Badminton shot classification in compressed video with baseline angled camera

Sam Careelmont

Kongens Lyngby 2013
IMM-M.Sc.-2013-39

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-M.Sc.-2013-39

Summary (English)

The goal of the thesis is to classify shots played during a recorded badminton match. The classification consists of a time chronological list of every shot's start and end position. Such classification offers tactical information which can be used for coaching or display during broadcast matches. The common camera position in badminton match recordings is behind the baseline of the court. Compressed video files are considered to accommodate home recordings and saved broadcasts.

An algorithm is developed that outputs a classification given a certain input video. The algorithm extract moving objects in every frame and connects them through time. Next, a shuttle trajectory model is fitted to the extracted data. A classification is made based on the best fitting model appearance. The moving object extraction uses background subtraction and connected component labelling. Connecting the components through time is achieved by a custom matching step. The shuttle trajectory model uses a physics based shuttle trajectory description and a camera calibration to establish a model of the image plane behaviour of a shuttlecock.

The results show that on average 8 percent of the shots' start and end position can be classified correctly. The overall performance is not satisfying but intermediate steps show promising results. Moreover, various bottlenecks are identified and an extensive list of suggestions for further research is given.

Summary (Danish)

Målet for denne afhandling er at klassificere slagspillet i løbet af en optaget badmintonkamp. Klassifikationen består af en tidskronologisk liste af hvert slags start -og slut position. Sådan en klassifikation tilbyder taktiske oplysninger som kan blive brugt til træning eller visning i udsendte kampe. Den almindelige kameraposition i badmintonoptagelse er bag banens baglinje. Komprimeret videoer er undersøgt for at medregne hjemmeoptagelse og gemte udsendelser.

En algoritme er udviklet som udlæser en klassifikation forudsat en korrekt indlæst video. Algoritmen tager bevægende objekter ud af hvert delbillede og forbinder dem igennem tid. Følgelig er en fjerboldforløbs model tilpasset de udtaget data. En klassifikation er lavet baseret på den bedst tilpassede model. Den bevægende objektfrilæggelse bruger baggrundssubtraktion og sammenhængende-komponent-mærkering. Forbindelse af komponenterne gennem tid er opnået ved et specialbygget matchetrin. Fjerboldforløbs modellen anvender en fysisk baseret fjerboldforløbsbeskrivelse og en kamerakalibrering for at skabe en model af fjerboldens adfærd i billedplanet.

Resultaterne viser at i gennemsnit er 8 procent af slagenes start -og slut positionerne klassificeret korrekt. Den generelle præstation er ikke tilfredsstillende men deltrinnene viser lovende resultater. Endvidere er forskellige flaskehalse identificeret og en omfattende liste med forslag for videregående undersøgelse er givet.

Summary (Dutch)

Het doel van deze verhandeling is het classificeren van slagen gespeeld tijdens een opgenomen badminton wedstrijd. The classificatie bestaat uit een tijd chronologische lijst van elke slag zijn begin end eind positie. Dergelijke classificatie biedt tactische informatie die gebruikt kan worden voor coaching of vertoning tijdens uitgezonden wedstrijden. De gebruikelijke camera positie in badminton opnames is achter de basislijn van het veld. Gecomprimeerde video bestanden worden beschouwd om thuis opnames en opgeslagen uitzendingen te kunnen behandelen.

Een algoritme is ontwikkeld dat een classificatie uitvoert gegeven een zekere invoer. Het algoritme extraheert bewegende objecten in elk video frame en verbindt ze doorheen de tijd. Vervolgens wordt een shuttle traject model gepast aan de geëxtraheerde gegevens. Een classificatie wordt gemaakt van de best passende modelvorm. De bewegende objecten worden geëxtraheerd middels achtergrond aftrekking en het labelen van verbonden componenten. Het verbinden van deze componenten doorheen de tijd gebeurt aan de hand van een op maat gemaakte stap. Het shuttle traject model gebruikt een fysische shuttle traject beschrijving en een camera calibratie om een model te vormen van het shuttle gedrag op het afgebeelde vlak.

De resultaten tonen dat gemiddeld 8 procent start -en eindposities van de slagen correct geclassificeerd werden. De algemene prestatie is niet voldoende maar deelstappen van het algoritme tonen veelbelovende resultaten. Daarenboven zijn verschillende knelpunten geïdentificeerd en een uitgebreide lijst van suggesties voor verder onderzoek is opgesteld.

Preface

This thesis was prepared during an exchange stay at the department of Informatics and Mathematical Modelling at the Technical University of Denmark. The thesis fulfils the requirements for acquiring an M.Sc. in Computer Science from Ghent University.

The thesis deals with the classification of badminton shots in compressed video recorded with a baseline angled camera.

The thesis consists of seven chapters. The introduction specifies the considered problem, motivates the proposed goals and elaborates on related work. The next four chapters describe the developed method. Chapter six summarizes the most important results. Concluding remarks and future work are listed in the last chapter.

Lyngby, 01-June-2013

Sam Careelmont

Acknowledgements

This thesis was realized with the direct and indirect help from many people. I would like to express my sincere thanks to:

- My supervisors at the Technical University of Denmark Prof. Anders Lindbjerg Dahl and Prof. Henrik Aanæs for offering me the chance to be part of their research group, following my progress during weekly meetings and guiding me towards the essence.
- My supervisor at Ghent University Dr.Ir. Peter Van Hese for tracking my progress through e-mail and providing helpful advice.
- Prof. Wilfried Philips for coordinating my thesis and offering me the chance to carry out my research abroad.
- Prof. Filip De Turck for allowing me to pursue my last master's year as an exchange student at the Technical University of Denmark.
- My parents, family and friends for their unconditional support.
- The Belgian state, the Danish state and the European Union for creating an educational climate with lots of learning opportunities.

Contents

Chapter 1

Introduction

Spectator participation in 776BC Olympia required attendance of the games. Three millennia later in a world vastly changed, the 1936 Berlin Olympic games showcased the world's first live broadcast of a sport event. Ever since this ground breaking event there has been an increase in sport coverage on television. Contemporary spectators are even given the choice of on demand sport transmissions through the internet. Sport videos have been and are playing an important role in the popularisation of sport.

Spectators aside, coaches and athletes also stand to benefit from video technology and it is known that sport video sequences are widely used for coaching instruction. An athlete's technical skills can be analysed by looking at an isolated movement. On a more global scale the coach can analyse and rectify the movements of the athlete and the tactical choices. No matter which approach is taken, video sequences offer useful information for post-processing any sport event. One of particular interest is badminton. Since it has a large tactical component, viewing and analytical opportunities for both spectators and coaches arise. A broadcast badminton game can be made more interesting by adding advanced statistics of the shots played. Coaches can use recordings to analyse the tactical implications of the athlete's shot decisions. This can be done before a match in order to prepare the athlete for a certain opponent, or afterwards to learn for future matches.

Through these analytic implications badminton game videos become very interesting. What if the game analysis could be done automatically? In this thesis research is carried out to track the shuttle in compressed video sequences with a baseline angled camera. The end goal is to have a solid base to perform a stable shot classification of every shot played in a complete badminton match.

The use of compressed videos with baseline angled camera instead of 3D images retrieved from multiple high definition cameras has some reasons. Firstly, the baseline angled camera has been the standard camera angle in badminton broadcasts for decades. Secondly, working with low quality images enlarges the pool of test data significantly. Older recordings can also be used for analysis. Consequently, our method might be useful for archiving purposes of large amounts of badminton videos. Also lots of data triggers statistical meaningful options to perform fundamental sport research. Sport scientists would be able to describe the evolution of the badminton game or generalize the current established tactics. Another advantage of working with compressed video are the possibilities for home use. Imagine recording a badminton match yourself with a smartphone, analysing it at home and uploading the match report on a social network. From the broadcasting point of view, robust methods offer small broadcast companies the possibility to display match statistics without the need to buy expensive high definition cameras.

1.1 Related work

A general overview of video technology use for coaching is given by Wilson [Wil08]. The use of video for coaching goes together with the increased availability of home computers. It started with plain recordings of sport events and manual reviews of the coaches. Today's systems should help the coach in a smart way and as close as possible to real time.

Yi et al. [YRC04] focus on a general semantic understanding of sport videos and extract motion trajectories in compressed videos. The generality of their method outlines the steps needed to perform a trajectory extraction.

More sport specific research was done for baseball [SK04], volleyball [CCL07] , basketball [CTC+09] and soccer [KSH98], [YXL+03]. These all try to make inferences about the real world trajectory of the ball based on 2D image sequences. The goal of this thesis is very alike except for the tracked object. Balls have a parabolic flight while a shuttlecock follow a more specific path (see chapter 4). Hence, these papers offer ideas for trajectory extraction and real world trajectory inferences, yet not every step is directly applicable or useful to solve this thesis

problem.

Several efforts have been made to analyse broadcast court-net sport videos. Zhu et al. [HFdW08] focus on the organization of extracted highlights based on a variety of features and support vector regression. Han et al. [ZHX+07] also try to detect certain key events by the use of player tracking and a Bayesian-based classifier. The goal of these studies differs from ours as no coaching use is aimed. However, some processing steps apply to our research. For example, an automatic ground homography is extended to a full camera calibration by assuming that height changes of real world objects only influence the vertical coordinate in the frame. Despite the good results these papers achieve, the overall outcome is not applicable for tactical analysis of a a badminton game.

Lastly, other research was specifically dedicated to badminton video analysis. [CW07] detects the shuttlecock trajectory based on 2D seriate images. All moving regions are detected using temporal difference. Trajectories are put together using properties of the moving regions in each frame. After the identification of all trajectories, the shuttlecock trajectory is filtered out by mainly looking at the trajectory length. By using knowledge about the stroke characteristics, the stroke type is determined. The method yields very good results but the camera was placed at an optimal position. Also, only six types of strokes are distinguished. Automated service scene detection in badminton game videos was done by Yoshikawa et al. [YKWO10]. As opposed to our research, a ceiling camera was used instead of a baseline angled camera.

To summarize, trajectory extraction from single view videos has been done for other sports than badminton. Badminton shot classification has been done before, but no real applicable results were described.

More relevant work, related to one of the specific steps in the method, is introduced in the corresponding sections.

1.2 Research overview and goals

This thesis can be abstracted as follows: some non-perfect input is processed by some algorithm which outputs some raw number format. The non-perfect input consists of compressed video of badminton matches with a baseline angled camera. The raw output format is a formal description of all rallies played during the badminton match that is recorded on the input video. Next a more detailed description of the input material and output format is given. An extensive description of the processing algorithm can be found in the chapter 2.

1.2.1 Input material

As justified in the general introduction, compressed videos with a baseline angled camera are considered because it is the common broadcast format and easy to record a match yourself with these properties. Besides these advantages, the restrictions on the input material trigger numerous challenges.

Lossy video compression is assumed since most common codecs (MPEG-4 Part 2 and H.264/MPEG-4 AVC) include a quantization step in a standard fashion [LD03]. Consequently, the input material can suffer from degraded colour information. Moreover, the frame rate is limited to a maximum of 30 frames per second. These limitations might hinder the detection of the features of interest (e.g. shuttlecock detection, see figure 1.4(c)) and the use of certain methods. An example of the latter is the infeasibility of depth estimation by shuttle size. Size is an unstable factor when a shuttle is observed with, for example, motion blur.

Other challenges come with the camera stipulations. Due to the nature of single camera recordings, a 3D world gets projected on a 2D plane. In another parlance, positional information is lost during the creation of the video material. Here, the camera is positioned some meters behind the short side of the court. The camera is assumed to have a certain tilt, no or a small pan and zero roll. In this setup, one can intuitively see that width information is kept quite intact in the x-component of the projected image while height and depth are encoded in the y-component of the projected image. Hence, a mathematical transformation is needed to find more exact relationships between real world and projected coordinates. Also, shuttle position evolution over time needs to be handled to overcome some problems. An illustrative example of this problem is given in figure 1.5.

Another consequence of the camera position is the possibility of shuttlecock trajectories with non-registrable parts. Since only objects within a certain real world space are captured, high flying shuttles might not be recorded (see figure 1.4(d)).

The last type of challenges is inherent to the content of the videos. Namely, the shuttlecock can be occluded by players. Also, every part in the recorded scene that has the same color as the shuttlecock (white), possibly interferes with a distinct shuttlecock observation in the projected image domain. Examples are white court lines, white coloured advertising signs or spectators wearing some white clothing (see figure 1.4(b)).

1.2.1.1 Test data

The videos are selected to vary in size, encoding, color and general quality. To this end, every video comes from another tournament. Video descriptions are found in table 1.2.1.1, video properties in table 1.2.1.1 and example screenshots in figure 1.1. Videos 1 to 3 are used as a training set, videos 4 to 6 as a test set. The training videos help choosing optimal values for the algorithm parameters and accommodate intermediate testing of substeps. The test videos only occur in the results chapter and are used to test the general performance of the algorithm.

video id	video description
1	Macau Open 2007, Men's Singles Semi Final, Game 1, Rally at 3-3 (Taufik Hidayat versus Park Sung-hwan)
2	Asia Championships 2008, Men's Singles Semi Final, Game 2, Rally at 7-6 (Sony Dwi Kuncoro versus Park Sung-hwan)
3	China Open 2011, Women's Singles Quarter Final, Game 2, Rally at 10 16 (Tine Baun versus Wang Xin)
4	French Open 2007, Men's Singles Final, Game 2, Rally at 8-11 (Lee Chong Wei versus Bao Chunlai)
5	Australian Open 2013, Men's Singles Final, Game 1, Rally at 3-3 (Tian Houwei versus Xue Song)
6	Beijing Olympics 2008, Women's Singles Final, Game 1, Rally at 6-8 (Xie Xingfang versus Zhang Ning)

video id	codec	bitrate (Kbps)	size # frames	fps	length	colorspace	chrome subsampling
1	MPEG-4 Part 2	1061	720x544	25	34s 520ms	YUV	4:2:0
2	MPEG-4 Part 2	498	640x480	25	24s 320ms	YUV	4:2:0
3	H.264/MPEG-4 AVC	319	636x360	25	20s 440ms	YUV	4:2:0
4	MPEG-4 Part 2	638	720x544	25	18s 840ms	YUV	4:2:0
5	H.264/MPEG-4 AVC	328	640x360	25	34s 840ms	YUV	4:2:0
6	H.264/MPEG-4 AVC	795	640x480	25	20s 40ms	YUV	4:2:0

Note that the compressed videos all use chroma subsampling in a YUV colorspace. This results in frames where the color data is encoded as YCbCr. OpenCV uses a BGR encoding by default and converts the input video data using following

transformation:

$$\begin{cases} R = Y + 1.403(Cr - 128) \\ G = Y - 0.344(Cr - 128) - 0.714(Cb - 128) \\ B = Y + 1.773(Cb - 128) \end{cases} \tag{1.1}$$

1.2.2 Output format

The output is a list containing all rallies played during the input badminton match. Formally, the list contains chronologically ordered tuples $S_{i,j}$ (see definition 1.1). An example of the output format can be found in figure 1.3.

Definition 1.1 Let $S_{i,j}$ be a tuple $\in (z_{start}, z_{end})$ with
$i :=$ index of rally
$j :=$ index of shot
$z_{start} \in \mathbb{N}[1, 9]$
$z_{end} \in \mathbb{N}[1, 9]$

z_{start} and z_{end} indicate a zone as depicted in figure 1.2. Formally, the z_{start} and z_{end} number can be defined as the area wherein the perpendicular projection of the real world shuttle position falls at the start and end of the flight, respectively.

This encoding facilitates future developments in several ways. Firstly, the start and end position of the shuttle trajectories encompass the most fundamental ground truth needed for further deductions. Secondly, the tuples can be extended with new elements without affecting existing statistical post processing built on the previous encoding. Lastly, identifying trajectories in this way is one of the least time consuming method for manual annotation. This is an advantage in the creation process of ground truth data, which is used to test the algorithms performance.

Extensions are beyond the scope of this thesis. But one can easily think of a graphical presentation layer built on top of the raw data. This layer could visualise patterns in the played shots. Other ideas are situated around statical reasoning and querying possibilities.

