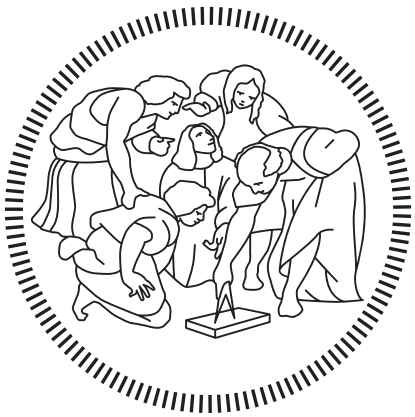
POLITECNICO DI MILANO
School Of Industrial and Information Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in
Automation and Control Engineering



Single View Ball Tracking and 3D Trajectory Reconstruction in Basketball Videos

Supervisor:

PROF. VINCENZO CAGLIOTI

Master Graduation Thesis by:

LUCA PIROTTA
Id - 864602

Academic Year 2017-2018

"The measure of who we are is how
we react to something that doesn't go our way."

– Gregg Popovich

ACKNOWLEDGMENTS

To Prof. Vincenzo Caglioti for giving me the opportunity to work on this project, and for allowing me to combine my passion for sports with engineering.
To my Family, for everything.
To my friends and my team, the Walruses, for all the adventures, the laughs, and the affection.

CONTENTS

Abstract xi
Estratto xiii
Preface xv
1 STATE OF THE ART 1
1.1 Computer Vision in Sports: A General Overview 1
1.1.1 Detection, Recognition and Tracking 1
1.1.2 The Challanges in Sport 5
1.1.3 Camera Calibration in Sport Scenes 6
1.1.4 Ball Tracking 7
1.1.5 Computer Vision Impact in Sports: Current Industrial Applications 8
1.2 Current Solutions in Single Camera Detection and Tracking 10
2 VIDEO STABILIZATION 13
2.1 Homography 13
2.1.1 Transformation Hierarchy 15
2.2 Projective Transformation Estimation 17
2.2.1 DLT Algorithm 18
2.3 Point Feature Extraction and Matching 20
2.3.1 Feature Extraction: FAST Features Algorithm 21
2.3.2 Feature Matching: Fast Retina Keypoints (Fast Retina Keypoint (FREAK)) Descriptor 22
2.4 Estimate Transform from Correspondances 24
2.5 Video Stabilization Results 26
3 CAMERA CALIBRATION 27
3.1 General rules 27
3.1.1 Camera Model 28
3.1.2 Camera Parameters 28
3.2 Five-Points Geometry Based Camera Calibration 30
3.2.1 Assumptions and Reference System Establishment 30
3.2.2 Camera Parameter Estimation 33
3.2.3 Implementation in a Basketball Scene 36
3.2.4 Court Identification 38
3.2.5 Line Detection 40
4 BALL DETECTION AND TRACKING 43
4.1 Moving Object Detection 43
4.1.1 Ball Detection with Gaussian Mixture Model 45
4.1.2 Ball Candidate Filtering 47
4.2 Trajectory Based Ball Tracking 48
4.2.1 Trajectory Generation 48
4.2.2 Track Analysis and Processing 50

5 3D TRAJECTORY EXTRACTION 53
5.1 Real World Ball Position from Geometry 53
5.2 Real World Ball Position from Physics 56
6 TRAJECTORY BASED APPLICATIONS 61
6.1 Shooting Position Extraction 61
6.2 Make or Miss Detection 61
6.3 Automatic Shot Chart Generation 62
6.3.1 Rebound Position Estimation 63
7 CONCLUSION AND FUTURE DEVELOPMENTS 65

BIBLIOGRAPHY 67
A APPENDIX: 71
A.1 Kalman Filter: Basics and Equations 71

LIST OF FIGURES

Figure 0.1 Hawk-Eye Technology application xvi
Figure 0.2 NBA SportVU Traking System xvi
Figure 1.1 Canny edge detector 2
Figure 1.2 Object recognition applicatin 3
Figure 1.3 Object recognition applicatin trained on dog species 3
Figure 1.4 Object recognition error 4
Figure 1.5 Kalman filter tracking example 5
Figure 1.6 Tennis Court Line Detection 7
Figure 1.7 Point Correspondence Homography 7
Figure 1.8 Ten Yard Line 8
Figure 1.9 Tracking results [5] 12
Figure 2.1 Video stabilization 13
Figure 2.2 Hierarchy of Projective Transformation 17
Figure 2.3 Homography from Pts correspondaces 20
Figure 2.4 FAST Corner Detection 22
Figure 2.5 Decriptor Extraction 23
Figure 2.6 FREAK Decriptor 24
Figure 3.1 Pinhole Camera Model 28
Figure 3.2 Camera Intrinsic Parameters 30
Figure 3.3 Camera Position and Rotation Angles 31
Figure 3.4 Camera and world refence frames 32
Figure 3.5 Vanishig points from court lines 34
Figure 3.6 Vanishing Line 35
Figure 3.7 Angle ω represented on ground plane 36
Figure 3.8 Camera Position on ground plane 37
Figure 3.9 Projection of noncoplanar point 38
Figure 3.10 Birds eye view of the detected camera position 38
Figure 3.11 Automatic court line and points algorithm flowchart 39
Figure 3.12 Court Mask 39
Figure 3.13 Edge Detection and Hough line and points extraction 41
Figure 4.1 Tracking Algorithm Flowchart 43
Figure 4.2 Gaussian of 3 distribuitions 46
Figure 4.3 Ball detection X and Y Candidate plot 49
Figure 4.4 Detected Parabolas in Y-Candidate Plot 50
Figure 4.5 Processed X and Y Trajectories 51
Figure 4.6 Detected ball over frames 51
Figure 5.1 Camera Model 53
Figure 5.2 Perpendicular Trajectory 54

Figure 5.3 Secants of a parabola 54
Figure 5.4 Secants of a parabola on a real case 55
Figure 5.5 Backprojection of a Line 56
Figure 5.6 Extracted 3D trajectory 59
Figure 6.1 Extracted Shot Chart in a correct case 62
Figure 6.2 Trajectory of a Missed Shot 62
Figure 6.3 Basketball Shot Chart 63
Figure 6.4 Statistical Rebound Position Estimation 64

LIST OF TABLES

Table 1.1 Example of a main CV application in today professional sport. 9

LISTINGS

ACRONYMS

DLT Direct Linear Transform

SVD Singular Value Decomposition

FAST Features from Accelerated Segment Test

FREAK Fast Retina Keypoint

GMM Gaussian Mixture Model

RGB Red, Green, Blue

HSV Hue, Saturation, Value

ABSTRACT

In recent years machine-vision techniques have revolutionized the way sport is analysed, televised and judged by referees.
In the NBA companies such as SportVU or Second Spectrum have managed to provide profession analysts with data, delivering huge potential for advanced analysis.
To do so, a muti-camera system is present in each of the 29 NBA arenas, but the price of this service is far beyond the budget of most of European teams, not to say amateur teams.
In this thesis an algorithm for tracking the ball with the use of a single camera is presented.
A system to automatically track the ball in a long-shot video is shown.
The proposed system detects and tracks the ball from its motion between frames.
A preliminaty video stabilization step is carried out to minimize the errors due to camera motion.
A candidate trajectory is generated from the detected ball candidates in each frame and a Kalman filter based approach extracts the final ball trajectory.
With the aid of camera calibration, thanks to geometry properties, the approximate 3D ball trajectory is visualized from the detected 2D trajectory.
The extracted 3D trajectory and the applications to trajectory-based tactics analysis may help coaches and players in game analysis.

ESTRATTO

Negli ultimi anni le tecniche di machine-vision hanno rivoluzionato il modo in cui lo sport è analizzato, trasmesso e arbitrato.
Nell'NBA aziende come SportVu e Second Spectrum sono riuscite a fornire agli analisti numerosi dati, rendendo disponibile un potenziale enorme per l'analisi avanzata.
Per fare ciò, un sistema multi-camera è presente in ognuna delle 29 arene NBA, ma il prezzo di questo servizio è oltre il budget di molti dei club europei, per non parlare degli amatori.
In questa tesi è presentato un algoritmo per il tracciamento della palla con una sola telecamera.
E' presentato un sistema per tracciare automaticamente la palla in un video di una sequenza di tiro dalla distanza. Il sistema proposta segue la palla attraverso il suo movimento tra i frame del video.
Un passo preliminare di stabilizzazione del video è proposto per minimizzare gli errori dovuti al movimento della telecamera.
La traiettoria è generata da tutte le possibili palle trovate nei frame e la traiettoria finale è estratta grazie all'aiuto del filtro di Kalman.
Con la calibrazione della telecamera e grazie alle proprietà geometriche, la traiettoria in 3D della palla è visibile dai suoi punti in 2D.
Questa traiettoria in 3D e l'analisi tattica basata su di essa potrebbe aiutare allenatori e giocatori nell'analisi di situazioni di gioco.

PREFACE

Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. The ultimate goal is to automate the methods for acquiring, processing, analyzing and understanding images in the same way as the human visual system does. These methods extract high-dimensional data from the real world in order to produce numerical or symbolic information.
Although is still far from the capabilities of the human vision system, computer vision is making fast progress in areas of pattern recognition, feature extraction and image classification.
A part from the academic research world, computer vision is becoming more popular and many industries are employing its wide range of application to solve problems in areas of medicines, military, sports, photography, and many other fields.
In recent years the strong demand to apply computer vision to perform the analysis in sports has greatly increased. Nowadays massive amount of video footages are available, and videos of sporting events are of special interest because they can provide differnt kind of data. The most cutting edge computer vision application it's for sure the in-game review of ambiguous referee decision. System such as Hawk-Eye [22] in tennis, to cite one of the most famous, record in three dimension the ball trajectory. This can shows exactly where the ball hit the ground and has given athletes and referees an important tool to correct errors. Even if Hawk-Eye is not infallible, its accuracy is within 3.6 millimetres, it is generally trusted as an impartial second opinion. Since 2012 also soccer has benefited from this system and the Goal-line technology have been present and used in all major leagues.
In addition to this application sport analysts, scouts, managers and coaches require sophisticated analysing systems to take sports to a higher level. This need to evaluate team and players in order to improve their performance has driven researchers into the challenge of improving sports analytics.
In basketball advanced statistic analysis is making great strides every year. While common audience are satisfied with a superficial analysis of game data, coaches and scouts are interested in a more deep study of all game situations. The most significant basketball events are caused by the player-ball interaction, thus detection and tracking of ball/player provide useful information.
In professional basketball NBA (National Basketball Association) the information is gathered by a multi-camera technology. The six cameras give the exact position of the player, ball and referees 25 times

per second. This tracking data is later used by teams to provide new insights and analysis about the game.
Such a system is very expensive and data are unavailable to pubblic. Many teams in Europe can't afford this type of deep study and must rely on video footage taken from broadcasters or even from the stands in minor leagues.
With a single camera player identification is still possible and has good performances. The ball detection though is still very difficult from a single view. In this thesis the problem of ball detection and tracking has been treated, with the aim of achieving a cost effective but accurate result.

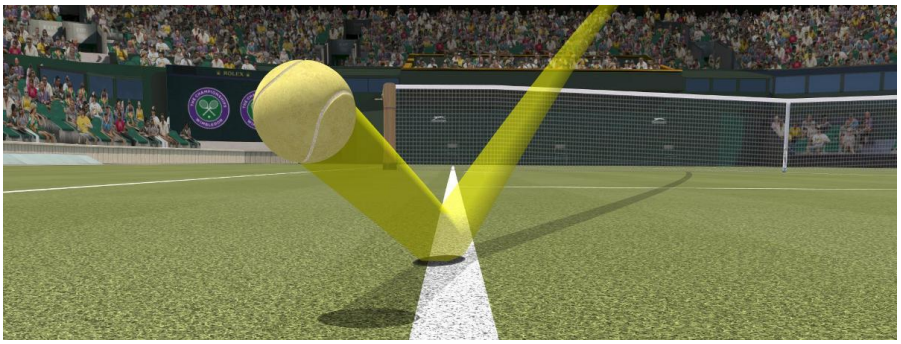


Figure 0.1: Hawk-Eye Technology application. Image shows the recorded 3D trajectory of a ball used for check the referee decision, or make the viewer more involved.

Figure 0.2: Overview of the NBA SportVU Traking System.

OBJECTIVE OF THE THESIS

The aim of this thesis is to apply computer vision techniques in the sports of basketball. The main objective is to reconstruct the three dimensional ball trajectory from a single-view video of a shot.
The solution must be simple, implementable and accurate enough in order to allow the extraction of the ball position. The video used for the analysis can either be a broadcast or a handled video from the audience.
A preliminary step of video stabilization and camera calibration is fundamental to achieve a good result. In particular calibration is carried out with just 5 points and with the help of geometry, giving the system a wide range of applicability.
An algorithm to automatically track the ball is developed in order to find the basketball trajectory in the image. This trajectory is converted in 3 dimension using the calibrated camera.

In this thesis, at first, the state of the art concerning ball detection and tracking, as well as the camera calibration, will be analysed and discussed.

The adopted methods for video stabilization, automatic line detection and point extraction will be discussed.

Calibration and ball-tracking algorithm, and how they are applied to retrieve the 3D trajectory, is shown in detail

Finally a comment on how the method can be used to extract statistics will be analysed

STRUCTURE OF THE THESIS

The thesis is divided in 7 chapters:

1. In the first chapter the state of the art regarding camera calibration, detection and tracking of ball in single view sport scenes will be analysed.

2. In the second chapter an overview on the math and geometry used in the thesis will be shown. Afterwards all the tools necessary for video stabilization will be introduced, and lastly video stabilization results will be analysed.

3. In the third chapter camera calibration problem is proposed and the employed algorithm is presented and evaluated.

4. In the fourth chapter tha ball tracking algorithm is treated in detail and its results will are shown.

5. In the fifth chapter will be displayed both the geometrical and physical model used to extract the 3D trajectory from the detected 2D trajecotry.

6. In the sixth section some applications are shown, as well as some potential applications.

7. In the final part, possible improvements and future implementation will be discussed.

1 STATE OF THE ART

In this chapter, the state of the art concerning object tracking in sports, detection and camera calibration will be analysed and reported.
At first a brief introduction to the problem of tracking, detection and calibration will be presented. Then the focus will be on how tracking of players and ball and motion analysis are being applied in today commercial application of computer vision. Then a look of how these techniques are still being further developed in recent studies will be presented. Finally, an overview on tracking algorithms from single view will be discussed.

1.1 COMPUTER VISION IN SPORTS: A GENERAL OVERVIEW

1.1.1 *Detection, Recognition and Tracking*

The computer vision terms object detection and object recognition are often used interchangeably. Another term, that frequently goes along with detection and/or recognition is object tracking. These three branches can collaborate to make a more reliable application but how they are distinct and how they relate one to another? Is tracking a direct consequence of detection?
Looking at a dictionary a distinction between the 3 can be made:

- Detection: *"The act or process of discovering, finding, or noticing something."*
- Recognition: *"The act of knowing who or what someone or something is because of previous knowledge or experience."*
- Tracking: *"To follow or watch the path of something or someone."*

Semantics aside, by further analysing the algorithms associated with each one of the three processes can make an even clearer discrimination. The objective is to have a basic idea about what the algorithms are designed to do.

▷ Detection

From the definition, it is straightforward to get the basic idea: detection notices the something *is* there. That 'something' that has to be

discovered may be anything.
Hence the goal of object detection is to acknowledge the presence of an object within an image (or a video frame) by inspecting the pixels of the image. To be able to separate a particular subset of pixels, the object, from a larger set, representing the background. Even though the main objective is the same, video and photos are treated differently.

Since phots are static, motion cannot be used to detect photo's objects. From a real-life photograph is often difficult to know the exact boundaries of an object. Edge detection methods (i.e. Canny Edge detection, 1.1) can help determine item is scenes. Edges define boundaries and can be found by looking at changes in grayscale levels across an image. Knowing edges allows to detect simple and even complex overlapping objects.

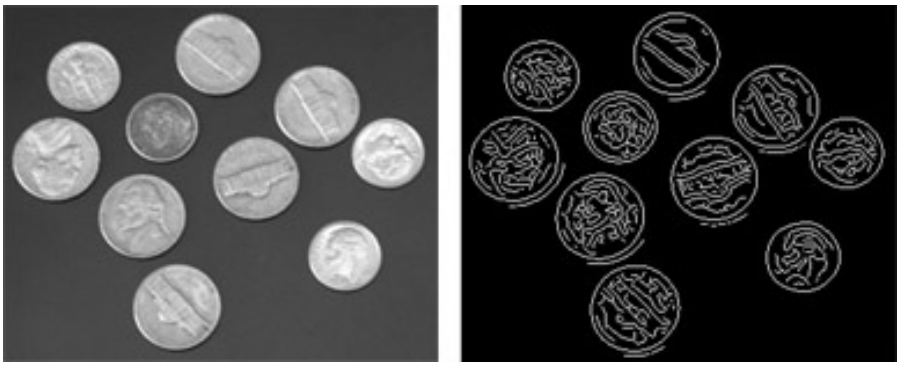


Figure 1.1: Image segmentation using the Canny method.

In video frames motion helps detection. The main behind detection in a video is to check whether something shows up in a frame that wasn't in the previous one, this is done by pixel inspection. An algorithm that notice this pixel difference can be implemented and has the ability to register that as a detection. Example of detection techniques for video can include background subtraction methods (a popular way to create a foreground mask) such as MOG (meaning mixture of Gaussian) and absdiff (absolute difference).

▷ Recognition

Based on what is assessable from definition, object recognition is a process for identifying the nature of an object. In other words recognition answer the question: *what's there?* Recognition applications can be based on matching, learning, or pattern recognition algorithms with the goal being to classify an object. Therefore the question is more specific: *what is the object?*. Recognition (algorithm knows what the object is) is substantially different from detection (algorithm notice presence of something).

How is possible for the algorithm to know what object is present? A computer cannot be taught to differentiate from dogs species, to cite a common example. Is impossible to write a program detailing the nature of a Golden Retriever from any other species. So, the solution is to employ programs that learn by themselves (this gives the computer *previous knowledge or experience*).

These are algorithms that can learn from experience and regognize an object precisely are called machine learning algorithms. The way an algorithm acquires knowledge about something (i.e. dog species) is through training data. A suitable training data to understand dog breeds may consist of tens of thousands of images of various species of dog; the algorithm extract visual features from an image, then use those features to associate one image, an unknown image seen for the first time, with another that has previously "seen" during its training. In this way learns to recognize different kinds of dogs. As an obvious consequence if an algorithm has never been trained with images labelled as "Golden Retriever" it will not be able to recognize that particular breed. On the other hand, it may still be smart enough to perceive the image as a dog. Example of the application can be seen in 1.2 and 1.3. There's a confidence level associated with recognition 1.2 and here the algorithm can regognize 3 classes: dog, bike and car. In 1.3 dog breed can be assessed.

