M.Sc. Thesis
Master of Science in Engineering

DTU Compute
Department of Applied Mathematics and Computer Science

Automatic Localization of Impact Position in Golf Swing using Computer Vision

Janus Nørtoft Jensen
Morten Hannemose

Kgs. Lyngby 2016

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Matematiktorvet, building 303B,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Abstract

Millions of people worldwide play golf. In order to improve they use key values to evaluate their game.

In this thesis, we have analyzed videos of golf strokes and studied the feasibility of detecting the impact position between club and ball without the use of markers. We have estimated other key parameters such as the initial ball position, impact time and club trajectory.

Data for this thesis was obtained by recording videos of golf strokes with an iPhone and data from a Doppler radar from TrackMan A/S.

The results show that it is possible to estimate the impact position with satisfactory accuracy. Furthermore, we have managed to locate the initial ball position with high precision, and estimated the time of impact with sub-frame accuracy.

Preface

This thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark (DTU) under the supervision of Associate Professor Henrik Aanæs and Associate Professor Anders Bjorholm Dahl in fulfillment of the requirements for acquiring an MSc Eng degree in Mathematical Modelling and Computation. The project was done in cooperation with TrackMan A/S with CTO Fredrik Tuxen and Benjamin Braithwaite as the company supervisors.

The scope of the project was 32.5 ECTS points and it was carried out from March to August 2016.

Kgs. Lyngby, August 13, 2016

Janus Nørtoft Jensen Morten Hannemose

Acknowledgments

We would like to thank our advisors for valuable advice and feedback throughout the project period.

We are grateful for the office and food that was generously provided to us by TrackMan A/S. This has made the project a delight to work on.

A great thanks goes to Ben, our office mate and advisor, for a pleasurable working environment and great conversations. Sorry for our poor singing voices that you had to endure on a daily basis.

We also wish to acknowledge the two golf players, Mads and Matt, for their beautiful shots in our datasets.

Personally I, Morten, would like to thank Janus for helping me with various daily necessities in the first few months of the project due to my temporarily reduced physical capabilities that come with a fractured hip.

We would also like to thank Morten's mother, Annette, for lending her car to us during the entire project period.

Finally, we would like to thank you, the reader, for taking the time to read this thesis.

Contents

List of abbreviations

DS1 Dataset 1

DS2 Dataset 2

DS2n Dataset 2 normal lens

DS2z Dataset 2 zoom lens

FPS Frames per second

GMM Gaussian mixture model

GRF Gibbs random field

MAP Maximum a posteriori

MRF Markov random field

PCA Principal component analysis

PDF Probability density function

PnP Perspective-n-Point

RANSAC RANdom SAmple Consensus

RMSE Root mean square error

ROI Region of interest

SVD Singular value decomposition

CHAPTER 1

Introduction

Golf is a sport enjoyed by millions of people worldwide. Players range from professionals to people whose primary or sole competitor is themselves and who play primarily for fun. What is common for most players is that they are interested in improving their game. Traditionally trainers have told players how to swing the club and move their body to achieve better results. Over the last decade, this process has become more and more data-driven using modern electronic aids. One of the leading companies within this field is TrackMan A/S.

TrackMan A/S is a Danish company specialized in using Doppler radar technology for various ball sports like golf, baseball, and tennis. Their golf radar tracks the ball and club trajectories and from these, they are able to derive a various range of different parameters such as club speed, ball speed, shot distance, spin rate etc. Such numbers are very useful for a golf player trying to improve his game. Some values they are however not able to obtain precisely using only radar. They are interested in examining if some of these can be determined with greater precision using computer vision. It is of special interest to determine where on the club head the ball is hit. This information is a key factor for a player when determining if they hit the ball optimally and consistently.

1.1 Problem statement

In this thesis, we will use computer vision to analyze videos of golf strokes recorded with an iPhone combined with Doppler radar data.

Specifically, we will study the feasibility of predicting the impact location of the ball on a club head in a stroke without the use of markers. In order to do this, we will also need to estimate other key values such as the initial ball position, shaft angles, and club head trajectory.

We aim to compute the impact time between the club and ball with sub-frame accuracy. This can be used both in the prediction of the impact location and for synchronizing the video and radar data.

CHAPTER 2

Theory

2.1 Related work

Currently, there exist a few commercial products that are able to detect the impact position in golf. They are all based on a combination of high-speed cameras and markers. Some of these products are the Foresight HMT (Foresight 2016), Qualisys (Qualisys 2016) and Gears Golf (Gears 2016).

We've been unable to find any previously published literature on detecting the impact position. There exists very little published literature on detecting a golf club in videos.

Gehrig et al. (2003) detect the golf club in videos by detecting motion in frames and detecting the straight lines in the motion. They fit a 6th-degree polynomial robustly to the endpoints of these lines using RANSAC. The resulting polynomial constitutes a global model of the golf swing. Their work was done on a dataset of videos recorded orthogonally to the shooting direction.

Woodward and Delmas (2005) designed a system using cheap web cameras with one camera in front and one to the side of a putter hitting a golf ball. Colored stickers are attached to the putter and a colored ball is used. These elements are detected in both cameras and triangulated which yields 3-D positions of the ball and the face of the putter.

Some work on modeling a golf swing has been done. It has shown that the double pendulum is a good model of the entire swing (Cochran and Stobbs 1968; Jorgensen 1999). This is because the arms act as the upper pendulum and the golf club acts as the lower pendulum. If only the lower part of the swing is considered, a single pendulum or circle is also a reasonable model (Tuxen 2014).

2.2 Notation

This section briefly outlines notation used in this thesis.

Matrices (and images) are denoted by bold uppercase letters e.g. $\mathbf{M}$, and vectors are denoted by bold lowercase letters, e.g. $\mathbf{v}$.