(a) Video 1: Macau Open 2007

(b) Video 2: Asia Championships 2008

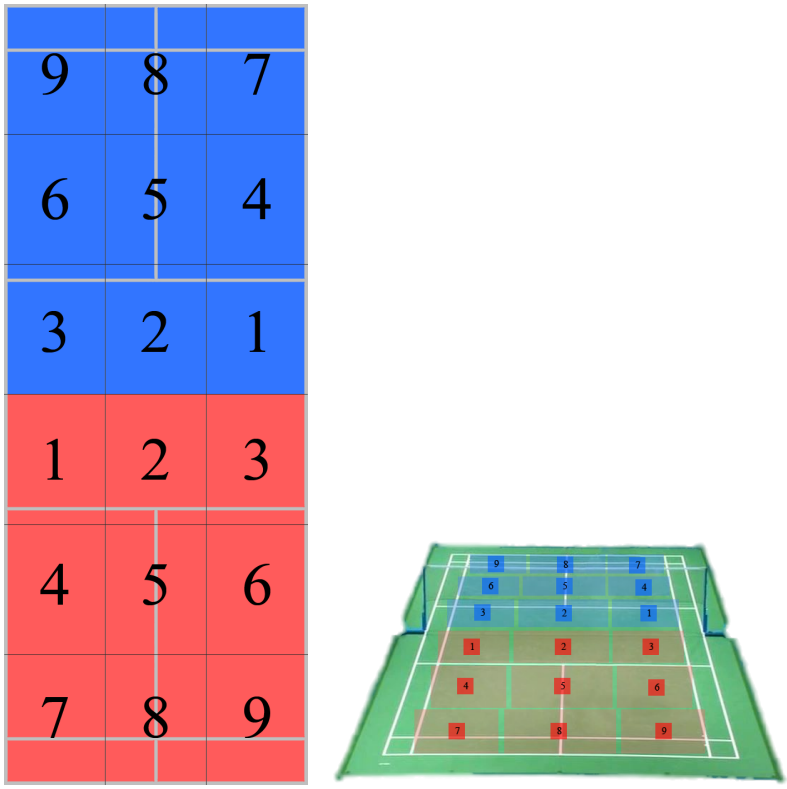
(c) Video 3: China Open 2011

(d) Video 4: French Open 2007

(e) Video 5: Australian Open 2013

(f) Video 6: Beijing Olympics 2008

Figure 1.1: Test videos



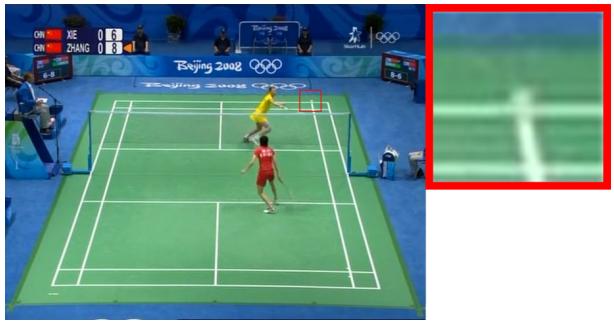
(a) Ground plane (b) Court from video extract

Figure 1.2: Visual representation of possible values for z_{start} and z_{end}.

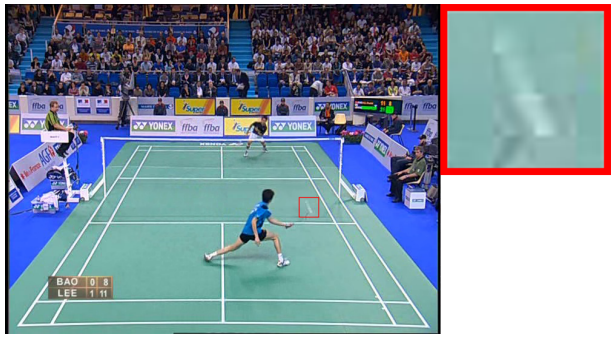
j	LAUNCH	ARRIVAL
1	1	3
2	3	1
3	1	8
4	8	5
5	5	1
6	1	1
7	1	9
8	9	4
9	4	6
10	6	1
11	1	9
12	9	4
13	4	3
14	3	7
15	7	6
16	6	7
17	7	4
18	4	6
19	6	7
20	7	9
21	9	1
22	1	8
23	8	8
24	8	1
25	1	7
26	7	9
27	9	3
28	3	7
29	7	9
30	9	4
31	4	6
32	6	6
33	6	4
34	4	1
35	1	9

Figure 1.3: Annotation example of Macau Open 2007, Men's Singles Semi Final, Game 1, Rally at 3-3.

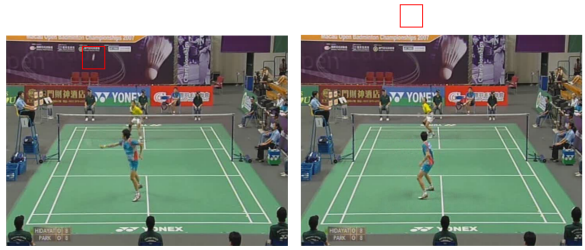
(a) An easy frame for shuttle detection. There is a high contrast between the shuttle and the background.



(b) The shuttle is close to the lines. These have the same colour, yielding possible confusion.



(c) The shuttle is stretched out due to motion blur. The low video resolution causes an extreme pixelation in this case.



(d) The shuttle disappears from the canvas.

Figure 1.4: Test videos

(a) The shuttle is at the far end of the court, flying towards the close end of the court in a straight trajectory.

(b) The shuttle is at the close end of the court, flying in a cross trajectory towards the closest player The shuttle will be hit in the right corner of the court as the player is late to intercept the shuttle.

(c) The shuttle has a similar flight as the shuttle in figure 1.5(b). However, the shuttle will be hit in the middle of the back court.

Figure 1.5: All shuttles in these frames can be detected as a blob on a dark background with some image plane coordinates. Solely this information is not sufficient to deduce the position of the shuttle in the real world. Humans can deduct information from the moving image sequence and the players movements. The algorithm has to consider players or shuttle evolution through time to infer more about the real world shuttle position.

1.3 The worldwide sport of badminton

Badminton is especially popular in Denmark, China, South Korea, Indonesia and Malaysia. However, the sport is played and watched all over the world. The Fifa World Cup Final 2010 was watched by 700 million people worldwide [Hib], while the number of viewers for the Olympic Badminton Final 2012 is estimated at 1.4 billion [citb]. About 20 percent of the world population was watching a badminton game. This underlies the immense popularity of the sport.

Chapter 2

Method

The processing algorithm can be split into two subalgorithms. The first takes the input video and outputs a list of trajectories. We call this the 'Trajectory extractor' (see figure 2.1(a)). The second starts from the list of trajectories and outputs a classification, further referred to as the 'Trajectory classifier' (see figure 2.1(b)).

The trajectory extractor consists of following steps: camera calibration, background subtraction, shuttle blob extraction and trajectory matching. To start, the camera calibration finds the parameters of the camera that produce images as observed. Thereupon, the background subtraction detects moving pixels by extracting the foreground of the image. These foreground pixels are clustered in connected components and non-interesting blobs are left out during the shuttle blob extraction step. At last, blobs are connected over time to create trajectories. Every step is described in detail in chapter 3.

The trajectory classifier is made up of following steps: false positive detection, initial parameter estimation, model fitting and classification. The false positive detection tries to rule out obvious outliers from the list of trajectories. Next, an initial parameter estimation is carried out to assist the consequent step where a 3D shuttle model is fitted to the trajectories. Finally, the eventual classification is performed. Chapter 5 describes the subalgorithm more specifically.

Chapter 4 describes in detail how the shuttlecock model is constructed. This chapter precedes chapter 5 since the latter refers extensively to the shuttlecock model in every subsection.

2.1 Terminology

Throughout this thesis, certain vocabulary is recurring. To not confuse the reader and to fully understand the arguments made, a list of important terminology is given:

General terms of a badminton match

Definition 2.1 A **rally** is a collection of shots that belong together because they are part of the same point in a certain badminton match

Definition 2.2 A **shot** is the flight of a shuttle from the moment a player hits the shuttle until the flight halted. A flight halt occurs when the opponent hits the shuttle, the shuttle hits the court net or the shuttle touches the ground.

Terms applying to the algorithms

Definition 2.3 A **blob trajectory** is a time chronological list of blob centroid coordinates that are matched together

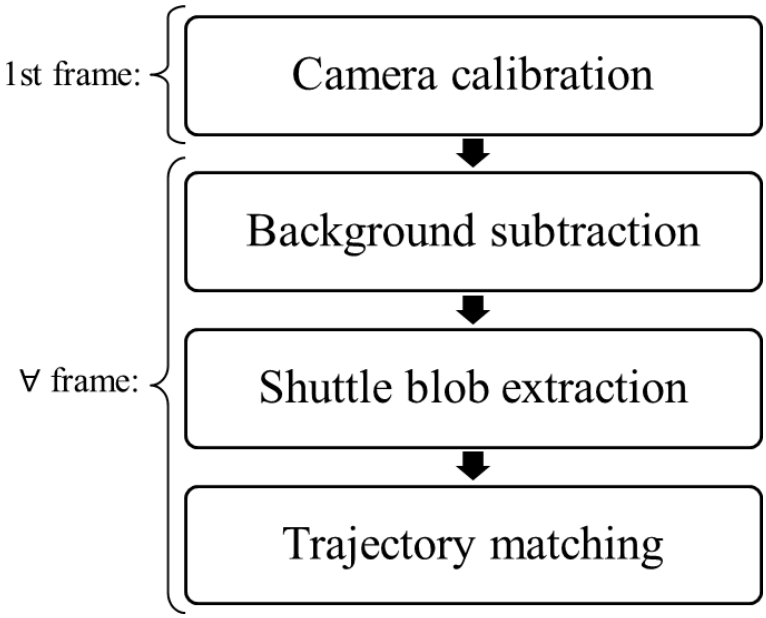
Definition 2.4 A **ground truth trajectory** is a time chronological list of shuttle centroid coordinates as observed during one shot in a badminton match video

Definition 2.5 A **model trajectory** is a time chronological list of shuttle centroid coordinates calculated by a shuttle model given some parameters

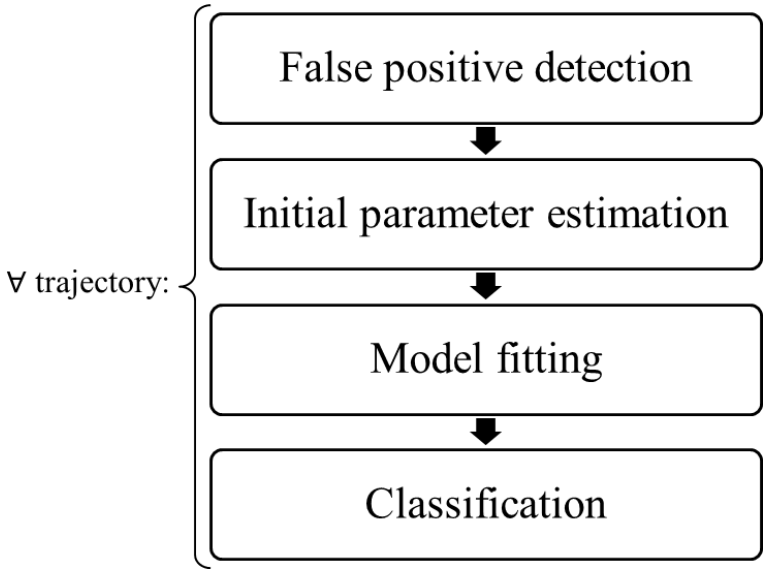
Definition 2.6 An **observation** is an extracted blob that is part of a certain blob trajectory

Definition 2.7 An **observed trajectory** is a blob trajectory that contains a ground truth trajectory

Definition 2.8 A **shuttle trajectory** is used to refer to the general notion of a characteristic shuttle flight



(a) Trajectory extractor subalgorithm (Chapter 3).



(b) Trajectory classifier subalgorithm (Chapter 5)

Figure 2.1: Visual overview of building blocks in the processing algorithm.

Chapter 3

Trajectory Extractor

3.1 Camera calibration

Video sequences are created by a camera in a certain position and with certain internal settings. This camera projects an observed world into a 2D image plane. The most common way to mathematically model this phenomenon is the pinhole camera model [HZ03]. This pinhole model describes a transformation that transforms real world coordinates into image domain coordinates. Having such a transformation allows to interfere about image plane implications of real world objects. Concretely, we use the transformation to make a court width estimate (see 3.3.2.1), to estimate how image plane observations translate into 3D information (see 5.2) and to calculate how a shuttle flight is seen in a video (see 4.3, 5.1.1). Due to the stationary camera, the camera calibration only needs to be performed once. This step is thus executed during the first frame, assuming that the badminton court is visible in this frame.

An illustration of the pinhole model can be seen in figure 3.1, equation 3.1 describes the transformation that forms the basis of the model. Real world

coordinates x, y and z are be transformed to image plane coordinates u, v:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.1}$$

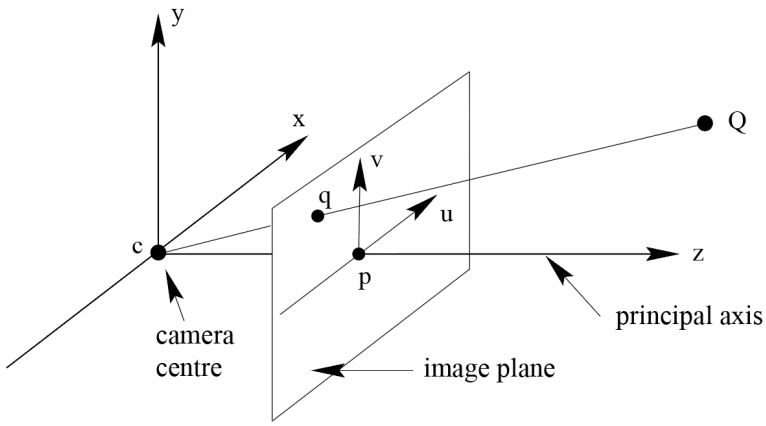


Figure 3.1: The pinhole camera model. Figure adapted from [HZ03].

To obtain this matrix, the same method is used as presented by [CTC+09]. Since the camera matrix is involved in the mapping between elements of two projective spaces, it can also be regarded as a projective element. This means that it has only 11 degrees of freedom since any multiplication by a non-zero scalar results in an equivalent camera matrix:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.2}$$

The real world court model is based on the official dimension of a badminton court (see [cit12] and figure 3.2) and the default main camera positioning (see figure 3.4). The eight points used to calculate the camera matrix are visualized in figure 3.3. The coordinates of these points are given by:

$p_1 = (3.05, 0.0, 6.7)$
$p_2 = (-3.05, 0.0, 6.7)$

$p_3 = (-3.05, 0.0, -6.7)$
$p_4 = (3.05, 0.0, -6.7)$
$p_5 = (3.05, 0.0, 0.0)$
$p_6 = (-3.05, 0.0, 0.0)$
$p_7 = (-3.05, 1.55, 0.0)$
$p_8 = (3.05, 1.55, 0.0)$

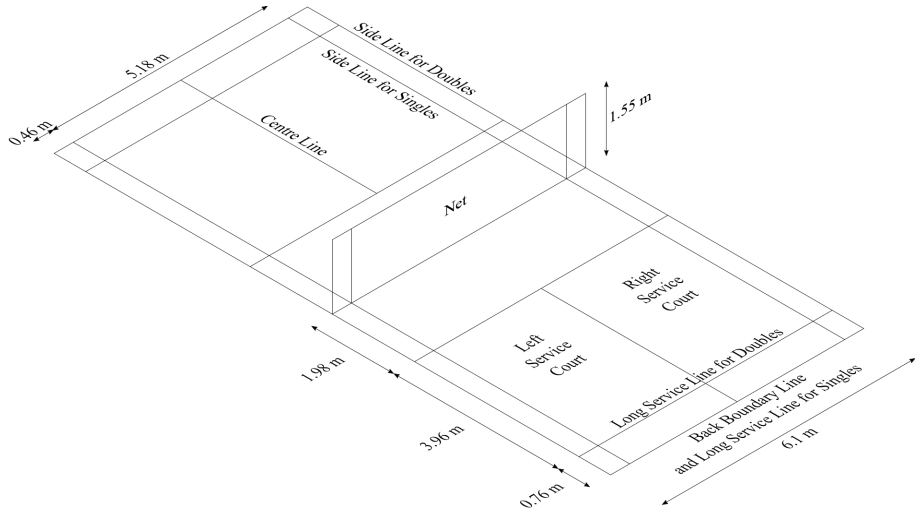


Figure 3.2: Dimension of a badminton court.