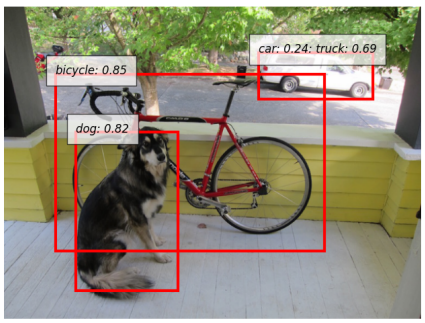


Figure 1.2: A recognition application trained to recognize simple objects.

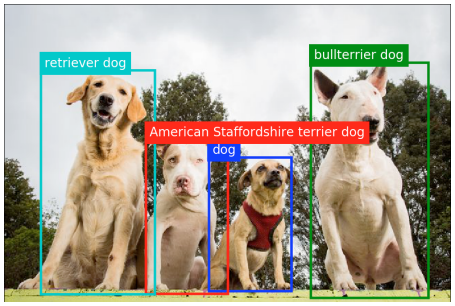


Figure 1.3: Another recognition application trained on dog species.

But object recognition doesn't always perform with such reliable accuracy. 1.4 present an example of misdetection.

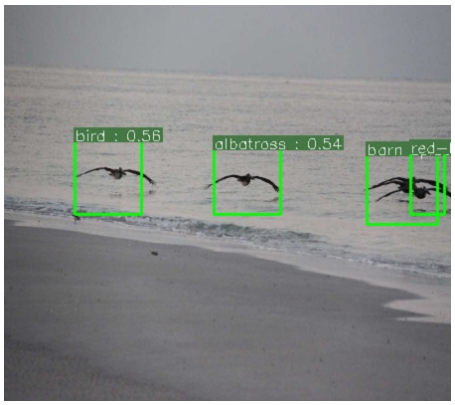


Figure 1.4: Example of application trained on birds species that is uncertain about objects in the right part of the image.

▷ Tracking

A tracking algorithm wants to know where something is headed. An ideal tracker is meant to follow the object of interest wherever it will go inside the image. The goal is to keep watch on something in its path in successive video frames. Tracking is often built in collaboration with detection and recognition.

Tracking always handles frames in relationship to one another, in this way there's a location history of the object, which allows us to know how its position has changed over time.
A Kalman filter A.1, a set of mathematical equations, can be used to determine the future location of an object. By using a series of measurements made over time, this algorithm provides a means to estimating past, present and future states. State estimation is useful for tracking and in many cases there is necessity to predict the future states, meaning an object's next position before it even makes it. Prediction come handy when object gets obstructed and if the ultimate goal is to maintain the identity of an object across frames, having an idea of the future location of the object helps to handle occlusions. Occlusion is where an object gets temporarily blocked. A robust tracking algorithm can handle the temporary obstruction and maintain its lock on the object of interest 1.5. The hardest part of tracking is in fact making sure the algorithm is locked onto the same thing so that tracking doesn't get lost. This is a big problem when more than one object is present in the video. To sum up:

- Object detection can notice that something is there.
- Object recognition techniques can be used to know what that something is (label an object).
- Object tracking can enable us to follow the path of a particular object.

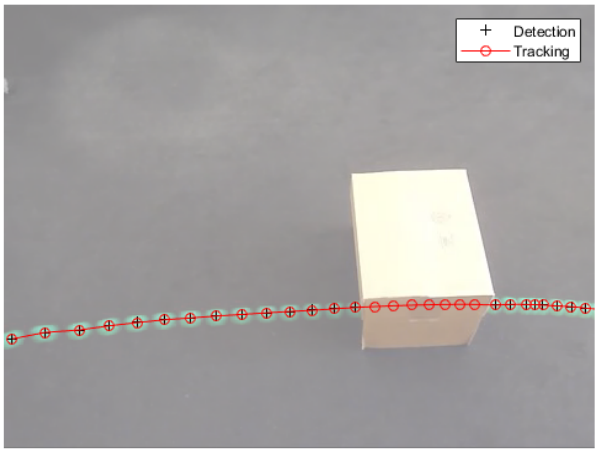


Figure 1.5: Example of application of a Kalman Filter based tracking algorithm confronted with a detection one. It can be seen that tracking can estimate ball position even when it is occluded.

1.1.2 *The Challanges in Sport*

The world of sports involves fast and accurate motion. Ball and player tracking is fundamental to analyse a sports scene where athletes interact with the ball. This kind of motion is not only difficult for athletes to master, but can be also challenging for coaches and trainers to analyse, and for audiences to follow.

The nature of most sports makes almost impossible to extract data by the use of sensors fixed to players or their equipment. This makes way for a set of opportunities for the application of computer vision to help the athletes, coaches and audience. Hence, Computer vision plays a key role in the world of sports. Major amount of sport activities carried out at any level, amateur or pro, involve a ball. Across many sports, when computer vision methods are applied, a crucial ability is to be able to keep track of where the ball is.

Modern muti-camera systems can keep track of the ball in almost every sport. Because of that, the bulk of current commercial applications of computer vision in sport rely on multi-camera systems to capture motion of objects in a game. These systems require special costly hardware and are difficult to achieve in the reality. Another diffused research topics include analysis of spatial movement of a group of players for application like: coaching in team sports, identifying key stages in a game for highlight extraction or automatic control of broadcast cameras.

No matter what the purpose of the video analysis is, all of computer vision application involving one or more cameras require a preliminary step: Camera Calibration.

1.1.3 *Camera Calibration in Sport Scenes*

During the broadcast of sport events, video sequences are processed for a wide variety of purposes such as mosaicking or visual object insertion to cite a few in addition to tracking. Some of these tasks require a highly precise calibration. The objective is to find the camera pose consisting of: position and orientation of the camera and its intrinsic parameters (focal length, pixel size and the principal point). Moreover, if the calibration must be extraordinarily precise estimation of the lens distortion is likely to be required. At sports events cameras are usually mounted on fixed tripods and are previously calibrated by means of standard off-line calibration methods to extract the intrinsic parameters. Only focal length is unknown, since it can be modified by the camera operator by changing the camera zoom.

Under the assumption that the camera can be calibrated from a model of the court some algorithms have been implemented [9]. The lines tracing the sport court boundaries and zones can be used to calibrate the camera. The court is a planar surface with known and standardized dimensions and some number of court lines (usually in white colour) can be extracted. If a minimum number of court primitives, i.e. the lines, are visible in the scene, then the homography relating the image plane to the actual court model can be found (see 1.6 and 1.7 for some examples). From this homography camera pose can be recovered. These method has been successfully applied to tennis and soccer [23] [27] [9].

It would be more flexible to achieve calibration from a single image with the less corresponding points possible. From a single image, the most diffused calibration technique is based on vanishing points or vanishing lines [29] [7]. From at least 6 corresponding points, whose coordinates are known in both the 3D real world and the 2D image, the camera projection matrix can be computed directly by solving a set of linear equations [15].

In the recent past so-called minimal algorithm have been found. Those method achieve camera calibration with a number of point smaller than 6. Using 5 points coming from two orthogonal 1D objects, with at least three collinear points each, focal length and orientation of a zooming camera can be found [21]. By solving systems of multivariate polynomials there are algorithms than can work with less than 5 points [3]. Although these systems are reliable, minimal and fast they require computational time, return more than an unique solution and a strong background in computer algebra is required to

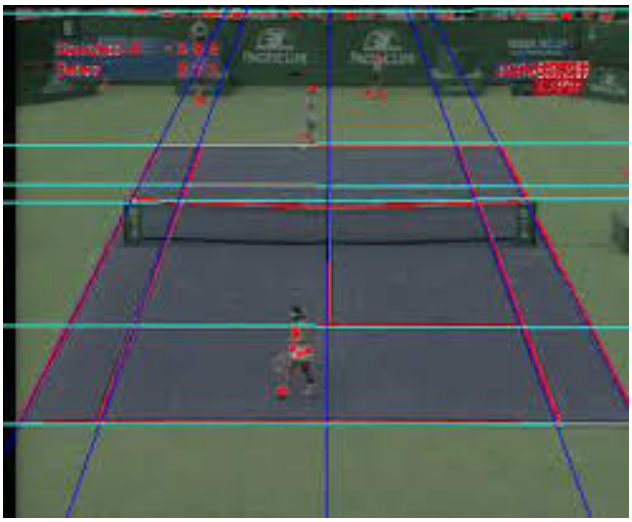


Figure 1.6: Example of primitives extraction from an image of a tennis court.

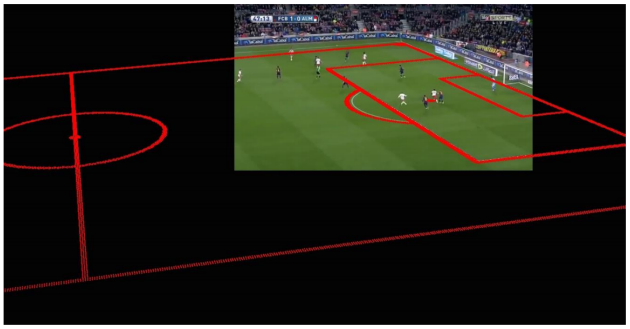


Figure 1.7: Homography estimation between court model and detected image points.

understand them. So they are of troublesome comprehension, especially for nonexperts.

Whatever calibration technique is used one of the most common consequent application to calibration, if the background is of uniform color, is the addition of graphic as if it was drawn on the field. The 10 yard line addition in american football broadcast has revolutionised the way the game is percieved from television [28]. Figure 1.8 shows an example of graphics applied to an NFL game thanks to calibration.

1.1.4 *Ball Tracking*

As stated before, tracking the ball is fundamental. The first commercialy available multi-camera system was developed by Hawk-Eye [22] for tracking balls in 3D in tennis and cricket. Initially the system was based on broadcast cameras, now it is deployed with 6 to 10 cam-

Figure 1.8: Example of a virtual graphic overlaid on a football pitch.

eras around the court to capture live images. They are static, easy to calibrate and with a high frame rate. The motion of a tennis ball once it his it can be retrieved by laws of physics, and there is a lot of prior knowledge about the appearance if the ball and the area where it needs to be tracked. The system first identifies possible ball candidates in each camera image, by identifying ball shaped elliptical region within the expected ball size range. Then it links the ball tracks along multiple frames, and then possible paths are matched between cameras to generate a 3D path [22].

More recently, soccer has implemented systems to check whether the ball has crossed the goal line. Again one of the main developer is Hawk-eye; their system relies on 7 cameras and it is installed in more than 108 soccer grounds [11].
These are just some example of how computer vision is currenty used today in professional sport.

1.1.5 *Computer Vision Impact in Sports: Current Industrial Applications*

The use of computer vision and graphics technologies to provide insight of sports events has been a part of the evolution of sports TV. Over the years developments have been embraced by viewers, broadcasters and insiders. However, from the time that technology first came into being it has allowed coaches, player and refereeing decisions to be examined in detail, inevitably leading to debate. Examples of comments made on those technologies can be found in [8].
Player and ball tracking technology has revolutionised training and scouting for players in sports such as football and basketball. Some insights into how the data is used in basketball can be found in articles of NBA insiders.[1].
The table 1.1 presents an overview on computer visions system used in today sport.

Company	Application Area	Cameras	Diffusion
STATS SportVU	Player and ball tracking for sports analytics in Football, Basketball etc.	3 to 6 HD cameras at fixed position. (Automatic Tracking)	Used by all 30 NBA Teams and many UEFA Champions League Teams
ChyronHego	Player and ball tracking for coaching/training and broadcast in sports	6 HD Cameras. (Requires Human Operator Intervention)	Used by all 20 Premier League Teams, diffused in Bundesliga, La Liga, UEFA Champions League, FIFA World Cup and also in MLB.
Hawk-Eye	Broadcast Enanchement and referee assistance	Up to 10 fixed Cameras for Tennis and Cricket. 7 Cameras for GLT in Football (Automatic Tracking)	Tennis: Since 2006 used in over 80 ATP tournaments around the world and at all Grand Slam events. Football: Used by the Premier League, Bundesliga, Eredivisie, Serie A and FIFA.
Second Spectrum	Player and ball tracking for sports analytics in Football, Basketball and Advanced Statistics Delivery	6 HD cameras at fixed position. (Automatic Tracking)	Official tracking data provider of the NBA, also used by some MLS and UEFA Football clubs, and many sport journals like ESPN and Sports Illustrated.

Table 1.1: Example of a main CV application in today professional sport.

1.2 CURRENT SOLUTIONS IN SINGLE CAMERA DETECTION AND TRACKING

Almost all commercial applications utilize multiple view high-speed cameras covering each game field zone. Applying similar systems in all leagues and sports would require an unrealistically large number of calibrated cameras and money. This is why research also focuses on trying to achieve same results from broadcast footage captured by a single camera.

In this section the largest research areas at the moment will be discussed. These fields are very vast, the focus will be on detection and tracking of players and balls. Since sport is a human centred activity, the first step in automatic analysis is to locate and segment each person of interest, and where it is possible, the further step is to follow each tracked person over time. Several challenges influence these tasks. The body posture makes hard to use a standard human/pedestrian detector because in an active sport it may change in a swift way. In any contact sports, which can be either team sports or a single sport, occlusion is another considerable problem. Another big problem is the way the sport is captured by cameras. Sports videos range from close-up views to wide views including spectators at large stadiums, presenting a wide range of different scenarios.

- ▷ Players Detection
  - Detection with static cameras:
    Using static cameras it is possible to build a background model and detect players with simple methods like image differencing and background subtraction [24].

  - Detection with Moving Cameras:
    - Colour Based Background Elimination:
      If the courts have uniformly coloured surfaces similar approaches can be applied with moving cameras using a colour-based elimination of the ground [20]. These methods are fast and suited for real-time performance. However, they lack precision since noise and missed detections happens due to the many moving objects in a scene. Similar colours in foreground and background, and effects such as changing lighting conditions and shadows are another cause of problems. Although many methods for post-processing of foreground images have been proposed, it remain an open research challenge [24].

    - Trained Classifiers for Pedestrian Detection:
      Applying trained classifiers has become the most pop-

ular strategy for pedestrian detection, so it has been implemented also in sports. However, sports players generally have much larger variations in pose, the viewpoint and distance between pedestrian videos and sport videos might change significantly. The AdaBoost algorithm for training a classifier with HOG features, [10] shows that it is important to train the system with samples from the same video types. In the article the detector trained on standard pedestrian images is applied on an Australian Football footage and performs very well. With a similar approach, the AdaBoost have been applied to basketball scenes. Here the results are better but the false positive rate is too high to consider the method suitable fot the task [16].

* Detect Players by Shirt Numbers
  This approach consist in classifying each object based on a unique feature, like the jersey number. These features are often only visible in a subset of frames, hence the classification can be combined with temporal tracking of the player to keep track of the identity [19].

▷ Ball Detection

Detection of the ball is particularly challenging due to fast motion and the small size of a ball compared to both humans and field sizes. Most ball detection algorithms get acceptable results by detecting moving objects and sorting these ball candidates on area, colour and shape [13] [31] [4]. However, often a ball is partly or fully occluded by the players, or moving too fast. In both those cases a pure detection algorithm will not be useful. Combination of detection and tracking instead, allow for a better recognition of the ball position based on its trajectory.

▷ Ball Tracking

Constructing trajectories of a ball is often of interest. This can be used for analysing shot types, play dynamics, and event detections or for virtual replays. In sports like tennis, where the ball is fully visible in most frames, it is possible to construct tracklets, which can be connected in a later association step [32]. Tracking the ball is harder in team sports. Several players can occlude the ball with their body, when a player is in possession of the ball with their hands or feet. Combining knowledge on the positions of players and ball, the possession of the ball can be modelled and used in trajectory determination [30]. Many

physics methods have been proposed. Starting from a set of ball detections a physical model is fitted and the trajectory is generated [5]. Differently for what happens with players, the ball in many sports cannot be assumed to be moving on ground plane. These leads to many problems and to solve ambiguities the best way is still to segment the video and analyse particular situations.

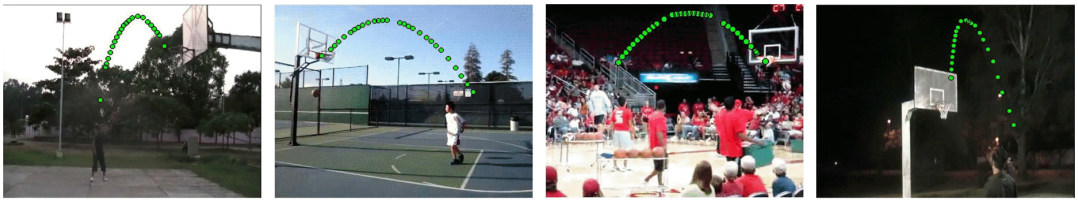


Figure 1.9: Results obtained in [5] by applying a physical model to determine ball trajectory

2 VIDEO STABILIZATION

Video stabilization algorithm can be seen in 2.1 and consist of 4 main steps:

1. Compute distinctive keypoints in both images (i.e. corners).
2. Compare the keypoints between the two images to find matches.
3. Use the matches to find a general mapping between the images (i.e. a homography).
4. Apply the mapping on the first image to align it to the second image.

To fully understand each of the above steps it is necessary to introduce some basic concepts.
This chapter provides a brief background about the homography transformation and feature extraction and matching. The last part will introduce the video stabilization algorithm used in this work.
More precisely, at first are displayed the homography and its mathematical representation. Then, a depiction on hierarchy of linear transformations is shown as well as a simple method through which the homography can be computed.
Afterward, feature detection and matching is discussed. An explanation on how interest point detectors are used in order to measure the image similarity and compute the homography between video frames will be given in the last part.

2.1 HOMOGRAPHY

The section deals with the theoretical aspects of the homography and its mathematical representation. Before talking about homographies, an introduction about homogeneous notation for 2D points, lines, and

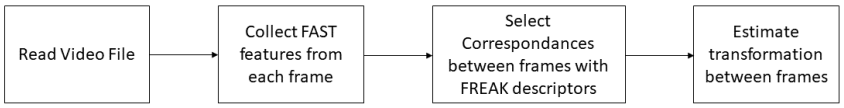


Figure 2.1: Video stabilization workflow.

the 2D projective plane is mandatory.

In a $\mathbb{R}^n$ a vector space, any pair of coordinates (x,y) represent a 2D that correspond to the vector $\mathbf{x} = (x, y)^T$ in $\mathbb{R}^2$. In the same way, any line in a 2D plane can be represented by three values (a,b,c), which in turn correspond to the vector $\mathbf{l} = (a, b, c)^T$. Additionally, since $ax + by + c = o$ and $(ka)x + (kb)y + kc = o$ represent the same line $\forall k \neq 0$, the two vectors $(a, b, c)^T$ and $k(a, b, c)^T$ are considered to be equal and related by a scaling factor k. The equivalence class of vectors based on this equivalence relationship is called *homogeneous* vector.
Exluding vector $(o, o, o)^T$, the equivalence classes of vectors in $\mathbb{R}^3$ makes the spaces known as *Projective Space* $\mathbb{P}^2$.
In the same way of lines also points have a representation in $\mathbb{P}^2$: the set of homogeneous vectors $(kx, ky, k)^T$ $\forall k \neq 0$ in $\mathbb{R}^3$ such that $(x_1, x_2, x_3)^T$ represents the point $(x_1/x_3, x_2/x_3)$ in $\mathbb{R}^2$.