When we have a point $\mathbf{p}_v$ in homogeneous coordinates and refer to $p_{v,x}$ and $p_{v,y}$ we mean the first coordinate divided by the scale and the second coordinate divided by the scale, i.e.

$$\mathbf{p}_v = \begin{bmatrix} s \cdot p_{v,x} \\ s \cdot p_{v,y} \\ s \end{bmatrix}.$$

The matrices $\mathbf{R}_x(\theta)$, $\mathbf{R}_y(\theta)$ and $\mathbf{R}_z(\theta)$ denote right-handed rotations around the x,y and z-axis respectively, i.e.

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix},$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix},$$

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Whenever we apply functions like max and median on matrices, they are applied as pixel-wise operations. E.g. if $\mathbf{A} = (A_{ij})$ and $\mathbf{B} = (B_{ij})$ are two matrices then

$$\mathbf{C} = \max(\mathbf{A}, \mathbf{B}) \Rightarrow C_{ij} = \max(A_{ij}, B_{ij}).$$

The Hadamard product or element-wise product, is denoted by $\circ$, i.e.

$$\mathbf{C} = \mathbf{A} \circ \mathbf{B} \Rightarrow C_{ij} = A_{ij} B_{ij}.$$

Throughout the thesis we use a right-handed coordinate system where the x-axis is positive to the right, the y-axis is positive downwards and the z-axis is positive outwards in the depth direction.

2.3 Golf terminology

In this section, we will briefly outline some golf terminology which is used in the thesis.

Golf clubs

A golf club consists of a shaft and a head and is made of various materials. The part of the club the furthest away from the shaft is sometimes called the toe, and the part closest to the shaft is called the heel. Golf clubs are divided into different types, and a golfer usually has a wide assortment of these different types. Our datasets contain strokes with the following types:

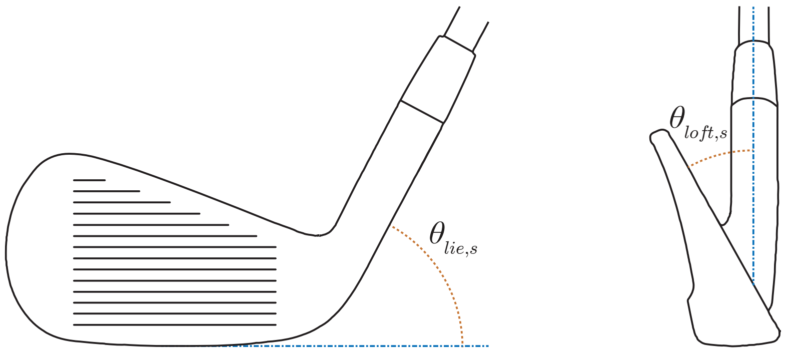
Drivers have big hollowed out heads, rounded faces, and very light shafts, and produce the longest ball distances.

Irons have solid metal heads and flat angled club faces which allow for different ball flights. They have shorter ball distances compared to drivers.

Wedges also have solid metal heads and even flatter club faces than irons. They produce short and high ball flights.

Static club angles

The shape of a golf club is very important in determining its properties. A club is manufactured with a static lie angle $\theta_{lie,s}$ and a static loft angle $\theta_{loft,s}$. The static lie angle is the angle between the shaft and the ground line when the lines on the club face are parallel with the ground line as illustrated in Figure 2.1(a). The static loft angle is the angle between the hitting surface and the shaft. This angle is shown in Figure 2.1(b).



(a) Static lie angle $\theta_{lie,s}$.

(b) Static loft angle $\theta_{loft,s}$.

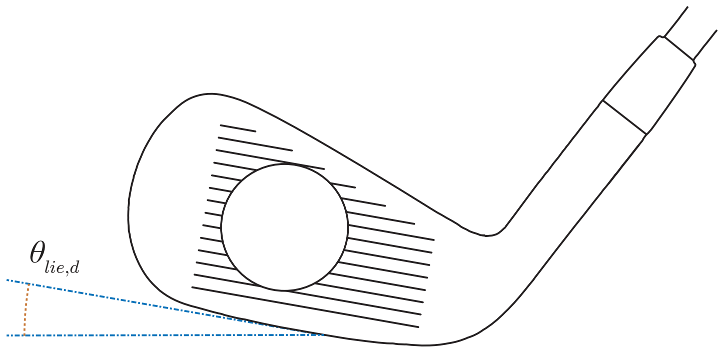
Figure 2.1: Illustration of how the static lie and loft angles are defined.

Dynamic club angles

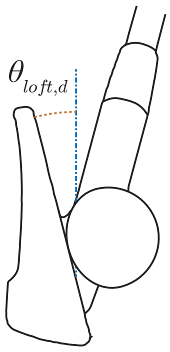
The pose of the golf club at impact time is essential for a golf stroke. This pose is determined by the dynamic lie angle $\theta_{lie,d}$, the dynamic loft angle $\theta_{loft,d}$ and the dynamic face angle $\theta_{face,d}$. The dynamic lie angle is the angle between the lines on the club face and horizontal at impact. The dynamic loft is the angle between the hitting surface and vertical at impact, and the dynamic face angle is the angle between the hitting surface and the x-axis at impact. All of these are visualized in Figure 2.2 on the following page.

When we refer to the neutral position we mean the position where

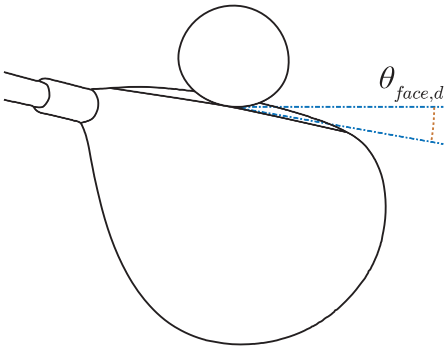
$$\theta_{lie,d} = 0^\circ, \quad \theta_{face,d} = 0^\circ, \quad \theta_{loft,d} = \theta_{loft,s}.$$



(a) Dynamic lie angle.



