for an ellipse [19]. The number of trials required to choose a set with probability p which contains no outliers (based on current *bestScore*) is calculated by the function $RequiredTrials$. $Length$ returns length of the array. For our experiment we use $N = 2000$, t=0.01.

Line

For line fitting we consider multiple hypotheses for a selected region. After edge filtering, we perform a Hough transform [10] on the filtered image to obtain multiple line hypotheses. We narrow down the search corresponding to these line segments returned by the Hough transform. We run RANSAC based line detection on edges in a narrow rectangular region around these hypotheses. Only one hypothesis is used for each region to avoid duplicates. We finally choose the best hypothesis based on a verification score (i.e., a line with strongest edge support, described in Section 3.3.4).

Algorithm 3.3 summarizes the procedure to generate line candidates. Function $GetHoughLines$, returns the lines corresponding to local maxima in the Hough transform of the edge image. Function $FitRansacLine$ fits the line to 2D points using RANSAC. $DIST\_TH$ is the threshold applied to choose edges in the neighborhood of the approximate line hypothesis obtained by the Hough transform. t is the threshold used to filter outliers in RANSAC. The RANSAC part of this algorithm is quite similar to the procedure described in Algorithm 3.2 (think of it as choosing only 2 random points at a time and replacing function $FitEllipse$ with $FitLine$).

Algorithm 3.2: *FitRobustEllipse*

Input: A set of points X, MaxTrials M, Threshold t
Output: Ellipse parameters C

$N \longleftarrow 0$
$p \longleftarrow 0.99$
$numTrials \longleftarrow 0$
$bestScore \longleftarrow 0$
$bestinliers \longleftarrow \phi$
$bestM \longleftarrow \phi$
while $numTrials \geq N$ **do**
    $ptToFit \longleftarrow$ randomly choose 5 points from X
    $M \longleftarrow FitEllipse(ptToFit)$
    $inliers \longleftarrow \{x_i \in X : Dist(M,\ x_i) < t\}$
    $ninliers \longleftarrow Length(inliers)$
    if $ninliers > bestscore$ **then**
        $bestScore \longleftarrow ninliers$
        $bestInliers \longleftarrow inliers$
        $bestM \longleftarrow M$
        $N \longleftarrow RequiredTrials(bestScore, p)$
    end
    $numTrials \longleftarrow numTrials + 1$
    if $numTrials > M$ **then**
        break
    end
end
$C \longleftarrow FitEllipse(bestInliers)$

Algorithm 3.3: *FitRobustLine*

Input: A set of points X, $DIST\_TH$, Threshold t
Output: Line hypothesis L

$H \longleftarrow GetHoughLines(X)$
foreach $h_i \in H$ **do**
    $x_f \longleftarrow \{x_i \in X : Dist(h_i, x_i) < DIST\_TH\}$
    $L_i \longleftarrow FitRansacLine(x_f, t)$
    $L \longleftarrow \{L\ \ L_i\}$
end

3.3.4 Verification

Even after using RANSAC to filter outliers, the fit may not correspond to the actual edges used to estimate an ellipse, especially when only part of the ellipse is visible in the image. We develop a measure of "goodness" for the estimate we have obtained. We need to discard invalid or improbable matches for both lines and ellipses.

Ellipse

One of the basic checks for ellipse verification is length of major, minor axis and orientation of the ellipse. We compare these quantities to the expected dimensions of the transformed geometric model. Please note that the orientation measure becomes unstable for ellipse that are near circular. Orientation should not be used when the ratio of the length of the minor and major axis of the model is close to one. This is the first stage of filtering. Then, we calculate a measure of fitness based on the alignment of normals at the fitted ellipse boundary to the gradient in the image. See Algorithm 3.4 for details. C is the fitted ellipse and M is the corresponding ellipse model projected onto the image (based on the same homography estimate which was used to initialize ellipse fitting). Function $GetEllipseParameters$ returns the center ($\mathbf{z}$), length of major and minor axis (a, b) and the rotation angle (α), given the algebraic equation of the ellipse. In our experiment, we choose $lTh = 0.15$ and $\alpha Th = \pi/9$.

To generate a quantitative verification score, we again divide the ellipse in angular spaces from 0 to 2π with an interval of $\Delta\theta = 0.02$. $GetModelNormal$ returns the expected normal at a particular angle for model ellipse M. Function $GetClosestEdgeGradient$ gets the image gradient at the edge point closest to the fitted ellipse boundary in the image. This also returns the edge point's distance d from the ellipse boundary. Taking the dot product of the two normals gives us measure of alignment of image gradient with normals at the boundary of the fitted ellipse. Edge points far from the fit ellipse boundary should be penalized. Hence, the score is inversely proportional to the square of the distance. We sum this score for the whole angular range of the visible ellipse in the image. We reject ellipse detection below a certain threshold, to make sure only accurate ellipse fits are used

for homography estimation.

Algorithm 3.4: *GetEllipseVerificationScore*

Input: Ellipse C, model M, lTh, aTh
Output: Score s
$s \longleftarrow 0$
$\Delta\theta \longleftarrow 0.02$
$[\mathbf{z}\ a\ \ b\ \ \alpha] \longleftarrow GetEllipseParameters(C)$
$[\mathbf{z}'\ a'\ \ b'\ \ \alpha'] \longleftarrow GetEllipseParameters(M)$
$aRatio \longleftarrow \|a - a'\|/a'$
$bRatio \longleftarrow \|b - b\|/b'$
if *(aRatio* $> lTh$*) OR (bRatio* $> lTh$*) OR (*$\|\alpha - \alpha'\| > aTh$*)* **then**
  return
end
foreach $\theta \in \{0,\ 0 + \Delta\theta ... 360\}$ **do**
  $\mathbf{n_m} \longleftarrow GetModelNormal(\theta)$
  $[\mathbf{n_e}\ d] \longleftarrow GetClosestEdgeGradient(\theta)$
  $s \longleftarrow s + \mathbf{n_m}.\mathbf{n_e}/d^2$
end

Line

Line verification is necessary to choose the best line among a set of candidates. Also, we reject lines which do not have enough edge support in the frame. We summarize the procedure for line verification in Algorithm 3.5. M is the model line projected onto the image (based on the same homography estimate which was used to initialize line fitting). n_m denotes the normal to the model line. L is a set of candidate matches obtained using line fitting (described in Section 3.3.3). One of the simplest tests for discarding false detections is to check the orientation of the fitted line. We accept a line fit only if its normal aligns with the expected model normal within a given threshold θ_{th} (we choose $\theta_{th} = \pi/13$). Lines not meeting the criterion are not considered for further score evaluation. To calculate a score, we sample points on the line and get the normal for the closest edge point to it, similar to the ellipse case (described in the last section), using function *GetClosestEdgeGradient*. *GetEndPoints* returns the end points of the fitted line. Note that we fit a line with no defined end points. End points returned by this

function are corresponding points to the expected ends of the projected model line segment visible in the image.

Once we have the score corresponding to each line candidate, the one with the highest score is chosen as the final line match.

Algorithm 3.5: *GetLineVerificationScore*

Input: Lines L, model M, θ_{th}
Output: Score $\mathbf{s}$
$\mathbf{s} \longleftarrow [0, 0, ...0]$
$\mathbf{n_m} \longleftarrow GetLineNormal(M)$
$l_m \longleftarrow GetLineSegmentLength(M)$
$\Delta t \longleftarrow 1/l_m$
foreach $l_i \in L$ **do**
  $\mathbf{n} \longleftarrow GetLineNormal(l_i)$
  if $\mathbf{n_m}.\mathbf{n} < \cos\theta_{th}$ **then**
    continue
  end
  $[\mathbf{x_b}\ \mathbf{x_e}] \longleftarrow GetEndPoints(M, l_i)$
  foreach $t \in \{0,\ 0+\Delta t,\ ...1\}$ **do**
    $\mathbf{x_t} \longleftarrow \mathbf{x_b} + t(\mathbf{x_e} - \mathbf{x_b})$
    $[\mathbf{n_e},\ d] \longleftarrow GetClosestEdgeGradient(x_t)$
    $\mathbf{s}[i] \longleftarrow \mathbf{s}[i] + \mathbf{n_e}.\mathbf{n_m}/d^2$
  end
end

Chapter 4

Using Ellipse Correspondences for Homography Estimation

Four or more point correspondences between two images provide sufficient constraints to estimate the homography between two images (or planes), which can be computed using the Direct Linear Transformation algorithm (DLT) [25]. In the case of noisy data, appropriate normalization is a crucial step in the DLT. Line correspondences can be used in a similar way by the duality principle. Dubrofsky and Woodham [9] show that point correspondences can be combined with line correspondences to obtain a better estimate. They describe a method to transform lines with the same normalizing transformation as points which allows them to treat lines and points in a similar way in the same linear framework. Ellipse features occur in many images due to circular symmetry in various real world objects. These features also have the advantage of being distinctive and robust [8, 48].

In this chapter, we describe a method to incorporate ellipse feature correspondences between two views along with line and point correspondences to estimate the homography. Using some basic properties of an ellipse, we generate additional point and line correspondences. We show that this simple strategy can combine point, line and ellipse correspondences in a single flexible framework.

4.1 Line based homography estimation

Lines can be treated similarly to points in the original DLT algorithm.