Now that basic notion have been introduced the homography transformation can be defined. The homography, also called projective transformation, is an invertible linear transformation relating 2D homogeneous points and lines to the projective plane $\mathbb{P}^2$.

Definition 2.1. *A homography is an invertible mapping h from $\mathbb{P}^2$ to itself, such that three points x_1, x_2 and x_3 represented by homogeneous 3-vectors, lie on the same line if and only if mapped points $h(x_1)$, $h(x_2)$ and $h(x_3)$ lie on the same line as well.*

Therefore an homography can be represented by a 3x3 non-singular matrix **H** such that $x' = \mathbf{H}\, x$.

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{2.1}$$

H is homogeneous matrix with 9 entries and 8 degree of freedom, because only ratios of its element are important. If *(x,y)* and *(x′,y′)* are 2 dimesional inhomogeneous points in world and image plane respectively, the homography can be expressed as:

$$x' = \frac{x_1'}{x_3'} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{2.2}$$

$$y' = \frac{x_2'}{x_3'} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{2.3}$$

Each points correspondance brings 2 equations, therefore to estimate $\mathbf{H}$ at least $n \geqslant 4$ correspondances are necessary, with the condition that these four points are coplanar but not collinear.

2.1.1 *Transformation Hierarchy*

All n dimensional projective transormations form a group. This include all invertible n x n real matrices related by a scalar factor. The set is known as *Projective Linear Group*, and is expressed as *PL(n)*. Taking into account our case where $n = 3$, subgroups of PL(3) include the affine group, which is the subgroup consisting of matrices for which the last row is (0, 0, 1), and the Euclidean group, which is a subgroup of the affine group for which in addition the upper left hand 2×2 matrix is orthogonal. One may also identify the oriented Euclidean group in which the upper left hand 2×2 matrix has determinant 1. These subgroups are all related in hierarchical order, from the most specialized (isometries) to the most general one, projective transformations.

▷ Isometries

Also known as *Euclidean Transformations*, are a set in $\mathbb{R}^2$ that perserves the euclidean distance between points.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \varepsilon\cos\theta & -\sin\theta & t_x \\ \varepsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2.4)$$

with $\varepsilon = \pm 1$. As it can be seen from 2.4 the transformation implies a rotation and a translation. Isometries have 3 degrees of freedom: one for the rotation and 2 for the translation; they can be determined from 2 exact point correspondance.

▷ Similarities

Similarity is an isometry combined with a scaling. In fact it preserves the shape but not the point distances. Its matrix representation is very close to the one of an euclidean transformation.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s\cos\theta & -\sin\theta & t_x \\ s\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} s\mathbf{R} & \mathbf{t} \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2.5)$$

with s called isotropic scaling factor. Because of this term, similarity have an additional degree of freedom with respect to isometries. Nevertheless they can still be computed from just 2 point correspondance.

▷ Affine Transformations

Also known as *affinity*, this transformation is the composition of a non-singular linear transformation followed by a translation. An affine transformation does not necessarily preserve angles between lines or distances between points, though it does preserve ratios of distances between points lying on a straight line

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{t} \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.6}$$

with $\mathbf{A}$ being a 2 x 2 non-singular matrix. So, A has 4 degrees of freedom, which added to the 2 of the translation, results in 6 degrees of freedom. Hence, at least 3 point correspondaces are required.

▷ Projective Transformations

As introduced in section 2.1 a projective transformation can be defined as a non-singular linear transformation of homogeneous coordinates and has 8 degrees of freedom.

$$\begin{pmatrix} x_1{}' \\ x_2{}' \\ x_3{}' \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{t} \\ V^T & v \end{pmatrix} x \tag{2.7}$$

where $\mathbf{V} = (v_1, v_2)^T$. Projectivity matrix may be decomposed into a chain of more specialized transformations. In fact $\mathbf{H}$ may be rewritten as:

$$\mathbf{H} = \mathbf{H}_S \mathbf{H}_A \mathbf{H}_P = \begin{pmatrix} s\mathbf{R} & \mathbf{t} \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} \mathbf{K} & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} s\mathbf{I} & 0 \\ V^T & v \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{t} \\ V^T & v \end{pmatrix} \tag{2.8}$$

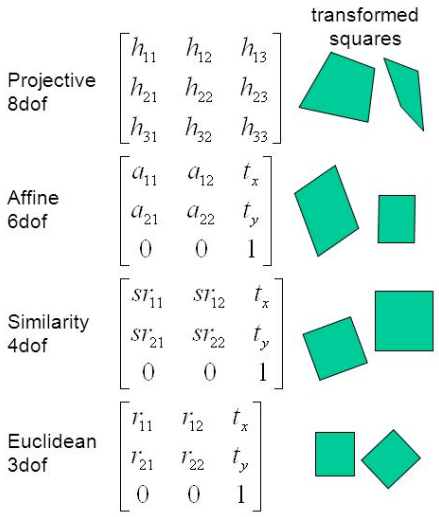


Figure 2.2: Hierarchy of Projective Transformation.

as it can be stated from previous sections, $\mathbf{H}_S$ is similarity with 4 degrees of freedom, $\mathbf{H}_A$ is an affinity without translation (so it only brings 2 degrees of freedom), and $\mathbf{H}_P$ is a particular transformation that moves the lined at infinity, called *elotion* and has 2 degrees of freedom.
Developing the product between matrices and taking into account that $v \neq 0$ the result $\mathbf{A} = s\,\mathbf{RK} + \mathbf{t}\mathbf{V}^T$ is a non-singular matrix and $\mathbf{K}$ is a normalized upper triangular matrix.
Adding up all the numbers, it can be verified that $\mathbf{H}$ has eight degrees of freedom. Therefore is verified that at least four exact non-collinear point correspondences are crucial to compute a projectivity matrix.

2.2 PROJECTIVE TRANSFORMATION ESTIMATION

The homography found with just four points is a minimal solution since there always is noise in the correspondences which makes them inexact. Therefore, the more point correspondances the better. To limit all the possible inaccuracies coming from measurement noise a constrained optimization of $\mathbf{H}$ is required.
There are two main classes of cost functions:

- Cost function minimizing algebraic error
- Cost function minimizing the geometric and statistical image distance

In case of estimating homography between two different viewpoints, the reprojection error for both images can be used as a cost function that needs to be minimized:

$$\sum_i d(x_i, \hat{x}_i)^2 + \sum_i d(x'_i, \hat{x}'_i)^2 \quad \text{s.t} \quad \hat{x}'_i = \mathbf{H}\hat{x}_i \quad \forall i \tag{2.9}$$

Optimazing cost function in Eq 2.9 is a maximum likelihood estimation problem that determines a set of perfectly matched point to point correspondance between $\hat{x}_i{}'$ and $\hat{x}_i$.

2.2.1 *DLT Algorithm*

When a set of n 2D point correspondances is given, with $n \geqslant 4$, the linear algorithm to compute $\mathbf{H}$ is called Direct Linear Transform (DLT) [14]. Altough the homography matrix is derived from $x' = \mathbf{H}x$, and x_i and x'_i are homogeneous vector, this does not automatically implies that the vectors x' and $\mathbf{H}x$ are equal. They have same direction but may have different magnitudes. As a consequence of the equal direction, the transformation equation can be expressed in terms of vector cross product:

$$x' \times \mathbf{H}\, x_i = 0 \tag{2.10}$$

From this, the homography matrix $\mathbf{H}$ can be found by solving a linear system with $x_i = (x_i, y_i, z_i)^T$ and $x'_i = (x'_i, y'_i, z'_i)^T$. Since (x'_i, y'_i) and (x_i, y_i) are both measured in image coordinate system, is assumed that both z_i and z'_i are equal to 1 ($z_i = z'_i = 1$).
Subsequently :

$$\mathbf{H}x_i = \begin{pmatrix} h_{11}x_i + h_{12}y_i + h_{13} \\ h_{21}x_i + h_{22}y_i + h_{23} \\ h_{31}x_i + h_{32}y_i + h_{33} \end{pmatrix} \tag{2.11}$$

and by simple mathemathic:

$$x' \times \mathbf{H}x_i = 0 \implies \begin{pmatrix} y_i'(h_{11}x_i + h_{12}y_i + h_{13}) - (h_{21}x_i + h_{22}y_i + h_{23}) \\ (h_{11}x_i + h_{12}y_i + h_{13}) - x_i'(h_{31}x_i + h_{32}y_i + h_{33}) \\ x_i'(h_{21}x_i + h_{22}y_i + h_{23}) - y_i'(h_{11}x_i + h_{12}y_i + h_{13}) \end{pmatrix} = 0$$

$$\implies \begin{pmatrix} 0 & 0 & 0 & -x_i & -1 & -y_i & y_i'x_i & y_i'y_i & y_i \\ x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & x_i'y_i & -x_i \\ -y_i'x_i & y_i'y_i & -y_i & x_i'x_i & x_i'y_i & x_i & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = 0$$

$$\implies A_i h = 0 \tag{2.12}$$

Only the first two rows of the 3 X 9 matrix $\mathbf{A}_i$ are linearly independent, therefore the 8 unknowns of $\mathbf{h}$ that form the homography matrix can be estimated from the first 2 rows. Matrix $\mathbf{A}$ has rank 8 and one-dimensional null-space that produce the solution $\mathbf{h}$, and this fact is not influenced by how many point correspondances are taken.
Obviously, the linear system coming from 2.12 will be over-determined for $n \geqslant 4$.

In fact, from $n \geqslant 4$ point correspondances 2.12 results in (2n-1) linear equations. Typically, this system does not have an exact solution except for the trivial one, h = 0, still an aprroximate non-zero solution can be found with the aid of a cost function as mentioned in the beginning of the chapter 2.2.
Undesired and useless solution $\mathbf{h}$ = 0 can be avoided by introducing an additional constrate on $\|\mathbf{h}\|$, e.g. $\|\mathbf{h}\|$=1.
It can be demonstrated that the this optimization problem is equivalent to the problem of finding the unit eigenvector of the $\mathbf{A}^\top\mathbf{A}$ that corresponds to the smallest eigenvalue of A. This is achieved by Singular Value Decomposition (SVD).

DLT algorithm is fast and easy to compute, but has the disadvantage that its homography estimation is dependent on the choice of the image coordinate system, which may be taken differently. In other words, using different image coordinate systems result in different homography estimates.
To get rid of this problem the point correspondances are first normalized into the same image coordinate system and then passed to the DLT algorithm.

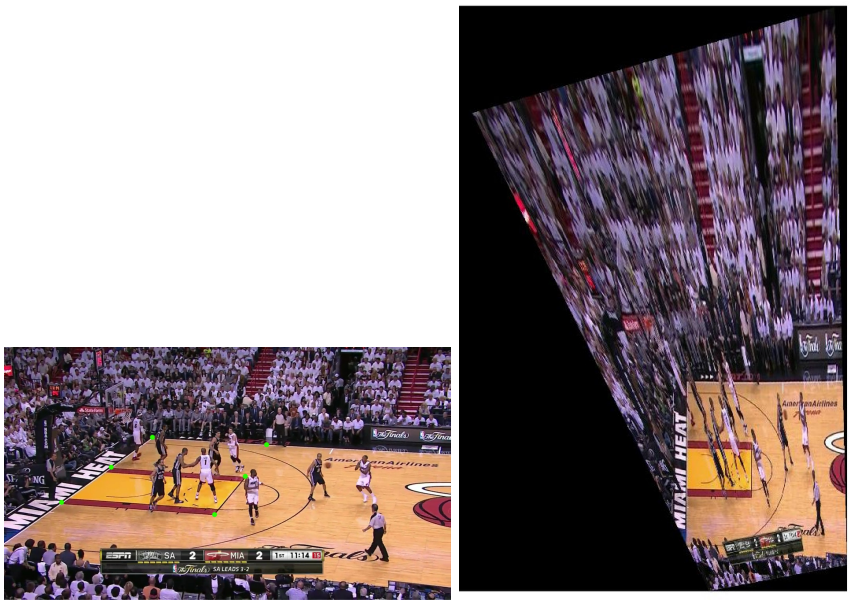


Figure 2.3: Homography from points correspondaces.

2.3 POINT FEATURE EXTRACTION AND MATCHING

If the objective is to find the homography between a pair of images, establishing a set of correspondances between the two is a key element. To achieve this, searching for specific patterns or specific features which are unique, which can be easily tracked, which can be easily compared is fundamental. In another words, the computer must find a distinctive element which can be compared across several images, a feature. In an image corners are considered to be good features.

A corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness. As said, corners are the important features in the image, and they are generally termed as interest points which are invariant to translation, rotation and illumination. Corners are only a small percentage of the image, but they contain the most important features in restoring image information. They can be used to minimize the amount of processed data for motion tracking, image stitching, building 2D mosaics, stereo vision, image representation and other related computer vision areas. The process of finding corners is called *Feature Detection*.

There are several feature detectors and many of them are really good, but they are not quick as Features from Accelerated Segment Test (FAST) [25] [26]. In fact FAST can be used in real time frame rate applications and when limited computational resources is necessary. In this thesis it is used for video stabilization.

2.3.1 *Feature Extraction: FAST Features Algorithm*

Here is presented a schematic description of the FAST methods:

- Feature Detection using FAST [25]
  1. Select a pixel p in the image which is to be identified as an interest point or not. Let its intensity be I_p.
  2. Select appropriate threshold value t.
  3. Consider a circle of 16 pixels around the pixel under test (see fig. 2.4).
  4. Now the pixel p is a corner if there exists a set of n contiguous pixels in the circle (of 16 pixels) which are all brighter than I_p + t, or all darker than I_p - t. n was chosen to be 12.
  5. A high-speed test was proposed to exclude a large number of non-corners. This test examines only the four pixels at 1, 9, 5 and 13 (First 1 and 9 are tested if they are too brighter or darker. If so, then checks 5 and 13). If p is a corner, then at least three of these must all be brighter than I_p + t or darker than I_p - t. If neither of these is the case, then p cannot be a corner. The full segment test criterion can then be applied to the passed candidates by examining all pixels in the circle.

     This detector in itself exhibits high performance, but there are several weaknesses:

     (a) It does not reject as many candidates for n < 12.

     (b) The choice of pixels is not optimal because its efficiency depends on ordering of the questions and distribution of corner appearances.

     (c) Results of high-speed tests are thrown away.

     (d) Multiple features are detected adjacent to one another.

- Machine Learning and Corner Detector [26]
  1. Select a set of images for training (preferably from the target application domain).
  2. Run FAST algorithm in every images to find feature points.
  3. For every feature point, store the 16 pixels around it as a vector. Do it for all the images to get feature vector P.
  4. Each pixel (say x) in these 16 pixels can have one of the following three states:

$$S_{p\to x} = \begin{cases} d, & \mathrm{I}_{p\to x} \leqslant \mathrm{I}_p - t \quad (\mathtt{darker}) \\ \mathrm{s}, & \mathrm{I}_p - t < \mathrm{I}_{p\to x} < \mathrm{I}_p + t \quad (\mathtt{similar}) \\ \mathrm{b}, & \mathrm{I}_p + t \leqslant \mathrm{I}_{p\to x} \quad (\mathtt{brighter}) \end{cases} \tag{2.13}$$

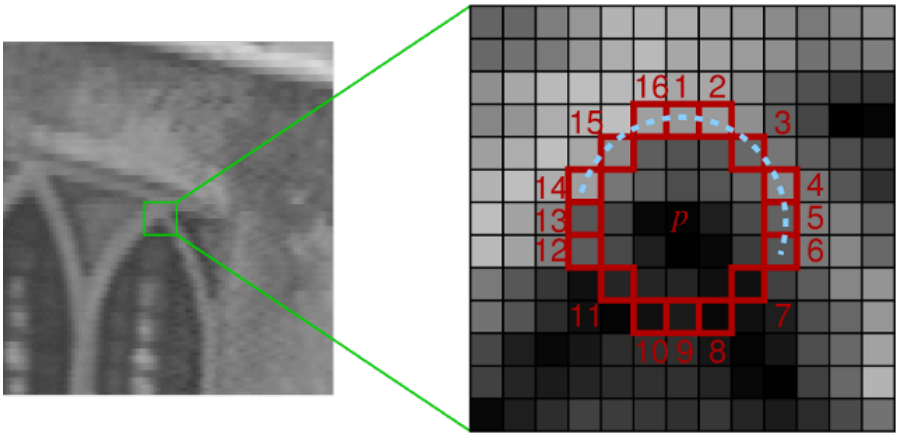


Figure 2.4: FAST Corner Detection from [25] [26].

5. Depending on these states, the feature vector P is subdivided into 3 subsets, P_d, P_s, P_b.
6. Define K_p boolean, which is true if p is a corner and false otherwise.
7. Use a decision tree classifier to query each subset using the variable K_p for the knowledge about the true class. It selects the x which yields the most information about whether the candidate pixel is a corner, measured by the entropy of K_p.
8. This is recursively applied to all the subsets until its entropy is zero.
9. The decision tree so created is used for fast detection in other images.

2.3.2 *Feature Matching: Fast Retina Keypoints (FREAK) Descriptor*

Section 2.3.1 dealt with the computation of distinctive keypoints in two images, this one will talk about matching the found keypoints by comparison between the two images.
Namely, given a small patch around a keypoint taken from the first image, and a second small patch around a keypoint taken from the second image, how can be determine if it's indeed the same point?
The problem summarises in finding the similarity between the two image patches by comparison. The resul must be robust with respect to rotation and translation.
This is where patch descriptors come in handy. A descriptor is some function that is applied on the patch to describe it in a way that is invariant to all the image changes (e.g. rotation, illumination, noise etc.). A descriptor is "built-in" with a distance function to determine the similarity, or distance, of two computed descriptors. So to compare

two image patches, first compute their descriptor and measure their similarity. The similarity is calculated by measuring the descriptor similarity, which in turn is estimated by computing their descriptor distance.
The common pipeline for using patch descriptors is (see also fig 2.5):

1. Detect keypoints in image (distinctive points such as corners).
2. Describe each region around a keypoint as a feature vector, using a descriptor.
3. Use the descriptors in the application (for comparing descriptors – use the descriptor distance or similarity) function.

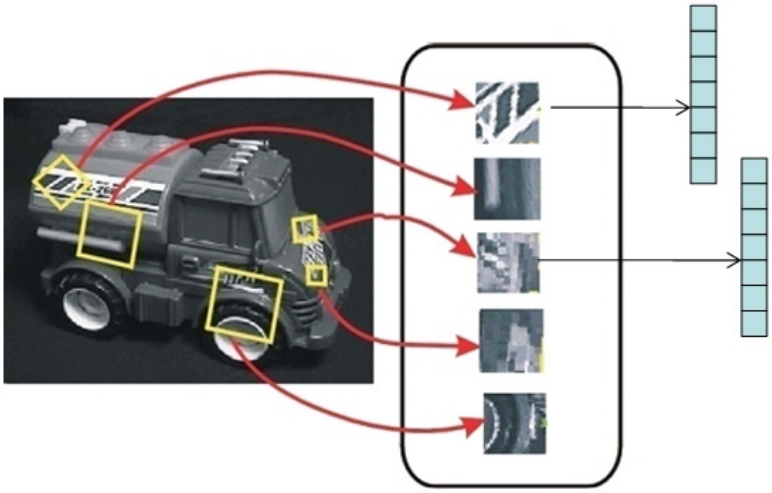


Figure 2.5: Extracting descriptors from detected keypoints.