(b) Dynamic loft angle.



(c) Dynamic face angle.

Figure 2.2: Definitions of the dynamic lie, loft and face angles.

Anchor point

We define a new point on the club: the anchor point. This point is the visual intersection of the head with the line going through the middle of the shaft. This point is shown in Figure 2.3 on the next page. Because it is defined as a visual intersection, it changes w.r.t. how the club is viewed. We will use this point as our reference point to measure the impact location.

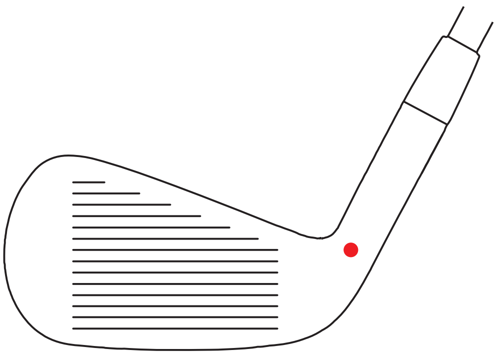


Figure 2.3: The red point is the anchor point of the golf club.

2.4 Canny edge detector

The Canny edge detector is a way of detecting edges in an image as the name suggests (Canny 1986). Let $\mathbf{I}$ be the grayscale image that we want to detect edges in. First, the image gradients are calculated in the x and y-direction. We denote these by $\mathbf{G}_x$ and $\mathbf{G}_y$. The magnitude of the gradient

$$\mathbf{G}_{mag} = \sqrt{\mathbf{G}_x + \mathbf{G}_y},$$

is used as the edge measure.

Each pixel in $\mathbf{G}_{mag}$ is compared to its neighbors in the direction of the gradient $(\mathbf{G}_x, \mathbf{G}_y)$ and $(-\mathbf{G}_x, -\mathbf{G}_y)$, and only those which are larger than their neighbors are kept. This process is called non-maximum suppression and ensures that the detected edges are thin.

The resulting image $\mathbf{E}$ is then thresholded with two different thresholds τ_1 and τ_2 with $\tau_1 > \tau_2$ to obtain the binary images

$$\mathbf{E}_1 = \mathbf{E} > \tau_1, \quad \mathbf{E}_2 = \mathbf{E} > \tau_2.$$

The edges in $\mathbf{E}_1$ are called strong edges and the ones in $\mathbf{E}_2$ are called weak edges. Connected components from $\mathbf{E}_2$ are then extracted. A component is deemed an edge segment if one or more of its pixels are strong edges, i.e. found in $\mathbf{E}_1$. The union of edge segments then constitutes the final edge image.

2.5 Hough transform

The Hough transform was originally an algorithm for detecting straight lines in images where edges have been detected (Duda and Hart 1972; Hough 1962). This is achieved through a voting scheme where each edge pixel votes for all possible lines going through it. The space of lines is called the Hough space and its axes are given by the parameters in the model, thus its dimensionality is the same as the degrees of freedom in the model. The lines receiving the most votes can then be extracted as the dominant straight lines in the image. With suitable parameterizations, this can be generalized to other models.

Hough transform for circles

A circle may be parameterized by the model

$$(x-a)^2 + (y-b)^2 = r^2,$$

where (a, b) is the center and r is the radius of the circle. For this model, the Hough space is thus 3-dimensional with axes a, b and r. An edge pixel with coordinates (x_0, y_0) will vote for circles of the type

$$(x_0-a)^2 + (y_0-b)^2 = r^2.$$

If r is kept fixed we see this is a circle with center (x_0, y_0) that is voted for in the Hough space. When r is increased the votes are cast for a circle with the same center but with increased radius. Doing this for all r's in the Hough space it is seen that the point (x_0, y_0) votes for a cone in the Hough space. After the voting process, the largest peaks in the Hough space can be extracted as the dominant circles. A resolution for each parameter in the Hough space must be chosen initially.

2.6 RANdom SAmple Consensus (RANSAC)

RANdom SAmple Consensus (RANSAC) is an algorithm which is widely used in computer vision for fitting parameterized models to data which may contain gross errors, where classical techniques like least squares estimation fail (Fischler and Bolles 1981).

The idea behind the algorithm is to fit the given model to a minimal subset of data points, i.e. the smallest number of points necessary to fit the model parameters and then decide the amount of data points which agree with this fitted model, that is, split the data points into inliers and outliers based on some distance function d. The inliers are sometimes denoted the consensus set. This process is repeated a number of times after which the model with the largest consensus set is chosen as the best model. In Section 2.6 on the facing page, we describe how the number of iterations can be chosen. Finally, the parameters may be improved by fitting the model to all data points in the largest consensus set via e.g. least squares. The algorithm is summarized in Table 2.4 on the next page. This process, thus yields a model only fitted to the points which the models fit well. The RANSAC algorithm can be seen as sampling randomly in an infinite-resolution Hough space, where points that have high values in Hough space have high probability of being sampled.

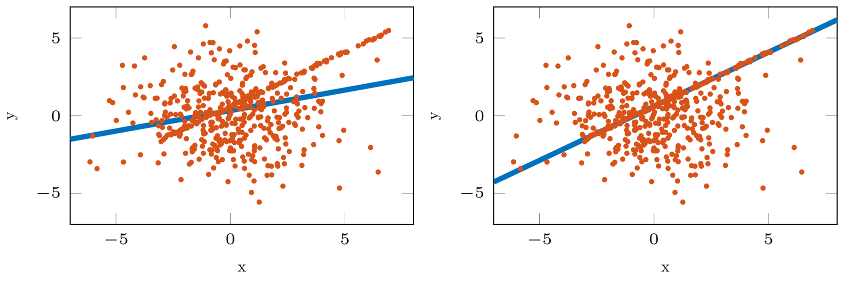
In Figure 2.5 an example of the different solutions obtained by classical least squares and RANSAC is shown when the fitted model is a straight line. In the example, there is a clear straight line, but also a very large proportion of outliers. The least squares solution uses all points for fitting and thus emphasizes the outlier points a lot while the RANSAC solution ignores them and successfully fits the straight line.