4.1.1 The DLT algorithm

Assume that $\mathbf{p}_i = \begin{bmatrix} x_i & y_i & w_i \end{bmatrix}^T$ and $\mathbf{p}'_i = \begin{bmatrix} x'_i & y'_i & w'_i \end{bmatrix}^T$ are two corresponding points related by a homography written in homogeneous coordinates. The homography matrix, H, by definition relates these two points as

$$\mathbf{p}'_i = H\mathbf{p}_i \quad \forall i \in \{1...n_p\} \tag{4.1}$$

where n_p is the number of point correspondences and H is a 3x3 matrix given by

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{4.2}$$

Equation 4.1 can be rewritten in the form

$$A_i\mathbf{h} = 0 \tag{4.3}$$

where $\mathbf{h} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix}^T$ and A_i is a 2×9 matrix given by

$$A_i = \begin{bmatrix} \mathbf{0}^T & -w'_i\mathbf{p}_i & y'_i\mathbf{p}_i \\ w'_i\mathbf{p}_i & \mathbf{0}^T & -x'_i\mathbf{p}_i \end{bmatrix} \tag{4.4}$$

The matrices, A_i, corresponding to each point correspondence can be stacked to form a matrix $A = \begin{bmatrix} A_1 & A_2 & \cdots & A_{n_p} \end{bmatrix}^T$ which satisfies the relation

$$A\mathbf{h} = 0 \tag{4.5}$$

In the case of noisy points an approximate solution can be obtained by minimization of the cost function (algebraic distance) $\|A\mathbf{h}\|$. This is the DLT algorithm for point correspondences (see Hartley and Zisserman [25] for details).

4.1.2 Normalization for points

The DLT algorithm is sensitive to the choice of the coordinate frame (origin and scale). Hartley and Zisserman [25] suggest a normalization step to make the data well conditioned. A similarity S is applied to transform points such that their centroid is at the origin and the average distance from the origin is $\sqrt{2}$

$$\tilde{\mathbf{p}}_i = S\mathbf{p}_i \quad \forall i \in \{1...n_p\} \tag{4.6}$$

where S is defined as

$$S = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

Corresponding points are also normalized by a similar transform S'. The homography matrix $\tilde{H}$ is computed using DLT on these normalized correspondences. It is denormalized to get the homography estimate for the original correspondences.

$$H = S'^{-1}\tilde{H}S \tag{4.8}$$

4.1.3 Normalization for lines

A line $ax + by + c = 0$ can be represented as a vector of coefficients $\begin{bmatrix} a & b & c \end{bmatrix}^T$. Using this representation, the transformation of a line $\mathbf{l}_i = \begin{bmatrix} p_i & q_i & r_i \end{bmatrix}^T$ under the homography H is given by

$$\mathbf{l}'_i = H^{-T}\mathbf{l}_i \quad or \quad \mathbf{l}_i = H^T\mathbf{l}'_i \tag{4.9}$$

This is analogous to the point case described above and a similar relation as Equation 4.4 can be obtained. Additional rows corresponding to the line correspondences are appended to the matrix A in Equation 4.5. However, including lines in the same framework as points requires lines to be normalized with the same similarity transform S. Dubrofsky and Woodham [9] extend point normalization to

lines as

$$\tilde{\mathbf{l}}_i = s \begin{bmatrix} p_i \\ q_i \\ sr_i - t_x p_i - t_y q_i \end{bmatrix} \tag{4.10}$$

The corresponding lines are also normalized using the transform S'. This gives a set of normalized line correspondences. Now, lines can be treated uniformly along with the normalized points to estimate the homography.

4.2 Adding ellipses

The coefficients of a conic cannot be treated in a similar way to lines and points. However, the constraints obtained from ellipses using existing points and lines in the scene can be transformed into additional line and point correspondences.

4.2.1 Pole-polar relationship

Let C be a matrix of coefficients of a conic such that any point $\mathbf{x}$ lying on the conic satisfies the relationship $\mathbf{x}^T C\mathbf{x} = 0$. The transformed conic under a homography H is given by

$$C' = H^{-T} C H^T \tag{4.11}$$

The polar line with respect to any point $\mathbf{x}$ in the plane is defined as $\mathbf{l} = C\mathbf{x}$. It is straightforward to prove that if two points correspond in two images (transformed by a homography), their polar lines with respect to the corresponding conics in the images also transform under the same homography [8]. Let $\mathbf{x}$ and $\mathbf{x}'$ be two matching points, C and C' be matching conics in the images and $\mathbf{l} = C\mathbf{x}$ be the polar corresponding to pole $\mathbf{x}$ with respect to conic C. The polar in the corresponding image is given by

$$\begin{aligned} \mathbf{l}' &= C'\mathbf{x}' \\ &= (H^{-T} C H^{-1})(H\mathbf{x}) \\ &= H^{-T} C\mathbf{x} \\ &= H^{-T}\mathbf{l} \end{aligned} \tag{4.12}$$

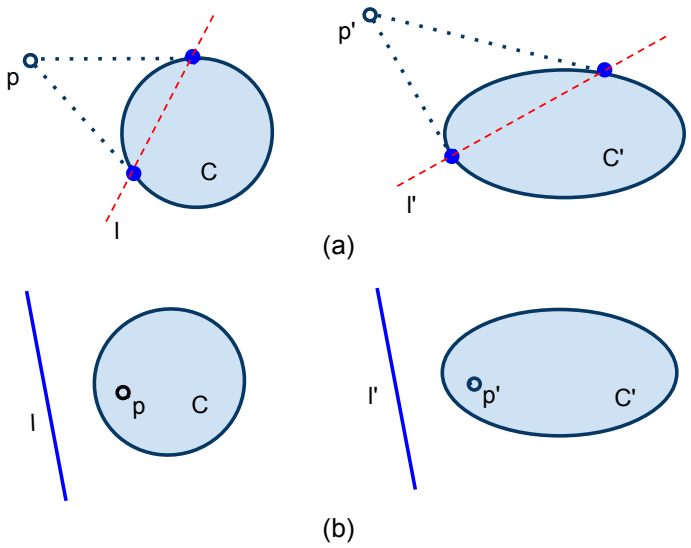


Figure 4.1: New correspondences generated using points and ellipses. There are two images transformed by a homography with point correspondences p, p' and ellipse correspondences C, C'. (a) When points lie outside the ellipse, we get two new point correspondences (shown with solid blue dots) and (b) when points lie inside or on the ellipse we calculate one new line correspondence (shown as blue solid line).

We can similarly prove that if two lines correspond then their poles with respect to a corresponding ellipse also correspond. Despite the fact that Equation 4.12 is true for a general conic, we use it here only in the context of ellipses. We use these results to generate additional correspondences as described in the next Section.

4.2.2 Generating new correspondences

For each corresponding point and line, we generate additional correspondences using an ellipse as follows (also summarized in Figure 4.1).
For a corresponding

1. *point lying outside the ellipse*: calculate the two points of intersection of its polar with the ellipse (two new points correspondences, see Figure 4.1(a)).

2. *point lying inside or on the ellipse*: generate a new line correspondence by calculating its polar with respect to the ellipse (one new line correspondence, see Figure 4.1(b)).

3. *line intersecting the ellipse*: use the corresponding intersection points (two new point correspondences).

4. *line not intersecting the ellipse or a tangent*: calculate the corresponding pole (one new point correspondence).

We repeat the same procedure for each ellipse in the scene. Once all new correspondences are calculated, we append them to the existing matches. These augmented point and line correspondences, now also including constraints due to ellipses, are used to estimate the homography (as described in Section 4.1).

4.2.3 Minimal case

Using this approach, two points and an ellipse (if at least one point is lying outside the ellipse) are enough to solve for a homography transformation uniquely. In this case we get at least two new point correspondences which we can use along with existing points (and possibly other new line correspondences generated by points lying inside the ellipse). We also verify experimentally that the situation is symmetric with respect to lines. If there are two lines in the scene (if at least one of these physically intersects the ellipse), we have enough constraint to solve for the homography.

Chapter 5

Homography Estimation and Tracking

The final step of the system involves fine tuning the homography estimate using non-linear optimization. The objective function minimized is based on the residual area between the detected shapes and the projected model. This is inspired by the maximum likelihood estimate of a homography using point matches between two views [25]. This is also closely related to parametric correspondence for chamfer matching [2] which is also effectively minimizing the integral area between the curves. Using this approach we are able to bring ellipses, points and line features into the same framework for minimization of geometric error. This is a complex objective function with no clear functional form. We use the DIviding RECTangle (DIRECT) [30] method for function minimization as described in Section 5.3. Before moving onto the final fine-tuning step, we review key-frame based homography estimation. We also describe the homography estimate between consecutive frames used to assist robust homography estimation over a long sequence.

5.1 Linear homography estimation

We use two alternative approaches for homography estimation between the current frame and the model using linear (DLT based) methods. The first method is based on features correspondences with key-frames and the second is based on point

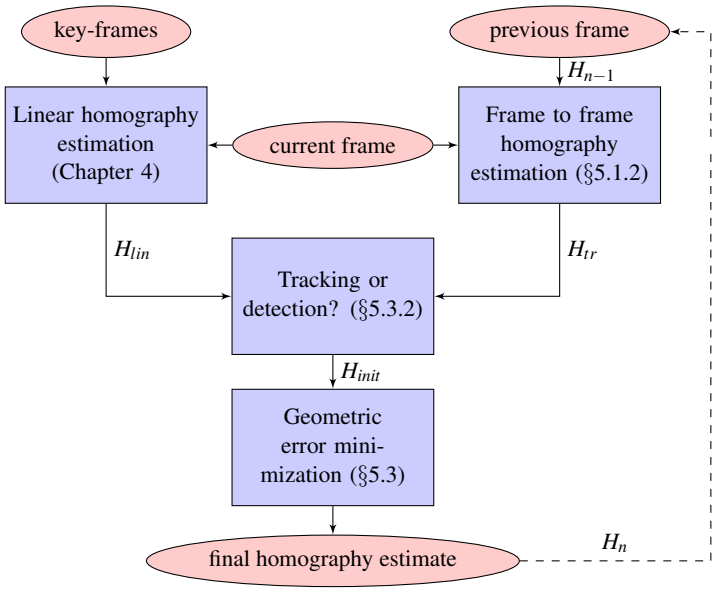


Figure 5.1: Tracking pipeline.

correspondences with previous frames. Since each frame is independently compared with key-frames, a homography estimate based on a key-frame can be seen as obtained from detection. On the other hand, homography estimation by finding correspondences between the previous frame and the current frame is a tracking based approach (described in the next section). We can choose one of these as an initial value for non-linear optimization (as described in Section 5.3.2). This approach is summarized in Figure 5.1.