Among the many type of descriptors available (i.e. HOG based descriptors such as SIFT[18] or SURF[2]), the one used in this thesis is FREAK descriptor.
A binary descriptor is composed out of three parts:

- A sampling pattern: where to sample points in the region around the descriptor.
- Orientation compensation: some mechanism to measure the orientation of the keypoint and rotate it to compensate for rotation changes.
- Sampling pairs: which pairs to compare when building the final descriptor.

FREAK suggests to use the retinal sampling grid which is also circular with the difference of having higher density of points near the center.

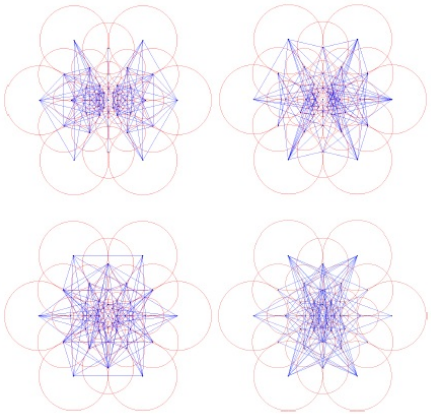


Figure 2.6: FREAK Sampling Pairs.

Each sampling point is smoothed with a Gaussian kernel where the radius of the circle illustrates the size of the standard deviation of the kernel. With few dozen sampling points, thousands of sampling pairs can be considered. FREAKs tries to learn the pairs by maximizing variance of the pairs and taking pairs that are not correlated. Interestingly, there is a structure in the resulting pairs – a coarse-to-fine approach which matches our understanding of the model of the human retina. The first pairs that are selected mainly compare sampling points in the outer rings of the pattern where the last pairs compare mainly points in the inner rings of the pattern. This is similar to the way the human eye operates, as it first use the perifoveal receptive fields to estimate the location of an object of interest. Then, the validation is performed with the more densely distributed receptive fields in the fovea area. The sampling pairs are illustrated in the fig. 2.6, where each figure contains 128 pairs (from left to right, top to bottom): FREAKS takes advantage of this coarse-to-fine structure to further speed up the matching using a cascade approach: when matching two descriptors, we first compare only the first 128 bits. If the distance is smaller than a threshold, we further continue the comparison to the next 128 bits. As a result, a cascade of comparisons is performed accelerating even further the matching as more than 90% of the candidates are discarded with the first 128 bits of the descriptor.

2.4 ESTIMATE TRANSFORM FROM CORRESPONDANCES

Once features have been detected and matched, the last step is to estimate the transformation between two images. To do so RANdom SAmple Consensus (RANSAC) [14] algorithm is applied. RANdom SAmple Consensus (RANSAC) is an iterative method to estimate parameters of a model by random sampling of observed data. It is a non-deterministic algorithm in the sense that it produces a reasonable

result only with a certain probability, with this probability increasing as more iterations are allowed. The RANSAC algorithm is essentially composed of two steps that are iteratively repeated:

1. Sample subset containing minimal data items is randomly selected from the input dataset. A fitting model and the corresponding model parameters are computed using only the elements of this sample subset. The cardinality of the sample subset is the smallest sufficient to determine the model parameters.

2. The algorithm checks which elements of the entire dataset are consistent with the model instantiated by the estimated model parameters obtained from the first step. A data element will be considered as an outlier if it does not fit the fitting model instantiated by the set of estimated model parameters within some error threshold that defines the maximum deviation attributable to the effect of noise.

The set of inliers obtained for the fitting model is called consensus set. The RANSAC algorithm will iteratively repeat the above two steps until the obtained consensus set in certain iteration has enough inliers. An advantage of RANdom SAmple Consensus (RANSAC) is its ability to do robust estimation of the model parameters, namely it can estimate the parameters with a high degree of accuracy even when a significant number of outliers are present in the data set. The main disadvantage of RANdom SAmple Consensus (RANSAC) is that there is no upper bound on the time it takes to compute these parameters (except exhaustion). When the number of iterations computed is limited the solution obtained may not be optimal, and it may not even be one that fits the data in a good way. In this way RANdom SAmple Consensus (RANSAC) offers a trade-off; by computing a greater number of iterations the probability of a reasonable model being produced is increased.

When applied to two consecutive frames RANdom SAmple Consensus (RANSAC) will search for the valid inlier correspondences in the 2 set of point correspondences. From these it will then derive the affine transform that makes the inliers from the first set of points match most closely with the inliers from the second set. This affine transform will be a 3-by-3 matrix of the form:

$$\begin{pmatrix} a_1 & a_3 & 0 \\ a_2 & a_4 & 0 \\ t_x & t_y & 1 \end{pmatrix} \tag{2.14}$$

where a define the scaling factor, shearing effects and rotation of the transform, while t is a translation. This transform can be used to warp the images such that their corresponding features will be moved to the same image location.

2.5 VIDEO STABILIZATION RESULTS

By applying the steps described in 2.1 all the input basketball video may be stabilized. If the video frame rate is high or if the camera does not undergo sudden and wide motion, the algorithm have excellent results. The core part of the images are well alligned. The only disadvatages are:

- The algorithm can only alter the imaging plane. Thus it is ill-suited to finding the general distortion between two frames taken of a 3-D scene
- Algorithm does not provide a panoramic possibility, hence if the camera motion is wide, it will not be able to perform for all frames.

Even though the affine transform is limited to altering only the imaging plane, in the case of a basketball video where the background plane does not change significantly between frames, the transformation will be able to correct camera motion.

3 CAMERA CALIBRATION

3.1 GENERAL RULES

The process of camera calibration is an essential step in many computer vision tasks. As introduced in 1.1.3 camera calibration is the process of estimating intrinsic and/or extrinsic parameters that control the relationship between the 2D image and the 3D information of the imaged object . Intrinsic parameters deal with the camera's internal characteristics, such as, its focal length, skew, distortion, and image center. Extrinsic parameters describe its position and orientation in the world. Knowing intrinsic parameters is an essential first step for 3D computer vision, as it allows you to estimate the scene's structure in Euclidean space and removes lens distortion, which degrades accuracy.

Calibration techniques can be divided in 3 macro-categories:

1. 3D reference object-based calibration:
   Conducted by capturing a 3D reference object with precisely known geometry in 3D space.
   This method grants high accuracy and efficiency in estimating the intrinsic and extrinsic parameters and just one image of the reference object is sufficient to accomplish calibration. However, it also has the disadvantage that in a multicamera setup, a 3D reference object is hard to observe without self-occlusion by all cameras simultaneously.

2. 2D plane-based calibration:
   Performed by observing a planar pattern shown at more than 2 orientations. This category of calibration is very flexible, since the 2D plane pattern (e.g. checkerboard pattern printed on paper) is easy to prepare and no special calibration setup is necessessary. However, these techniques oblige to capture several images of calibration objects with fixed camera intrinsic parameters.

3. 1D linear object-based calibration:
   This category is nowadays very popular. It is possibile when in image is present a 1D object having three collinear points with known distances and requires at least six observations of the 1D object rotating or undergoing a planar motion. This method is simpler with respect to the other 2 and works also in a multicamera sutup. Similar to 2D methods, techniques in this category suffer from the disadvantage that calibration cannot be

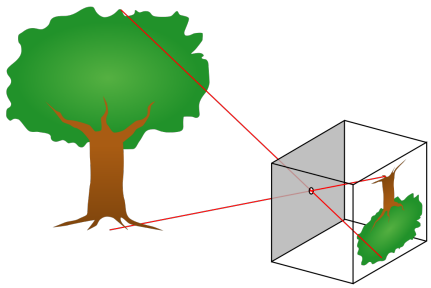


Figure 3.1: The Pinhole Camera Model.

accomplished from a single image, nor from a sequence of images involving camera zooming, which happens frequently in a basketball video.

None of the above categories is suitable fot the case evalueted in this study. There are method that does not involve the usage of a calibration object, as cited in 1.1.3, but those method are also not suitable. Before introducing the calibration method used here and its relative mandatory assumptions, the camera model is presented.

3.1.1 *Camera Model*

The simplest type of camera that can be found in the real world is the pinhole camera. It is a simple lightproof box with a very small hole in the front which is also called an aperture, and some light-sensitive film paper laid inside the box on the side facing this pinhole (see fig. 3.1). The principle of work is simple: objects from the scene reflect light in all directions; the size of the aperture is so small that among the many rays that are reflected off at P, a point on the surface of an object in the scene, only one of these rays enter the camera. These rays form an image of the object rotated by 180 degrees. In geometry, the pinhole is also called the center of projection; all rays entering the camera converge to this point and diverge from it on the other side. To each point in the scene corresponds a single point on the film.

3.1.2 *Camera Parameters*

The pinhole camera parameters are represented in a 3x4 matrix called the *camera matrix* **P**. This matrix maps the 3-D world scene into the image plane. The calibration algorithm calculates the camera matrix using the extrinsic and intrinsic parameters under the relationship:

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \implies \lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{K}\,\mathbf{R} \left[\mathbf{I} \mid \mathbf{-T}\right] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.1}$$

where λ is a scale factor, $[x, y, 1]^T$ are the image points, while $[X, Y, Z, 1]^T$ are 3D points coordinates. The extrinsic parameters **R [I | -T]** (I is a 3x3 identity matrix)represent the location of the camera in the 3-D scene. The intrinsic parameters **K** represent the optical center and focal length of the camera.

- Extrinsic Parameters:

  The extrinsic parameters consist of a rotation, **R**, and a translation, **T**. The origin of the camera's coordinate system is at its optical center and its x- and y-axis define the image plane. The camera's extrinsic matrix describes the camera's location in the world, and what direction it's pointing. In other words, this matrix describes how to transform points in world coordinates to camera coordinates.

$$\mathbf{R} \left[\mathbf{I} \mid \mathbf{-T}\right] = \begin{pmatrix} r_{11} & r_{12} & r_{13} & -T_x \\ r_{21} & r_{22} & r_{23} & -T_y \\ r_{31} & r_{32} & r_{33} & -T_z \end{pmatrix} \tag{3.2}$$

  R is a 3x3 rotation matrix whose columns are the directions of the world axes in the camera's reference frame. The vector **T** = $[T_x, T_y\ T_z]^T$ is the camera center in world coordinates;

- Intrinsic Parameters:

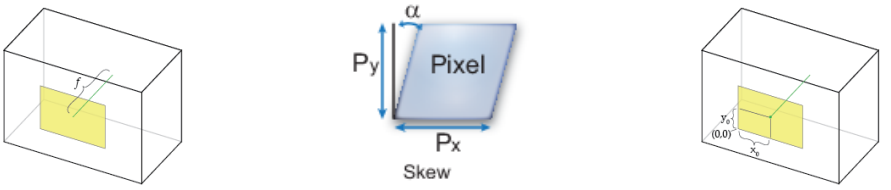
  The intrinsic parameters include the focal length, the optical center, also known as the principal point, and the skew coefficient. The camera intrinsic matrix, K, is defined as:

$$\mathbf{K} = \begin{pmatrix} f_x & s & p_{0x} \\ 0 & f_y & p_{0y} \\ 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

  - (f_x, f_y) = Focal Lengths
    The focal length is the distance between the pinhole and the image plane. The focal length is measured in pixels. In

a true pinhole camera, they have the same value, $f_x = f_y = f$. In practice, f_x and f_y and can differ. As a consequence the resulting image will have non-square pixels.

- s = Pixel Skew
  Axis skew causes shear distortion in the projected image.
- (p_{0x}, p_{0y}) = Optical center (the principal point).
  The camera's principal axis is the line perpendicular to the image plane that passes through the pinhole. Its itersection with the image plane is referred to as the principal point. The exact definition depends on which convention is used for the location of the origin. Note that p_0 can be in any point of the image plane.



(a) Focal Length **(b)** Skew **(c)** Principal point, defined with image origin at bottom-left

Figure 3.2: Camera Intrinsic Parameters

3.2 FIVE-POINTS GEOMETRY BASED CAMERA CALIBRATION

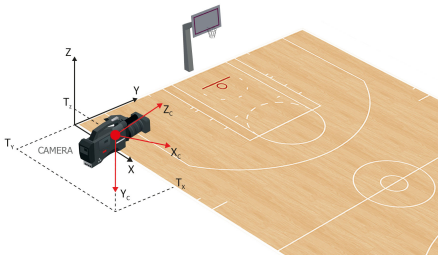
3.2.1 *Assumptions and Reference System Establishment*

The geometry definition of the camera position and rotation angles is illustrated in 3.3.

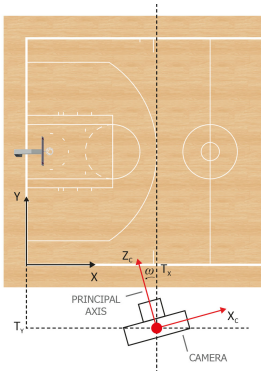
The camera position, also known as the optical center, is denoted by (T_X, T_Y, T_Z) in the world coordinate system XYZ. Define the camera coordinate system XcYcZc, of which the origin is at the camera center and the principal axis is in agreement with the Zc-axis. Figure 3.4 defines the relationship among the world (X, Y , Z), camera (Xc, Yc, Zc), and image (x, y) coordinate systems.

The camera considered is central projection, the relationship from a 3D world point to a 2D image point is given by 3.1 and the rotation matrix **R** is describing:

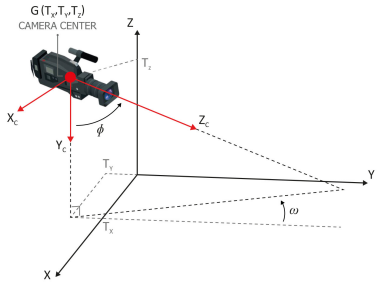
1. A first rotation around Y_c-axis.
2. Then a rotation around the new position of the X_c-axis.



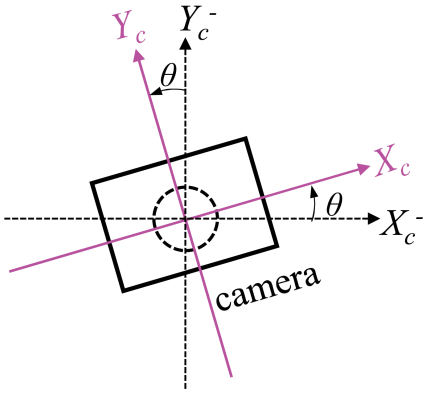
(a) Camera Position in Real World Coordinate System XYZ



(b) Camera Position in XY Plane and Rotation angle ω definintion around Yc axis



(c) Rotation angle Phi around X_c axis



(d) Definition of theta, rotation angle around Z_c axis

Figure 3.3: Camera Position and Rotation Angles

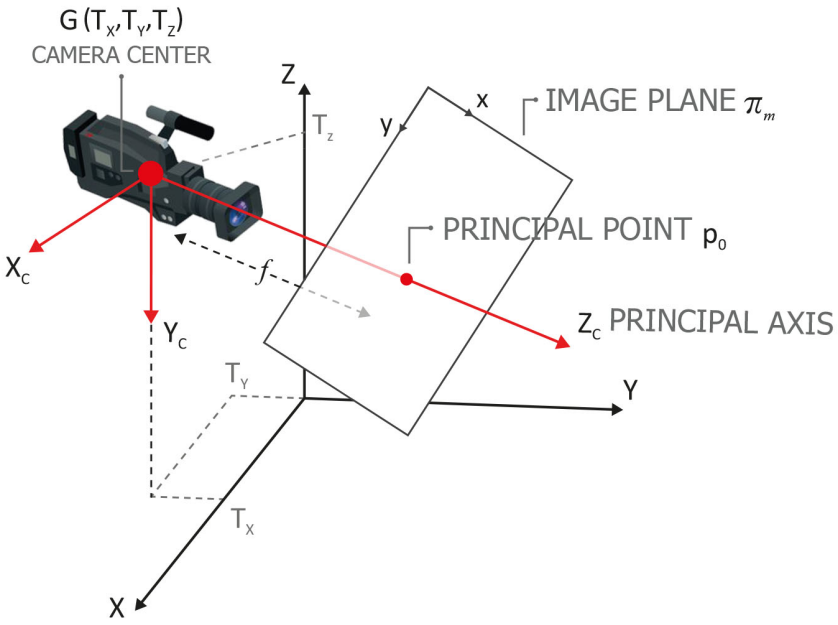


Figure 3.4: Relationship between the camera reference frame (Xc, Yc, Zc) and the world reference frame (X, Y , Z).

3. Finally a rotation around the new Z_c-axis.

As mentioned in the previous section 3.1.2, the projection matrix **P** has 12 entries, and (ignoring scale) 11 degrees of freedom, so it is necessary to have 11 equations to solve for **P**. Each point correspondence leads to two equations, therefore six such correspondences are required to solve for P, at a minimal. In the following, the method through which can be estimated camera parameters using only five-point correspondences from a geometric analysis-based approach is provided.

The focal lengths are supposed to be equal $f_x = f_y = f$, the skew is considered to be null $s = 0$ and the principal point p_0, the point where the Z_c axis meets the image plane, is believed to be in the projection centre.
Hence, developing 3.1 and substituting into it equations 3.2 and 3.3, the trasformation between a world point and an image point becomes:

$$\begin{aligned} x &= f\frac{r_{11}(X-T_x)+r_{12}(Y-T_y)+r_{13}(Z-T_z)}{r_{31}(X-T_x)+r_{32}(Y-T_y)+r_{33}(Z-T_z)} \\ y &= f\frac{r_{21}(X-T_x)+r_{22}(Y-T_y)+r_{23}(Z-T_z)}{r_{31}(X-T_x)+r_{32}(Y-T_y)+r_{33}(Z-T_z)} \end{aligned} \tag{3.4}$$

3.2.2 *Camera Parameter Estimation*

This method uses five-point reference in the image and geometric properties to find the camera matrix **P** [6]. Four out of the 5 points are coplanar but not collinear (A, B, C, and D), while the fifth is not coplanar (E). Note that any four coplanar but noncollinear points and a fifth noncoplanar one can be utilized as a reference object for the calibration method. The 3D real-world positions and 2D image coordinates of the five reference points are known. In the 2D image coordinate system $p = (x, y)$, the five points corresponding to the reference points A, B, . . . , E are denoted by $pa = (x_a, y_a), pb = (x_b, y_b)$, etc. The reference points A, B, C, and D are for simplicity assumed to lie on the XY plane (π_0), the plane of Z=0 in the XYZ coordinate system (i.e. the basketball court).
Consider the mapping between π_0 and the image plane π_m, this is a projective transformation that can be described by a plane to plane mapping **H**, an homography.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{3.5}$$

From 2.1 it is known that **H** has 8 d.o.f, up to an arbitrary scale, so it can be derived from 4 point correspondances. Using the four coplanar points A,B,C and D of the reference object and the DLT algorithm, **H** can be easily be found. [1]. Once the projective transformation is found, the subsequent step is to find the rotation matrix and camera position.