Table 2.4: Outline of the original RANSAC algorithm.

Given a model $M(\theta)$ which has n degrees of freedom, a set of m data points $\mathbf{p} = \{p_1, p_2, \ldots, p_m\}$ where $m \geq n$, a distance function d and a threshold T repeat the process below N times

1. Sample n data points from $\mathbf{p}$ randomly.
2. Estimate the model parameters θ and model $M(\theta)$ from the sample.
3. Compute the consensus set for the model, i.e. determine the p_i's for which
$$d(p_i, M(\theta)) < T,$$

The model with the largest consensus set is chosen and the parameters may be improved by fitting a new model to this set.



(a) Line fitted with least squares. (b) Line fitted with RANSAC.

Figure 2.5: Example of two fitted lines when using least squares and RANSAC.

Determining the number of iterations

We want to determine how many iterations N are needed to fit the right model. If n points are needed to fit the model parameters, we need to have at least one iteration where all n points are inliers from the underlying model. In most cases, is it not feasible to try out every combination of n points and most often not necessary (Hartley and Zisserman 2004).

Determining number of iterations with a priori knowledge

If we have a priori knowledge about the probability ϵ that a given point is an outlier, or have an educated guess about this, we can determine N for a given probability p that at least one subset contains only inliers. The value of p is usually set to a value very close to one. We can deduce that $1-\epsilon$ is the probability of a point being an inlier and thus that $(1-\epsilon)^n$ is the probability that a subset of n points are all inliers. The probability of drawing a sample with at least one outlier is thus $1-(1-\epsilon)^n$. The probability that this happens in N trials, i.e. that no subset has only inliers is

$$(1-(1-\epsilon)^n)^N.$$

Since p is the probability that at least one subset contains only inliers we have that

$$\begin{aligned}(1-(1-\epsilon)^n)^N &= 1-p \Rightarrow \\ N &= \frac{\log(1-p)}{\log((1-(1-\epsilon)^n))},\end{aligned} \tag{2.1}$$

which is then rounded up. Table 2.6 shows $\lceil N \rceil$ for different ϵ and n and $p = 0.99$.

Table 2.6: Number of iterations $\lceil N \rceil$ needed for different ϵ and n when $p = 0.99$.

n	ϵ					
	0.05	0.1	0.2	0.3	0.4	0.5
2	2	3	5	7	11	17
3	3	4	7	11	19	35
4	3	5	9	17	34	72
5	4	6	12	26	57	146
6	4	7	16	37	97	293
7	4	8	20	54	163	588

Determining number of iterations adaptively

In most cases we do not know ϵ or have an educated guess hereof. We can however utilize that the RANSAC algorithm naturally gives an upper bound $\tilde{\epsilon}$ of ϵ. If we have m data points and the largest consensus set after an iteration has size s we have the upper bound

$$\tilde{\epsilon} = 1 - \frac{s}{m},$$

which then can be using to determining an upper bound $\tilde{N}$ of N by plugging it into Equation (2.1). Hence, in each iteration we calculate $\tilde{\epsilon}$ and $\tilde{N}$ and terminate the algorithm when the number of iterations exceeds $\tilde{N}$. Initially, we have $s = 0$ and $\tilde{N}$ is set to ∞. Table 2.7 on the next page shows an example with $m = 25$ and 18 points in the largest consensus set.

Table 2.7: Example of adaptively choosing the number of iterations. In this example $n = 2$, $m = 25$ and the largest consensus set has size 18. It is seen that the algorithm terminates after iteration 7.

Iteration	0	1	2	3	4	5	6	7
s	0	3	8	8	12	12	15	18
$\tilde{\epsilon}$	1	0.88	0.68	0.68	0.52	0.52	0.40	0.28
$\tilde{N}$	∞	318	43	43	18	18	11	7

2.7 Markov random fields (MRFs)

In images, neighboring pixels often include information about each other. When segmenting an image this information can be modeled and utilized with an MRF, to obtain a segmentation with some degree of homogeneity, i.e. a good chance that neighboring pixels belong to the same class. In this section, we present the MRF modeling framework in the context of image segmentation. MRFs are also used for various other tasks in computer vision, and we refer the reader to Li (2009) for more information. This section is largely based on Aanæs (2015, Chapter 8).

Random field

Random fields are generalizations of stochastic processes in the sense that the underlying random variables may depend on other things than just time. Formally a random field $\mathcal{F}$ is a set of random variables F_i associated to a set of sites $\mathcal{S}$, i.e.

$$\mathcal{F} = \{F_i \,:\, i \in \mathcal{S}\},$$

where the sites $\mathcal{S}$ are elements in some topological space. In an image, the set of sites is the set of pixels. The random variables F_i can take values f_i in a set of labels $\mathcal{L}$. For segmentation of foreground and background it is typical to use $\mathcal{L} = \{0, 1\}$ or $\mathcal{L} = \{-1, 1\}$.