The coordinate system of the real world court model is aligned with the coordinate system of the pinhole model to ease the interpretation of the calculated results. This coordinate system is right handed. The y-axis is used to describe the height by convention, while the z-axis corresponds with the depth. This implies that the long side of the court is located along the z-axis and the short side along the x-axis. Due to the right handed orientation of the coordinate system, the right side of the court has negative x-coordinates. These coordinate system characteristics can be seen in figure 3.3.

Each of the real world model points has a corresponding point in the image plane domain. These points can be selected by user input or automatically found by an algorithm. Every point-to-point correspondence gives rise to equations (3.3) and (3.4).

$$p_{11}x + p_{12}y + c_{13}z + p_{14} = u(p_{31}x + p_{32}y + p_{33}z + 1) \tag{3.3}$$

$$p_{21}x + p_{22}y + c_{23}z + p_{24} = v(p_{31}x + p_{32}y + p_{33}z + 1) \tag{3.4}$$

A linear system is build with 6 point-to-point correspondences from the court plane and 2 point-to-point correspondences from the top of the net. Solving this linear system in a least squares manner results in an estimate for the projective matrix. To solve the least squares problem, QR-factorization is applied.

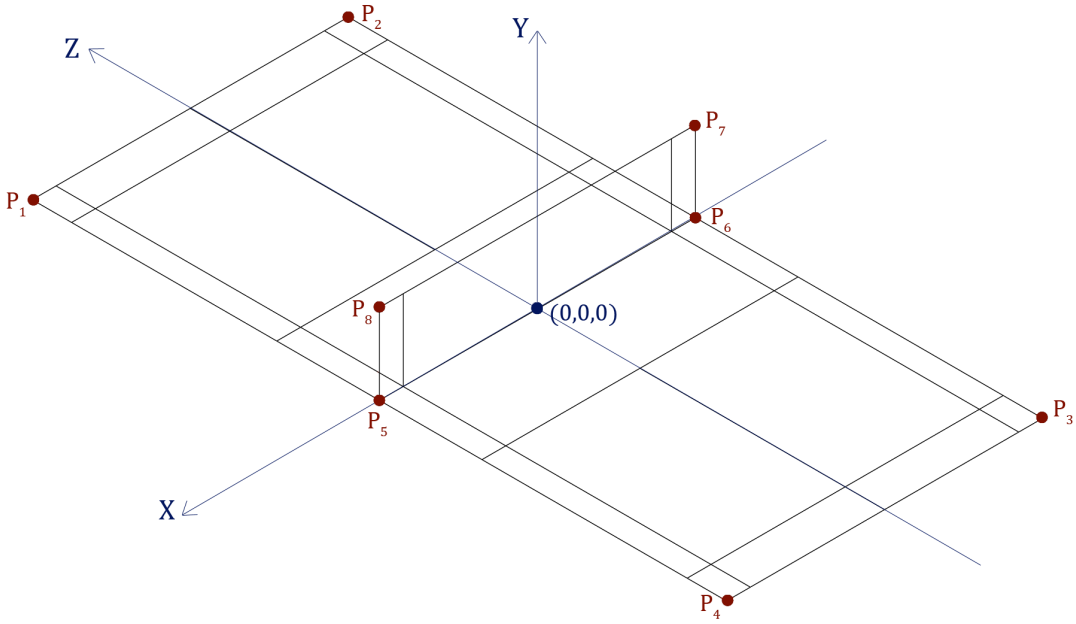


Figure 3.3: Court model and calibration points.

Figure 3.4: The main view of a badminton match recording is created by camera 1. Figure from [cita].

3.1.1 Investigation of test video camera matrices

The linear system to find the camera matrix should be well conditioned to yield feasible results (see [Hea96] and [HZ03]). The accuracy of the obtained solutions is found sufficient by visual observation (see figure 3.5). To support these findings, the maximum reprojection error (see equation 3.5) is calculated for all test videos (see table 3.1.1).

$$\eta = \max_i ||calibrationPoints_{ImagePlane}(i) - calibrationPoints_{ProjectedModel}(i)||_2 \tag{3.5}$$

video id	**maximum reprojection error**
1	0.8765
2	1.9931
3	1.3893
4	1.9393
5	0.6648
6	1.5825

The reprojection error is not higher than 2 for any of the test videos. This entails that a projected model point differs at maximum 2 pixels from the annotated image plane point. Knowing that the manual point annotation has pixel precision (see 3.1.2), the accuracy of the P matrix estimate is sufficient. After all, the reprojection error is influenced by both the camera calibration matrix estimation and the error introduced to human annotations (i.e. fiducial localization error).

The camera matrix is also decomposed in order to investigate some intrinsic and extrinsic parameters of the camera [Kov]. The advantage of this verification method is the human interpretability of these parameters. Table 3.1.1 shows the position of the camera in the real world court model (figure 3.3), focal length and rotation angles. The real world position coordinates are according to the real world court model. The focal length is given by $10 \cdot K_{11}/dpcm$ with K_{11} the first element of the intrinsic matrix and *dpcm* the number of dots per centimetre of the video. Since K_{11} encodes the focal length in meter (same unit as real world court model) divided by the pixel width, a multiplication by 10 is done to get the focal length in mm. Lastly, the rotation angles are calculated from the rotation matrix using the method described by Slabaugh [Sla]. Pan, tilt and roll describe the rotations around the y,x and z axis in the model coordinate system, respectively (figure 3.3).

video id	real word position	focal length	pan, tilt, roll
1	(0.3, 5.0, -22.6)	19 mm	-1.7°,-6.8°,-0.3°
2	(0.0, 9.8, -33.0)	24 mm	-3.2°,-5.4°,-1.0°
3	(0.2, 9.9, -33.1)	17 mm	1.5°,-23.6°,0.3°
4	(0.0, 4.5, -15.2)	17 mm	0.6°,-8.7°,0.0°
5	(0.6, 7.0, -28.6)	15 mm	4.5°,-3.8°,1.7°
6	(-0.3, 10.2, -30.2)	17 mm	2.0°,-5.4°,0.1°

The camera is positioned from 8.5 to 26.4 meters behind the court (see figure 3.2 for court dimensions), right in the middle and 5.0 to 10.2 meters high. The roll is negligible, the pan is slightly left or right and the tilt varies between -23.6° to -3.8°. The focal length corresponds to typical wide-angle lenses. All these parameters are realistic values and accord with the observed images (see figure 1.1).

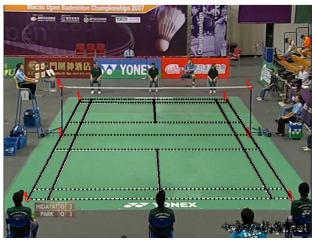
3.1.2 Manual selection of calibration points

Because of the implementation complexity of the automatic model fitting, an interface is created on which users can indicate the court and net corners points. By means of this, camera calibration can be performed without an operative version of the automatic calibration point selection.

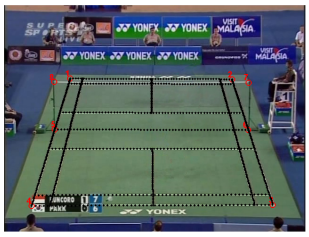
3.1.3 Automatic calibration point selection

An automatic playing field detection algorithm is implemented based on previous research from Farin et al. [FHD05], [FKE+04]. However, the implementation has some incompletenesses and is not optimized for speed. Therefore, we stick to the manual selection of calibration points. Nevertheless, the description of the automatic method might aid further developments based on the existing code.

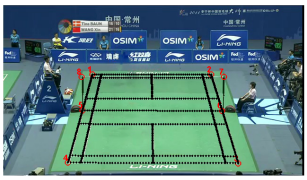
The method of Farin et al. focuses on finding a ground homography that maps court points to a court model. The ground homography is useful to find the real world position of objects on the court plane (for example: players' feet). A homography estimation only needs four point correspondences. To get the camera projection matrix, these points can be used in combination with manually annotated points or the method can be extended to find more calibration points. Consequently, the next paragraphs focus on the extraction of calibration points in the image plane rather than the homography details.



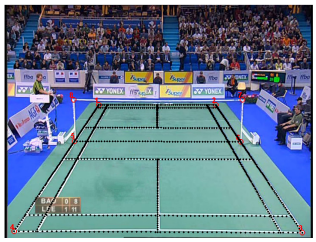
(a) Video 1: Macau Open 2007



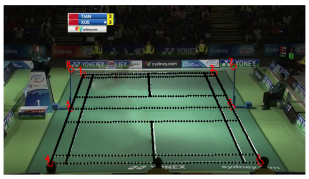
(b) Video 2: Asia Championships 2008



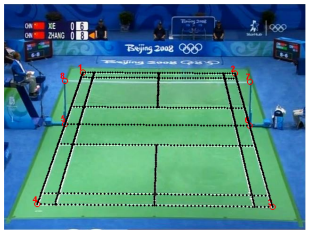
(c) Video 3: China Open 2011



(d) Video 4: French Open 2007



(e) Video 5: Australian Open 2013



(f) Video 6: Beijing Olympics 2008

Figure 3.5: Model points and court model projected on image plane using the calculated P matrix.

3.1.3.1 White pixel detection

Court lines are assumed to be white. Therefore, pixels are first classified using a luminance threshold. To prevent large white areas to be selected, a local constraint is added. The luminance difference of the considered pixel and some pixels away neighbouring pixels in the horizontal or the vertical direction should be bigger than a certain threshold. Formally: let θ_1 and θ_b be thresholds, μ the estimated line thickness upperbound (eg. 8 pixels) and l(x,y) the luminance of a pixel at position (x,y). Then a pixel is classified as a court candidate pixel as follows:

$$cc(x,y) = \begin{cases} 1: & l(x,y) \geq \theta_1 \ \wedge \ l(x,y) - l(x-\mu,y) > \theta_2 \ \wedge \ l(x,y) - l(x+\mu,y) > \theta_2 \\ 1: & l(x,y) \geq \theta_1 \ \wedge \ l(x,y) - l(x,y-\mu) > \theta_2 \ \wedge \ l(x,y) - l(x,y+\mu) > \theta_2 \\ 0: & else. \end{cases} \tag{3.6}$$

To exclude small white areas from the selection, a structure tensor is calculated. For a candidate pixel (P_x,P_y), a structure matrix S is calculated according to following formula:

$$S = \sum_{x=P_x-b}^{x=P_x+b} \sum_{y=P_y-b}^{y=P_y+b} \nabla l(x,y).(\nabla l(x,y))^T \tag{3.7}$$

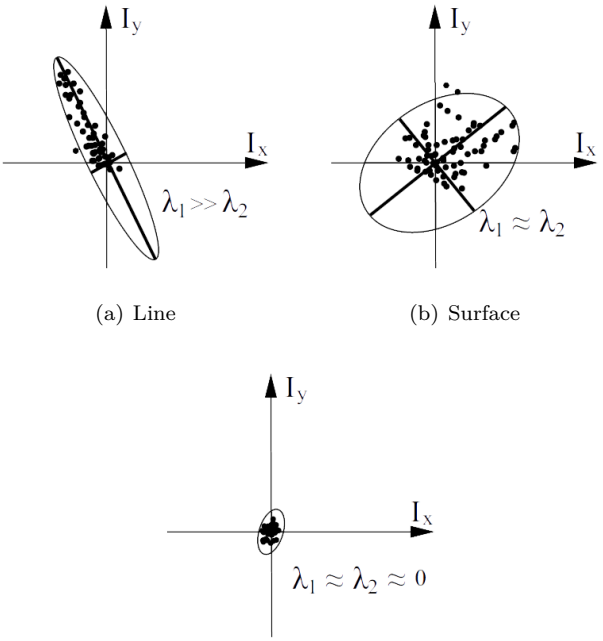
with $2b+1$ the window size and ∇ the image gradient.

The eigenvalues of this matrix yield information about the structure of the pixels inside the considered window. This can intuitively be understood since the eigenvalues indicate the main directions of variation of the luminance values gradients. See figure 3.6 for an overview of the possible cases.

Line color problem
Note that not every court has white playing lines. An algorithm could be invented which searches for the dominant court color and the court line color. Next, the same steps as in the white pixel case can be executed by replacing white pixels with the found color pixels.

3.1.3.2 Line extraction

Hough transform
After the pixel selection task, a Hough transform is performed to get parametrized



(a) Line

(b) Surface

(c) Point

Figure 3.6: Structure tensor visualizations (illustration from [Nie10])

lines (and not just a list of possible court pixels). A Hough transform transforms the image to a parametrized form where every pixel has an associated sinusoid. Points lying on a straight line will then produce sinusoids crossing at the parameters for that line. The result is a set of lines specified by (r, θ) where r is the distance of the line to the origin and θ the angle between the line normal and the horizontal axis.

Parameter refinement
As the candidate court lines are quite thick (> 1 pixel lines), the Hough transform detects several close-lying lines for every court line. Therefore, the line parameters of all lines are refined, minimizing their distance to the court candidate pixels. This yields an overdetermined linear system of equations which has to be solved with a robust least trimmed squares estimator. After this step, lines are considered equal if their angle and distance is small enough. The refinement and duplication deletion steps are repeated until a stable number of lines is achieved.

Alternative
An alternative for the Hough transform is a RANSAC-like algorithm [FHD05]. This works also well and is a bit faster. Since the camera calibration only has to be done for the first frame, speed is not a huge issue. A court tracking mechanism is sufficient for the consequent frames. Nevertheless, the RANSAC-like algorithm omits the necessity for the difficult parameter refinement (as described above) and can thus lower the implementation complexity.d

3.1.3.3 Model fitting

The remaining lines are ordered in a set of vertical and a set of horizontal lines. Within every set, the lines are sorted respectively from left to right and from top to bottom. Correspondences between court lines in the image and court lines in the model are now found by iterating all quadrangles that preserve the indices order after sorting. From these quadrangles, the four intersections are calculated as correspondence points.

Possible assignments are tested by calculating the homography, projecting model court lines to the image plane domain based on this homography and calculating the matching between these projected lines and the extracted lines. The points from the assignment with the largest matching score are selected as the calibration points. Remark that the researchers [FKE+04] speed up this process by introducing a quick parameter rejection test.

3.2 Background subtraction

Since a moving shuttlecock is the object of interest, an essential step is to extract motion from the input video sequence. So, in this step, the moving pixels in a frame are detected.

The background subtraction problem can be formulated as follows: identify all foreground objects given a certain video frame from a stationary camera. In abstracto, the absolute difference between the considered frame and the static background is compared against a certain threshold ($|frame_i - background_i| > Threshold$). Every pixel with a value above the threshold is classified as part of the foreground.

A good background subtraction algorithm can track different changes in the background: illumination changes (both sudden and gradual), motion changes (small camera movements, high frequency moving object such as a waving flag) and long term changes in background geometry (for example: a parked car drives away after some time).

3.2.1 Basic methods

In this section, some basic background subtraction methods are described. Non of them is eventually used in the algorithm, they are merely stated here as an introduction to the used mixture of Gaussian method.

The most basic method to solve the background subtraction problem uses the previous frame as an estimate of the static background: $|frame_i - background_{i-1}| > Threshold$. This frame difference method is very sensitive to the chosen threshold and only works well with very specific video properties. Better methods try to make a better estimate of the static background by considering more frames. The mean filter method uses the mean of the previous n frames: $|frame_i - \frac{1}{n}\sum_{j=1}^{n} frame_{i-j}| > Threshold$, while the median filter method exploits the median of the previous n frames: $|frame_i - median(frame_{i-j})| > Threshold$ with $j \in \{n, n-1, ..., 2, 1\}$. In the running average method the background is calculated as weighted combination of the current frame and the previous background: $|frame_i - background_i| > Threshold$
with $background_i = \alpha * frame_i + (1 - \alpha) * background_{i-1}$ and α is known as the learning rate with a typical value of 0.05.