The points at infinity in the 2D projective space form a line, which is usually called the line at infinity l_∞, horizon line or vanishing line. Therefore, from two points at infinity the vanishing line of the ground plan can be found. From l_∞ is possible to estimate the rotation angle θ. From equation 3.5 comes:

$$x = \frac{h_{11}X + h_{12}Y + h_{13}}{h_{31}X + h_{32}Y + h_{33}} \qquad y = \frac{h_{21}X + h_{22}Y + h_{23}}{h_{31}X + h_{32}Y + h_{33}} \tag{3.6}$$

Substituting $X = \infty$ and then $Y = \infty$ in the first and second part of equation 3.6 respectively, the two vanishing points $v_1 = (v_{1x}, v_{1y})$ and $v_2 = (v_{2x}, v_{2y})$ of the image plane can be found. They can also be found from 2 pairs of parallel lines in orthogonal directions, as figure 3.5 shows.

1 For image point $(p_a, p_b, ..., p_e.)$ extraction, see section 3.2.4.

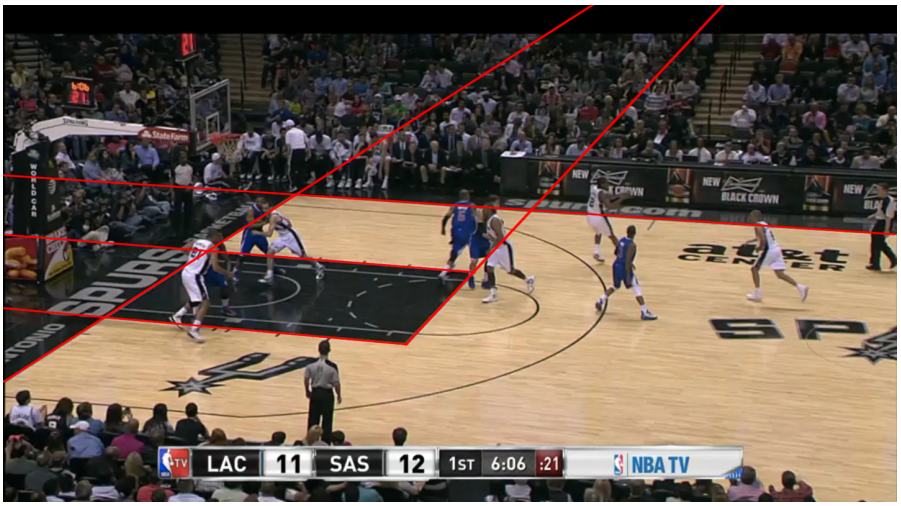


Figure 3.5: Vanishing points v_1 and v_2 identification from orthogonal court lines

The line passing thorough v_1 and v_2 is the vanishing line l_∞, indicated by l_1 in the picture. As it can be seen θ is the angle between the horizontal line going through the vanishing point and the line at infinity, positive in anticlockwise direction.

$$\theta = \tan^{-1}\left(\frac{\Delta_y}{\Delta_x}\right) = \tan^{-1}\left(\frac{v_{1y} - v_{2y}}{v_{1x} - v_{2x}}\right) \tag{3.7}$$

To compute the second rotation angle ω, first is required the projection of the principal axis on the ground plane. Imagine a plane π_2 that:

- Contains the camera center G.
- Contains the principal point p_0.
- Is perpendicular to the ground plane π_0.
- Contains G'= $(T_x, T_y, 0)$, that is the vertical projection of G on π_0.

Note that l_∞ is constructed by intersecting a plane π_1 and the image plane π_m, with π_1 parallel to the ground plane and passing through the camera center G [14]. The consequences of all these conditions on the creation of π_2 bring to the conclusion that π_2 will also be perpendicular to π_m. As a result, the line l_2 formed by itersecting π_2 and π_m will be perpendicular to l_1. Hence, l_2 can be drawn on the image plane and the image point p_1 is derived, it is the intersection of l_2 with bottom part of the image (see fig. 3.6).

Since **H** is a plane to plane to plane mapping, from its inverse is possible to obtain the projection of image points p_0, p_1 onto ground π_0, termed P_0 and P_1 respectively.

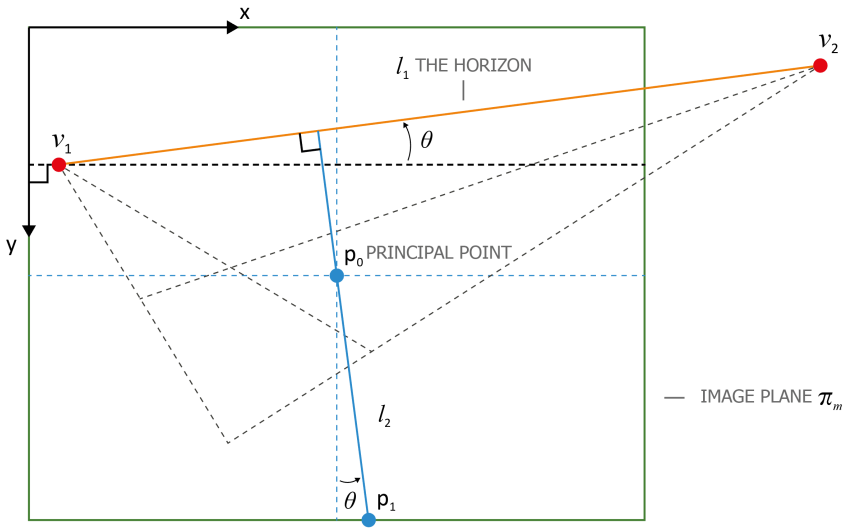


Figure 3.6: Vanishing line l_1 obtained from vanishing points and line $l_2 \perp l_1$ through p_0 in image plane.

$$\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \mathbf{H}^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.8}$$

As shown in figure 3.7, the line L_2 formed by connecting P_0 and P_1 contains also G′. Let $P_0 = (X_0, Y_0, 1)$ and $P_1 = (X_1, Y_1, 1)$, rotation angle ω can be computed by:

$$\omega = \tan^{-1}\left(\frac{X_1 - X_0}{Y_0 - Y_1}\right) \tag{3.9}$$

The next step is to compute the camera position $G = (T_x, T_y, T_z)$ with the help of the fifth noncoplanar reference point $E = (X_E, Y_E, Z_E)$, which correspond to p_e on image. By 3.8 project E onto ground plane at $P_e = (X_e, Y_e, 0)$. Let E′ be the vertical footprint of E on π_0, $E' = (X_E, Y_E, 0)$. Accordin to the pinhole camera model, the line L_e formed by connecting P_e and E will pass through camera center G. As a consequence, the line L_3 coming from the intersection of P_e and E′, will pass through G′. This implies that lines L_2 and L_3 both pass via G′, fig. 3.8.

Being m_2 and m_3 the slopes of L_2 and L_3, respectively, T_x, and T_y can be obtained from:

$$\begin{aligned} T_x &= \frac{m_2 X_0 - m_3 X_e - Y_0 + Y_e}{m_2 - m_3} \\ T_y &= m_2(T_x - X_0) + Y_0 = m_3(T_x - X_e) + Y_e \end{aligned} \tag{3.10}$$

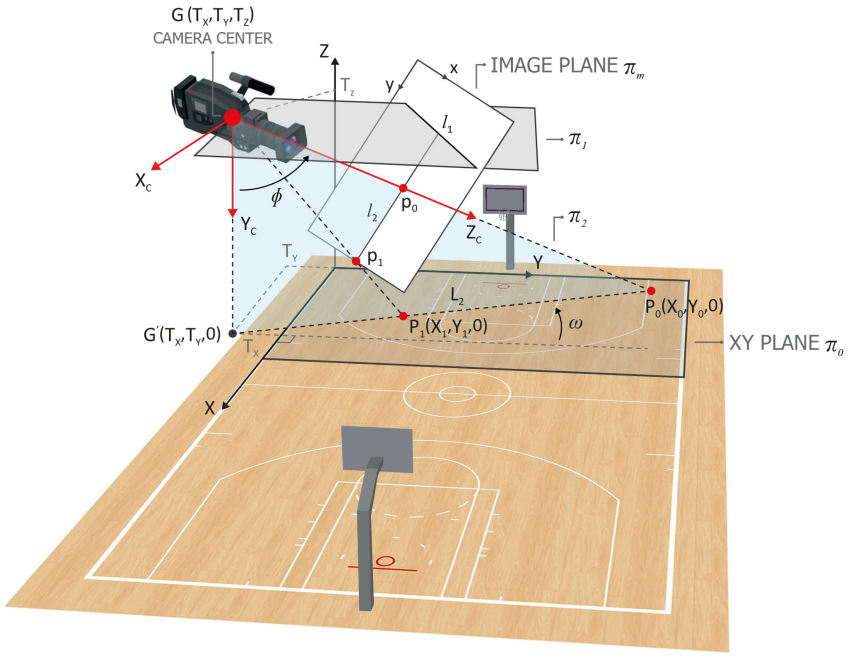


Figure 3.7: Angle ω represented on ground plane. Plane π_1 goes through the camera centre G, intersects $l_\infty = l_1$ in the image, and is parallel to court plane π_0. The plane π_2 is by construction perpendicular to π_0 and π_1, and also contains G and G′. The line L_2 joins the projection of p_0 and p_1 in XY plane.

Then compute coordinate T_z using the similar triangles $\Delta P_e E E'$ and $\Delta P_e G G'$ (see fig. 3.9) via:

$$T_z = Z_e \frac{|\overline{P_e G'}|}{|\overline{P_e G'}|} \tag{3.11}$$

Now that the camera position has been found, with it and P_0 the angle ϕ can be computed by:

$$\phi = \tan^{-1}\left(\frac{(X_0 - T_x)^2 + (Y_0 - T_y)^2}{T_z}\right) \tag{3.12}$$

Only the focal length f is missing, but it can be estimated with DLT algorithm by inserting the coordinates of the five corresponding points into 3.6.

3.2.3 *Implementation in a Basketball Scene*

In a basketball image, the four coplanar points are selected on the intersection of the court lines, while the fifth noncoplanar point is on

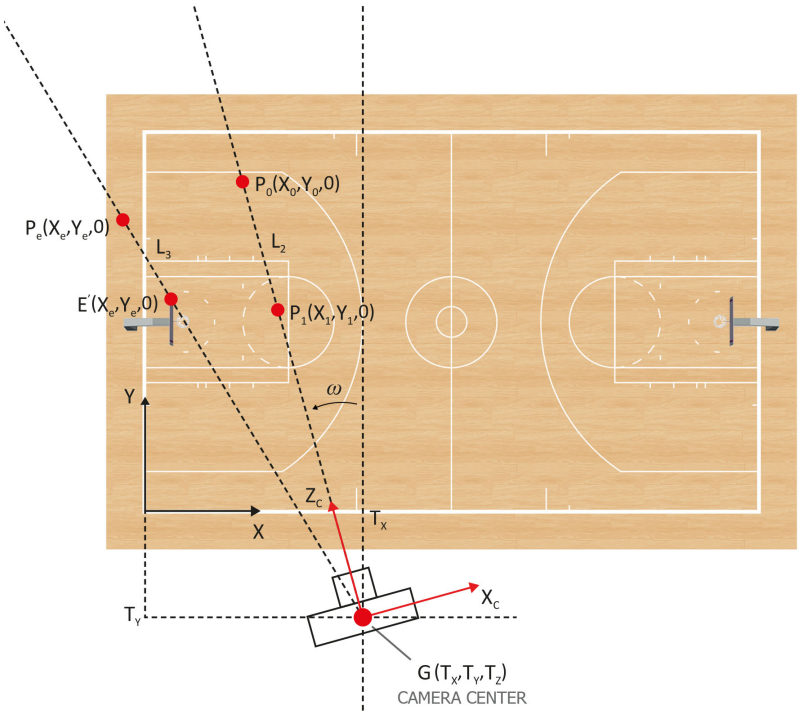


Figure 3.8: Camera Position on ground plane coming from intersection of L_2 and L_3.

a characteristic object, the top backboard corner. With the known 3D court model, camera parameters can be estimated by our proposed calibration method using the five extracted corresponding points, of which both 2D and 3D coordinates are known.

One significant advantage of this approach over other 3D calibration methods, which require at least six corresponding points, is that the 3D calibration methods will fail to treat the case where only one noncoplanar point is visible in the image due to the limitation of the camera viewing field or object occlusion. This implementation can still perform well. Another advantage is that since the video is stabilized, in every frame the 5 image point do not move, hence the only changing parameter can be the focal length, giving speed and robustness to the algorithm. On the other hand, it has the disadvantage that since basketball arenas have court in many different colours, it may happen that the line cannot be identified so image points $p_a - p_e$ cannot be derived from their intersection. This is solved by selecting manually points on visible court lines and fitting lines between them. This process does not decrease the algorithm performances, but just slows it down.

The last section shows a brief overview on the algorithm to automatically detect court points and lines.

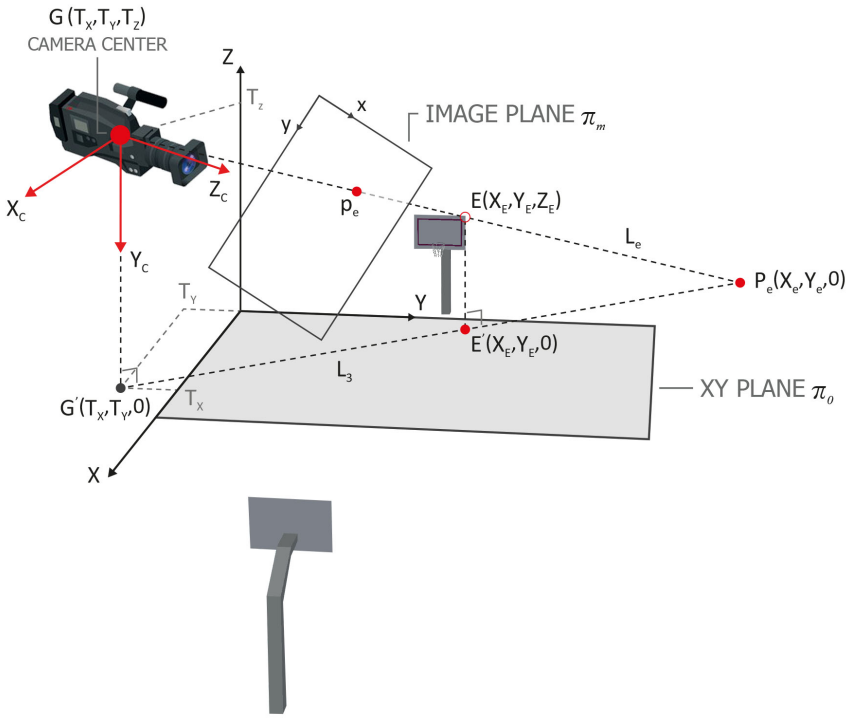


Figure 3.9: Projection of p_e onto XY plane. Image shows also similar triangles $\Delta P_e EE'$ and $\Delta P_e GG'$, and line L_3.

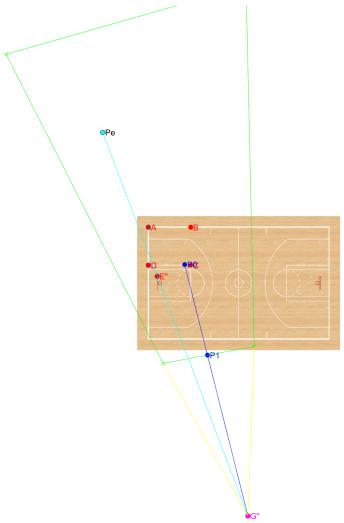


Figure 3.10: Birds eye view of the detected camera position.

3.2.4 *Court Identification*

As previously stated, is possible to automatically detect interesting court points in the majority of basketball scenes. A NBA court in fact has an homogeneous color exluding the area, which is in a different

color and that's why it is called *paint* in basketball slang. Therefore, it's relatively easy to identify the pixels value that contribute to the most common colors in the image. The algorithm flowchart is presented in figure 3.11.

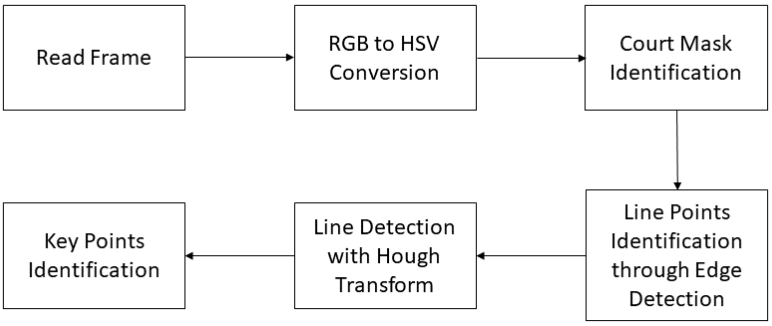


Figure 3.11: Automatic court line and points algorithm flowchart.

The first step is to convert the Red, Green, Blue (RGB) image to the Hue, Saturation, Value (HSV) colour domain. Focusing on the Hue plane the so called hue-map can be extracted, which is an image where all pixels are in the range [0,1]. In a basketball image non-court pixels are usually greater values compared to court ones, so, by image thresholding is possible to isolate court pixel. This result in a binary mask in which the black pixels are the non-court region, while the white one represent the court.

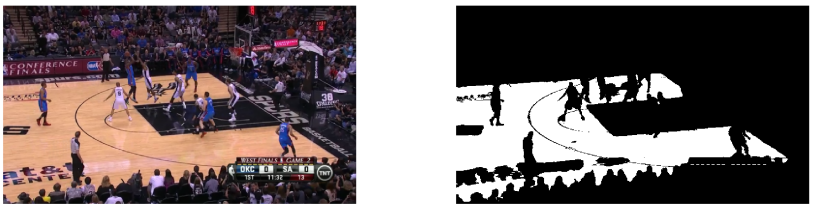


Figure 3.12: Court Mask generation.

This method is fast and farly accurate, but it has the disadvantage that the threshold is not unequivocally defined. To solve this issue many values have been tested and the result is that a good enough value of threshold in between [0.1, 0.25], but note that it may not be good for all images, hence sometimes it has to be changed based on the hue-map. Another handicap of this solution is again occlusion by players. It may happen that in an image more than one player reside in the *paint*. If his jersey and skin colour passes the thresholding, this will lead to false positives. The area will not be detected as it will appear mostly white in the binary mask. Fortunately, basketball is

an highly dynamic sport and this issue lasts for a short number of frames. Therefore is solved by neglecting the image if the court lines are not detected correctly. This has the obvious drawback that the image will be discarded completely and that frame will not bring any information, and as a consequence ball cannot be detected in those frames.

3.2.5 *Line Detection*

To lines that are to be identified are the baseline, the sideline and the paint lines. Those lines are chosen because they allow to identify a minum number of 4 keypoints that allows tho extract the plane to plane homography **H**. Also, the lines are less likely to be occluded and the points can be deducted from their intersection, giving stability to the algorithm.