Markov random field (MRF)

The Markov part of MRFs indicates that they are random fields with a Markov property. The regular Markov property states that the conditional probability of a future state given the past states is only dependent on the present state, that is $P(x_{t+1} \,|\, x_t, x_{t-1}, \ldots, x_1) = P(x_{t+1} \,|\, x_t)$. In a random field, the sites $\mathcal{S}$ are not necessarily ordered in time but they are however related via a neighborhood structure $\mathcal{N}$, which the Markov property can be generalized to. A neighborhood structure is a set of neighborhoods N_i for each site i, where the neighborhoods have the properties that

- no site can be a neighbor to itself, i.e. $i \notin N_i$,

- if a site i is a neighbor of j then j is a neighbor of i, i.e. $i \in N_j \Leftrightarrow j \in N_i$.

The Markov property is generalized to sites by

$$P(F_i = f_i \,|\, F_j = f_j,\ j \in \{\mathcal{S} \backslash i\}) = P(F_i = f_i \,|\, F_j = f_j,\ j \in N_i).$$

Images have a natural 4-neighborhood structure where the neighborhood of a pixel is the pixel above, below, to the left and to the right of it. Furthermore a MRF must satisfy that,

$$P(f) > 0,\ \forall f \in \mathbb{F},$$

where $\mathbb{F} = \mathcal{L}^m$ is the set of all labelings, and $f = \{F_1 = f_1, F_2 = f_2, \ldots, F_m = f_m\}$ is a specific labeling, hence the constraint says that all labelings must be possible.

An MRF is just a distribution on a field. However, what we are interested in is an image segmentation, i.e. an optimal labeling. Typically, this is found as the maximum a posteriori (MAP) solution to the MRF. It turns out that this solution is much easier to compute using Gibbs random fields (GRFs), which we define in the next section. Fortunately the Hammersley-Clifford theorem states these are equivalent with respect to the same neighborhood structure (Hammersley and Clifford 1971).

Gibbs random fields (GRFs)

A GRF is a random field which follows a Gibbs distribution. Before we introduce the Gibbs distribution, we must first define the notion of cliques. A clique c is a subset of sites in $\mathcal{S}$. It can be either a single site i or a set of sites which are all neighbors to each other. The cliques containing only single sites $\mathcal{C}_1 = \mathcal{S}$ are called the set of 1-cliques, the cliques containing two sites $\mathcal{C}_2 = \{\{i, j\} \,|\, i \in \mathcal{S},\ j \in N_i\}$ are called the set of 2-cliques and so on. The collection of all cliques is then $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots$. For images with the natural 4-neighborhood structure $\mathcal{C}$ consists only of 1- and 2-cliques, since no three pixels are all pairwise neighbors.

Let $f = \{F_1 = f_1, F_2 = f_2, \ldots, F_m = f_m\}$ be a labeling. Then the Gibbs distribution is given by

$$P(f) = \frac{\exp\left(\frac{-U(f)}{T}\right)}{\sum_{f' \in \mathbb{F}} \exp\left(\frac{-U(f')}{T}\right)}, \tag{2.2}$$

where

$$U(f) = \sum_{c \in \mathcal{C}} V_c(f).$$

T is a scalar called the temperature, and $U(f)$ is called the energy function and is defined as a sum of clique potentials $V_c(f)$ which, given f, are energy functions of the labels assigned to the sites in clique c. From Equation (2.2) on page 12, it can easily be seen that

$$\arg\max_{f\in\mathbb{F}} P(f) = \arg\min_{f\in\mathbb{F}} U(f),$$

hence finding the optimal labeling comes down to minimizing an energy function.

Image segmentation

In order to use MRFs for image segmentation, we have to define the clique potentials $V_c(f)$ to obtain an energy function $U(f)$. As mentioned earlier, if we model the MRF using the 4-neighborhood structure there are only 1-cliques and 2-cliques. If $V_c(f)$ is independent on the position of clique c then $U(f)$ can be rewritten as a sum of terms corresponding to cliques of certain size

$$\begin{aligned} U(f) &= \sum_{c\in\mathcal{C}} V_c(f) \\ &= \sum_{i\in\mathcal{C}_1} V_1(f_i) + \sum_{\{i,j\}\in\mathcal{C}_2} V_2(f_i, f_j), \end{aligned}$$

thus we need to define the functions V_1 and V_2 determining the 1-clique and 2-clique potentials respectively. When we only have two labels $\mathcal{L} = \{0, 1\}$ the 1-clique potentials are defined as

$$V_1(f_i) = \begin{cases} -\log(p_0(x)) & \text{if } f_i = 0 \\ -\log(p_1(x)) & \text{if } f_i = 1 \end{cases},$$

where $p_0(x)$ and $p_1(x)$ are two probability density functions (PDFs) belonging to label 0 and label 1, as a function of pixel value. These are usually defined based on the image data. If only the 1-clique potentials were used, we would thus get a pixel-wise classification without the smoothness prior. The 2-clique potentials are defined as

$$V_2(f_i, f_j) = \begin{cases} -\beta_{ij} & \text{if } f_i = f_j \\ \beta_{ij} & \text{if } f_i \neq f_j \end{cases},$$

where β_{ij} are predefined scalars. The 2-clique potentials can be regarded as smoothness priors, where we see that we reward the energy function $U(f)$, which we want to minimize, with a negative value if two neighbors have the same label and punish it with a positive value if two neighbors have different labels. Hence if the β_{ij}s are set to high values, the optimal labeling will be very homogeneous.

It has been shown that for two labels the MAP-MRF solution can be found as minimum cut in a graph (Kolmogorov and Zabih 2004).