5.1.1 Homography with key-frame

We use the homography estimate from the previous frame to assist feature detection in the current frame. We also identify the closest key-frame to establish feature correspondences (as described in Chapter 3). Once these feature (point, line and ellipse) correspondences are known, we can use them for linear homography estimation (as described in Chapter 4). Since we already have the transforms for key-frames available, we can calculate the homography for the current frame.

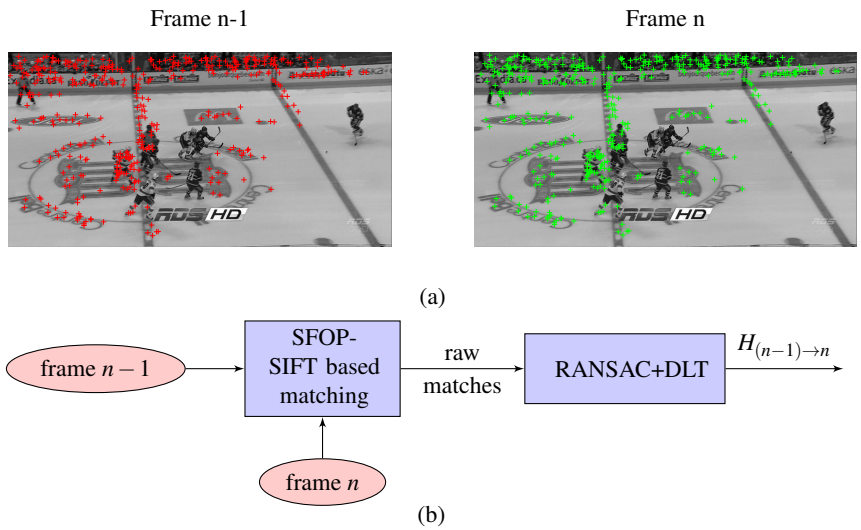


Figure 5.2: Homography estimation between consecutive frames. (a) Filtered point matches between a pair of consecutive frames. (b) Method used for estimating frame to frame homography, $H_{(n-1)\to n}$.

5.1.2 Homography between consecutive frames

It is possible to obtain a large number of point matches between two consecutive frames. We use these point matches to estimate the homography using the DLT algorithm (see Section 4.1.1). The point correspondences are obtained using SFOP-SIFT key-point matching (see Section 3.2 for details). We also use RANSAC [18] to filter out the spurious matches (Figure 5.2(a) shows filtered correspondences used for homography estimation). Please note that not all the point matches lie on the image of the rink. Since the camera is mounted on a tripod, it is only allowed to pan, tilt and zoom. With an additional assumption that the optical center of the camera coincides with its center of rotation (see Section 1.1.1), all the images obtained using such a camera transform under homography (see Section 2.3 in Hartley and Zisserman[25] for details). Hence, the homography between two frames of the video is same as the homography between the images of the rink (a planar surface) in these frames. This allows us to use all the point matches (corresponding to stationary objects in the scene) for homography estimation, even if

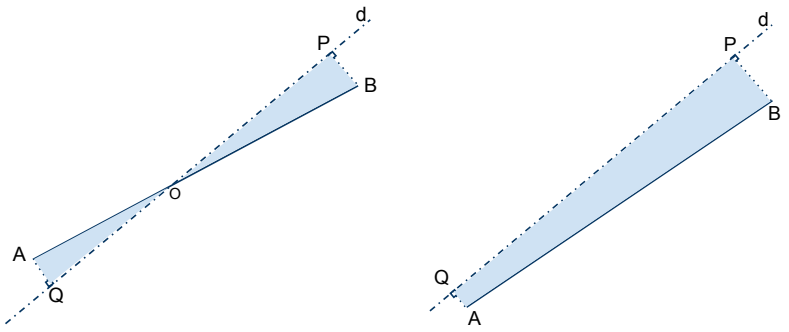


Figure 5.3: The area between line segments is calculated by drawing perpendiculars from the ends of model line segment AB onto the detected line d intersecting it at points P and Q respectively. a) When the lines intersect at a point (let's say O) the residual area is the sum of areas of the two right angle triangles OAQ and OPB and b) when there is no intersection the residual area is given by the area of the trapezoid $ABPQ$.

they do not lie on the rink. If we filter out the point matches lying outside the rink, we can completely relax this assumption of the camera with a fixed optical center. However in the current system, we leave this implementation for future work.

We use the homography estimate (obtained after minimization of geometric error) for the previous frame and multiply it with the frame to frame homography to obtain the tracking based estimate. Figure 5.2(b) summarizes the method.

5.2 Residual area

For area minimization we assume that we have already localized all the feature matches. Let the correspondences between model and image features for lines, ellipses and points be $\{l_i \leftrightarrow l_i'\}$, $\{C_i \leftrightarrow C_i'\}$ and $\{x_i \leftrightarrow x_i'\}$ respectively. For a given homography H, the projections of these features, $\hat{l}_i'$, $\hat{C}_i'$ and $\hat{x}_i'$ on the image are given by

$$\hat{l}_i' = H^{-T} l_i \tag{5.1}$$

$$\hat{C}_i' = H^{-T} C_i H^{-1} \tag{5.2}$$

$$\hat{x}_i' = H x_i \tag{5.3}$$

Once we have projected model features to the image plane, we can compute the residual area (see more details in the following subsections). We propose the weighted sum of these areas as our cost function for optimization

$$A_{res}(H) = A_l(H) + w_c A_c(H) + w_p A_p(H) \tag{5.4}$$

where A_l and A_c are the sum of residual area between all the lines and ellipses respectively. A_p is the sum of geometric errors corresponding to the point correspondences. w_c and w_p are weights to adjust the relative importance of each feature type in the optimization.

5.2.1 Area between line segments

Two lines can possibly have an infinite area between them. We restrict this area to the region visible in the image. Figure 5.3 shows the simple strategy to calculate the area between a projected model line segment and a detected line on the image. If a projected model line segment is partially visible in the image, only the visible part of the line segment is considered for area calculation. We sum the residual area corresponding to all the visible model line segments referred to as A_l in Equation 5.4

5.2.2 Area between ellipses

The residual area between two ellipses can be simply defined by *sum of areas* $-2\times$ *intersecting area*. There are various approximate and exact methods which can be used to calculate the intersecting area between two ellipses. One of the simpler methods is to approximate an ellipse with a convex polygon and calculate the area of intersection between them. We can also use Monte Carlo integration [50] to find the intersection area. An exact, analytical method is described by Eberly in [12]. This approach relies on first finding the points of intersection between two ellipses. Once these points are known, the intersection area can be described in terms of the area between the elliptical arcs (between two points of intersection) and the line segments joining their ends. This would require minor modifications to be used in the scenario when the area of intersection does not completely lie inside the image (as shown in Figure 5.5) as we would also need to take care of

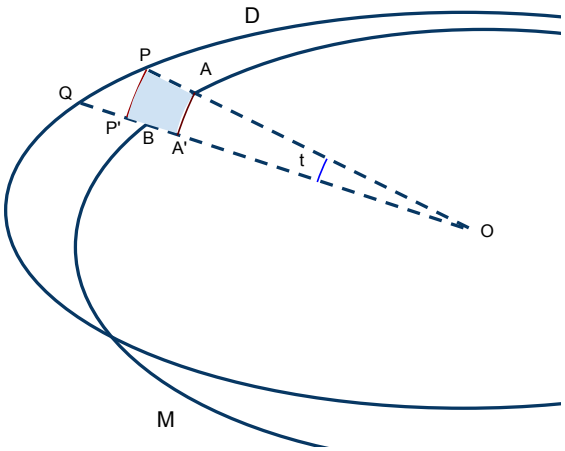


Figure 5.4: Approximating residual area between ellipses. Given a detected ellipse D and a model ellipse M with center O, we calculate the area for a block arc $APPA'$ (shaded area) subtending an angle t at the center. We use it as an approximation for the actual area contained within two ellipses for this angle range. For all the small arcs (we choose t=0.01 radians) for ellipse D, we sum these areas to get the total residual area.

points of intersections with the image boundaries.

We approximate ellipses with small circular arcs similar to the method described in [11]. These arcs are centered on one of the ellipse centers. Our method is approximate but we do not have to explicitly solve the equations to find the points of intersections. We directly approximate the residual area using a numerical approximation. Intersection with the image rectangle is also handled in the same framework. Figure 5.4 shows the area approximated for an arc using this approach. Summing these small areas over the whole angular range gives us the residual area for an ellipse correspondence. We sum these residual areas for all visible ellipses to calculate A_c (see eq. Equation 5.4).