The common method with which lines can be extracted is :

1. Invert colours in court mask to have pixels identifying court in black colur
2. Operate Canny edge detection to find contours of object
3. Use Hough transform, with some thresholds, to find the straight lines in teh image

This works in most cases, but in some it misdetect lines. Hence, another type of approach is chosen.

At first is determined whether the image represent a left side view or a right side view of the court. Thi is done by simply counting the court mask pixel and assessing if the more of them lies on the right side or the left side. Then, by finding the minimum and maximum rows and columns in the image that contains the majority of court pixels, the extent of the court in the image is determined. This information helps roughly defining the image zones where the lines should be found. Therefore, by edge detection the potential interesting line points are identified. It may happen that some of the detected points do not belong to a line but to another object, such as players or referee. This is caused again by the fact that the court mask is not perfect, but has holes due to presence of people on the game field. However, the lines can still be estimated by fitting the points with the help of the Hough transform. These allows to detect the lines as long as the main part of the points belong to the line and not to other objects.

Once court lines have been found, by simple line intersection, the image points p_a, p_b, p_c, p_d necessary for homography estimation can be detected. As far as concerns point p_e it was chosen to leave it

to the user to detect due to the fact that the blackboard lines where not easy to detect, or required to much computational time. Figure 3.13 shows the detected lines and points in the image, and it can be seen that they are not perfectly precise. It was chosen to accept giving up a bit of precision to gain computational speed.

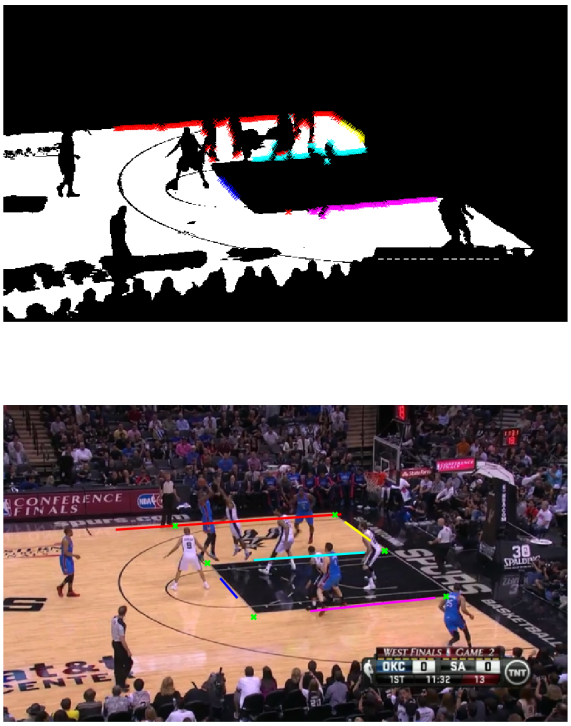


Figure 3.13: Edge Detection and Hough line and points extraction.

4

BALL DETECTION AND TRACKING

The differences between detection and tracking were introduced in chapter 1.1. In this chapter the adopted algorithm to detect and track a ball from a basketball video is presented. The flowchart of the proposed framework for ball detection and tracking and its analysis is shown in figure 4.1. The first part deals with detection, introducing in the beginnig the main approaches, then dissecting the proposed algorithm. Afterwards the trajectory extraction is analysed.

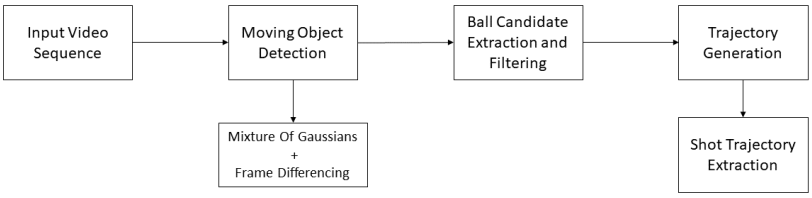


Figure 4.1: Flowchart of the algorithm used for ball detection and tracking.

4.1 MOVING OBJECT DETECTION

Moving object detection is to recognize the physical movement of an object in a given place or region. By acting segmentation among moving objects and stationary area or region, the moving objects motion could be tracked and thus could be analyzed later. Consecutive frames from a video are compared by various methods to determine if any moving object is detected. All detection techniques are based on modelling the background of the image, i.e. set the background and detect which changes occur. This may be difficult when the scene contains many objects of different size and shape, shadows and illumination changes. So a good detector should both operate in real-time and adapt to these changes throughout all the duration of the video sequence. Traditional moving object detection methods could be categorized into three major approaches:

1. *Background Subtraction*:
   it is a widely used approach for detecting moving objects in videos from static cameras. The rationale in the approach is that of detecting the moving objects from the difference between the current frame and a reference frame, often called "background image", or "background model".
   Since in a real environment the scene udergoes many changes,

this method is not applicable. This can be improved by the temporal average filter, which is a method that estimates the background model from the median of all pixels of a number of previous images. This method can not be considered very efficient because they do not present a rigorous statistical basis and requires a buffer that has a high computational cost.

2. *Frame Differencing*:
the idea behind this algorithm is to take an image as background, denoted by B, and take the frames obtained at the time t, denoted by $I(t)$, and make a comparison between the two. Hence, using simple arithmetic calculations, moving object can be extracted. In fact, through image subtraction technique for each pixels in $I(t)$, take the pixel value denoted by $P[I(t)]$ and subtract it with the corresponding pixels at the same position on the background image denoted as $P[B]$.

$$P[F(t)] = P[I(t)] - P[B] \tag{4.1}$$

This difference image would only show some intensity for the pixel locations which have changed in the two frames. However, this technique only works when all pixels in background are perfectly still, which is not the case of a real scene. A simple improvement is to apply a threshold to the difference image, to filter out background moving object:

$$|P[F(t)] - P[F(t+1)]| > Threshold \tag{4.2}$$

Accuracy depends on speed of movements in the scene.

3. *Optical Flow*:
Optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movemement of object or camera. It is 2D vector field where each vector is a displacement vector showing the movement of points from first frame to second. Optical flow works on assumptions that:
   a) The pixel intensities of an object do not change between consecutive frames.
   b) Neighbouring pixels have similar motion.

   Consider a pixel $I(x, y, t)$ in first frame. It moves by distance (dx, dy) in next frame taken after dt time. So since those pixels are the same and intensity does not change, it can be said:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{4.3}$$

Then take taylor series approximation of right-hand side, remove common terms and divide by dt to get the so called Optical Flow equation:

$$f_x u + f_y v + f_t = 0 \qquad where \quad f_x = \frac{\sigma f}{\sigma x}\ ; \qquad f_y = \frac{\sigma f}{\sigma y} \qquad u = \frac{dx}{dt}\ ; \qquad v = \frac{dy}{dt} \tag{4.4}$$

To solve this equation several method have been implemented, but they will not be treated for sake of brevity.

From the above methods, in case of ball detection, Optical Flow performed poorly, while frame differencing and background subtraction suffered from too many false positive to take them into consideration. Hence an improving background subtraction by usign Gaussian Mixture Model (GMM) was considered.

4.1.1 *Ball Detection with Gaussian Mixture Model*

A basketball video of an indoor basketball arena normally contains many moving objects in the background such as spectators, banners etc. The illumination condition keeps varying. To overcome these situations, a robust moving object segmentation algorithm is required which can withstand the effects of illumination changes and the presence of multiple moving objects in the background.

GMM [17] is one of the most popular technique to construct the background model for segmentation of moving objects from background. In this evolution of background subtraction technique each pixel is represented by a mixture of Gaussians and during run time each pixel is updated with new Gaussians. The principle of GMM is to model each pixel by using a statistical model composed by a mixture of Gaussian distributions. Therefore, the every pixel of the video is composed by a mixture of K Gaussian distributions, each one represented by four parameters:

1. ω: weight
2. μ: mean
3. σ^2: variance,

K is the number of Gaussian distribution used, a number from 3 to 5. The larger the value of K, the stronger is the ability to reject background noise of the algorithm. However a big K increases the computational cost, that is why in this work the choice is to use $K = 3$.
Being $X_1, X_2, \ldots, X_t$ the sequence of observed pixel $f(x, y)$, the probability of observing the current pixel value X_t at time t can be estimated as:

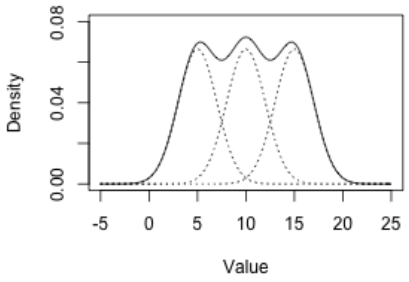


Figure 4.2: A Gaussian mixture of three normal distributions.

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t}\eta(X_t, \mu_{i,t}, \theta_{i,t}) \tag{4.5}$$

where $\theta_{i,t}$ is the covariance matrix of the i^{th} Gaussian at time t. The weight $\omega_{i,t}$ indicates what portion of data is accounted for by the i^{th} Gaussian at time t, and as a consequence $\sum_{i=1}^{K} \omega_{i,t} = 1$.

$$\eta(X_t, \mu_{i,t}, \theta_{i,t}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\theta|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_t)^T\theta^{-1}(X_t-\mu_t)} \tag{4.6}$$

id a Gaussian probability density function with n the dimension of the Gaussian distribution. In this project the the background modeling is performed on grayscale frames so $n = 1$ and $\theta_{i,t} = \sigma^2$.

At time instant $t+1$ a new frame incomes and all the GMM parameters are updated according to the new pixel value. The pixels are matched with each Gaussian component and are considered as a match when:

$$|X_{t+1} - \mu_{i,t}| < k\sigma_{i,t} \tag{4.7}$$

with k is a constant threshold. If match is found, weights are updated comforming to:

$$\begin{aligned} \omega_{i,t} &= (1-\alpha)\omega_{i,t-1} + \alpha M_{(i,t)} \\ M_{(i,t)} &= \begin{cases} 1 & \text{if match} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4.8}$$

Other parameters are updated as follows:

$$\begin{aligned}
\mu_{i,t} &= (1-\rho)\mu_{i,t-1} + \rho X_{(i,t)} \\
\sigma^2_{(i,t)} &= (1-\rho)\sigma^2_{(i,t-1)} + \rho|X_{(i,t)} - \mu_{i,t}|^2 \\
\rho &= \alpha\eta(X_t|\mu_{i,t}, \sigma_{i,t})
\end{aligned} \tag{4.9}$$

$\alpha \in (0,1)$ is the learning rate that has the function to control the update speed, while ρ is the parameter learning rate. In the case that no match is found a Gaussian distribution characterized by initial large variance and small weight ($\mu = X_{t+1}$) will replace the other with minimal weight.

The background model estimation is the last step of the method. For each pixel the pixel, all the existing mixture are sorted in descending order according to the value $\frac{\omega}{\sigma}$ and the first B distributions are chosen as model for the background.

$$B = argmin_b(\sum_{k=1}^{b} w_k > c_f) \tag{4.10}$$

with c_f = measure of the minumum portion of the data that should be considered by the background.

The pixels p are matched to the Gaussian models and:

$$p = \begin{cases} \text{matches background model} \implies \text{is a background pixel} \\ \text{is a foreground pixel otherwise} \end{cases} \tag{4.11}$$

In the proect the GMM algorithm is applied, and after that, a frame differencing with different threshold between the cases refines the detected blobs.

4.1.2 *Ball Candidate Filtering*

Once that the moving object has been found, morphological operations of opening and closing are performed to remove unwanted noise.

- Open: smoothes the objects contours.
- Close: fill the holes and joins the brakes.

The foreground extracted is good but still not good enough to allow ball regognition. Due to the presence of players and a considerable amount of other moving objects, the segmention result is often wrong. In fact, the ball image result deformed (it looks more like an ellipse

rather than a circumference), and sometimes it merges with other scene objects. So the ball must be filtered from other moving objects present in the frame. Geometrical properties of the ball are exploited to get rid of unwanted object and remain with a fewer number of objects. All this objects are potentially the ball, so they are called *ball candidates*.

- *Shape Filter*:
  as said, in a basketball video the ball is often blurred and resamble an ellipse. Thus, from all blobs remaining, the ones with an eccentricity between 0 and 0.7 are mantained. Note than a circle has eccentricity equal to 0, while the an ellipse has it greater than 0 but less than 1.

- *Circularity Filter*:
  many object have ball-like shape and pass through the first filter. Circulariy is defined as the ratio of the area of the blob and the circle having the same perimeter. This measure can be normalized and defines as:

$$\gamma_N = 1 - \frac{4\pi A}{P^2} \tag{4.12}$$

  Again γ_N for a circle is 0 and are kept as potential ball candidates the blobs that swing between $0 < \gamma_N < \kappa$, with κ differing from case to case ($\kappa \in [0.5, 0.8]$)

- *Size Filter*:
  Simply filter out smaller blobs.

4.2 TRAJECTORY BASED BALL TRACKING

Some non ball object still pass through the filters, hence the ball is tracked along frames using it motion along x and y directions.
Plotting the detected ball candidates against the number of frame results in two different graphs. It can be seen from figure 4.3 that:

- In the x plot the ball moves in a straight line.

- In the y plot the ball forms a parabolic path.

This behaviour is used to generate the trajectories of the ball from the set of detected ball candidates.

4.2.1 *Trajectory Generation*

The objective is to recognize the straight line and the parabola paths from all the detected ball candidates. The algorithms, starts with a

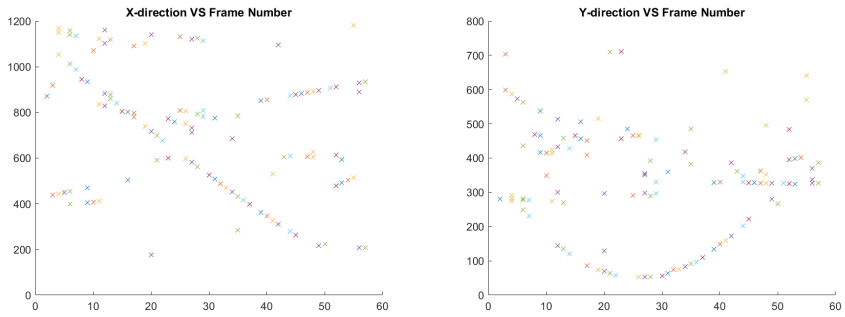


Figure 4.3: X and Y-Candidate Plot.

pair of nearby ball candidates. A Kalman filter (A.1) prediction is used to predict the ball locations along the trajectory and this does not depend on velocity and acceleration of the ball:

$$\begin{aligned} x_i &= Ax_{i-1} + w_{i-1} \\ z_i &= Hx_i + v_i \end{aligned}$$

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad ; \quad H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{4.13}$$

where x_i is the state vector, representing the estimated ball location in the i^{th} frame; z_i is the measurement vector which is the position of the detected ball candidate. A is the system evolution matrix, w the process noise, H the measurement matrix and v the measurement noise.
From the Kalman filter two situation may rise:

1. The prediction is verified $\implies$ function is updated and the detected ball candidates are added to the trajectory

2. The prediction is not verified $\implies$ Kalman filter process continues for a fixed number of times. If the prediction is still not verified the process stops and record another possible trajectory.

The previous algorithm generates more than one acceptable trajectory (fig. 4.4), hence it still requires a refinement to exploit the actual ball trajectory. From experiment it has been found that the shot sequence last between 30 and 60 frames, and in these the ball flying time lasts for about 15 to 20 consecutive frames. Under the hypotesis that the ball trajectory should be amongst the longest found, all the candidates with length < 10 are eliminated a priori.

Then the remaining trajectories are processed one by one. A parabola is fitted through all the points in the y-candidate trajectory and if all

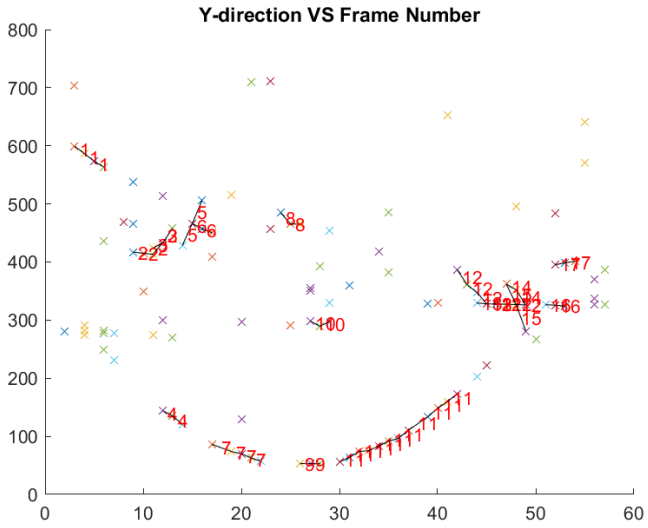


Figure 4.4: Detected Parabolas in Y-Candidate Plot.

those points have a distance from the parabola d within a threshold, the same is done with a line in the x-candidate plot. Also the prediction error is considered as a redundant refinement. The prediction error is defined as the average distance (in pixel) between each ball location to the predicted location in a frame. The trajectories having prediction error greater than a threshold are discarded.
The remaining trajectory is the ball trajectory, as it can be seen in the figure 4.5.

4.2.2 *Track Analysis and Processing*

From the parabola and the straight line in the y-candidate plot and x-candidate plot respetively the ball trajectory in the image may be composed. From the same plots, ball position in the frames where the ball was not detected can also be estimated. Then, combining both estimated and detected ball positions the final trajectory is derived and plotted over the a video frame, as it can be seen in 4.6.

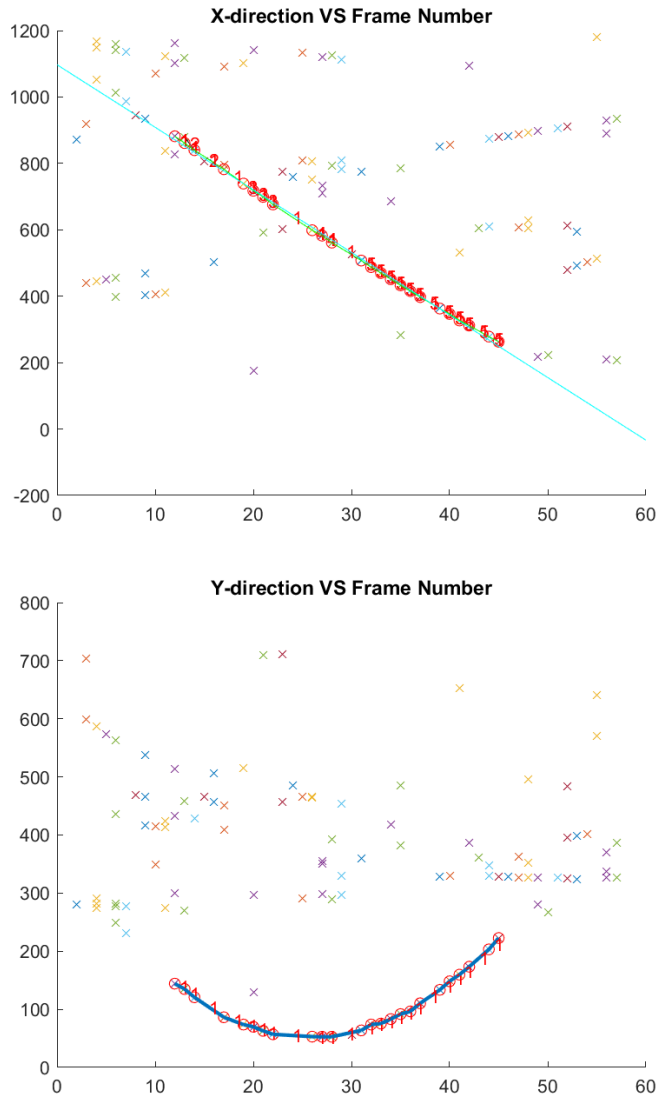


Figure 4.5: Identified x and y trajectories.