2.8 Gaussian mixture model (GMM)

A multivariate normal distribution has PDF

$$f(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{k/2}\,|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right),$$

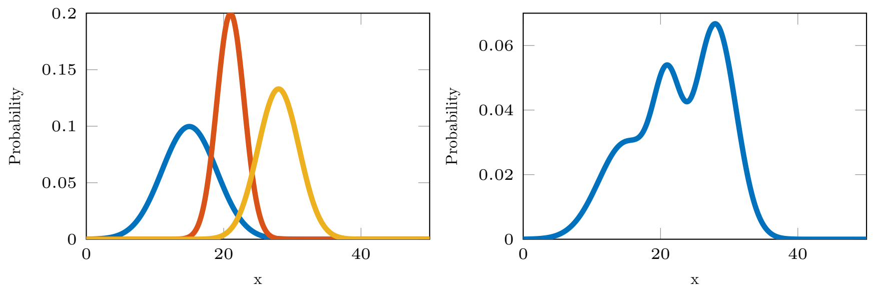
where $\mathbf{x}$ is a k-dimensional random vector, $|\boldsymbol{\Sigma}|$ is the determinant of the symmetric $k \times k$ covariance matrix $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ is the k-dimensional mean vector.

Using this we can define the PDF of a GMM as a weighted sum of N multivariate normal distributions.

$$p(\mathbf{x}) = \sum_{n=1}^{N} w_n f(\mathbf{x}|\boldsymbol{\mu}_n,\boldsymbol{\Sigma}_n), \quad \text{s.t.} \sum_{n=1}^{N} w_n = 1,$$

where w_n, $\boldsymbol{\mu}_n$ and $\boldsymbol{\Sigma}_n$ is the weight, mean vector and covariance matrix for the nth normal distribution. Figure 2.8 shows an example of a GMM for three one-dimensional normal distributions.

A GMM can be fitted to a set of data points using the expectation-maximization algorithm (Dempster et al. 1977).



(a) Plot of $f(x)$ for three one-dimensional normal distributions. (b) GMM of the normal distributions in (a)

Figure 2.8: Example of a GMM for three one-dimensional normal distributions. The normal distributions have the following parameters $\mu_1 = 15$, $\mu_2 = 21$, $\mu_3 = 28$, $\sigma_1 = 4$, $\sigma_2 = 2$ and $\sigma_3 = 3$, and the weights for the GMM are $w_1 = 0.3$, $w_2 = 0.2$, $w_3 = 0.5$.

2.9 Dynamic programming

Dynamic programming is a method that can be used to solve problems that are very computationally expensive to solve exhaustively. The problem is broken down into smaller subproblems where each problem is only solved once, which makes it computationally efficient. In dynamic programming, the structure of the problem is also used to design how to store the solutions of the subproblems in memory.

Checkerboard problem

Let each square on an $n \times m$ checkerboard have an assigned value $d(i,j)$, where i is the row and j is the column. We want to find the path with the maximal sum that goes from left to right on this checkerboard, but we can only move diagonally up right, straight right, and diagonally down right. That means that from (i,j) we can only move to $\{(i-1,j+1),(i,j+1),(i+1,j+1)\}$. In order to break this problem into subproblems, we will consider the maximal path that goes through (i,j), and designate the sum of this path as $q(i,j)$. Since there are only three possible squares that we can come from in order to reach this position, so the maximal one of them will be the one that the path has to go through to be maximal. We can now compute $q(i,j)$ as follows.

$$q(i,j) = \begin{cases} \infty & \text{if } i < 1 \text{ or } i > m \\ d(i,j) & \text{if } j = 1 \\ d(i,j) + \max(q(i-1,j-1), q(i,j-1), q(i+1,j-1)) & \text{otherwise} \end{cases}$$

This is very simple to calculate for increasing values of i as each calculation only relies on previously computed values of q. Once $q(i,m), i \in \{1,2,\ldots,n\}$ has been calculated, we have computed the value of the maximal paths that end in each square in the last column. The value of the globally maximal path will then be $\max\limits_i q(i,m)$. The squares that the path goes through can then be determined by looking at where the maximum value was taken from in the calculation of q. The path can then be computed by.

```
$p_m \leftarrow \arg\max\limits_i q(i,m)$
**for** $j = \{m-1, m-2, \ldots, 1\}$ **do**
    $r \leftarrow \{p(j+1)-1, p(j+1), p(j+1)+1\}$
    $p_j \leftarrow \arg\max\limits_{i \in r} q(i,j)$
**end for**
```

This can be used to find the path with maximal sum through an image without having to try all paths.

2.10 Geometrical least squares for straight lines

Given a set of points, we want to fit a line through the points which minimizes the sum of squared distances from the points to the line. We parameterize the line as

$$ax + by = r.$$

Given that $a^2 + b^2 = 1$, we can compute the signed distance d_i from a point to a line by

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ -r \end{bmatrix}$$

With this parameterization, we can multiply the line with a scalar s without changing the line. This yields

$$s \cdot \mathbf{d} = \begin{bmatrix} s \cdot d_1 \\ s \cdot d_2 \\ \vdots \\ s \cdot d_n \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} s \cdot a \\ s \cdot b \\ -s \cdot r \end{bmatrix} = \mathbf{X} \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} = \mathbf{Xl}$$

If we use the line multiplied by s we can compute the sum of squared distances as

$$s^2 \sum_{i=1}^{n} d_i^2 = \|s \cdot \mathbf{d}\|_2^2 = \|\mathbf{Xl}\|_2 .$$

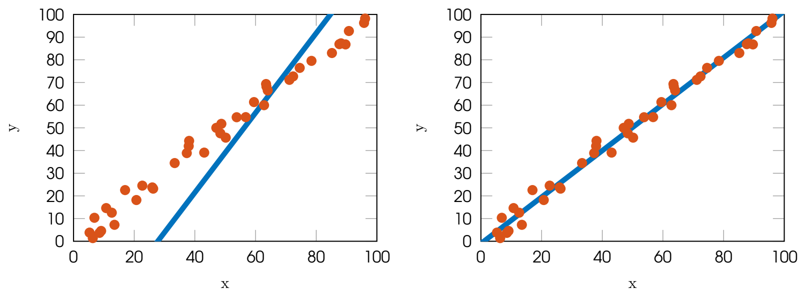
We can find a non-trivial solution that minimizes this using singular value decomposition (SVD) (Björck 1996, p. 185).

$$\arg\min_{\mathbf{l}} \|\mathbf{Xl}\|_2 \qquad s.t. \;\; \|\mathbf{l}\|_2 = 1$$

The problem with this approach is that the error minimized is not only the sum of squared distances but also the scale. This results in very bad fits when r is close to zero, as illustrated in Figure 2.9(a) on the facing page. In order to mitigate this we solve the geometrical least squares problem, where the Euclidean distance from the points to the line is minimized.

$$\arg\min_{l} \sum_{i=1}^{n} d_i^2$$

The first principal component of the points solves this problem (Groen 1996). The result of this is shown in Figure 2.9(b) on the next page.