The residual area is only calculated for bigger ellipses on the rink. We treat smaller circular spots on the rink as point matches (as described in Section 5.2.3).

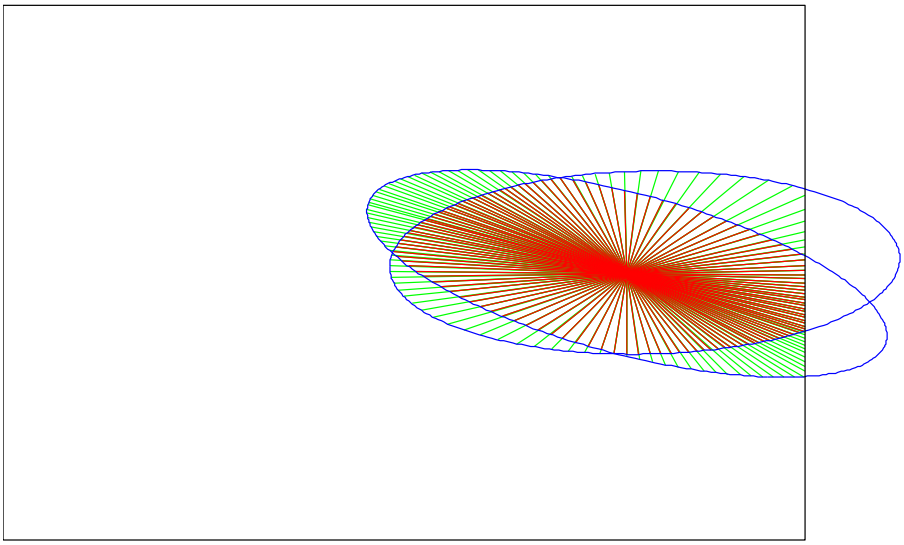


Figure 5.5: The residual area between two ellipses. The shaded region in red shows the intersection area and the region shaded in green corresponds to the residual area. The rectangle represents the image boundary. Please note that we only account for the area visible in the image.

5.2.3 Point correspondences

The geometric model of the rink does not contain any texture information. It is not an appearance model of the actual rink. Unlike lines and ellipses, we do not have direct point correspondences between the model and the image. However, we have a reliable estimate of the homography between key frames and the model. We can use this to back project the point correspondences from the matching key frame to the model. Once we have these correspondences, we can calculate the geometric error in the point correspondence. The area term for points in Equation 5.4 is defined as

$$A_p(H) = \sum_i d(\hat{x}'_i, x'_i)^2 \tag{5.5}$$

5.3 Geometric error minimization

Once we have the area calculation framework in place, the final optimization problem can be formulated as

$$H_{est} = \underset{H}{\operatorname{argmin}}(A_{res}(H)) \tag{5.6}$$

where H_{est} is the homography estimate. We do not have a functional form for this objective function. We use a sampling based algorithm called DIRECT. It does not require any knowledge about the gradient of the objective function. It is meant to solve the global optimization problem with bounded constraints (i.e., we need to specify the range of all the input parameters). Another important merit of the DIRECT algorithm is that it is an *anytime* optimization technique, which implies that we always have a solution (with equal or lower error than the last iteration) available in case we have to stop the optimization due to time constraints.

As DIRECT requires no knowledge of the gradient, it samples the input domain. The value of the function at these sampled points is used to determine where to look next. It divides the domain into hyper-rectangles and samples their centers to move towards the optimum (hence the name DIviding RECTangle or DIRECT). It eventually converges to a global minimum. However, a large number of samples may be required for high dimensional (dimension of the homography vector) problem like ours. See [16] for further details.

We use a Matlab implementation of DIRECT by Finkel [17]. To use this implementation all we need is: an objective function, a real valued function which takes a homography matrix as input to return value of the residual area; an initial value of the homography matrix to start with and an allowed range (min and max value) for every dimension of the homography vector.

5.3.1 Objective function

We pass the objective function as a parameter to the optimization algorithm. We outline our objective function *GetResidualArea* in Algorithm 5.1. In this algorithm, $TransformModel$ is a simple function which transforms the model to the current frame using an input homography. $CorrespondingModelProjection$ re-

trieves the corresponding model for each feature. Please note that we only use the features detected in the image. *FeatureResidualArea* routine is based on different definitions of residual area for different kinds of features (as described in Section 5.2).

Algorithm 5.1: *GetResidualArea*

Input: Homography matrix H, set of detected features F and model M

Output: A

$A \longleftarrow 0$

$M_t \longleftarrow TransformModel(M,H)$

foreach $f_i \in F$ **do**

  $m \longleftarrow CorrespondingModelProjection(M_t,i)$

  $A_i \longleftarrow FeatureResidualArea(f_i,m)$

  $A \longleftarrow A + A_i$

end

5.3.2 Choosing the initial value

In most cases, we start with a close initial estimate from linear homography estimation from the last step (described in Chapter 4). However, this makes our system sensitive to the errors in detection (i.e., a line being confused with a parallel line or a bad fit for a noisy partial ellipse). To avoid this we switch to an estimate from tracking if certain conditions are met. As we run this sequentially, we have the homography estimate available from the last frame and a frame to frame homography estimate based on point matches. Now a question arises, what should be our criteria for choosing between these two options as an initial value. We solve this problem by deciding which value we trust more. Intuitively, to avoid drifting we would like to assign more weight to the current estimate. Algorithm 5.2 outlines the procedure. To estimate H_{init} for the current frame n, we use the previous homography estimate H_{n-1}, frame to frame homography $H_{(n-1)\rightarrow n}$ and the linear estimate based on detection H_{lin}. Parameter $w \geq 1$ defines the weight given to detection compared

to tracking. For our experiments we choose $w = 2$.

Algorithm 5.2: *GetInitialValue*

Input: $H_{(n-1)\to n}$, H_{n-1}, H_{lin}, F, M, w

Output: H_{init}

$H_{tr} \longleftarrow H_{(n-1)\to n} \times H_{n-1}$

$A_{lin} \longleftarrow GetResidualArea(H_{lin}, F, M)$

$A_{tr} \longleftarrow GetResidualArea(H_{tr}, F, M)$

if $A_{tr} \geq w \times A_{lin}$ **then**

  $H_{init} \longleftarrow H_{tr}$

else

  $H_{init} \longleftarrow H_{lin}$

end

5.3.3 Specifying bounds

With the camera intrinsics not known, we do not have a structured way to quantify the camera motion in terms of components of the homography vector. We choose a relaxed bound so that our fitting algorithm is not sensitive to this choice. However, choosing too large a range can make this optimization slow. To get approximate values of the bound, we manually annotate a few frames in the sequence to find their homography with respect to the model. We average the absolute value of these ground truth homography vectors. We use 15% of the average value as our allowable range of homography correction during this non-linear step, let's say $\mathbf{b}$. The range is given by $[\mathbf{h}_{init} - \mathbf{b}\ \mathbf{h}_{init} + \mathbf{b}]$, where $\mathbf{h}_{init}$ is the vector representation of the matrix H_{init}.

5.3.4 Termination criteria

To limit the time spent on each frame, we fix number of iterations. This does not guarantee a global minima but since DIRECT is an *anytime* optimization technique, we always get a fit with an equal or lower error than what we started with. In our experiments, we use $10,000$ as the limit on number of iterations.

Chapter 6

Results

To test the system, we use a broadcast hockey video dataset (introduced in Section 1.1.2). In this chapter, we describe experiments and report results obtained on this dataset. First, we describe a method to generate ground truth data and define an error measure for the homography used for evaluation in the following experiments. Second, we talk about the accuracy of line and ellipse detections (as described in Section 3.3). We also evaluate the effect of including ellipse matches in the DLT framework along with points and lines (as described in Chapter 4). We report the reduction in error obtained after geometric minimization of the residual area (as discussed in Section 5.3). We then discuss the results showing the importance of using key-frames (described in Section 3.1.2). Finally, we discuss the robust homography estimates obtained for a 1000 frame sequence.

6.1 Ground truth

It is hard to generate the ground truth for all the frames in the dataset. The ground truth in this case means the best possible homography fit for a frame. A good fit has to be visually evaluated by a user, as we do not have a clear way to quantitatively measure it. To simplify this issue, we only annotate a subset of frames from the 1000 frame sequence with ground truth. The process works as follows: we select a set of frames and manually choose point correspondences between these frames and the geometric model to estimate the homography using the DLT algo-

rithm for points. This initial estimate of homography is used to detect line and ellipse features on these frames. We further refine the estimate by using geometric minimization of the residual area error (as obtained by using Algorithm 5.1). The error measure does not go to zero even for these ground truth frames as features never align perfectly with the projected model. We refer to this error as the *ground truth residual area*.

6.2 Error measure

These annotated frames represent a close approximation to the perfect transformation between the geometric model and the video. We make sure the frames we choose have line and ellipse detections which are closely aligned with the actual features in the image. We project the geometric model using the homography and calculate the residual area between projected features (only lines and ellipses, no points) and the detections in the ground truth frames. In the subsequent discussion, this error is referred as the *residual area error* for a given homography estimate in a particular frame.

Please note that this error measure is a comparable quantity only for different homography estimates for a single frame. This is not an absolute measure of the error in homography estimate as it is dependent on the features visible in that particular frame. However, using this measure makes sure that all the different homography estimates for a frame are compared using a single set of detections (detections in that ground truth frame).

6.3 Experiments

First, we describe experiments to evaluate different sub-modules of the system. We finally test the complete system on a long sequence.