Each of these basic methods can be extended using selectivity. Histograms of

pixel values through time are created. Thresholds are placed on these histograms to classify pixels of the current frame in foreground and background pixels. Next, foreground pixels are ignored in the calculation of a new background model.

For all basic methods, the value of each background pixel is a chronological average of previous pixel values. Hence, these algorithms do not take advantage of any spatial correlations in the background. Therefore, calculating the background model for each color channel does not significantly improve the performance in comparison with the use of a greyscale version of the video. Also, the results of these mostly depend on the chosen threshold.

3.2.2 Mixture of Gaussians method

Stauffer and Grimson describe [SG99] a method that models each pixel as a mixture of Gaussians (= weighted sum of Gaussian distributions): $P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, u_{i,t}\, \Sigma_{i,t})$ with X_t the considered pixel values (= RGB components) at time t, K the number of Gaussians (typically 3 or 5) and $\omega_{i,t}$ a weight associated to the ith Gaussian at time t with a mean $u_{i,t}$ and standard deviation $\Sigma_{i,t}$. The weight at a given time should sum up to one: $\sum_{i=1}^{K} \omega_{i,t} = 1$.

The Gaussian density function is given by: $\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_t)^T \Sigma^{-1}(X_t-\mu_t)}$
The pixel values are assumed to be independent and have the same variance for computational reasons. Thus, the covariance matrix is of the form $\Sigma_{i,t} = \sigma_i^2 I$.

For every new frame, pixel values are tested against their Guassians. Whenever the pixel value lies within 2.5 standard deviations, it is classified as a match. The weights of the distributions are changed in order to give higher weight to matched Gaussians: $\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha(M_{k,t})$ with α a constant learning rate. $M_{k,t}$ equals 1 if the pixel value matches the kth Gaussian at time t, 0 otherwise.

The mean and variance are only updated for matched Gaussians: $\mu_t = (1-\rho)u_{t-1} + \rho X_t$, $\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t)$ with $\rho = \alpha\eta(X_t|\mu_k, \sigma_k)$
If the pixel value matches none of the K Gaussians, the least probable one is replaced by a new Gaussian. This new distribution gets a low weight, large variance and a mean equal to the pixel value.

The distributions for every pixel are ordered to find the Gaussians with the

most supporting evidence (= big weights) and the least variance. The first B distributions are chosen to be the background model.

$$B = argmin_b \left(\sum_{k=1}^{b} \omega_k > T \right)$$

B depends on the desired portion of the data one wants in the background model. This portion is described by the threshold value T. Pixels that do not match one of these distributions in the background model, are considered as foreground.

The online adaptive mixture model copes with light changes, while the multiple Gaussians model the variety of surfaces that might contribute to one pixel value. For many applications, this method yields robust results. This mixture of Gaussians background subtraction algorithm is used for the background subtraction step.

3.2.3 Implementation

The running average method is implemented and tested by visual observation. The algorithm only works average when using an appropriate threshold (different for every video sequence). It is clear that a more complex background modelling is needed.

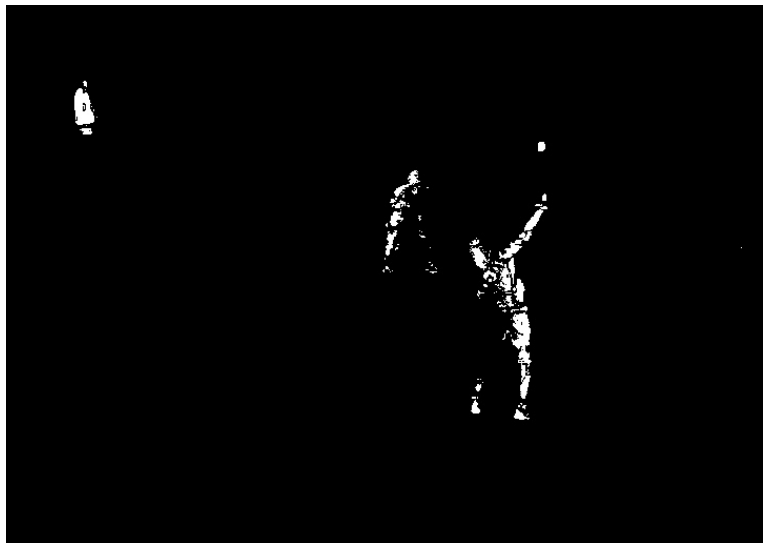
The mixture of Gaussians background subtraction uses the OpenCV library implementation of the algorithm which is based on the work of Kaewtrakulpong et al [KB01]. The described method changes the way the Gaussians are updated to improve the speed and accuracy of the original algorithm. The update equations use a recent window update to give priority to recent data and thus allowing the Gaussians to adapt to changes in the environment. These changes boil down to removing the $\eta(X_t|\mu_k, \sigma_k)$ term from ρ, resulting in $\rho = \alpha$. Nothing is changed to the main principles of the background modelling.

A visualization of the result can be seen in figure 3.7.

3.2.4 Performance

The background subtraction is tested by manually counting the number of frames where the shuttlecock is among the extracted foreground pixels. Besides α, the

(a) Original frame



(b) Foreground mask

Figure 3.7: An example the mixture of Gaussian background subtraction result

parameter values are set to default: the number of Gaussians in the mixture $K = 5$, initial weights $\omega_{k,1} = 0.05$ and the background ratio $T = 0.7$.

Following table summarizes the results for the background subtraction run with learning rate $\alpha = 0.05$:

video id	A = total # frames	B = background initialization (#frames)	C = pre rally (#frames)	D = shuttle occluded (# frames)	E = shuttle out of image (#frames)	F = shuttle not detected (#frames)	detection %
1	863	24	0	4	72	25	96.72
2	608	9	33	14	24	6	98.86
3	511	9	0	64	0	43	90.18

The detection percentage is calculated as follows:
$detection\% = \frac{\#\ frames\ shuttle\ is\ detected}{\#\ frames\ shuttle\ is\ detectable} = 1 - F/(A - B + C + D + E)$

B represents frames where the court is not completely visible (due to transits, players on court, ...) leading to wrong background subtraction. C encompasses the frames when the rally is not started yet (the shuttlecock is not moving). In D frames the shuttle is not detectable because a player or other object is position in front of the shuttle (for example: shuttle on a court line). E are frames where the shuttle is going so high that the shuttle is not captured by the camera.

Following table summarizes the results for the background subtraction run with learning rate $\alpha = 0.25$:

video id	A = total # frames	B = background initialization (#frames)	C = pre rally (#frames)	D = shuttle occluded (# frames)	E = shuttle out of image (#frames)	F = shuttle not detected (#frames)	detection %
1	863	4	0	4	72	15	98.08
2	608	3	33	14	24	6	98.86
3	511	9	0	64	0	43	90.18

The learning rate compromises between being fast enough to adapt to changes and slow enough to store a useful temporal history. Due to the stationary camera and the relatively stable background, the mixture of Gaussians background subtractions shows little performance variation for different values of α (see tables 3.2.4 and 3.2.4). α is further referred as α_{MOG}.

Three main observations are made in case the shuttlecock pixels are not extracted. Firstly, a not detected shuttlecock is often an isolated case, only leaving out one shuttlecock observation on a whole trajectory. Secondly, non detected shuttlecocks appear the most when the shuttlecock is accelerating very fast or under a steep angle. This happens the most when the shuttle leaves the racket and thus starts a new trajectory. These cases yield observed trajectories without the first shuttlecock image plane positions. Lastly, the algorithm has more problems if the resolution of the video goes down (see video 3). However, most non detected shuttlecocks are very hard cases where even a human would fail to detect the shuttlecock without seeing the images in a sequence (see figure 3.8). Investigating these extreme cases for an ever bigger value of α_{MOG} ($= 0.5$), shows no improvements.

The above observations and the good performance of the algorithm with the chosen parameters justify the limited investigation of background subtraction algorithms. The possible gain one can make in this step is marginal. α_{MOG} is

set to 0.25 since it slightly performs better and requires a lower initialization time.

Figure 3.8: Frame 149 from video 3. The location of the shuttlecock is indicated by the red square.

3.3 Shuttle blob extraction

3.3.1 Blob extraction

The background subtraction step gives a mask which indicates the foreground pixels in a frame. However, pixels by themselves are not powerful enough to accommodate object extraction. Therefore, pixels are connected into components, the so-called connected-components or blobs.

Traditional component-labelling algorithms use two passes and relabel components according an equivalence relation induced by 8-connectivity. During the first pass, the neighbours of each foreground point F are investigated. If none of them has a label, F is given a new one. In the other case, the labels of the neighbours are equivalent and F is assigned the minimal equivalent label. So any foreground pixel F has a pair of arrays that hold the current labels and the other minimal equivalent labels of those current labels. The second pass is executed to make the necessary label replacements, so that every F has its minimal label.

Blob extraction is implemented using the cvBlob library [N´]. It implements the component-labeling algorithm proposed by Chang et al [CCL04] that runs in linear time to find 8-connectivity components. To achieve this, the image is scanned from top to bottom and from left to right and encountered contours are traced back to their starting point to resume scanning.

(a) Foreground mask on which the blob extraction algorithm is applied

(b) Blobs contours are outlined in red. The blue crosses indicate the centroids

Figure 3.9: Visualization of blob extraction result.

3.3.2 Blob filtering

The blob extraction (see 3.3.1) gives a list of connected-components. These components encode more information than single pixels and can thus be investigated further. From this list of blobs, the blob corresponding to the shuttle is to be found. Therefore, several properties of the blobs are examined and used as discriminating properties. Note that the output of this step is a list of possible shuttle blobs rather than just one shuttle blob since there are always non shuttle blobs that pass the blob filtering. Ideas for blob discriminating properties were found in [CTC+09].

3.3.2.1 Size properties

Image plane proportions of projected real world objects differ for every video and are dependent on the camera matrix. Therefore, the size constraints in the blob extraction step are deducted from a court width estimate. To achieve this estimate, the width of the court in pixels (d_{cw}) is calculated by transformidng a real world court line to the projected space. The service line on the closest court part is used to accommodate the slightly bigger projections of shuttles flying on that side of the court.

It is known that the court width measures 6.1 meters in the real world. A shuttlecock measures 0.07 meters on its longest side. Next, the ratio between the shuttlecock size and the court width is multiplied by the image domain court width estimate. This gives a good estimate of the image domain shuttle size. Consequently, blobs are only considered as shuttlecock candidates if their length and width is smaller than the estimated image domain shuttle size multiplied by 3. The tripling is done to address motion blur which makes the image domain shuttle size bigger.

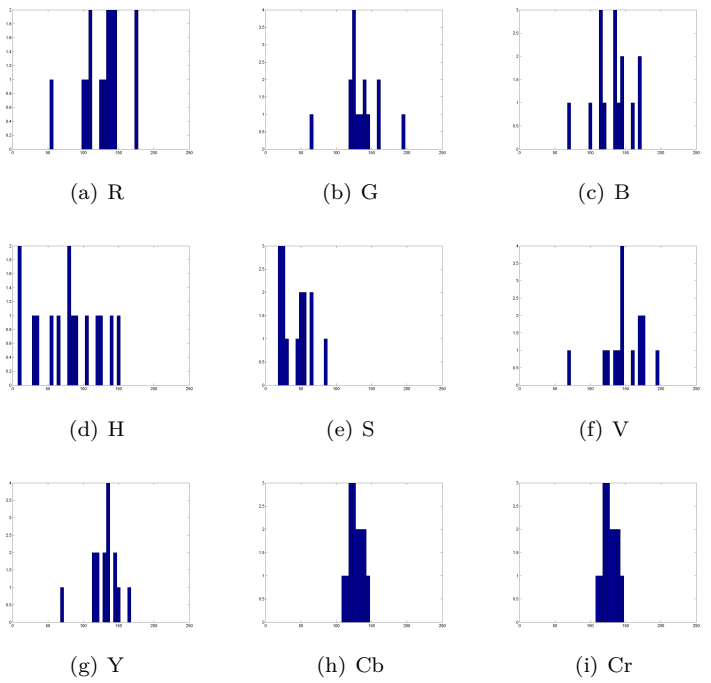
So every blob B on every frame is removed from the list of blobs if $B_{width} > d_{cw} * 0.0115 * 3$ or $B_{height} > d_{cw} * 0.0115 * 3$.

3.3.2.2 Color properties

To investigate the color properties, 15 shuttles are manually segmented. By looking closely at these images, a wide variation of colours is observed. Since a shuttle is made from goose feathers, it is not a dense object. Therefore a shuttle observation in a video will be intermixed with background colours depending on the shuttle's position. Looking at individual pixel color values is thus useless.

To find the best discriminative color properties, histograms of the average color values of the manually segmented shuttles (see figure 3.10) are compared. Useful color values for filtering are those that have little variation and thus show some typical value range for shuttles. However, to be discriminative it is also important that this range differs from what is observed in the histogram of possible non-shuttle blobs. To this end, the histograms of 15 manually segmented possible non-shuttle blobs are created (see figure 3.11). The most discriminative color value is the saturation. Every segmented shuttle has a saturation value below 90.

So every blob with an average saturation value higher than θ_{BFS} is removed from the list of blobs. Unfortunately, when testing the shuttle blob extraction with θ_{BFS} set to 90, insufficient results appear. By visual observation it is immediately seen that too many blobs are filtered out. Even when θ_{BFS} is set to 150, the shuttle blob detection rate for the first video drops from 98.08 to 95.66. The complex colour composition of shuttle blobs make blob filtering based on color properties a very hard task.

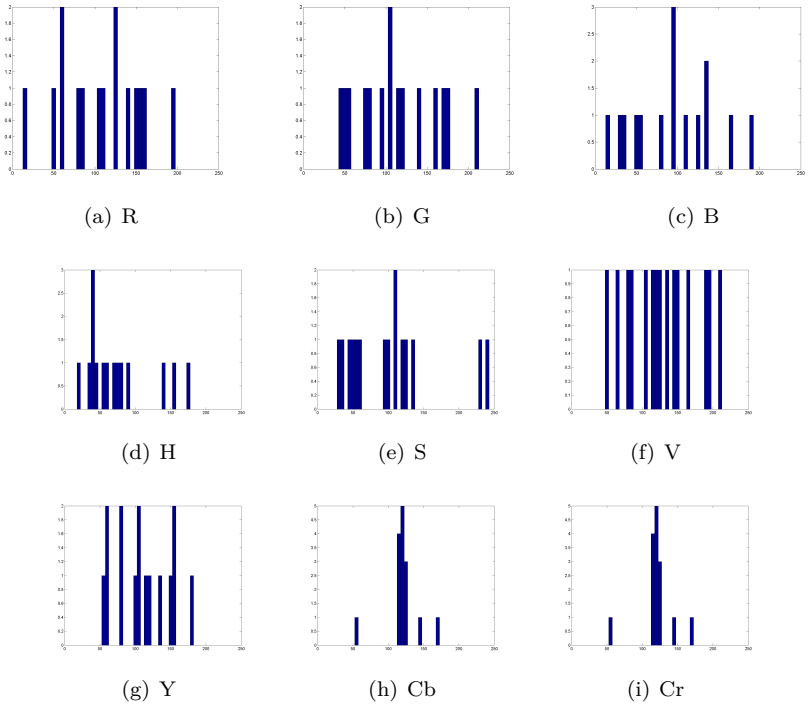


(a) R (b) G (c) B

(d) H (e) S (f) V

(g) Y (h) Cb (i) Cr

Figure 3.10: Average color value histograms of manually segmented shuttles



(a) R (b) G (c) B

(d) H (e) S (f) V

(g) Y (h) Cb (i) Cr

Figure 3.11: Average color value histograms of manually segmented possible non-shuttle blobs

3.3.3 Performance

This blob filtering is run with only the size constraint. The shuttle blob detection rate does not drop for video 1 and video 2. The rate only drops 0.05 for the third training video. Meanwhile, the average number of possible shuttle blobs per frame is reduced from 94, 68 and 75 to 87, 63 and 63, respectively.