(a) Line fitted with SVD as described above.

(b) Total least squares solution.

Figure 2.9: Example of two fitted lines when using SVD and total least squares.

2.11 Camera model

With the pinhole camera model, projection of the world points in homogeneous coordinates to an image plane is done by the 3×4 projection matrix

$$\begin{aligned}\mathbf{P} &= \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\\ &= \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\\ &= \begin{bmatrix} fR_{1,1} + p_x R_{3,1} & fR_{1,2} + p_x R_{3,2} & fR_{1,3} + p_x R3,3 & ft_1 + p_x t_3 \\ fR_{2,1} + p_y R_{3,1} & fR_{2,2} + p_y R_{3,2} & fR_{2,3} + p_y R3,3 & ft_2 + p_x t_3 \\ R_{3,1} & R_{3,2} & R_{3_3} & t_3 \end{bmatrix} \end{aligned} \tag{2.3}$$

Here $\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$ is rotation and translation to the camera's coordinate system and $\mathbf{K}$ is the projection to the image plane, where f is the focal length and (p_x, p_y) is the principal point. The rotation matrix can be defined by roll ϕ, pan θ and tilt ψ angles as

$$\begin{aligned}\mathbf{R} &= \mathbf{R}_z(\phi)\mathbf{R}_x(\theta)\mathbf{R}_y(\psi)\\ &= \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}\begin{bmatrix} \cos(\psi) & 0 & \sin(\psi) \\ 0 & 1 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) \end{bmatrix}\\ &= \begin{bmatrix} \cos(\phi)\cos(\theta) & -\sin(\phi)\cos(\psi) + \cos(\phi)\sin(\theta)\sin(\psi) & \sin(\phi)\sin(\psi) + \cos(\phi)\sin(\theta)\cos(\psi) \\ \sin\phi\cos(\theta) & \cos(\phi)\cos(\psi) + \sin(\phi)\sin(\theta)\sin(\psi) & -\cos(\phi)\sin(\psi) + \sin(\phi)\sin(\theta)\cos(\psi) \\ -\sin(\theta) & \cos(\theta)\sin(\psi) & \cos(\theta)\cos(\psi) \end{bmatrix}.\end{aligned}$$

CHAPTER 3

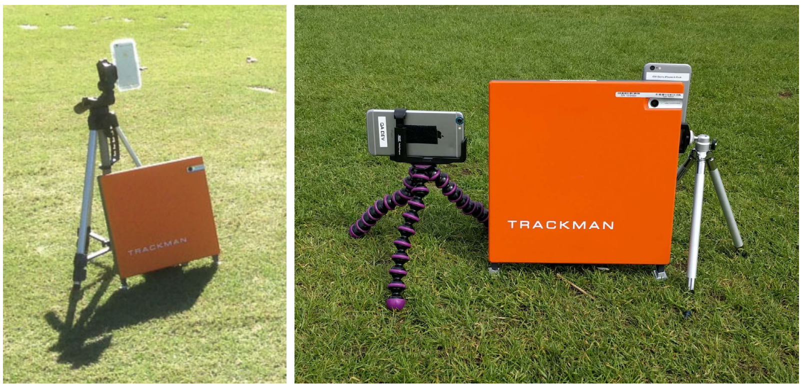
Data

3.1 Datasets

During our project we have worked with two datasets, which we will refer to as dataset 1 (DS1) and dataset 2 (DS2). During our experiments, we have recorded video with the back camera of iPhones, an overview is in Table 3.1 and Figure 3.2 shows the test setup for each dataset. We also have video available from the radar's internal camera, and we provide an overview of the technical specifications in Table 3.3 on page 21. In each dataset, we have a selection of different golf clubs. An overview is provided in Table 3.4 on page 21.

Table 3.1: iPhones used for each dataset.

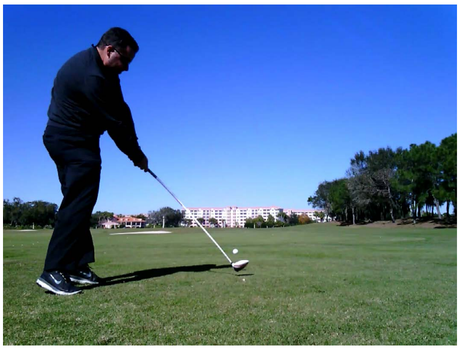
Dataset	Camera
DS1	iPhone 6s
Dataset 2 normal lens (DS2n)	iPhone 6 Plus
Dataset 2 zoom lens (DS2z)	iPhone 6 Plus with 2x zoom lens



(a) DS1

(b) DS2. On the left DS2z (pictured without zoom lens), and on the right DS2n.

Figure 3.2: Test setup for each dataset.


(a) Radar internal camera.