6.3.1 Line and ellipse detection

Our detection algorithm is dependent of the homography obtained for the last frame. Figure 6.1 shows the detections obtained for lines and ellipses for a few selected frames while running the whole system. Please note that these results

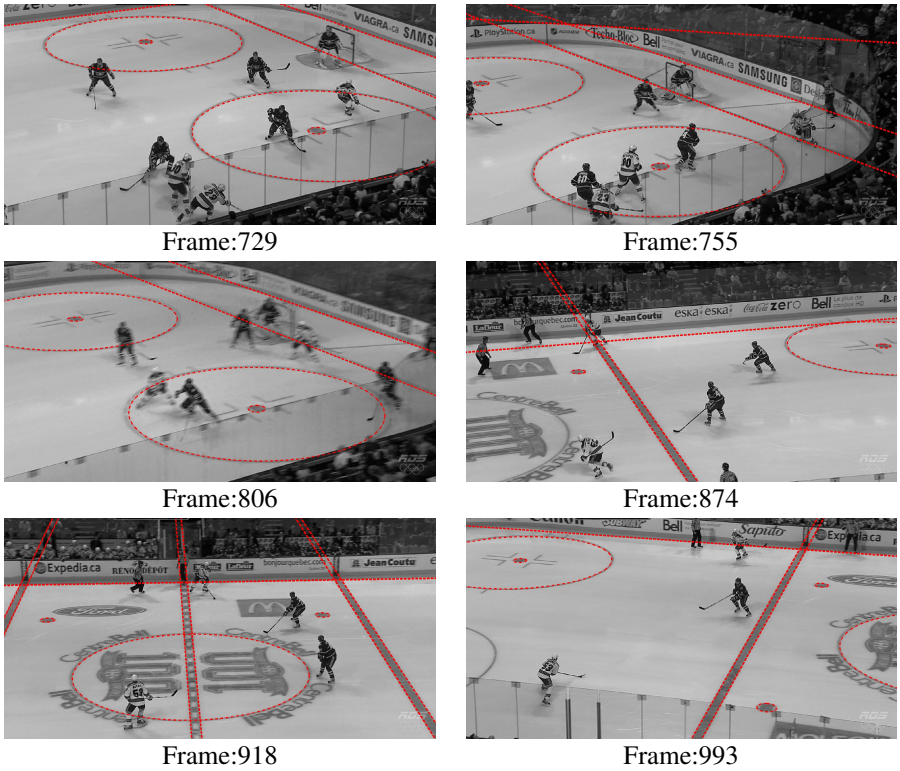


Figure 6.1: Detection of lines and ellipses in a few selected frames from the input sequence. There is a false line detection in Frame:755. In Frame:874, a line is not correctly localized. Lines and ellipses are well localized despite motion blur in Frame:806.

should not be examined in isolation, as accuracy of these detections is dependent on the accuracy of the homography estimate for the previous frame. Lines and ellipses are reliable features for hockey rink images and are detected accurately for most of the frames. We highlight some of the errors in detections. A few inaccurate detections (see Frame:755 and Frame:874) are observed, however the performance is not affected by motion blur (see Frame:806 in Figure 6.1).

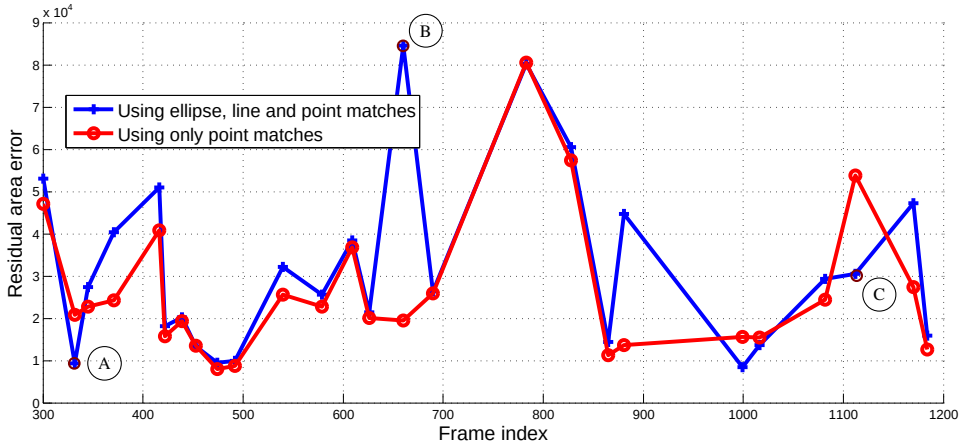


Figure 6.2: Error in the linear homography estimation using point, line and ellipse correspondences compared with the error only using points. The residual area error (defined in Section 6.2) is plotted for both the cases along the y-axis. The x-axis shows the frame number of the image. The performance after adding ellipses and lines is similar or worse in most of the cases. To see what these error numbers translate to in terms of visual error, we have plotted the linear homography estimates for selected frames (332, 660, and 1112, denoted by A, B, and C respectively) in Figure 6.3.

6.3.2 Using ellipse correspondences

As discussed in Chapter 4, the addition of ellipse correspondences gives rise to additional line and point correspondences. These can be used along with the existing line and point matches for linear homography estimation. We evaluate the effect of these additional constraints on the accuracy of the homography estimate. Figure 6.2 shows the residual area error (defined in Section 6.2 above) in the linear estimate of homography obtained only using points and compare it with the case when ellipse, line and point correspondences are combined. We expect that adding these additional constraints would lead to a reduced error. However, the results show that the performance after adding lines and ellipses is similar or worse in most of the cases. We are not sure how the error in parameters of ellipse and line

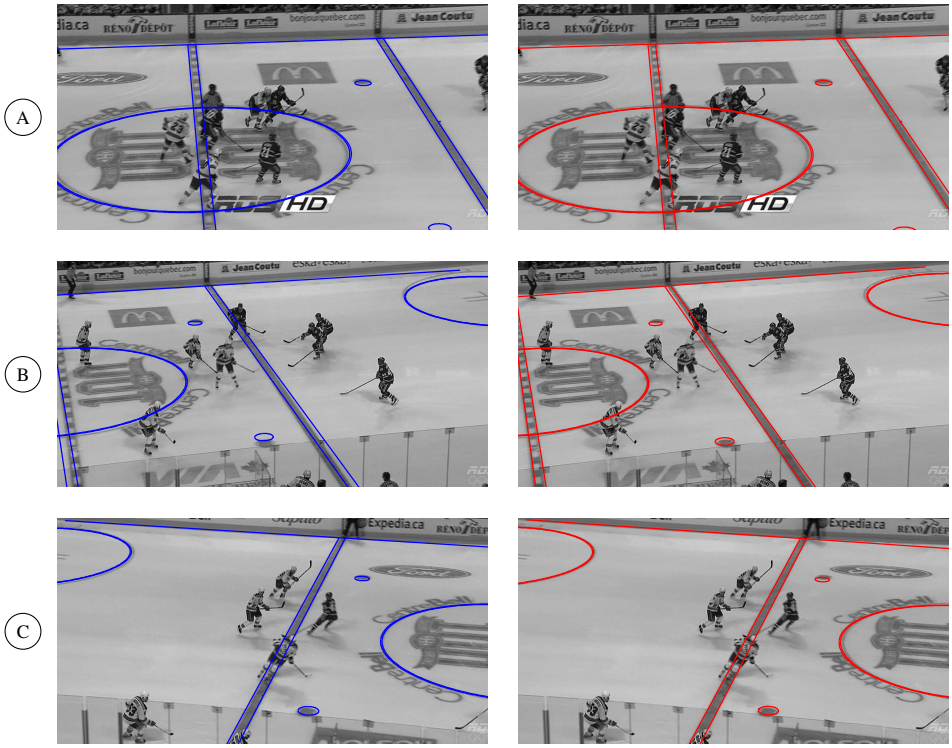


Figure 6.3: Rows A, B, and C correspond to the selected frames in Figure 6.2. Images show the video frames with the a transformed geometric model superimposed on top of them. The left column with blue lines shows the homography estimates obtained using ellipse, lines, and point correspondences while the right column with red lines shows the homography estimates just based on point correspondence.

detections affect the homography estimate. The analyses of this behavior is left for future work. We still use ellipse and line correspondences for linear homography estimation. These can be useful in cases when there are not enough point correspondences between an image and the corresponding key-frame. Such a situation may arise due to motion blur (see Figure 6.4 for an example).

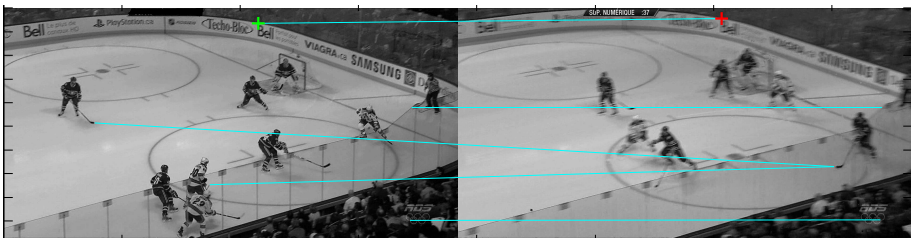


Figure 6.4: Key-point matches under motion blur. Frame 807 (right image) has only a single consistent match with its closest key-frame (left image). Green and red crosses show the point correspondence.

6.3.3 Geometric error minimization

As a final step of our system we refine our linear estimate of the homography using minimization of the geometric error. We evaluate the quantitative reduction in the residual area error due to this non-linear optimization. In Figure 6.5 we compare the error in homography estimation after the geometric error minimization to the linear homography estimate. We observe that there is a significant reduction in the error after the optimization step. We also find that the tracking is more stable (observe the variation in the error corresponding to the linear estimate in Figure 6.5).