3.4 Trajectory matching

Recall what we have so far. A badminton match video sequence is read and feed to a background subtraction algorithm. Consequently, the resulting foreground mask is used for blob extraction. Thus, there is a decent detection of moving blobs in a video sequence. With or without blob filtering, there are still multiple blobs detected for each frame while the shuttle blob is the only blob of interest. To tackle this problem, the possible shuttle blobs are connected through time in order to construct blob trajectories. These carry more information than just a single blob. Hence, it is easier to detect those blob trajectories that contain a ground truth trajectories over multiple frames rather than finding the shuttle blob in every frame.

3.4.1 Trajectory descriptor

A blob has centroid coordinates and an area of contour. So a blob B can be represented as a tuple $B \in (x, y, area)$. A blob trajectory is a chronological list of blob centroids (see 2.1). To construct the blob trajectories, we want to match every possible shuttle blob in every frame to a certain blob trajectory. This is done by adding every possible shuttle blob to its closest blob trajectory. To formally define the closest blob trajectory, it is necessary to have a distance measure between a blob B and a blob trajectory. One could simply take the latest blob B_{last} in the blob trajectory list and calculate the Euclidean distance between this B_{last} and B. However, a better choice is to introduce an additional structure associated with each blob trajectory and then calculate the distance between a blob B and this associated structure (see [YRC04]). This structure then abstracts some characteristics of the trajectory. Thus, blob trajectories are not only characterized by their last added blob. By doing this, the construction of blob trajectories is made more robust. Consider for example a blob trajectory which contains mainly shuttle blobs. Since the shuttle blob extraction step (see 3.3) is not perfect, a non shuttle blob can be matched to this blob trajectory.

Here, the trajectory abstraction prevents the unwanted match to destroy the general trend of the blob trajectory.

Define a trajectory descriptor T, associated with a blob trajectory, as a tuple $(x, y, area)$. A trajectory descriptor comprises characteristics of a blob trajectory over time by forming a running average of the blobs contained in the blob trajectory. The distance between a trajectory descriptor T and a blob B is given by

$$D(T,B) = (1-\omega_{TMDA})\sqrt{(T_x - B_x)^2 + (T_y - B_y)^2} + \omega_{TMDA}\sqrt{(T_{area} - B_{area})^2} \tag{3.8}$$

The value of ω_{TMDA} indicates the weight given to the area in the calculation of the distance measure. Since the goal is to match possible shuttle blobs that are on the same ground truth trajectory, the Euclidean distance is the most important part in the total distance measure. So, ω_{TMDA} will typically be lower than 0.4.

The trajectory descriptor is updated in an online fashion:

$$\begin{aligned} T_x^t &= B_x^{t-1} \\ T_y^t &= B_y^{t-1} \\ T_{area}^t &= (1 - \alpha_{TMA}) \cdot T_{area}^{t-1} + \alpha_{TMA} \cdot B_{area}^{t-1} \end{aligned} \tag{3.9}$$

The learning rate α_{TMA} denotes how fast the trajectory descriptor will update its area attribute. The aim is to get a running average of the blob areas on the trajectory to avoid big blobs being matches with small blobs and vice versa. Therefore, a typical value for α_{TMA} is 0.1.

3.4.2 Trajectory matching threshold

So for every possible shuttle blob, a distance to every blob trajectory can be calculated. Next, every possible shuttle blob is added to its closest blob trajectory. However, when a blob ($B \in (x, y, area)$) matches no blob trajectory with a smaller distance than a certain threshold (θ_{TMD}), the blob is added to a new blob trajectory.

A new trajectory descriptor is created for the trajectory:

$$\begin{aligned} T_x &= B_x \\ T_y &= B_y \\ T_{area} &= B_{area} \end{aligned} \tag{3.10}$$

θ_{TMD} has a big impact on the behaviour of the algorithm. If the threshold is too big, all blobs are connected to one trajectory. On the other hand, if the threshold is to low, a new blob trajectory is started for every blob since no blob is connected to a trajectory.

On the first sight, it might be a good idea to determine an image plane distance upper bound between two shuttle observations based on model trajectories. This upper bound would be a good value for θ_{TMD} since every two observations that have a bigger distance can never belong to the same ground truth trajectory. However, by calculating this theoretical value and running some tests, it is found that this upper bound is too big to yield feasible results. To many blobs are connected to the same blob trajectory. Thus, a trade-off between completeness and practical use has to be found. The optimal value of θ_{TMD} is determined by empirical testing. Note that θ_{TMD} depends on α_{TMA} since the latter value influences the calculated trajectory-blob distances.

To make θ_{TMD} independent of the video size and camera parameters, calculated blob-trajectory distances are normalized. The normalization uses the court width estimate as defined for the blob filtering.

3.4.3 Idle time

The trajectory matching step also has an idle time measurement to prevent different blob trajectories being merged together. For each frame, the trajectory idle time counter is increased when there is no blob added to the blob trajectory. If a blob trajectory reaches an idle time of θ_{TMIT}, the blob trajectory is marked as finished. θ_{TMIT} should be at least 3 so that shots, of which not all shuttle blobs are extracted, can still be matched together. To extract shots that disappear from the image plane, θ_{TMIT} should be even bigger. However, when θ_{TMIT} becomes too big, distinct blob trajectories are matched together. So a plausible value for θ_{TMIT} lies in the interval $[5 - 15]$.

3.4.4 Performance

The main goal of the trajectory extractor algorithm is to extract the blob trajectories which contain a ground truth trajectory. The performance of the trajectory extractor algorithm is defined as the number of ground truth trajectories that are extracted, regardless of the number of false positives. False positive are blob trajectories which do not contain parts of ground truth trajectories. Ground truth trajectories can either be found or stay undiscovered. In the former case, three situations need to be discriminated. Firstly, a ground truth trajectory can be found as distinguishable trajectory. In other words, the ground truth trajectory is contained in exactly one blob trajectory. Secondly, multiple ground truth trajectories are found in one blob trajectory (= composed blob trajectory). Lastly, one ground truth trajectory can be found fragmentary in multiple blob trajectories. If a ground truth trajectory is found in more than three blob trajectories, it is considered as not found. Figure 3.12 illustrates the three cases. In all cases, the ground truth trajectories are extracted as blob trajectories and can thus contain unwanted non shuttle blobs.

By visual observation, combinations of θ_{TMD} and ω_{TMDA} are tested. Infeasible combinations are quickly ruled out. It is found that $\theta_{TMD} = 0.08$ and $\omega_{TMDA} = 0.2$ is a good combination.

The trajectory extractor is run on the training videos (see 1.2.1.1) with $\alpha_{MOG} = 0.25$, $\theta_{BFS} = 256$ (implies blob size filtering only), $\theta_{TMD} = 0.08$, $\theta_{TMIT} = 5$, $\alpha_{TMA} = 0.1$ and $\omega_{TMDA} = 0.2$.

ground truth trajectories	**video 1**	**video 2**	**video 3**
not found	2	3	2
distinguishable	20	14	17
composed	11	8	0
fragmented	2	1	0
total	35	26	19
% distinguishable	0.57	0.54	0.89

The same test run on video 1 with $\theta_{TMIT} = 10$ yields exactly the same results. The exact choice of θ_{TMIT} is of little importance for the performance of the trajectory matching if the value lies within plausible boundaries (see 3.4.3). However, it is better to set θ_{TMIT} as low as possible to prevent non shuttle blobs to get matched to blob trajectories with mainly shuttle blobs.

The not found ground truth trajectories are mostly ground truth trajectories that incorporate a shot that crosses the net close to its center. The number of detected non shuttle blobs is higher here than in other regions of the image plane.

Consequently, shuttle blobs get matched to different blob trajectories that are still active in the area around the middle of the net. Other not found ground truth trajectories either originate from insufficient previous steps (eg. shuttle blobs not detected by the background subtraction algorithm) or from too big distances between the shuttle blobs with respect to θ_{TMD}.

It is important to note that non shuttle blobs remain an issue since these contaminate the extracted blob trajectories. Figure 3.13 shows such a problematic case.

(a) A distinguishable trajectory. The blob trajectory (red) contains a ground truth trajectory (green). Remark observation number 11, which is a matched non shuttle blob.

(b) A composed trajectory. The blob trajectory (red) contains two ground truth trajectories (green).

(c) A fragmented trajectory. One ground truth trajectory (green) is contained in two blob trajectories (red and magenta).

Figure 3.12: Each image shows a blob trajectory as found after the trajectory matching step. Red and magenta circles indicate the observed shuttle blobs. Green circles denote the annotated ground truth shuttle positions.

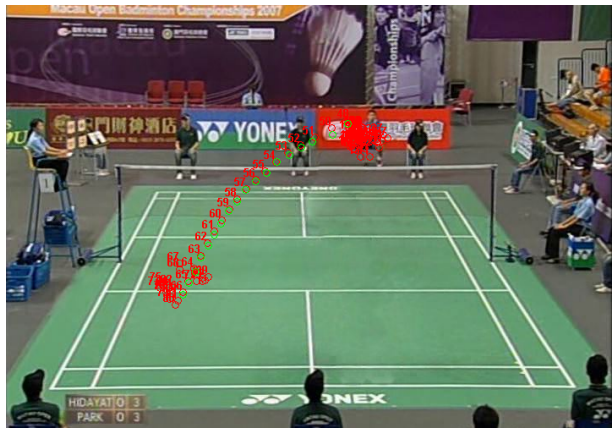


Figure 3.13: Red: constructed blob trajectory. Green: ground truth trajectory.

Chapter 4

Shuttlecock dynamics

The shuttlecock soars upward
In a parabola of whiteness,
Turns,
And sinks to a perfect arc.

(Amy Lowell)

In this chapter, a theoretical model of the shuttlecock flight is introduced. Such a model gives us information of the shuttlecock position in function of the time. These finding are extensively used in the subsequent chapter.

A lot of research has been done to describe the dynamics of a shuttlecock flight [CR12], [CPC09], [Coo02], [TCQC12]. The most import observation of this research is that a shuttle trajectory is not parabolic. It starts in a nearly-parabolic manner but has a steep drop after a certain overturn point is reached. This point is the terminal velocity and it occurs when the drag force rules out the acceleration of the shuttle.

Most papers ([CR12], [Coo02], [TCQC12]) use environmental condition parameters in the shuttlecock trajectory modelling. These parameters are not of interest for this thesis. Since badminton games recorded under different environmental conditions are considered, a more general model is needed. Chen et al. [CPC09]

model the shuttle trajectory by determining the terminal velocity in aerodynamics. By observing the shuttlecock as a moving object in a plane normal to the ground, they find equations that describe the vertical and horizontal displacement of the of shuttlecock in function of the time. In the next paragraphs, their method is described and extended to model the shuttle position in a three dimensional space. As last, the model is projected using the pinhole camera model (see 3.1).

4.1 2D model

Newton's second law of motion yields following formula for a shuttle in motion:

$$\overrightarrow{W} + \overrightarrow{F_v} + \overrightarrow{B} = m.\overrightarrow{a} \tag{4.1}$$

With $\overrightarrow{W}$ = gravitational force, $\overrightarrow{F_v}$ = aerodynamic drag force (also known as the air resistance force), $\overrightarrow{B}$ = buoyancy (air pressure acting up against the shuttle, a negligible value in comparison to gravitational force and drag force), m = shuttle mass and $\overrightarrow{a}$ = acceleration of the shuttle

The magnitude of the drag force can be written as

$$F_v = bv^n \tag{4.2}$$

Where b = a constant depending on air and shuttle properties, v = velocity of the shuttle and $n \in \mathbb{R}$

A vertically dropped shuttle has an increasing speed. However, according to the formula above, this also leads to an increasing resistance force. Thus, there is a point where the resistance force balances the shuttle weight and the shuttle acceleration becomes zero. This is point is called the terminal velocity v_T. Afterwards, the shuttle continues to drop with zero acceleration. Combining equation (4.1) and (4.2), neglecting buoyancy and setting $a = dv/dt = 0$, we get following expression:

$$mg - bv_T^n = 0 \tag{4.3}$$

From this follows the terminal velocity:

$$v_T = \left(\frac{mg}{b}\right)^{1/n} \tag{4.4}$$

By experimental measurements, it is known that $6.8\,m/s$ is a good estimate of the terminal velocity ([CPC09], [CR12]).

The resistance force can be modelled proportional to the shuttle speed (n = 1 in equation (4.2)) or to the shuttle speed squared ($n = 2$ in equation (4.2)). Experiments (see [CPC09]) have shown that the squared shuttle speed is the best modelling choice.

Suppose a shuttle is hit with an initial velocity $\overrightarrow{v_i}$ and an initial angle θ_i. Then, the initial velocity can be decomposed into a horizontal component $v_{xi} = v_i \cos(\theta_i)$ and vertical component $v_{yi} = v_i \sin(\theta_i)$.

The air resistance force could be modelled as a linear combination of the air resistance force in the vertical and horizontal direction. However, this would precludes analytical solutions to be found. Therefore, the vertical and horizontal motion are considered separately: $F_{vy} = bv_y^2$ and $F_{vx} = bv_x^2$.

The vertical directional motion is now characterized by

$$mg - bv_y^2 = m\frac{dv_y}{dt} \tag{4.5}$$

Integrating the expression, the vertical velocity is given by

$$v_y = \frac{v_{yi} - v_T \tan\left(\frac{gt}{v_T}\right)}{1 + \frac{v_{yi}}{v_T}\tan\left(\frac{gt}{v_T}\right)} \tag{4.6}$$

At the shuttlecock's highest point, $v_y = 0$ and the flight time is expressed as follows:

$$t = \frac{v_T}{g}\tan^{-1}\left(\frac{v_{yi}}{v_T}\right) \tag{4.7}$$

At that moment, the height is

$$y = \frac{v_T^2}{g}\ln\left|\frac{sin\left[\frac{gt}{v_T} + tan^{-1}\left(\frac{v_T}{v_{yi}}\right)\right]}{sin\left[tan^{-1}\left(\frac{v_T}{v_{yi}}\right)\right]}\right| \tag{4.8}$$

For the horizontal directional motion it holds that

$$-bv_x^2 = m\frac{dv_x}{dt} \tag{4.9}$$

From this, the horizontal velocity is found as

$$v_x = \frac{v_{xi} v_T^2}{v_{xi} g t + v_T^2} \tag{4.10}$$

and the horizontal distance as

$$x = \frac{{v_T}^2}{g} ln \left(\frac{v_{xi} g t + v_T^2}{v_T^2} \right) \tag{4.11}$$

4.2 3D model

To extend the above model to support a three dimensional space, the horizontal displacement is decomposed into two components: $x = \cos(\varphi_i) \cdot x_{2D}$ and $z = \sin(\varphi_i) \cdot x_{2D}$ with x_{2D} being the x as defined in the 2D model. Furthermore, initial position terms x_i, y_i, z_i are added. This gives following model:

$$coord_{3DModel}(t; params) = coord_{3DModel}(t; \{x_i, y_i, z_i, v_i, \theta_i, \varphi_i\}) = (x, y, z) \tag{4.12}$$

with

$$x = x_i + \cos(\varphi_i) \frac{{v_T}^2}{g} ln \left(\frac{v_i \cos(\theta_i) g t + v_T^2}{v_T^2} \right)$$

$$y = y_i + \frac{v_T^2}{g} \ln \left| \frac{sin \left[\frac{gt}{v_T} + tan^{-1} \left(\frac{v_T}{v_i \sin(\theta_i)} \right) \right]}{sin \left[tan^{-1} \left(\frac{v_T}{v_i \sin(\theta_i)} \right) \right]} \right|$$

$$z = z_i + \sin(\varphi_i) \frac{{v_T}^2}{g} ln \left(\frac{v_i \cos(\theta_i) g t + v_T^2}{v_T^2} \right)$$

The height is kept as the Y-coordinate to agree with the standard coordinate system of the orthographic projection model which is right handed. The initial angle φ_i is formally denoted as the angle between the x-axis and the initial ground plane direction of the shuttle trajectory. θ_i is the angle between this ground plane direction and the initial vertical direction. See figure 4.1 for a graphical representation of these angles.