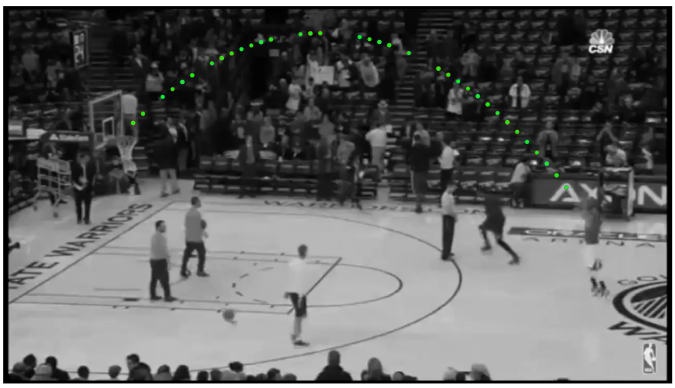


Figure 4.6: Combined XY Trajectory plotted over a video frame.

5

3D TRAJECTORY EXTRACTION

Once the 2D ball position have been extracted the 3D trajectory can be estimated. The calibrated camera and its relations are fundamental in this process. As previously stated, image points relate to 3D points with:

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \implies \lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{K}\,\mathbf{R} \begin{bmatrix} \mathbf{I} \mid \text{-}\mathbf{T} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{5.1}$$

where **P** is a 3x4 matrix and hence is not invertible. From a single image and without a depth map, extracting 3D ball position is impossible.

Therefore, under the assumption that a basketball shot forms a parabolic trajectory in real word[1], exploiting geometrical and physical properties of a parabola 3D ball position can be estimated.

5.1 REAL WORLD BALL POSITION FROM GEOMETRY

If only gravity affects the ball trajectory, in 3D it undergoes a pure ballistic motion, that has known physical equations and geometrical properties since it has been deeply studied through the years.

Definition 5.1. In this work is defined as *Trajectory Plane* the plane where the ball develop its motion. It is the plane containing all 3D ball positions.

1 In real situation, factors like ball spin, spin axis, air friction etc. affect ball trajectory, here those elements are considered negligible

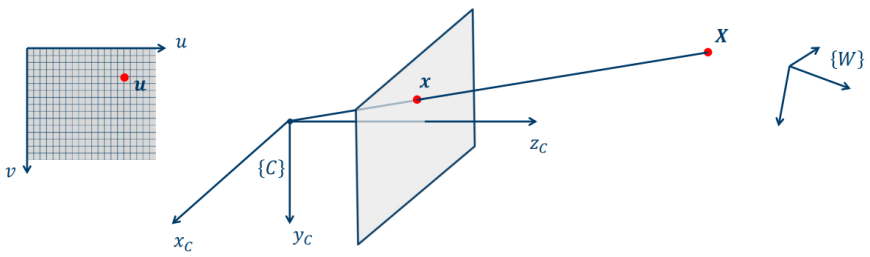


Figure 5.1: Relation between image point x and world point 3D.

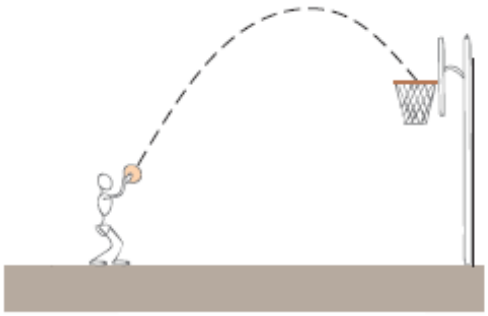


Figure 5.2: Trajectory plane perpendicular to image plane

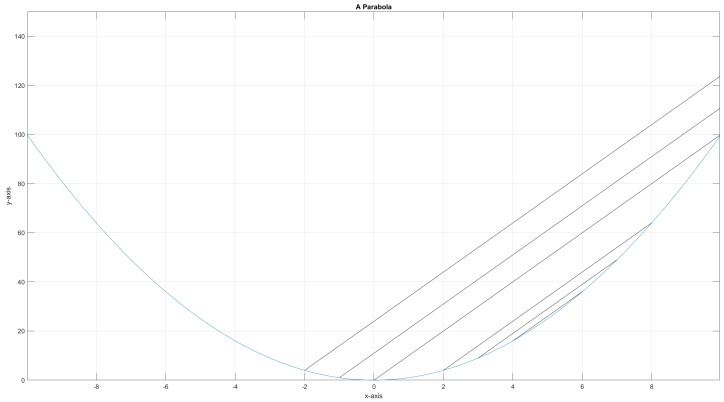


Figure 5.3: Example of a parabola with equation $y = x^2$, secants traced considering as central time $t = 5$.

In case that the camera is perpendicular to the trajectory plane, the ball is subjected to a perfect projectile motion even in the image plane. In fact, saying that the camera is perpendicular to the trajectory plane is the same as saying that the video is taken perpendicularly to both ground plane and image plane, see figure 5.2 for an immediate understanding. This is an extremely rare case, in reality camera can be in any position, and usually is located in the stands, so trajectory plane and image plane never coincide. In any case, the parabola properties used are the always the same irrespective of the trajectory plane:

1. Taken a point on the parabola correspondig to time instant t, and the secants in $(t-\tau_1, t+\tau_1)$, $(t-\tau_2, t+\tau_2)$, $(t-\tau_3, t+\tau_3)$ etc. these are all parallel to the tangent in t (see fig. 5.3).

2. A parabola has a single point at infinity. This is a property of conic sections: if the parabola is intersected with the line at infinity of the plane that contains it, the solution is a double point. To put it figuratively: the time increases, the further the two "arms" of a parabola become more parallel. In the limit, at in-

finity, they are parallel, and are also parallel to the axis of the parabola. It is known that parallel line meet at a common point at infinity, the parabola "arms" meet in a common point at infinity as well.

As a consequence, the first thing to do is to find the trajectory plane. From detected parabola points in 4.2.1, two family of secants are identified coming from two different time instants t_1 and t_2. As stated in property number 1 these lines are parallel, and as a result their images meet in two dinstinct points at infinity. The line joinig the 2 points is the line at infinity of the trajectory plane. Now property number 2 is exploited. The parabola meets the line at infinity in a double point that is the point at infinity in the direction of the gravitational force.

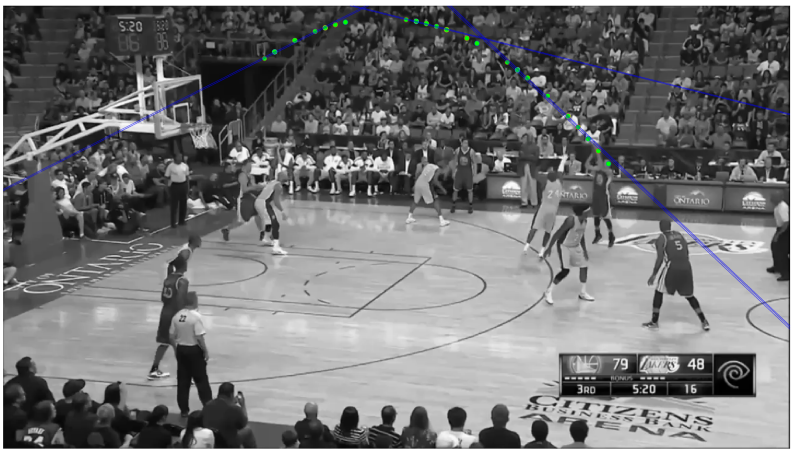


Figure 5.4: Real case parallel line from 3 central points. The lines will meet in 3 vanishing points outside the image, the line passing through them is the line at infinity of the trajectory plane.

From a calibrated camera the orientation of the trajectory plane can be defined form backpropagation of the line at infinity. Intersecting the viewing ray of the image points of the ball location along the parabola with the trajectory plane, 3D points of the parabola are found.

Only scale factor λ il left. To estimate this parameter are again used the parallell secants and the gravitational force. In fact, the separation between the parallel are ruled by the gravitational law. Consider the line tangent to the parabola at time t and the secant from $t - \tau_1$ to $t + \tau_1$, the distance between those lines is given by:

$$\text{dist} = \frac{1}{2} g \tau^2 \tag{5.2}$$

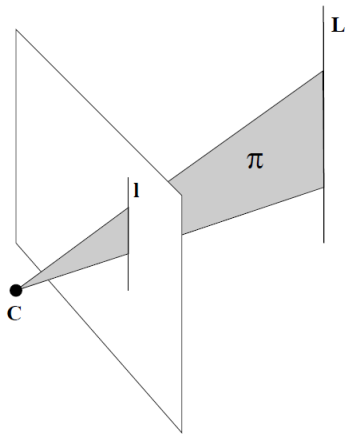


Figure 5.5: A line *l* in image plane is the backbrojection of a 3D line L. Plane π is defined by intersecting L and the camera center C.

Unforunetely, in the basketball case, if the camera is not perfectly calibrated and ball centre locations are not idendified precisely, this method bring to ill conditionated solutions, due to numerical errors. Hence another approach was chosen to estimate 3D points.

5.2 REAL WORLD BALL POSITION FROM PHYSICS

In a shot sequence, the basketball makes parabolic trajectories that can be modeled by:

$$\begin{aligned} X_t &= X_0 + V_X t \\ Y_t &= Y_0 + V_Y t \\ Z_t &= Z_0 + V_Z t + \frac{1}{2} g t^2 \end{aligned} \tag{5.3}$$

with (X_t, Y_t, Z_t) being the 3D ball coordinate at time t, (X_0, Y_0, Z_0) being the starting position, (V_X, V_Y, V_Z) the velocity and g is obviously the gravitational force. Define the 3D world coordinate as $W_t = (X_t, Y_t, Z_t)^T$ and the 2D image coordinate $m_t = (x_t, y_t, 1)$, from each point correspondance equation 5.1 can be rewritten as (recall DLT algorithm in section 2.2.1):

$$\begin{pmatrix} 0^T & -W^T & yW^T \\ W^T & 0^T & -xW^T \\ -yW^T & xW^T & 0^T \end{pmatrix} \begin{pmatrix} P^1 \\ P^2 \\ P^3 \end{pmatrix} = 0 \tag{5.4}$$

where P^{iT} is a the i^{th} row of matrix **P**. As in homography case, only the first two rows of equation 5.4 are linearly independent, so each point correspondance brings two equations:

$$\begin{pmatrix} 0^T & -W^T & yW^T \\ W^T & 0^T & -xW^T \end{pmatrix} \begin{pmatrix} P^1 \\ P^2 \\ P^3 \end{pmatrix} = 0 \tag{5.5}$$

and introducing $W_t = (X_t, Y_t, Z_t)^T$ and $m_t = (x_t, y_t, 1)$ the equation 5.5 becomes:

$$\begin{pmatrix} 0^T & -W_t^T & y_t W_t^T \\ W_t^T & 0^T & -x_t W_t^T \end{pmatrix} \begin{pmatrix} P^1 \\ P^2 \\ P^3 \end{pmatrix} = 0 \tag{5.6}$$

From N detected ball locations are obtained 2N equations. For each equation the 2D image coordinate $(x_t, y_t, 1)$, the elements P_{ij} of matrix P as well as time t are kown, a linear system can be written where the unknowns are $(X_0, V_x, Y_0, V_y, Z_0, V_z)^T = M^B_{6x1}$.
Therefore there is to compute 6 unknowns from a overdetermined

system with $M^{A}_{2Nx6}\ M^{B}_{6x1} = M^{C}_{2Nx1}$ and DLT algorithm is used to reconstruct 3D trajectory. The system obtained looks like:

$$
\begin{bmatrix}
p_{11}-x_1p_{31} & p_{11}t_1-x_1p_{31}t_1 & p_{12}-x_1p_{32} & p_{12}t_1-x_1p_{32}t_1 & p_{13}-x_1p_{33} & p_{13}t_1-x_1p_{33}t_1 \\
p_{21}-y_1p_{31} & p_{21}t_1-y_1p_{31}t_1 & p_{22}-y_1p_{32} & p_{22}t_1-y_1p_{32}t_1 & p_{23}-y_1p_{33} & p_{23}t_1-y_1p_{33}t_1 \\
p_{11}-x_2p_{31} & p_{11}t_2-x_2p_{31}t_2 & p_{12}-x_2p_{32} & p_{12}t_2-x_2p_{32}t_2 & p_{13}-x_2p_{33} & p_{13}t_2-x_2p_{33}t_2 \\
p_{21}-y_2p_{31} & p_{21}t_2-y_2p_{31}t_2 & p_{22}-y_2p_{32} & p_{22}t_2-y_2p_{32}t_2 & p_{23}-y_2p_{33} & p_{23}t_2-y_2p_{33}t_2 \\
 & & \vdots & & & \\
p_{11}-x_Np_{31} & p_{11}t_N-x_Np_{31}t_N & p_{12}-x_Np_{32} & p_{12}t_N-x_Np_{32}t_N & p_{13}-x_Np_{33} & p_{13}t_N-x_Np_{33}t_N \\
p_{21}-y_Np_{31} & p_{21}t_N-y_Np_{31}t_N & p_{22}-y_Np_{32} & p_{22}t_N-y_Np_{32}t_N & p_{23}-y_Np_{33} & p_{23}t_N-y_Np_{33}t_N
\end{bmatrix}
\begin{pmatrix} X_0 \\ V_x \\ Y_0 \\ V_y \\ Z_0 \\ V_z \end{pmatrix}
=
\begin{bmatrix}
x_1(p_{33}gt_1^2/2+1)-(p_{13}gt_1^2/2+p_{14}) \\
y_1(p_{33}gt_1^2/2+1)-(p_{23}gt_1^2/2+p_{24}) \\
x_2(p_{33}gt_2^2/2+1)-(p_{13}gt_2^2/2+p_{14}) \\
y_2(p_{33}gt_2^2/2+1)-(p_{23}gt_2^2/2+p_{24}) \\
\vdots \\
x_N(p_{33}gt_N^2/2+1)-(p_{13}gt_N^2/2+p_{14}) \\
y_N(p_{33}gt_N^2/2+1)-(p_{23}gt_N^2/2+p_{24})
\end{bmatrix}
\tag{5.7}
$$

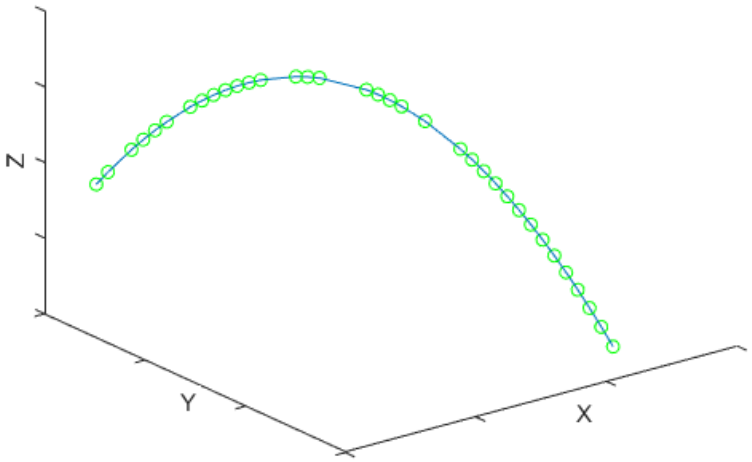


Figure 5.6: Extracted 3D trajectory example

In this case 3D ball trajectory can be extracted in a quite good manner, see figure 5.6.

Now that the 3D ball trajectory has been estimated, it can be used in the so called *Trajectory based applications*.

6 TRAJECTORY BASED APPLICATIONS

This section presents several applications based on the acquired 2D and 3D trajectories to demonstrate the utility of the proposed work. The trajectory-based applications usually have the scope to help coaches and players in game strategy developement, but also broadcasters.

6.1 SHOOTING POSITION EXTRACTION

The shooting location is defined as the position where the player leaves the ball. However, the starting position of the ball may not be perfectly detected, it may happen that the first detected ball appear after some frames. The solution adopted to estimate ball location works in this way:

1. Find ball trajectories as explained in chapter 4
2. Detect upright people using HOG features with Matlab's built-in function
3. Extend ball trajectory using the property of the x and y-candidate plots (4.2).
4. Find the intersection between extended ball trajectory and the bounding box detecting a player to define the shooter
5. The intersection point is considered in the 3D projection, and by considering the projection of the 3D point onto the XY plane the shooting location is estimated.

The main issue that of this system is that not always the matlab built-in application detects the shooter, so there is still room for improvements. This issued was not solved, and shooting position is extracted by taking the projection on the ground of the last location of the extended 3D trajectory.

6.2 MAKE OR MISS DETECTION

Rationally, the immediate step after shot trajectory extraction would be detecting if the shot went into the basket. A made shot presents just one trajectory, while a miss present two distinct and clearly visible parabolas. Slightly changing the algorithm described in section 4.2 in a way that in the y-candidate plot more than one parabola is identified, allows to detect a miss. In fact, if the tracking algorithm detects 2 parabolas, the shot can be considered a miss.

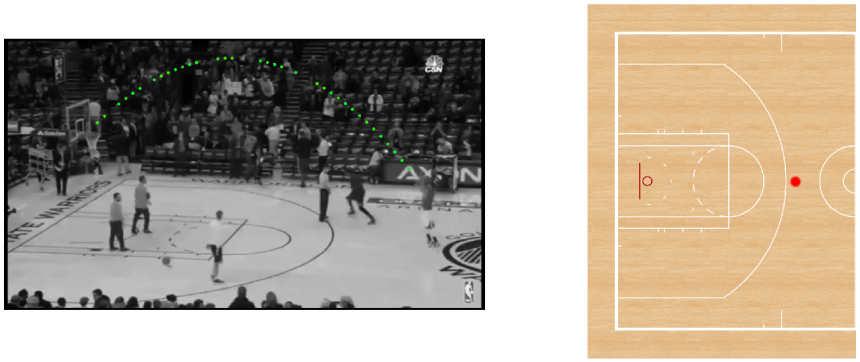


Figure 6.1: Extracted Shot Chart in a correct case.