6.3.4 Key-frames

Key-frames are crucial to establish a robust match between a frame and the geometric model. This saves the system from drift due to using the frame to frame correspondences. Figure 5.1 summarizes the tracking framework where we combine the frame to frame estimate with the key-frame based homography estimate. To test the importance of matching each frame with key-frames, we run the system on the same data relying purely on the frame to frame homography (or tracking) estimate as the initial value for the non-linear optimization. We obtain no homography estimate from the key-frames (or detection). The rest of the system is kept the same. Figure 6.7 shows the results for a few frames in the sequence. We observe that the error accumulates quickly and the system is not able to recover from the error.

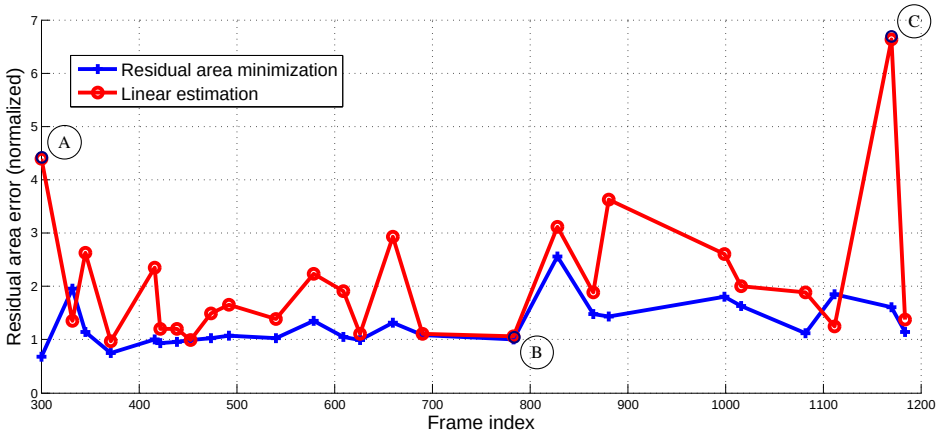


Figure 6.5: The error in homography estimation after minimization of the geometric error compared with the linear estimate. Along the y-axis we have the residual area error, normalized by the *ground truth residual area* (as defined in Section 6.2). The frame numbers are plotted along the x-axis. We also show homography estimates for three selected frames, denoted by A, B, and C, in Figure 6.6.

6.3.5 Tracking in a long image sequence

Finally we test our system, using all the components and running it over a long image sequence. Figure 6.8 shows a few frames from the sequence with the estimated homography. We observe for most of the cases features in the model are well aligned with the features in the image. We observe a larger error in Frame:817. However, the system is able to recover quickly (see Frame:871). This shows that we are able to robustly estimate the homography for a long sequence accurately. We also observe that there is no error accumulation. The last frame is well aligned with the projected features from the model (see Frame:1299). This shows that the system could possibly continue to track a longer sequence.

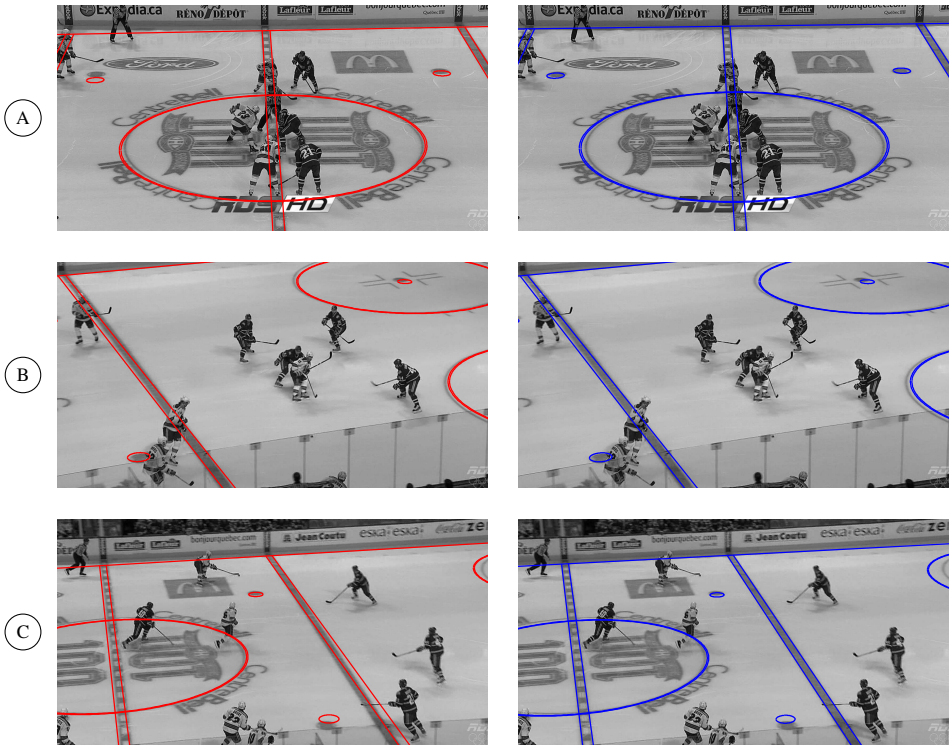


Figure 6.6: Rows A, B, and C correspond to the selected frames in Figure 6.5. The images show the video frames with the transformed geometric model superimposed on top of them. The left column with red lines shows the linear homography estimates while the right column with blue lines shows the homography estimates obtained after the minimization of geometric error.

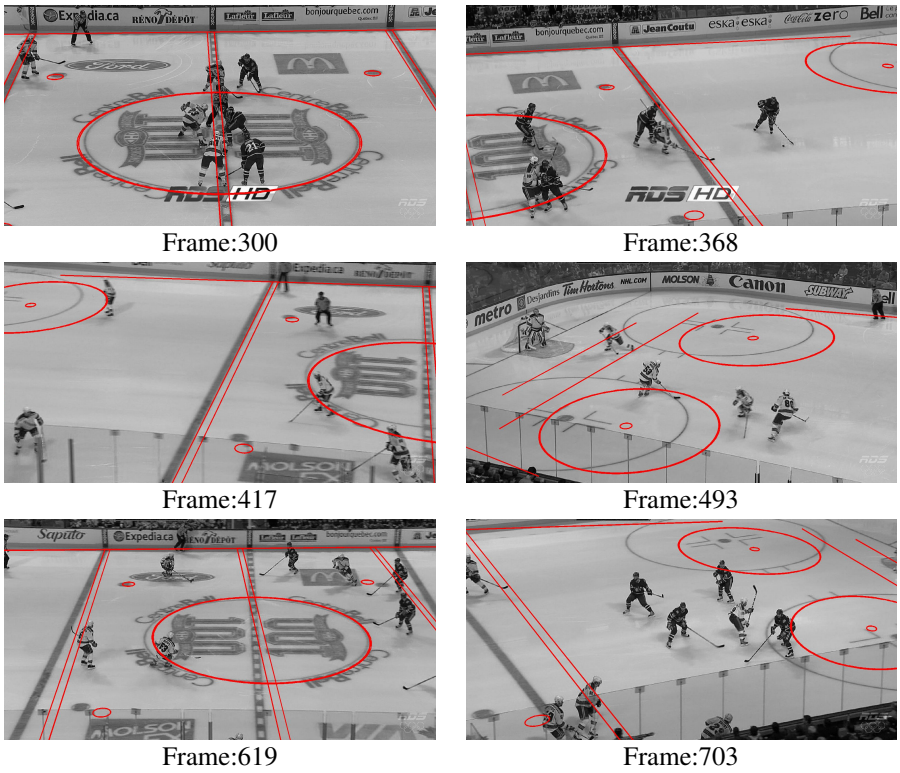


Figure 6.7: Homography estimation for a sequence without using key-frames. Tracking begins at Frame:300. Error accumulates quickly (see Frame 368). The system is not able to recover subsequently.