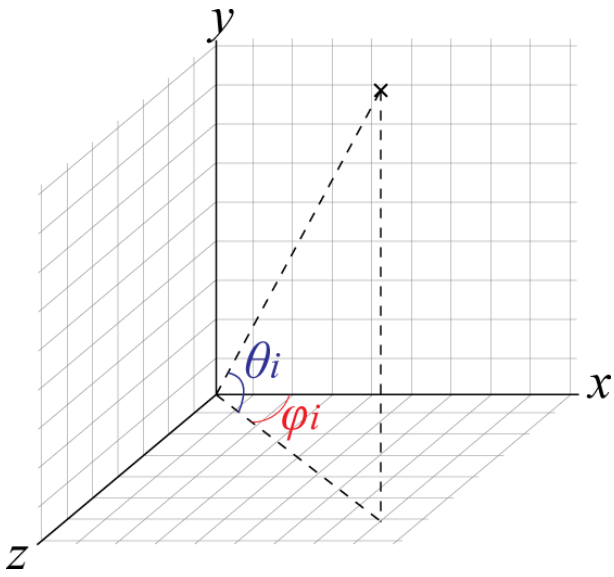


Figure 4.1: Initial angles in the employed coordinate system

Every shuttle trajectory possibly played during a badminton match can now be modelled. A trajectory is characterized by six initial parameters: three position parameters x_i, y_i, z_i, two direction parameters θ_i, φ_i and one velocity parameter v_i.

4.3 Projected model

The general mapping between model coordinates and image domain coordinates is described by equation 3.2. The 3D model of the shuttlecock dynamics is plugged in this equation:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & 1 \end{pmatrix} \begin{pmatrix} x_i + \cos(\varphi_i)\frac{{v_T}^2}{g} ln\left(\frac{v_i \cos(\theta_i) gt + v_T^2}{v_T^2}\right) \\ y_i + \frac{v_T^2}{g} \ln\left|\frac{sin\left[\frac{gt}{v_T} + tan^{-1}\left(\frac{v_T}{v_i \sin(\theta_i)}\right)\right]}{sin\left[tan^{-1}\left(\frac{v_T}{v_i \sin(\theta_i)}\right)\right]}\right| \\ z_i + \sin(\varphi_i)\frac{{v_T}^2}{g} ln\left(\frac{v_i \cos(\theta_i) gt + v_T^2}{v_T^2}\right) \\ 1 \end{pmatrix} \tag{4.13}$$

4.4 Visualizations

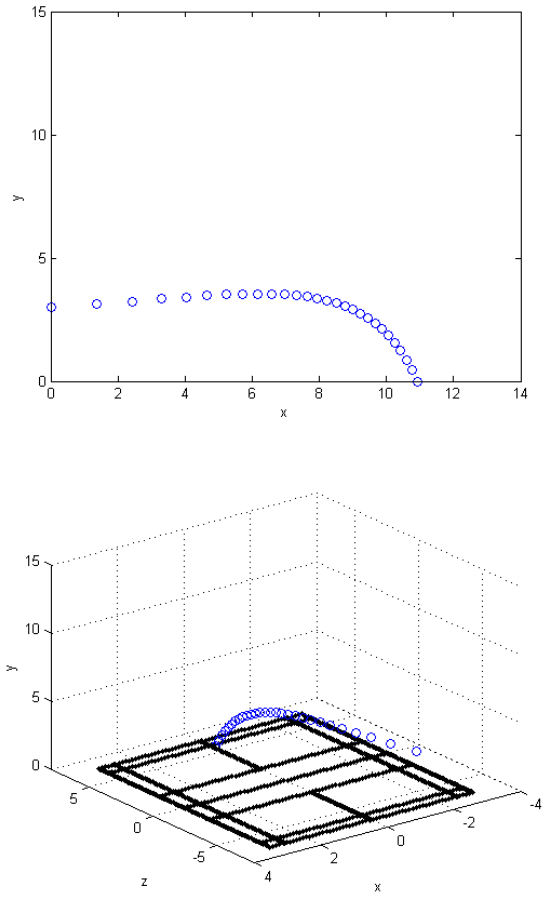
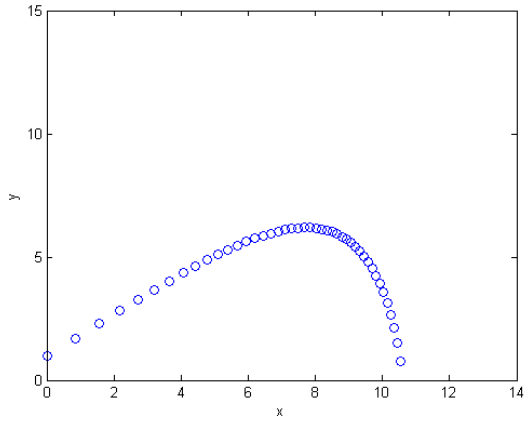


Figure 4.2: Plots of the calculated 2D and 3D model initiated with $x_i = -2$, $y_i = 3$, $z_i = -5$,$v_i = 40m/s$ $\theta_i = 5°$, $\varphi_i = 80°$. For the 2D model plot, y_i is added to the calculated y.

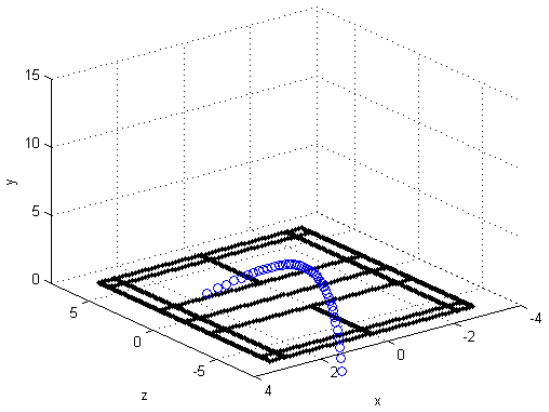
(a) Projected model on video 1

(b) Projected model on video 2

Figure 4.3: Image plane projection of the 3D model initiated with $x_i = -2$, $y_i = 3$, $z_i = -5$,$v_i = 40m/s$ $\theta_i = 5°$, $\varphi_i = 80°$.



(a) 2D model



(b) 3D model

Figure 4.4: Plots of the calculated 2D and 3D model initiated with $x_i = 2$, $y_i = 1$, $z_i = 1$,$v_i = 30m/s$ $\theta_i = 40°$, $\varphi_i = 270°$. For the 2D model plot, y_i is added to the calculated y.

(a) Projected model on video 3

(b) Projected model on video 6

Figure 4.5: Image plane projection of the 3D model initiated with $x_i = 2$, $y_i = 1$, $z_i = 1$,$v_i = 30m/s$ $\theta_i = 40°$, $\varphi_i = 270°$.

Chapter 5

Trajectory classifier

5.1 False positive detection

In an ideal situation the model fitting step itself would eliminate outliers. After all, blob trajectories that can not be matched to a model trajectory should not contain a ground truth trajectory. However, the model fitting step still yields false positives (a blob trajectory considered for further investigation that does not contain a ground truth) and false negatives (a blob trajectory that is not considered for further investigation that contains a ground truth trajectory). See section 3.4.4 for a more detailed discussion. Also, the model fitting is time intensive task, pre-filtering blob trajectories can have a positive influence on the overall speed of the algorithm. Thus, research is carried out to find properties of the blob trajectories that can foresee false positive detection in an earlier stage.

To ease the reading, blob trajectories that contain a ground truth trajectory are further mentioned as observed trajectories. In this context, blobs are referred as observations.

5.1.1 Trajectory length boundaries

Blob trajectories are filtered based on length. The length boundaries are calculated on the run when all possible model trajectories are generated (see 5.3.1). This is important since the model trajectory length boundaries are dependent on the P matrix and can thus differ from video to video. The shortest and longest model trajectories are used as boundaries. The shortest model trajectories are only two observations long. By visual observation, one can determine that these trajectories are infeasible to contain a ground truth trajectories. Therefore the lower limit is increased by one. The upper limit is increased by 10 percent. Observed blob trajectories can be longer than the theoretical trajectories due to non shuttle blobs matched to the trajectory.

The above described reasoning breaks down because of cases where a huge number of non shuttle blobs are matched to an observed trajectory. Such a case can be seen in figure 3.13. So the implemented trajectory length filter only exploits the lower bound.

5.1.2 Trajectory derivative properties

Ground truth trajectories have a characteristic course (see chapter 4). Some of these properties can be expressed in terms of slopes. If a shuttlecock leaves the racket with a positive angle ($\theta_i > 0$ in shuttlecock model described in chapter 4), then the shuttlecock trajectory has a close-to-constant positive slope in the first part and an increasing negative slope in the second part. If a shuttlecock leaves the racket with a negative angle ($\theta_i < 0$), then the shuttlecock trajectory has an increasing negative slope.

To interfere about these properties, a piecewise interpolation is made in the model domain. Therefore, sequences of three image plane observations are placed in a linear system (equation (5.2)) to calculate the 3D line that projects on these three observations. The linear system follows from the projective transformation of a 3D line (equation (5.1)). The idea for this 3D line construction originates from [CTC+09] where a similar transformation is done for a general 3D parabolic model.

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & 1 \end{pmatrix} \begin{pmatrix} xt \\ yt \\ zt \\ 1 \end{pmatrix} \tag{5.1}$$

$$\begin{pmatrix} p_{11}t_1 - u_1p_{31}t_1 & p_{12}t_1 - u_1p_{32}t_1 & p_{13}t_1 - u_1p_{33}t_1 \\ p_{21}t_1 - v_1p_{31}t_1 & p_{22}t_1 - v_1p_{32}t_1 & p_{23}t_1 - v_1p_{33}t_1 \\ p_{11}t_2 - u_2p_{31}t_2 & p_{12}t_2 - u_2p_{32}t_2 & p_{13}t_2 - u_2p_{33}t_2 \\ p_{21}t_2 - v_2p_{31}t_2 & p_{22}t_2 - v_2p_{32}t_2 & p_{23}t_2 - v_2p_{33}t_2 \\ p_{11}t_3 - u_3p_{31}t_3 & p_{12}t_3 - u_3p_{32}t_3 & p_{13}t_3 - u_3p_{33}t_3 \\ p_{21}t_3 - v_3p_{31}t_3 & p_{22}t_3 - v_3p_{32}t_3 & p_{23}t_3 - v_3p_{33}t_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u_1 - p_{14} \\ v_1 - p_{24} \\ u_2 - p_{14} \\ v_2 - p_{24} \\ u_3 - p_{14} \\ v_3 - p_{24} \end{pmatrix} \tag{5.2}$$

Solving the linear system (equation (5.2)), gives us the coordinates of the 3D line. Next the derivative for observation i is defined as:

$$D_i = \frac{-y}{\sqrt{x^2 + y^2}} \tag{5.3}$$

with x,y,z being the solution to equation 5.2 for observations i, $i+1$ and $i+2$.

Thereupon, the second derivative for observation i is calculated as follows:

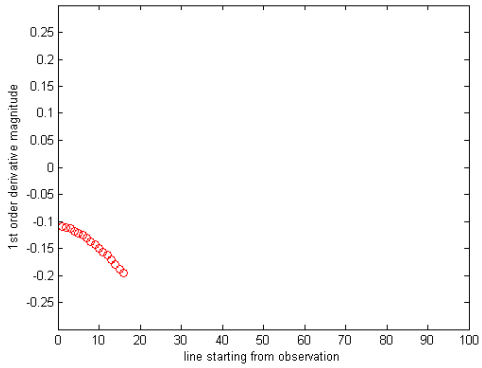
$$D_i^2 = \frac{D_{i+1} - D_i}{fps} \tag{5.4}$$

t_1, t_2 and t_3 are set to 0, $\frac{1}{fps}$ and $\frac{2}{fps}$, respectively. The abbreviation fps denotes the frame per seconds of the considered video.

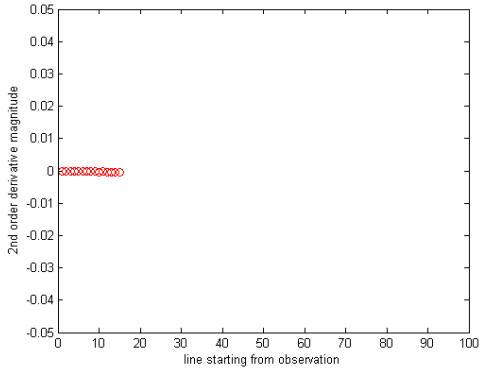
Figures 5.1 to 5.3 show some results of this method. It is seen that the second derivative is centred around zero for shuttlecock observations.

To summarize the information of a second derivative plot, following measures are defined: number of peaks (thresholding derivative values), number of flipped peaks (thresholding the discrete convolution of the derivative values with a vector $[-1\,1]$) and peak percentage (number of peaks divided by number of observations). These summary values are employed to extract blob trajectories that contain a ground truth trajectory. However, tests show that these summary values are insufficient for inlier extraction. This is caused by the fact that a lot of observed trajectories are contaminated with non shuttle blob observation. Moreover, the derivative calculation breaks down if the shuttlecock observations are mixed with non shuttle blob observations. A possible solution is to extract those observations which correspond with shuttlecock observation. Another option is to build in this derivative check in the trajectory match. Due to time constraints both of these suggestion could not be investigated. Nevertheless, they are worth being investigated in further research.

(a) Red: observed blob trajectory. Green: ground truth trajectory.



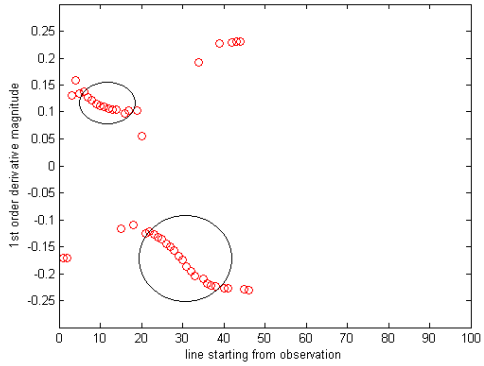
(b) The derivative is negative and decreasing as the shuttle drops.



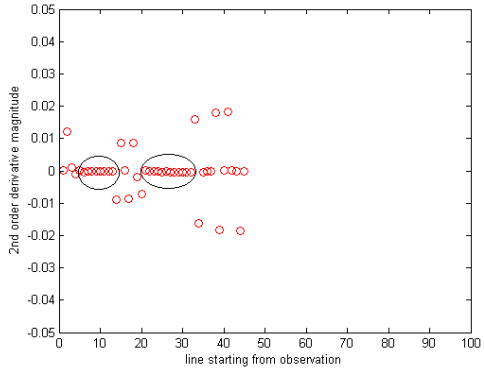
(c) The second derivative values are close to zero.

Figure 5.1: Trajectory derivative properties

(a) Red: observed blob trajectory. Green: ground truth trajectory. The blob trajectory contains non shuttle blobs.

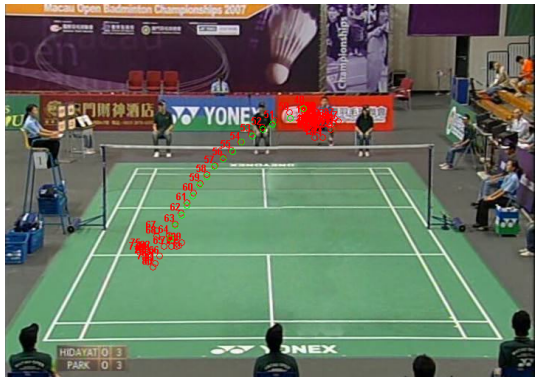


(b) The derivative is positive and decreases slowly when the shuttle is going up. When the shuttle is dropping, the derivative is negative and decreasing.

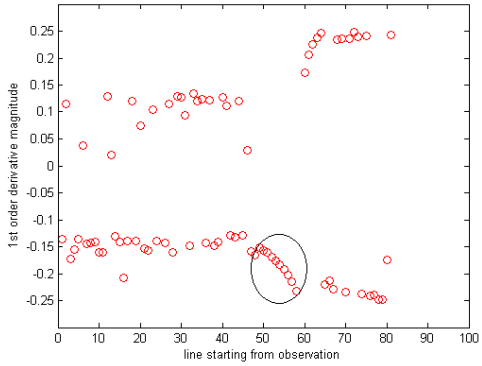


(c) The second derivative values for observations that are part of the ground truth trajectory are centred around zero.