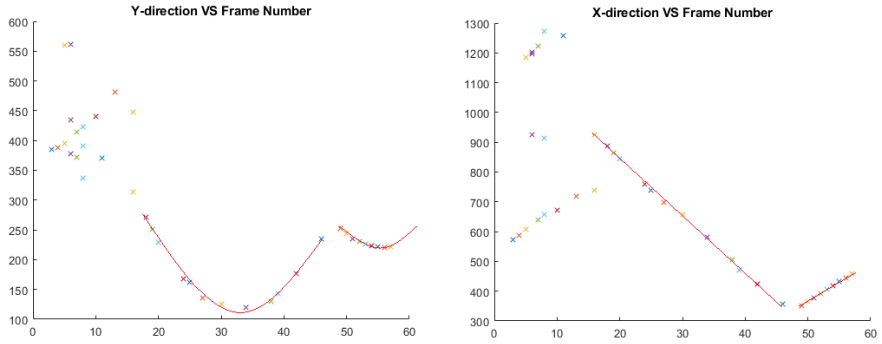


Figure 6.2: Trajectories of a Missed Shot.

Unfortunately, an error in basketball could result in very different ways. The shot may be so bad that it completely misses the rim, or it could be short and hit the rim "tangentially" and as a result the ball does not bounce back but just falls. Another important problem is introduced by the backboard; a missed shot might hit the rim and then bounce back on the backboard, then back onto rim or court. In other words, too many factors affect the ball and rim interaction.
In all these cases, the trajectory of the shot would not be identify as a parabola in the y-candidate plot. Therefore, this application is only suitable for particular type of shots, resulting in a limited analysis. A manual correction of misdetection would be needed to have a perfectly functioning system.

6.3 AUTOMATIC SHOT CHART GENERATION

Shot charts are the basic tool to understand where a particular player or a team shoots the ball. It is a fairly simple-to-read graph: it repre-

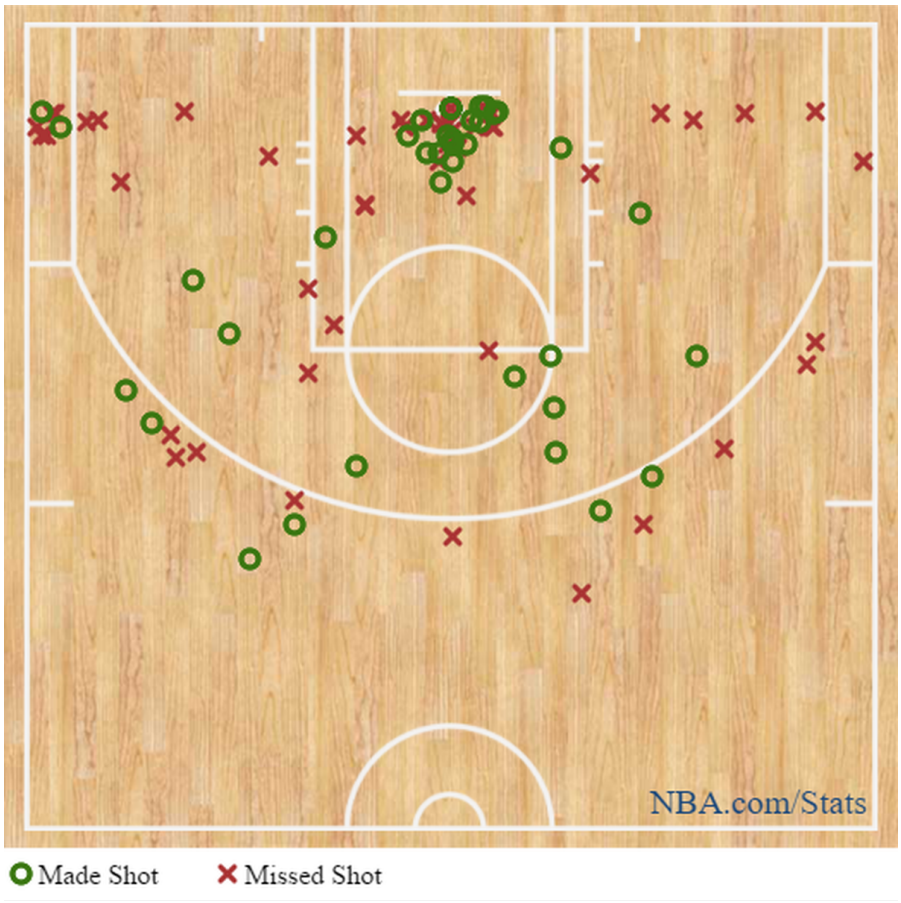


Figure 6.3: Example of a shot chart taken from an NBA game.

sent a birds eye view of the court with all shooting locations added. A shot is marked with a cross was missed or a circle if it was made. Taking advantage of the previous algorithms, altough they are limited, an automatic shot chart can be made from more than one shot sequence.

After having identified a shot as a make or a miss from the 2D trajectory, the algorithm extract the estimated shooting location from the 3D trajectory and plots it on the birds eye view of the court with a cross or a circle based on the make/miss detection.

Other possible application would require the NBA tracking dataset and machine learning algorithms. This application is not deeply treated beacause it is mostly related to statistical analys, but is just introduced briefly just to show another potential usage of the system.

6.3.1 *Rebound Position Estimation*

As noted before, since most missed shots necessarily involve the ball bouncing off some combination of rim and backboard, there is no

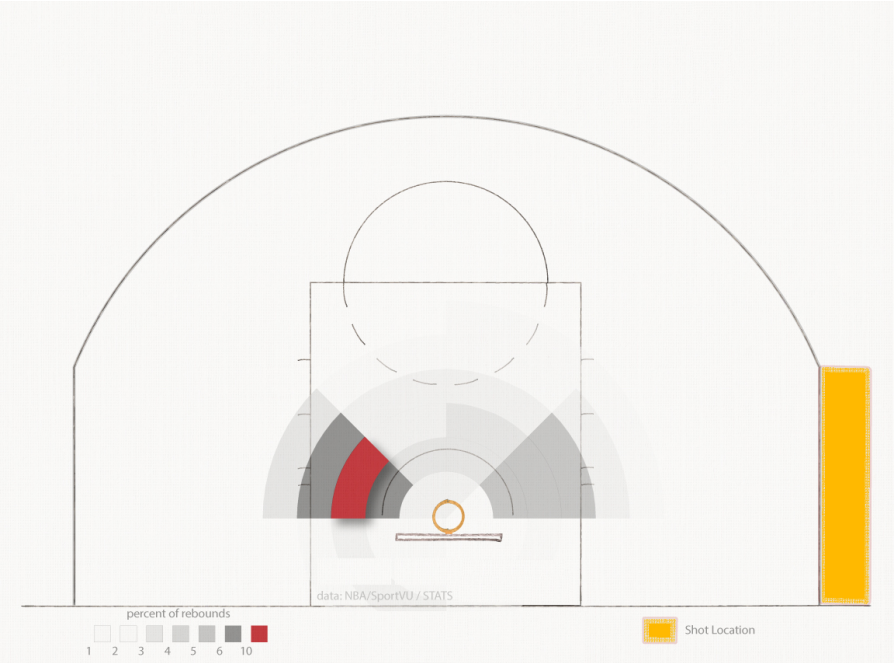


Figure 6.4: Statistical Rebound Position Estimation [12]

way to predict exactly where rebounds will go.

Thanks to tracking data this may be possible. The league's tracking system not only logs locations for *every* missed shot, it also charts the locations of their subsequent rebounds. Therefore, a statistical conclusion can be drawn from previous rebound and shooting posiotion, and the current estimated shooting location.
Of course the result will not be an absolute estimated location for rebound position, but it will be a proability map as it can be seen in 6.4. This is a consequence of the randomness that characterizes every field goal. The outcome will be guided by some fundamental principles, but some randomizing effect remains. A sure consequence of statistical analysis is that most rebounds falls araound the basket area, but the exact up to a millimeter location is not predictable.

7

CONCLUSION AND FUTURE DEVELOPMENTS

In this thesis, a 3D trajectory reconstruction from a single view video o basketball shots has been developed and successfully tested. Analysis on real application of this kind of system have been presented and have displayed what benefit it could bring to game study and broadcasting.

The calibration algorithm can be applied to any kind of sport, provided that a 5^{th} noncoplanar point can be found. In example, in volleyball or tennis the net post can be used, while in soccer one of the corner of the goal post. The calibration from 5 points and their automatic estimation from single image gives speed to the algorithm, and allows to find all the calibration parameters. Therefore it opens the possibility to use the system with a high amount of basketball videos. All videos analysed in this work was taken from uncalibrated cameras.

Detection and tracking processes are reliable and can withstand illumination variations, occlusion and fast movements of the ball; moreover, thanks to the introduction of the Kalman filter in trajectory processing the possibility of trajectory misdetection is greatly reduced.

The employment of physical properties of ball motion allows to extract the 3D trajectory with suitable results. Unfortunately, the geometry trajectory process bring to ill conditioned solution, and therefore it was disregarded.
During testing on videos, no particular problem are identified, except from stabilization problems, as explained, and ball extraction in videos where the background motion was too high. Those problem can be avoided with a video mosaicing stabilization and by adding more than one camera, respectively.

Although the system works accurately in many cases, performance can be increased, especially in calibration. Adoption of a camera calibrated offline could be a suitable solution to accuracy problems in the system. A further improvement could be brought by the insertion of another camera. This will bring to a more costly solution, but the ball 3D position may be estimated more precisely allowing the system to gain robustness.

Possible future developments may aim to track the ball and players in a whole game action: this is a difficult task since the system must deal occlusion and unpredictable ball movements. For example training and inserting a neural network that works alongside video detection algorithm could help identifying the trajectory; another possible improvement would be, as introduced in previous section, to use tracking data to make a prediction on rebound position.

BIBLIOGRAPHY

[1] David Aldrige. *SportVU cameras shift focus of what's possible with NBA stats*. http://www.nba.com/2013/news/features/david_aldridge/11/11/morning-tip-sportvu-cameras-in-arenas-problems-with-nets-qa-with-paul-george/. 2013 (cit. on p. 8).

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." In: *In ECCV*. 2006, pp. 404–417 (cit. on p. 23).

[3] M. Bujnak, Z. Kukelova, and T. Pajdla. "A general solution to the P4P problem for camera with unknown focal length." In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8 (cit. on p. 6).

[4] B. Chakraborty and S. Meher. "Real-time position estimation and tracking of a basketball." In: *2012 IEEE International Conference on Signal Processing, Computing and Control*. 2012, pp. 1–6 (cit. on p. 11).

[5] Bodhisattwa Chakraborty and Sukadev Meher. "A Trajectory-Based Ball Detection and Tracking System with Applications to Shooting Angle and Velocity Estimation in Basketball Videos." In: *2013 Annual IEEE India Conference, INDICON 2013*. Dec. 2013 (cit. on p. 12).

[6] H. Chen. "Geometry-Based Camera Calibration Using Five-Point Correspondences From a Single Image." In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.12 (2017), pp. 2555–2566. ISSN: 1051-8215 (cit. on p. 33).

[7] Roberto Cipolla, Tom Drummond, and Duncan P. Robertson. "Camera Calibration from Vanishing Points in Image of Architectural Scenes." In: *BMVC*. 1999 (cit. on p. 6).

[8] Harry Collins and Robert Evans. "You cannot be serious! Public understanding of technology with special reference to "Hawk-Eye"." In: *Public Understanding of Science* 3 (2008), pp. 283–308 (cit. on p. 8).

[9] D. Farin, J. Han, and P. H. N. de With. "Fast camera calibration for the analysis of sport sequences." In: *2005 IEEE International Conference on Multimedia and Expo*. 2005, 4 pp.– (cit. on p. 6).

[10] H. Faulkner and A. Dick. "AFL Player Detection and Tracking." In: *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. 2015, pp. 1–8 (cit. on p. 11).

[11] FIFA. *FIFA - Goal Line Technology Standards*. https://football-technology.fifa.com/en/standards/goal-line-technology/ (cit. on p. 8).

[12] Kirk Goldsberry. *How Rebound Work*. http://grantland.com/features/how-rebounds-work/. 2013 (cit. on p. 64).

[13] Josef Halbinger and Juergen Metzler. "Video-Based Soccer Ball Detection in Difficult Situations." In: *Sports Science Research and Technology Support*. Ed. by Jan Cabri, Pedro Pezarat Correia, and João Barreiros. Cham: Springer International Publishing, 2015, pp. 17–24 (cit. on p. 11).

[14] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. New York, NY, USA: Cambridge University Press, 2003. ISBN: 0521540518 (cit. on pp. 18, 24, 34).

[15] M. Hu, M. Chang, J. Wu, and L. Chi. "Robust Camera Calibration and Player Tracking in Broadcast Basketball Video." In: *IEEE Transactions on Multimedia* 13.2 (2011), pp. 266–279. ISSN: 1520-9210 (cit. on p. 6).

[16] Z. Ivankovic, B. Markoski, M. Ivkovic, D. Radosav, and P. Pecev. "AdaBoost in basketball player identification." In: *2012 IEEE 13th International Symposium on Computational Intelligence and Informatics (CINTI)*. 2012, pp. 151–156 (cit. on p. 11).

[17] Pakorn KaewTrakulPong. "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection." In: 2001 (cit. on p. 45).

[18] D. G. Lowe. "Object recognition from local scale-invariant features." In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2 (cit. on p. 23).

[19] W. Lu, J. Ting, K. P. Murphy, and J. J. Little. "Identifying players in broadcast sports videos using conditional random fields." In: *CVPR 2011*. 2011, pp. 3249–3256 (cit. on p. 11).

[20] Upendra Rao M. and U. C. Pati. "A novel algorithm for detection of soccer ball and player." In: *2015 International Conference on Communications and Signal Processing (ICCSP)*. 2015, pp. 0344–0348 (cit. on p. 10).

[21] I. Miyagawa, H. Arai, and H. Koike. "Simple Camera Calibration From a Single Image Using Five Points on Two Orthogonal 1-D Objects." In: *IEEE Transactions on Image Processing* 19.6 (2010), pp. 1528–1538. ISSN: 1057-7149 (cit. on p. 6).

[22] N. Owens, C. Harris, and C. Stennett. "Hawk-eye tennis system." In: *2003 International Conference on Visual Information Engineering VIE 2003*. 2003, pp. 182–185 (cit. on pp. xv, 7, 8).

[23] J. Puwein, R. Ziegler, L. Ballan, and M. Pollefeys. "PTZ camera network calibration from moving people in sports broadcasts." In: *2012 IEEE Workshop on the Applications of Computer Vision (WACV)*. 2012, pp. 25–32 (cit. on p. 6).

[24] V. Renò, N. Mosca, M. Nitti, T. DOrazio, D. Campagnoli, A. Prati, and E. Stella. "Tennis Player Segmentation for Semantic Behavior Analysis." In: *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*. 2015, pp. 718–725 (cit. on p. 10).

[25] E. Rosten and T. Drummond. "Fusing points and lines for high performance tracking." In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. 2005, 1508–1515 Vol. 2 (cit. on pp. 20–22).

[26] Edward Rosten and Tom Drummond. "Machine Learning for High-speed Corner Detection." In: *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*. ECCV'06. Graz, Austria: Springer-Verlag, 2006, pp. 430–443. ISBN: 3-540-33832-2, 978-3-540-33832-1 (cit. on pp. 20–22).

[27] Graham A. Thomas. "Real-time camera tracking using sports pitch markings." In: *Journal of Real-Time Image Processing* 2 (2007), pp. 117–132 (cit. on p. 6).

[28] TrackVision. *Sensor-free First Down Line System*. https://www.avid.com/~/media/avid/files/products-pdf/trackvision-fdl/avid_trackvisionfdl_ds_a4.pdf. 2016 (cit. on p. 7).

[29] Ling Ling Wang and Wen-Hsiang Tsai. "Camera calibration by vanishing lines for 3-D computer vision." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.4 (1991), pp. 370–376. ISSN: 0162-8828 (cit. on p. 6).

[30] X. Wei, L. Sha, P. Lucey, P. Carr, S. Sridharan, and I. Matthews. "Predicting Ball Ownership in Basketball from a Monocular View Using Only Player Trajectories." In: *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*. 2015, pp. 780–787 (cit. on p. 11).

[31] Lifang Wu, Xianglong Meng, Xun Liu, and Shiju Chen. "A New Method of Object Segmentation in the Basketball Videos." In: *18th International Conference on Pattern Recognition (ICPR'06)*. Vol. 1. 2006, pp. 319–322 (cit. on p. 11).

[32] Fei Yan, William J. Christmas, and Josef Kittler. "A Tennis Ball Tracking Algorithm for Automatic Annotation of Tennis Match." In: *BMVC*. 2005 (cit. on p. 11).

APPENDIX: A

A.1 KALMAN FILTER: BASICS AND EQUATIONS

The Kalman Filter is the most famous and used data fusion algorithm in field of information processing. Nowadays Kalman filters are employed in many fields and objects, from the smartphones to the satellite navigation. It is usually derived from linear algebra as a minimum least square estimator. Assumption: the system at time t comes from the prior state at time instant $t-1$ according to:

$$x_t = F_t x_{t-1} + B_t u_t w_t \tag{A.1}$$

Where:

- x_t is the state vector containing the terms of interest for the system.
- u_t is the inputs vector containing the control inputs.
- A_t is the state transition matrix. Applies the effect of each system state at time $t-1$£ on the system state at time t. In example, position and velocity at $t-1$ have a consequence on the position at time t.
- B_t is the control input matrix. Applies the effect of the control inputs u_t on the state vector
- w_t is the state vector containing the process noise for each parameter in x_t. It is assumed to derive form a zero mean multivariate normal distribution, with Q_t as covariance matrix

From the system measurements, the model can drawn:

$$z_t = H_t x_t + v_t \tag{A.2}$$

With:

- z_t is the measurements vector.
- H_t is the transformation matrix. Relates state vector parameters into measurements domain.
- v_t is the vector containing the measurement noise. Again, it is assumed to derive form a zero mean multivariate normal distribution, with R_t as covariance matrix. (Note that process noise and measurement noise are statistically uncorrelated)

In case no control input is applied: $B\,u_t = 0$. Once the model has been fitted into the Kalman filter the initial values and parameters can be estimated. At k_t state:

$$\begin{aligned}
&\textit{Time Update (Prediction)}\\
&x_k = Ax_{k-1} + Bu_k\\
&P_k^- = AP_{k-1}A^T + Q\\
&\\
&\textit{Measurement Update (Correction)}\\
&K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}\\
&x_k = x_k^- + K_k(z_k - H_k^-)\\
&P_k = (I - K_k H)P_k^-
\end{aligned} \tag{A.3}$$

The next and last step is iteration of the process: after initial estimate, the prediction is calculated, then measurement corrected and outputs at k will be the new input to time prediction for $k+1$ and so on. The initial state estimation matrix **A** and transformation matrix **H** for the basketball case were introduced in 4.13:

$$A = \begin{pmatrix} 1 & 0 & 1 & 0\\ 0 & 1 & 0 & 1\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 \end{pmatrix}; \quad H = \begin{pmatrix} 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{A.4}$$

Noises are initialized in order to have as covariance matrices the identity matrix of the size of the state and observation vector.