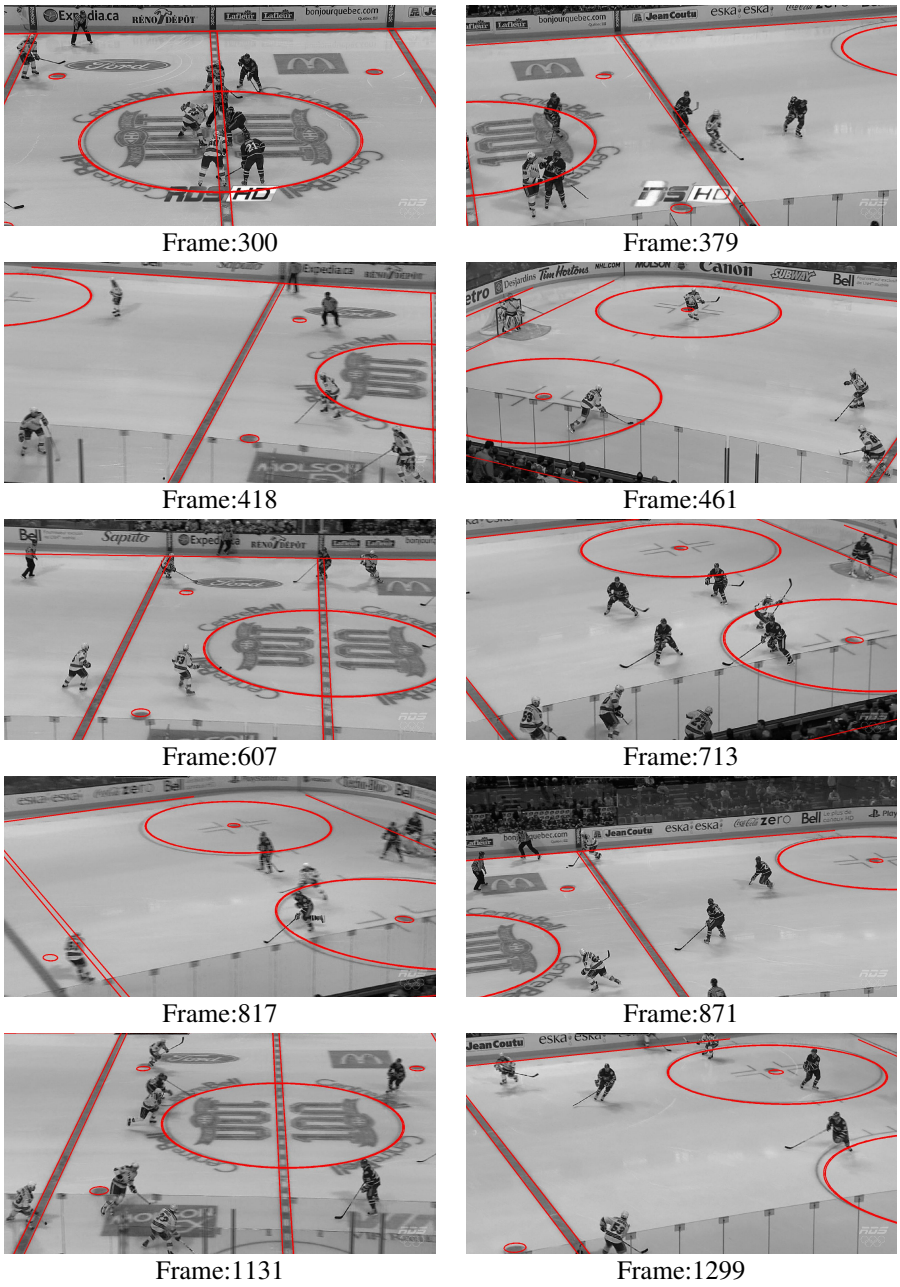


Figure 6.8: Final homography estimate for selected frames from 1000 frame sequence.

Chapter 7

Discussion and Future Work

We effectively combine the geometry, appearance and motion information to get the best possible estimate of the homography between a geometric model of the field (or rink) and each frame in the sports video. In this work, we focus on using the geometric shapes in the model as features to estimate the homography. To achieve this, we develop a method for incorporating ellipse features in homography estimation along with line and point features (which have been traditionally used to solve similar problems). We show that the minimization of an area based geometric error measure can be used to refine the linear estimate and stabilize tracking. We also combine the geometric model with an appearance model using the key-frame idea to add robustness to the system.

The results we present show that our system is able to robustly track long sequences of the order of 1000 frames. We also show the effects of using or removing different sub-modules which add to the accuracy and robustness. We still need to show the system's robustness for longer sequences and possibly the whole game (around an hour long). We would also like to try it on other sports like soccer, basketball and football.

In this chapter, we summarize the current limitations of our system along with possible future directions. Finally, we present a brief conclusion.

7.1 Future directions

We now summarize the major limitations and suggest improvements in the specific sub-modules described in this work.

7.1.1 Efficient minimization of geometric error

As described in Section 5.3.1, we have a complex geometric error function with no clear functional form. We use a sampling based technique for optimization called DIRECT. This randomly samples (with no gradient information) a high dimensional space which is not efficient and requires a large number of trials. We can make some simplifying assumptions about our cost function to reduce it to a simpler functional form so that standard techniques used for solving for the non-linear least-squares problems (e.g., Levenberg-Marquardt algorithm [39]) can be applied. Currently, we numerically estimate the residual area between two ellipses. We can also obtain an estimate for the area by sampling a few points on the ellipse boundaries (similar to the technique described in [44]). This may lead to a large reduction in running time.

7.1.2 Selecting key-frames

In the current implementation of the system we manually select key-frames. A set of five images was chosen from the sequence to cover the whole range of camera motion with some overlap between frames. However, this is dependent on the location of the camera and its focal length. We would like to eventually automate the process of key-frame selection. Another important question is to determine the number of key-frames. Further experiments need to be conducted to evaluate an optimal number for key-frames. There is an alternative to the key-frame approach which involves registering point features from different frames to a geometric model (as described by Hess and Fern[29]). This approach, originally implemented for football videos, can also be tried for the hockey video rectification problem.

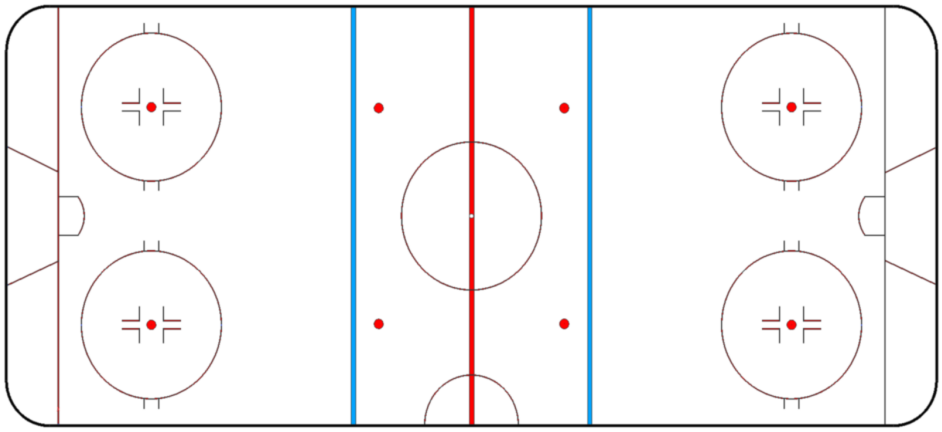


Figure 7.1: The complete geometric model of an NHL hockey rink [43]. This can be compared with Figure 3.2 to see the features which we excluded in the geometric model used for the current system.

7.1.3 Using a complete model

The model that we use currently is a simplified version of the actual geometric model of an NHL rink (see Figure 7.1). Some of the features were left out as they were harder to detect. However these additional features are distinct (e.g., lines near the center of the four big outer circles) and can add to the robustness to the detection module. Augmenting our model to incorporate all the features on the rink is a natural extension for future work.

7.1.4 Refining the geometric model using data

The model we obtain from the rule book is a close but not a perfect representation of all the hockey rinks we could possibly encounter. Some of the sports authorities only provide a rough guideline for a few measurements in the rink. Since, our whole tracking and detection framework is based on the assumption of a perfect geometric model, we need to make sure that it is correct. To solve this problem, we can start with an approximate model and system should be able to use the observed features to correct the model over time. Automatically generating the accurate geometric model is an interesting project for future work.

7.1.5 Motion model

We do not explicitly build a motion model for the homography we are tracking over time. We hypothesize smooth camera motion leads to a smooth variation in homography. A major road-block in interpolating homography is the high-dimensional nature of the transformation. Building models for homography corresponding to specific camera motions (i.e., pan, tilt and zoom) without calibrating the camera, is an interesting and challenging problem. Also, in this work, we only use information from the previous frame to predict current estimate of homography. We could possibly look at a wider horizon in time. In case of applications with no real-time requirements, even future frames could possibly be used to smooth and stabilize the output homography estimates over a sequence.

7.1.6 Removing lens imperfections

Lens imperfections are also a major challenge. Broadcast quality videos also have measurable radial distortion which is visible to a careful observer. But this does not normally affect the game viewing experience. This non-linear distortion makes the homography assumption invalid. Also, since straight lines in the model/world are not straight lines in the image, this also adversely affects the line detection algorithm. As the camera zooms in smoothly, we hypothesize that the lens distortion parameters also vary smoothly over time. We can track these parameters along with the homography transformation. This can possibly make our predictions of the player locations on the rink (based on the homography estimate and player tracking results) more accurate.

7.1.7 Robust feature detection

We observe that in our case a large portion of the inaccuracies in detection correspond to partial ellipses (not every ellipse is completely visible in the frame at a given point in time). Also, in some other sports (e.g., soccer) many of the shapes on the field are only partially circular. The ellipse fitting for a partial data is an interesting challenge. The line detection can be improved by using the approach described by Fan et al. [13]. This idea can possibly be extended to detect more accurate ellipse boundaries.

7.1.8 Normalization for lines

Analogous to the normalization for point correspondences in the DLT (as described in Section 4.1.2), line correspondences also need to be normalized as described by Zeng et al. [54]. This work also suggests a method to achieve this normalization. The approach we currently use (as suggested by Dubrofsky and Woodham [9]) normalizes the correspondences only with respect to points. This can possibly lead to instability in the solution when one of the line correspondences passes close to the origin [54]. For combining lines and points for homography estimation, we still need to work on an appropriate normalization with respect to both.

7.2 Conclusion

Rectification of the sports video captured from a moving camera is an important problem in automated sports video analysis because it provides a mapping between the events detected in the image coordinates and the world coordinates. In this work, we describe a new way to aggregate the information from heterogeneous features (points, lines and ellipses) and apply it to the specific problem of rectification of broadcast hockey videos. We also present an integrated system which combines various ideas related to feature detection, homography estimation and tracking. The problem of automatic rectification still holds great challenges and possibilities for interesting research. With all the limitations described above, our approach is a significant next step towards combining a wider variety of heterogeneous scene information for homography estimation and also building an application dealing with real world broadcast data.