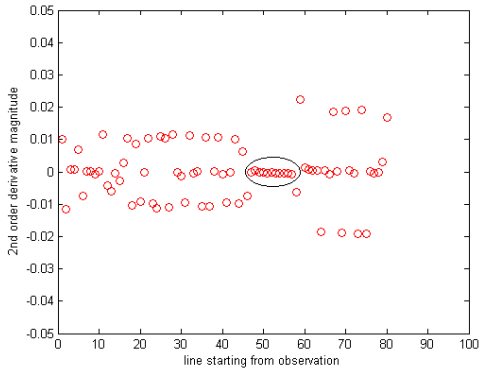
Figure 5.2: Trajectory derivative properties



(a) Red: observed blob trajectory. Green: ground truth trajectory. The blob trajectory contains non shuttle blobs.



(b) The derivative is negative and decreasing when the shuttle drops.



(c) The second derivative values for observations that are part of the ground truth trajectory are centred around zero.

Figure 5.3: Trajectory derivative properties

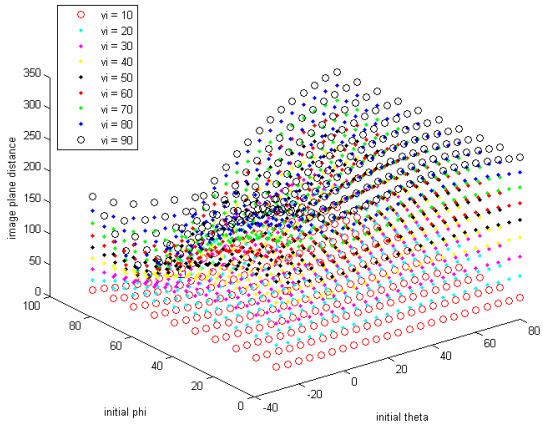
5.2 Initial parameter estimation

An initial parameter estimation aids the consequent model fitting step in several ways. Firstly, the estimation can speed up the model fitting by limiting the search space. Secondly, initial parameters can prevent that the model fitting gets stuck in local optima which are not of our interest. Prototypes of initial position estimation and initial angle estimation are implemented, yet not extensively tested. Therefore the description of these ideas are given in the future work section (see 7.1). In the following paragraph, research towards an initial speed estimate is described.

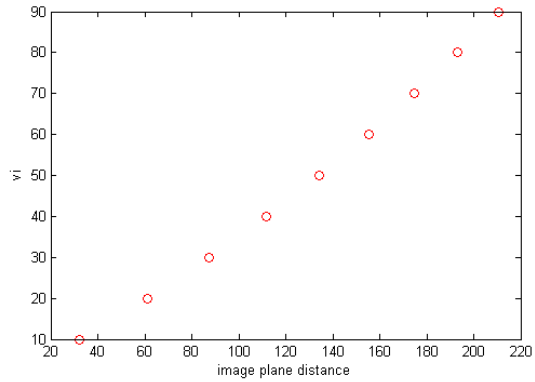
5.2.1 Initial speed estimation

Figure 5.4 shows the theoretical calculations of the relationship between the 3D speed and the image plane distance between the first two observations. To get these plots, model trajectories are generated for a range of initial values. Next, these model trajectories are projected on the image plane. Lastly, the image plane distance is calculated as the Euclidean distance between the first two projected image plane points.

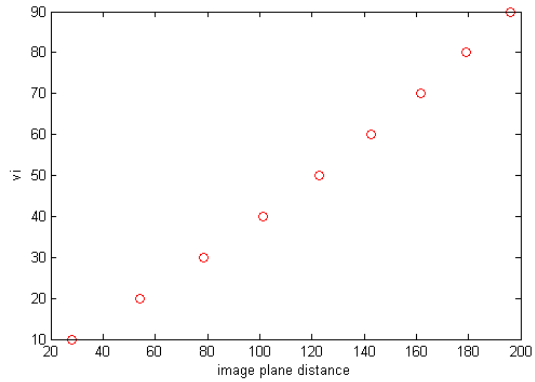
A linear relationship is found between the model trajectory initial speed and the image plane distance between the first two observations. However, the linear relationship depends on the initial position that is used to calculate the model trajectories. A good initial position and initial angles estimation are thus needed to exploit the linear relationship between image plane distance of the first observations and the initial velocity of the model trajectory. Moreover, shuttlecocks at the beginning of a shot are hard to observe by the background subtraction method (see 3.2.4). So even with a good initial position and initial angles estimation it is difficult to estimate the initial velocity.



(a) The image plane velocity is plotted for combinations of v_i, θ_i and φ_i.



(b) Initial velocity of a model trajectory in function of the image plane velocity for $\theta_i = 5°$ and $\varphi_i = 40°$.



(c) Initial velocity of a model trajectory in function of the image plane velocity for $\theta_i = 35°$ and $\varphi_i = 70°$.

Figure 5.4: Initial speed estimation

5.3 Model fitting

The shuttlecock model (chapter 4) is fitted to the blob trajectories. By means of this, inferences of the blob trajectory are made (see 5.4) using the the best matching model trajectory. If the model cannot be fitted with a certain accuracy, the blob trajectory is considered as an outlier and not further considered. In the other case, it is assumed that the blob trajectory contains a ground truth trajectory (and is thus an observed trajectory). By finding the theoretical fitting model parameters that correspond with an observed trajectory, real world positional information can be deducted for a certain sequence of image plane observations.

The mapping of the 3D shuttle model to the image plane domain is described by the projected model (see 4.3). This is a nonlinear function with parameters x_i, y_i, z_i, θ_i, φ_i and v_i, dependent variables u, v (= image plane coordinates) and independent variable t. A shorthand notation is introduced: $coord_{ProjectedModel}(t; params)$.

An observed trajectory consists of time chronological list of blob coordinates and is thus referred as $coord_{Observed}(t)$. To fit the projected model to an observed trajectory, one needs to minimize a norm between them. Formally:

$$\underset{params}{\operatorname{argmin}} \sum_{t} = \|coord_{ProjectedModel}(t; params) - coord_{Observed}(t)\| \qquad (5.5)$$

This a nonlinear regression problem. Common strategies to solve such a problem use linearization or numerical methods. Linearization can for example be done using the multivariate variant of Taylor's theorem. If the 2-norm is used, suitable numerical methods are the Gauss–Newton algorithm and the Levenberg–Marquardt algorithm (uses the GN algorithm). A more general numerical approach is found in the Nelder–Mead method. Unfortunately, no time was left during this research to investigate these methods.

5.3.1 Semi-exhaustive fitting

Since an exhaustive fitting is too time complex, a semi-exhaustive fitting is employed to find the best matching model trajectory. More concrete, a sequence of generated model trajectories is searched for the model trajectory that matches best the observed trajectory. Once this best matching model trajectory is found, one also knows the corresponding 3D model.

5.3.1.1 Trajectory database

A trajectory database is built up by calculating the model trajectories for every combination of certain parameter values. Table 5.3.1.1 lists the sets of parameter values from which the combinations are made. x_i and z_i are chosen so that every couple of these corresponds with the center of a zone (see 1.2.2). The values for y_i comprise feasible heights. v_i and θ_i are limited to physically possible values (see [CPC09]). Lastly, φ_i ranges over angles between 30° and 330°. The database has 68040 model trajectories.

parameter	**value set**
x_i	$\{1.295, 0, -1.295\}$
y_i	$\{1, 2, 3\}$
z_i	$\{-5.6, -3.4, -1.1, 1.1, 3.4, 5.6\}$
v_i	$\{5, 10, 15, 20, 25, 30, 50, 70, 90\}$
θ_i	$\{-30, -20, -10, 0, 5, 10, 20, 40, 60, 80\}$
φ_i	$\{30, 45, 60, 90, 120, 135, 150, 210, 225, 240, 270, 300, 315, 330\}$

5.3.1.2 Best match

The best matching model trajectory is found with equation 5.5 using a 2-norm. The Euclidean norm is a natural choice since image plane distances are envisaged. So, for every model trajectory from the trajectory database, the matching error with an observed trajectory is calculated as the sum of squared residuals. Consequently, the model trajectory with the lowest matching errors becomes the best matching model trajectory.

As pointed out earlier (see 3.2.4), shuttlecocks at the beginning of a shot are hard to observe by the background subtraction method. So observed trajectories might not have all the positions of the shuttlecock at the beginning of a shot. Therefore the matching error calculation is also extended with a time shift in the projected model:

$$\underset{params,\Delta t}{\operatorname{argmin}} \sum_{t} = \|coord_{ProjectedModel}(t+\Delta t; params) - coord_{Observed}(t)\| \quad (5.6)$$

where $\Delta t \in [0..3]$.

Another problem is that observed trajectories possibly contain non shuttle blobs. Especially at the beginning and the end of the observed trajectory, lots of non

shuttle blobs can be contained. To make the semi-exhaustive fitting robust against these false positives, the matching error calculation could incorporate a local matching error and ignore observed coordinates at the beginning and the end of an observed trajectory if the local error exceeds a certain thresholds. This idea however belongs to future research.

5.3.1.3 Outlier deletion

Ideally, if the matching error exceeds a certain threshold, θ_{ME}, the considered blob trajectory can be assumed to be an outlier. However, tests show that the semi-exhaustive model fitting as described above does not perform good enough for this assumption to hold.

5.3.1.4 Performance

The model cannot be fitted satisfactorily to observed trajectories that are either composed, fragmented or contaminated with a lot of non shuttle blobs (see 3.4.4). If the model fitting is performed to a more or less distinguishable trajectory, the fitted model does not always agree with the shot that is observed. Figure 5.5 to 5.8 show some fitted model trajectories.

Figure 5.5: Red: observed trajectory. Green: ground truth trajectory. The best matching model trajectory with parameters $x_i = 1.295$, $y_i = 2$, $z_i = 5.6$,$v_i = 5m/s$ $\theta_i = 20°$ and $\varphi_i = 300°$ is given in white.

Figure 5.6: Red: observed trajectory. Green: ground truth trajectory. The best matching model trajectory with parameters $x_i = -1.295$, $y_i = 2$, $z_i = 1.1$,$v_i = 10m/s$ $\theta_i = 60°$ and $\varphi_i = 315°$ is given in white.

Figure 5.7: Red: observed trajectory. Green: ground truth trajectory. The best matching model trajectory with parameters $x_i = 1.295$, $y_i = 1$, $z_i = 3.4$,$v_i = 15m/s$ $\theta_i = 20°$ and $\varphi_i = 240°$ is given in white.

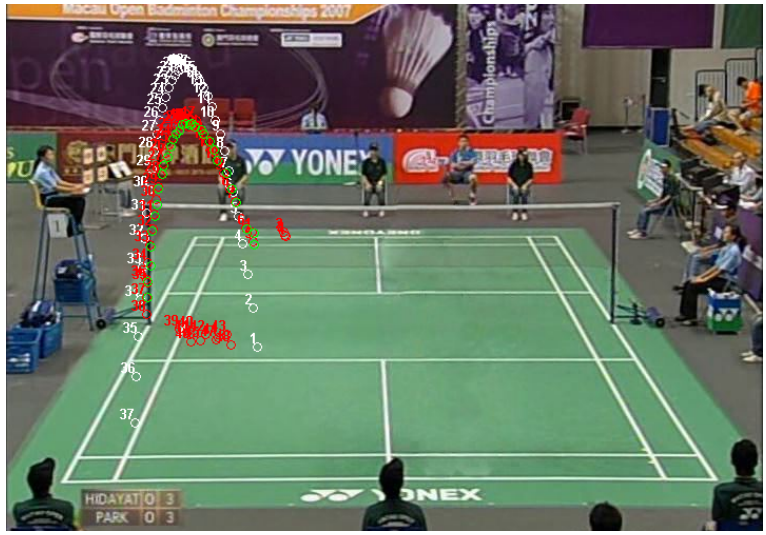


Figure 5.8: Red: observed trajectory. Green: ground truth trajectory. The best matching model trajectory with parameters $x_i = 1.295$, $y_i = 1$, $z_i = -5.6$,$v_i = 10m/s$ $\theta_i = 80°$ and $\varphi_i = 45°$ is given in white.

5.4 Classification

5.4.1 Zones

The classification step outputs an ordered list of start and end positions of the shots played during a badminton rally. As introduced in section 1.2.2, the start and end positions are denoted by a zone number. The zones are formally characterized by their center coordinates $zoneCenter_x$ and $zoneCenter_z$ in the real world model coordinate system (see 3.3). Zones at the farthest court side are depicted as the blue zones (see also figure 1.2) and are given in table 5.4.1. The red zones (closest court side) are found in table 5.4.1. These center coordinates are calculated using the official badminton court dimensions (see figure 3.2).

$zoneNumber$	$zoneCenter_x$	$zoneCenter_z$
1	-1.295	1.1
2	0	1.1
3	1.295	1.1
4	-1.295	3.4
5	0	3.4
6	1.295	3.4
7	-1.295	5.6
8	0	5.6
9	1.295	5.6

$zoneNumber$	$zoneCenter_x$	$zoneCenter_z$
1	1.295	-1.1
2	0	-1.1
3	-1.295	-1.1
4	1.295	-3.4
5	0	-3.4
6	-1.295	-3.4
7	1.295	-5.6
8	0	-5.6
9	-1.295	-5.6

5.4.2 Zone number deduction

Consider an observed trajectory $coord_{Observed}(t)$ of length $len_{Observed}$ and its best matching model trajectory $coord_{BestMatch}(t)$ of length $len_{BestMatch}$. The orthogonal projections on the court plane of the 3D coordinates of the latter

model are named by $ortho_{BestMatch}(t)$. Then a shot's start position is deducted as follows:

$$\underset{zoneNumber}{\operatorname{argmin}} \; \|zoneCenter - ortho_{BestMatch}(0)\|_2 \tag{5.7}$$

with blue zones if $\varphi_i > 180$ for the best matching model trajectory and red zones else.

A shot's end position is deducted as follows:

$$\underset{zoneNumber}{\operatorname{argmin}} \left|\left| zoneCenter - ortho_{BestMatch}\left(\frac{\min\left(len_{Observed}, len_{BestMatch}\right)}{fps}\right)\right|\right|_2 \tag{5.8}$$

with red zones if $\varphi_i > 180$ for the best matching model trajectory and blue zones else.

5.4.3 Performance

Recall the cases presented in the model fitting step (figures 5.5 to 5.8). For the first example (figure 5.5), both the correct start and end position are deducted. For the second and third example (figures 5.6 and 5.6), only start and end position are correctly found, respectively. The last case (figure 5.8) does not give any correct positional information. However, the zone number deduction runs correctly with respect to the considered model trajectory (see table 5.4.3). Thus, wrongly estimated start or end positions originate from the insufficient model fitting. For the forth example (figure 5.8), the ground plane direction of the best matching model trajectory lies in the wrong semicircle ($\varphi_i < 180$ while it should be $\varphi_i > 180$). Thus, evaluating the zone number deduction in this case is irrelevant.

example	zoneNumber start	zoneNumber end
1 (figure 5.5)	9	1
2 (figure 5.6)	1	1
3 (figure 5.7)	6	3
4 (figure 5.8)	7	3

Chapter 6

Results

Intermediate results on the training videos are described throughout the previous chapters. This chapter concentrates on the performance of the trajectory extractor and trajectory classifier on the test videos. These videos are not used for training purposes during the development phase of the algorithm. Details of the test video material are found in section 1.2.1.1.

6.1 Algorithm parameters summary

parameter	**range**	**explanation**	**section**
α_{MOG}	$(0-1]$	learning rate for the background subtraction	3.2.2
θ_{BFS}	[0..256]	saturation threshold for the blob filtering	3.3.2
θ_{TMD}	$\mathbb{R}$	trajectory matching distance threshold	3.4
θ_{TMIT}	$\mathbb{N}^*$	trajectory matching idle time threshold	3.4
α_{TMA}	$(0-1]$	learning rate for trajectory descriptor area	3.4
ω_{TMDA}	$[0-1]$	weight of area in blob-trajectory distance calculation	3.4
θ_{ME}	$\mathbb{R}$	classification matching error threshold	5.3.1

6.2 Trajectory extractor

The trajectory extractor is run on the test videos with $\alpha_{MOG} = 0.25$, $\theta_{BFS} = 256$ (implies blob size filtering only), $\theta_{TMD} = 0.08$, $\theta_{TMIT} = 5$, $\alpha_{TMA} = 0.1$ and $\omega_{TMDA} = 0.2$.

ground truth trajectories	**video 4**	**video 5**	**video 6**
not found	2	4	1
distinguishable	14	17	8
composed	0	8	8
fragmented	1	6	0
total	17	35	17
% distinguishable	0.82	0.48	0.47

6.3 Trajectory classifier

Since the false positive detection and initial parameters estimation are not developed far enough, the trajectory classifier is tested on the observed trajectories that where found by manual observation. So the test results give the performance of the model fitting step combined with the classification step.

	video 4	**video 5**	**video 6**
start zone correct	1	1	2
end zone correct	3	0	2
total zones	17·2	35·2	17·2
% correct	0.12	0.01	0.12

In general, the classifier does not perform satisfactorily. The performance on video 5 is very bad. The combination of the video resolution, camera position and lighting conditions make a difficult case from video 5.

Chapter 7

Conclusion

This thesis proposes a general outline for an algorithm to solve the badminton shot classification problem in compressed videos with a baseline angled camera. The proposed algorithm is new since previous research either uses different types of video material or does not focus on the shot classification problem. For the trajectory extractor subalgorithm, an existing background subtraction algorithm and an existing component-labeling algorithm are used in combination with general camera geometry theory. Also the trajectory matching idea is found in previous research. The trajectory extractor mainly introduces new ideas while the exploited physics based shuttle model is deduced from other research.

The total algorithm is a compound of several steps which interplay with each other. For example, a better blob filtering will serve the trajectory matching step. Vice versa, a better trajectory matching step lowers the need for a good blob filtering step. The same holds for the relationship between the inlier detection step and the model fitting. The former aids the latter but a better latter reduces the contributions of the former. Accordingly, it is of importance to consider the algorithm as a whole even when just improving single parts.

The general performance of the algorithm is inadequate. Nevertheless, several sub steps show promising performances and the chosen building blocks are justifiable. Moreover, the comprehensive identification of problems paves the way for future research.

In general, two big shortcomings can be distinguished. Firstly, the observed trajectories contain too many non shuttles blobs. Secondly, the accuracy of the model fitting is insufficient. As stated above, a certain problem can be addressed in different steps. To solve the first shortcoming one can improve the blob filtering, trajectory matching or the inlier detection. To deal with the second problem, the model fitting step itself can be ameliorated or the initial parameter estimation is improved to serve the model fitting.

7.1 Future work

The extensiveness of this section demonstrates the complexity of the problem and the steps that can be taken to refine or extend the proposed method.

7.1.1 Algorithm improvements

For the shuttle blob extraction (see 3.3):

- Try the ball tracking algorithm presented by Kopf et al. [KGFH11].
- Build appearance models to improve the shuttle blob extraction.

For the trajectory matching (see 3.4):

- Close blob trajectories that exceed a certain length as done for inactive trajectories. This prevents that infeasible long blob trajectories are constructed. However, it might be difficult choose an appropriate threshold.
- Make θ_{TMD} adaptive so that it is bigger for the first blobs that are matched to a trajectory. By doing this, one does not loose fast moving shuttle blobs that are located at the beginning of shot .
- A trajectory descriptor only characterizes a blob trajectory by an area and centroid coordinates. Future improvements may include color or structure information about the blobs on the trajectory. For example: mean luminance or the percentage of black pixels in the blob mask.
- Incorporate trajectory derivative properties (see 5.1.2) so that blobs are only matched to a blob trajectory if certain derivative properties are preserved.

For the false positive detection (see 5.1)

- Investigate the trajectory derivative properties further (see 5.1.2).
- Given an initial parameter estimate, create a system of non-linear equations by plugging in an observed trajectory in the projected model (equation 4.13). However, leave the terminal velocity (v_T) as a parameter and use a numerical method to solve the system. Now, the calculated value of the calculated terminal velocity can be used for inlier detection since it is know that the terminal velocity for a shuttlecock flight lies around 6.8 m/s.

For the initial parameter estimation (see 5.2):

- Use the players' feet for an initial position estimation. Since the feet are on the ground court plane, the exact position of the feet on the plane can be calculated using a homography. To estimate the height, one can use the property that an image plane point backprojects to a 3D ray ($X(\lambda) = P^{+}x + \lambda C$ with X the 3D points, x the 2D points, P^{+} the pseudo-inverse of P and C the camera center). So a 3D ray can be calculated from the first image plane observation. Next, one loops through several heights and checks if the 3D point constructed from this height and the court plane positional estimate lies on the calculated 3D ray.
- Try to use the 3D line estimation described in section 5.1.2 to estimate the initial angles of the best fitting model trajectory. The spherical coordinates of the line passing through the first three observations of a blob trajectory can be used for this purpose.

For the model fitting (see 5.3):

- Make the semi-exhaustive model fitting (see 5.3.1) faster by organizing the trajectories in a tree structure which can be searched more efficiently. If the database search is faster, more trajectories can be added to the database, which will serve the performance of the model fitting.
- If an initial parameters estimation is found, only search in local neighbourhood of these parameters during the semi-exhaustive model fitting (see 5.3.1).
- To make the semi-exhaustive fitting robust against non shuttle blobs in the observed trajectories, the matching error calculation could incorporate

a local matching error and ignore observed coordinates at the beginning and the end of an observed trajectory if the local error exceeds a certain thresholds.

- Use a non linear optimization algorithm or method to employ the model fitting. For example: the Gauss–Newton algorithm, the Levenberg–Marquardt algorithm or the Nelder-Mead method.

For the classification (see 5.4):

- There is an inherent redundancy in the shot annotations. Namely, it is known that the start position of shot i is the same as the end position of shot $i-1$. This information can be used to improve the classification. For instance: if the deducted start position of shot i differs from the end position of shot $i-1$ and there is a high certainty about the deduction made for shot $i-1$, than the start position of shot i is set to the end position of shot $i-1$.

7.1.2 Algorithm extensions

- Implement playing frame detection. One idea is to look at the ratio of court-coloured pixels to the number of pixels in a frame. (references: [KGFH11], [CTC+09], [ET03] and [YKWO10])
- From a badminton tactical view, it would be logical to add a speed element to the output tuples (see 1.2.2). It is sufficient to introduce a relative speed estimate in three categories. One for shuttle with a 'normal' speed and one for respectively shuttles that are significantly faster and slower than average.
- Extract other features than just the shuttle position. For example: the position of the racket can give information about the shuttle height. If the shuttle is hit on a high point ($y_i > 1$), then is the racket head is positioned above the racket shaft. However, it might be difficult to extract the racket in the compressed input videos.
- The strictly divided zones could be replaced by a more dynamic approach. Zones could be made flexible within some boundaries. Next, the found start and end points are clustered into nine zones on each side of the court. The classification step now uses the cluster centres to perform zone deduction (see 5.4.2). By doing so, the classification of shuttles with start or end position close to a zone boundary would better follow the human intuitive annotation.

- Preprocessing the videos before the analysis starts. For example: more contrast can be added so that shuttle extraction becomes easier.
- Leave the trajectory matching step (see 3.4) out and fit models on the raw blob positions rather than on constructed blob trajectories.

Appendix A

Install OpenCV on Windows

Following instructions are made for OpenCV 2.4.2 and Visual Studio 2010.

- Download the pre-compiled binaries for Visual Studio 2010 at `http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.2/`.
- Extract the binaries to `C:\opencv\`.
- Add two values to the Windows' system path variable. (Control Panel → System → Advanced system properties → Environment Variables)

  ```
  C:\opencv\build\x86\vc10\bin;
  C:\opencv\build\common\tbb\ia32\vc10;
  ```

- Restart computer.
- Open Visual Studio 2010. Create an empty 'Win 32 Console Application' in C++.
- Create a new property sheet using the Property Manager. (Right click on 'Debug | Win 32' node and select 'Add New Project Property')
- Edit the property sheet by double clicking on it. Click on 'VC++ directories' and add `C:\opencv\build\include` to 'Include Directories'.

- Add these paths to the 'Library Directories':

  ```
  C:\opencv\build\x86\vc10\lib
  C:\opencv\build\x86\vc10\bin
  ```

- Add `C:\opencv\modules\core` to the 'Source Directories'.
- Select 'Configuration Properties → Linker → Input' and add some 'Additional Dependencies'. Choose the libraries you want to use. For example:

  ```
  opencv_core242d.lib
  opencv_imgproc242d.lib
  opencv_highgui242d.lib
  opencv_ml242d.lib
  opencv_video242d.lib
  opencv_features2d242d.lib
  opencv_calib3d242d.lib
  opencv_objdetect242d.lib
  opencv_contrib242d.lib
  opencv_legacy242d.lib
  opencv_flann242d.lib
  ```

- Go to the project properties. (Project → Properties → Configuration Properties → Debugging → Environment). Switch 'Merge Environment' to 'Yes' and Add

  ```
  PATH=%PATH%;C:\opencv\build\x86\vc10\bin=
  ```

This example should run if the installation was successful:

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>

int main() {
  // show an image for 10 seconds
  cv::Mat image = cv::imread("Nyhavn.png");
  cv::namedWindow("MyImageWindow");
  cv::imshow("MyImageWindow", image);
  cv::waitKey(10000);
  return 0;
}
```

Bibliography

[CCL04] Fu Chang, Chun-jen Chen, and Chi-jen Lu. A linear-time component-labeling algorithm using contour tracing technique. In *Computer Vision and Image Understanding*, pages 206–220, 2004.

[CCL07] Hua-Tsung Chen, Hsuan-Shen Chen, and Suh-Yin Lee. Physics-Based Ball Tracking in Volleyball Videos with its Applications to Set Type Recognition and Action Detection. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 1, pages I–1097–I–1100. IEEE, April 2007.

[cita] Official Bidding Document Badminton Europe Events 2014-2018. `http://badmintoneurope.com/Clubs/CommonDrive/Components/GetWWWFile.aspx?fileID=14064`.

[citb] Round-up: Badminton tournament serves up a treat. `http://www.london2012.com/news/articles/round-badminton-tournament-serves-treat.html`.

[cit12] Spilleregler for badminton. `http://www.badminton.dk/Clubs/CommonDrive/Components/GetWWWFile.aspx?fileID=14117`, December 2012.

[Coo02] A. Cooke. Computer simulation of shuttlecock trajectories. *Sports Engineering*, 5(2):93–105, May 2002.

[CPC09] Lung-Ming Chen, Yi-Hsiang Pan, and Yung-Jen Chen. A study of shuttlecock's trajectory in badminton. *Journal of Sports Science and Medicine*, 8:657–662, 2009.

[CR12] ChakMan Chan and JennStroud Rossmann. Badminton shuttlecock aerodynamics: synthesizing experiment and theory. *Sports Engineering*, 15(2):61–71, February 2012.

[CTC+09] Hua T. Chen, Ming C. Tien, Yi W. Chen, Wen J. Tsai, and Suh Y. Lee. Physics-based ball tracking and 3D trajectory reconstruction with applications to shooting location estimation in basketball video. *J. Vis. Comun. Image Represent.*, 20(3):204–216, April 2009.

[CW07] Bingqi Chen and Zhiqiang Wang. A statistical method for analysis of technical data of a badminton match based on 2-D seriate images. *Tsinghua Science and Technology*, 12(5):594–601, October 2007.

[ET03] A. Ekin and A. M. Tekalp. Robust dominant color region detection and color-based applications for sports video. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I–21–4 vol.1. IEEE, 2003.

[FHD05] Dirk Farin, Jungong Han, and Peter H. N. De. Fast camera-calibration for the analysis of sports sequences. In *In Proceedings IEEE International Conference on Multimedia and Expo (ICME*, pages 482–485, 2005.

[FKE+04] Dirk Farin, Susanne Krabbe, Wolfgang Effelsberg, Peter H. N. de With, and Peter H. N. De. Robust Camera Calibration for Sport Videos using Court Models. 2004.

[Hea96] Michael T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill Higher Education, 2nd edition, 1996.

[HFdW08] Jungong Han, D. Farin, and P. de With. Broadcast Court-Net Sports Video Analysis Using Fast 3-D Camera Modeling. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11):1628–1638, November 2008.

[Hib] James Hibberd. FIFA: 700 mil worldwide watch World Cup final. http://www.hollywoodreporter.com/blogs/live-feed/fifa-700-mil-worldwide-watch-53991.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[KB01] P. Kaewtrakulpong and R. Bowden. An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection. 2001.

[KGFH11] S. Kopf, B. Guthier, D. Farin, and Jungong Han. Analysis and retargeting of ball sports video. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 9–14. IEEE, January 2011.

[Kov] Peter Kovesi. Decomposition of a camera projection matrix. `http://www.csse.uwa.edu.au/~{}pk/Research/MatlabFns/Projective/decomposecamera.m`.

[KSH98] Taeone Kim, Yongduek Seo, and Ki-sang Hong. Physics-based 3D Position Analysis of a Soccer Ball from Monocular Image Sequences. In *Proc. ICCV*, pages 721–726, 1998.

[LD03] Ze-Nian Li and Mark S. Drew. *Fundamentals of Multimedia*. Prentice Hall, 1 edition, November 2003.

[N´] Cristóbal Carnero Li Nán. cvBlob. http://cvblob.googlecode.com.

[Nie10] M. Nieto. *Detection and tracking of vanishing points in dynamic environments*. PhD thesis, Universidad Politécnica de Madrid, 2010.

[SG99] Stauffer and Grimson. Adaptive background mixture models for real-time tracking. *IEEE Conference on Computer Vision & Pattern Recognition (CVPR)*, 2:246–252 Vol. 2, 1999.

[SK04] H. Shum and Taku Komura. A spatiotemporal approach to extract the 3D trajectory of the baseball from a single view video sequence. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, volume 3, pages 1583–1586 Vol.3. IEEE, June 2004.

[Sla] Greg Slabaugh. Computing Euler angles from a rotation matrix. `https://truesculpt.googlecode.com/hg-history/38000e9dfece971460473d5788c235fbbe82f31b/Doc/rotation_matrix_to_euler.pdf`.

[TCQC12] Baptiste D. Texier, Caroline Cohen, David Quéré, and Christophe Claneta. Shuttlecock dynamics. *Procedia Engineering*, 34:176–181, January 2012.

[Wil08] Barry D. Wilson. Development in video technology for coaching. *Sports Technol.*, 1(1):34–40, 2008.

[YKWO10] F. Yoshikawa, T. Kobayashi, K. Watanabe, and N. Otsu. Automated Service Scene Detection for Badminton Game Analysis Using CHLAC and MRA. *World Academy of Science, Engineering and Technology*, (38):935–938, February 2010.

[YRC04] Haoran Yi, Deepu Rajan, and Liang T. Chia. Automatic extraction of motion trajectories in compressed sports videos. In *Proceedings of the 12th annual ACM international conference on Multimedia*, MULTIMEDIA '04, pages 312–315, New York, NY, USA, 2004. ACM.

[YXL+03] Xinguo Yu, Changsheng Xu, Hon W. Leong, Qi Tian, Qing Tang, and Kong W. Wan. Trajectory-based ball detection and tracking with applications to semantic analysis of broadcast soccer video. In *Proceedings of the eleventh ACM international conference on Multimedia*, MULTIMEDIA '03, pages 11–20, New York, NY, USA, 2003. ACM.

[ZHX+07] Guangyu Zhu, Qingming Huang, Changsheng Xu, Liyuan Xing, Wen Gao, and Hongxun Yao. Human Behavior Analysis for Highlight Ranking in Broadcast Racket Sports Video. *Multimedia, IEEE Transactions on*, 9(6):1167–1182, October 2007.