Bibliography

[1] Region descriptors, March 2006. URL http://www.robots.ox.ac.uk/~vgg/research/affine/descriptors.html. → pages 24

[2] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: two new techniques for image matching. In *Proceedings of the 5th international joint conference on Artificial intelligence - Volume 2*, pages 659–663, San Francisco, CA, USA, 1977. → pages 43

[3] H. Bay, V. Ferrari, and L. V. Gool. Wide-baseline stereo matching with line segments. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:329–336, 2005. ISSN 1063-6919. → pages 10

[4] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision–ECCV 2006*, pages 404–417, 2006. → pages 23

[5] A. Behrens and H. Röllinger. Analysis of Feature Point Distributions for Fast Image Mosaicking Algorithms. *Acta Polytechnica Vol*, 50(4), 2010. → pages 26

[6] W. Cai, Q. Yu, and H. Wang. A fast contour-based approach to circle and ellipse detection. In *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, volume 5, 2004. → pages 32

[7] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986. ISSN 0162-8828. → pages 10, 27, 28

[8] C. Conomis. Conics-based homography estimation from invariant points and pole-polar relationships. In *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pages 908–915, 2006. → pages 13, 37, 40

[9] E. Dubrofsky and R. Woodham. Combining line and point correspondences for homography estimation. *Advances in Visual Computing*, pages 202–213, 2008. → pages 12, 37, 39, 67

[10] R. O. Duda and P. E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972. ISSN 0001-0782. → pages 10, 11, 32

[11] D. Eberly. Approximating an ellipse by circular arcs. Technical report, Geometric Tools, LLC, 2008. URL http://www.geometrictools.com/Documentation/ApproximateEllipse.pdf. → pages 48

[12] D. Eberly. The area of intersecting ellipses. Technical report, Geometric Tools, LLC, 2009. URL http://www.geometrictools.com/Documentation/AreaIntersectingEllipses.pdf. → pages 47

[13] B. Fan, F. Wu, and Z. Hu. Line matching leveraged by point correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. → pages 10, 11, 66

[14] D. Farin, S. Krabbe, H. Peter, and W. Effelsberg. Robust camera calibration for sport videos using court models. In *Proceedings of SPIE*, volume 5307, page 80, 2004. → pages 9, 10, 18, 19

[15] Felzenszwalb, P. and McAllester, D. and Ramanan, D. A discriminatively trained, multiscale, deformable part model. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) Anchorage, Alaska*, June 2008. → pages 1

[16] D. Finkel. DIRECT optimization algorithm user guide. *Center for Research in Scientific Computation, North Carolina State University*, 2003. → pages 50

[17] D. E. Finkel. DIRECT: A global optimization algorithm, March 2003. URL http://www4.ncsu.edu/~ctk/Finkel_Direct/. → pages 50

[18] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. → pages 10, 11, 12, 16, 45

[19] A. Fitzgibbon, M. Pilu, and R. Fisher. Direct least square fitting of ellipses. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):476 –480, may. 1999. ISSN 0162-8828. → pages 12, 30, 32

[20] A. W. Fitzgibbon, M. Pilu, and R. B. Fisher. Fitzgibbon, pilu, fisher: Direct least squares fitting of ellipses, 1999. URL http://research.microsoft.com/en-us/um/people/awf/ellipse/. → pages 32

[21] W. Förstner, T. Dickscheid, and F. Schindler. SFOP keypoint detector. URL http://www.ipb.uni-bonn.de/sfop/. → pages 24

[22] W. Förstner, T. Dickscheid, and F. Schindler. Detecting interpretable and accurate scale-invariant keypoints. In *12th IEEE International Conference on Computer Vision (ICCV'09)*, Kyoto, Japan, 2009. → pages 23

[23] W. Förstner, T. Dickscheid, and F. Schindler. Detecting interpretable and accurate scale-invariant keypoints. In *12th IEEE International Conference on Computer Vision (ICCV'09)*, Kyoto, Japan, 2009. → pages 10

[24] W. Gander, G. Golub, and R. Strebel. Least-squares fitting of circles and ellipses. *BIT Numerical Mathematics*, 34(4):558–578, 1994. → pages 11, 12

[25] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press New York, NY, USA, 2003. → pages 2, 12, 13, 15, 26, 37, 38, 39, 43, 45

[26] J. Hayet, J. Piater, and J. Verly. Robust incremental rectification of sports video sequences. In *In: British Machine Vision Conference*, pages 687–696, 2004. → pages 16, 17

[27] J.-B. Hayet and J. Piater. On-line rectification of sport sequences with moving cameras. In *MICAI 2007: Advances in Artificial Intelligence*, volume 4827 of *Lecture Notes in Computer Science*, pages 736–746. Springer Berlin / Heidelberg, 2007. → pages 9, 10, 18, 19

[28] J.-B. Hayet, J. Piater, and J. Verly. Fast 2d model-to-image registration using vanishing points for sports video analysis. In *Image Processing, 2005. ICIP 2005. IEEE International Conferenceon*, volume 3, pages III – 417–20, 2005. → pages 19

[29] R. Hess and A. Fern. Improved video registration using non-distinctive local image features. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1 –8, 2007. → pages 9, 10, 18, 64

[30] D. Jones, C. Perttunen, and B. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79 (1):157–181, 1993. → pages 43

[31] J. Kannala, M. Salo, and J. Heikkilä. Algorithms for computing a planar homography from conics in correspondence. In *Proc. the 16th British Machine Vision Conference (BMVC*, volume 1, pages 77–86, 2006. → pages 13

[32] H. Kim and K. Sang Hong. Robust image mosaicing of soccer videos using self-calibration and line tracking. *Pattern Analysis & Applications*, 4:9–19, 2001. ISSN 1433-7541. 10.1007/s100440170020. → pages 10, 16, 18

[33] K. Kim, M. Grundmann, A. Shamir, I. Matthews, J. Hodgins, and I. Essa. Motion fields to predict play evolution in dynamic sport scenes. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 840–847. IEEE, 2010. → pages 1

[34] M. Kumar, C. Jawahar, and P. Narayanan. Geometric structure computation from conics. In *Proceedings of Indian Conference on Computer Vision, Graphics, and Image Processing (ICVGIP)*, pages 9–14, 2004. → pages 13

[35] F. Li and R. J. Woodham. Video analysis of hockey play in selected game situations. 27(1–2):45–58, 2009. → pages 1

[36] D. Lowe. Demo software: Sift keypoint detector, July 2005. URL http://www.cs.ubc.ca/~lowe/keypoints/. → pages 24

[37] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004. ISSN 0920-5691. 10.1023/B:VISI.0000029664.99615.94. → pages 10, 23

[38] F. Mai, Y. Hung, H. Zhong, and W. Sze. A hierarchical approach for fast and robust ellipse extraction. *Pattern Recognition*, 41(8):2512 – 2524, 2008. ISSN 0031-3203. → pages 11, 26

[39] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963. → pages 64

[40] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22 (10):761–767, 2004. → pages 23

[41] R. A. McLaughlin. Randomized Hough Transform: Improved ellipse detection with comparison. *Pattern Recognition Letters*, 19(3-4):299 – 305, 1998. ISSN 0167-8655. → pages 11

[42] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004. → pages 10, 23

[43] NHL. Official rules- rule1: Rink. http://www.nhl.com/ice/page.htm?id=26458. → pages 22, 65

[44] K. Okuma, J. Little, and D. Lowe. Automatic rectification of long image sequences. In *Asian Conference on Computer Vision*, 2004. → pages 9, 10, 16, 17, 19, 64

[45] P. D. Sampson. Fitting conic sections to "very scattered" data: An iterative refinement of the bookstein algorithm. *Computer Graphics and Image Processing*, 18(1):97 – 108, 1982. → pages 15

[46] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994. ISBN 0818658258. → pages 10, 16

[47] I. Sobel and G. Feldman. A 3x3 isotropic gradient operator for image processing. Presented at a talk at the Stanford Art ificial Project, 1968. → pages 27, 28

[48] A. Sugimoto. A linear algorithm for computing the homography from conics in correspondence. *Journal of Mathematical Imaging and Vision*, 13(2): 115–130, 2000. → pages 13, 37

[49] L. Wang, U. Neumann, and S. You. Wide-baseline image matching using line signatures. In *Proc. of IEEE International Conference on Computer Vision (ICCV), Kyoto, Japan, 2009*, 2009. → pages 11

[50] S. Weinzierl. Introduction to Monte Carlo methods. *ArXiv High Energy Physics - Phenomenology e-prints*, June 2000. → pages 47

[51] C. Xu, U. CS, B. Kuipers, U. CSE, and A. Murarka. 3D pose estimation for planes. In *ICCV Workshop on 3D Representation for Recognition (3dRR-09)*, 2009. → pages 17

[52] L. Xu, E. Oja, and P. Kultanen. A new curve detection method: randomized Hough transform (RHT). *Pattern Recognition Letters*, 11(5):331–338, 1990. ISSN 0167-8655. → pages 11

[53] H. Yuen, J. Illingworth, and J. Kittler. Detecting partially occluded ellipses using the Hough transform. *Image and Vision Computing*, 7(1):31 – 37, 1989. ISSN 0262-8856. → pages 11

[54] H. Zeng, X. Deng, and Z. Hu. A new normalized method on line-based homography estimation. *Pattern Recognition Letters*, 29(9):1236 – 1244, 2008. → pages 12, 67

[55] S. Zhang and Z. Liu. A robust, real-time ellipse detector. *Pattern Recognition*, 38(2):273–287, 2005. → pages 11