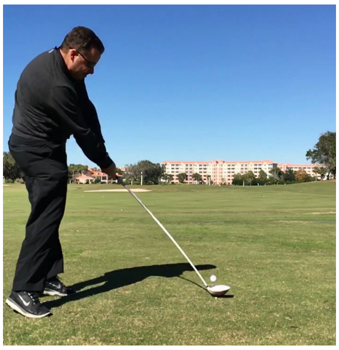
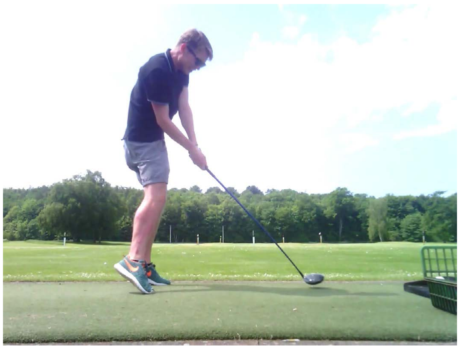
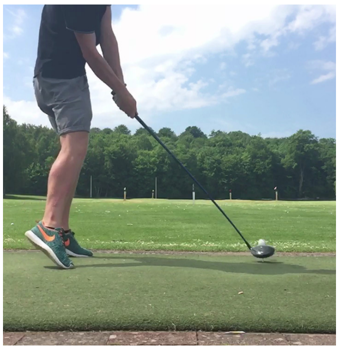

(b) DS1

Figure 3.5: Example of a frame from each camera in DS1.


(a) Radar internal camera.


(b) DS2n

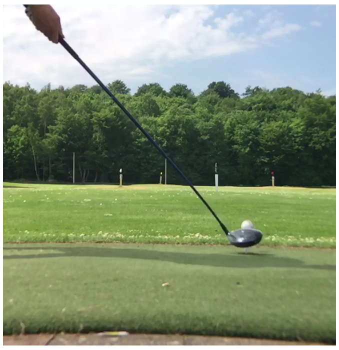

(c) DS2z

Figure 3.6: Example of a frame from each camera in DS2.

Table 3.3: Technical video information

Camera	FPS	Resolution	Bitrate	Codec
TrackMan 4	45	1280×960	6.6 Mbps	H.264
iPhone	240	720×720	24.5 Mbps	

Table 3.4: Clubs used and number of strokes.

Club	Number of strokes		Stroke range
	Total	With ground truth	
DS1			
Driver 1	17	17	1-17
Driver 2	9	9	18-26
Iron 1	16	14	27-42
Wedge 1	13	11	43-55
DS2			
Driver 1	30	29	16-30, 46-60
Driver 3	15	15	1-15
Iron 2	15	15	61-75
Wedge 2	15	15	31-45

3.2 Annotations

Ground truth impact point

In order to evaluate how well our method is performing it is necessary to compare with a ground truth. We obtain this by spraying the golf club head with foot deodorant prior to each stroke. This leaves a thin layer on the club face, which is removed on impact with the golf ball as seen in Figure 3.7 on the following page. This gives us a visual cue for the impact location.

In order to be able to quantify this, we create a coordinate system on the club face by measuring with a ruler where the lines on the club face are located. We have then annotated all the lines in each ground truth image and computed a homography between them and the measured lines. This is because the club face is not always parallel with the image plane of the camera taking the ground truth image.

Initial ball position

In each video, we have annotated the position and radius of the initial ball to be able to quantify the performance of this part of our method. Example of an annotation is in Figure 3.8(a) on the next page.

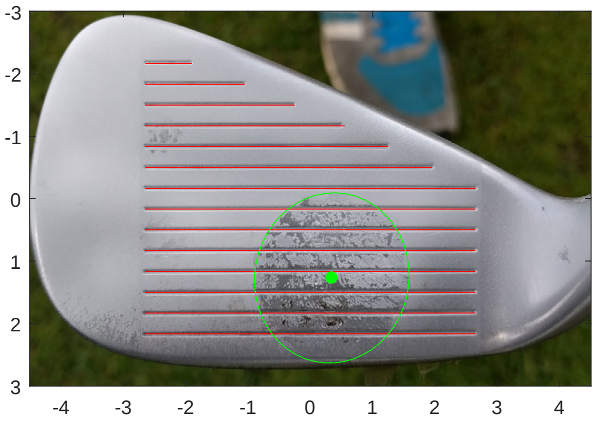


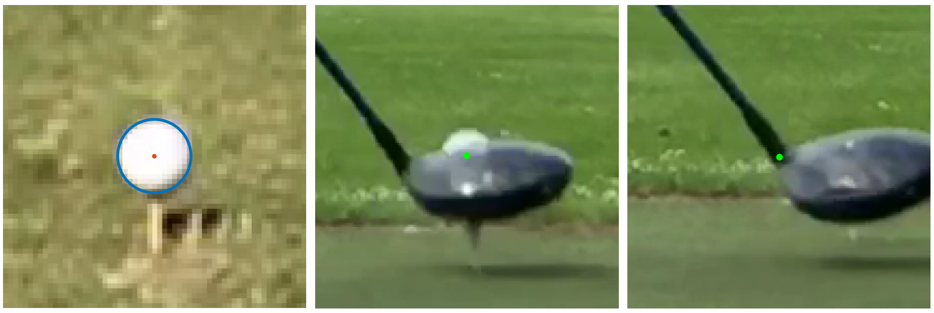
Figure 3.7: Ground truth example, where the image has been transformed using the homography. Red is standard lines, green is annotated impact ball. Axes are in cm.

Position of first visible post-impact ball

In our dataset, we have annotated the position of the ball in the first frame where it is visible after impact. An example can be seen in Figure 3.8(b).

Last anchor point before impact

We have annotated the anchor point in the last frame before impact. This can be seen in Figure 3.8(c).



(a) Initial annotated ball. The blue circle shows the perimeter and the red dot the center.

(b) Annotation of the ball first time it's visible after impact.

(c) Annotation of last anchor point before impact.

Figure 3.8: Examples of annotated elements.

CHAPTER 4

Methods

In this chapter, we describe our method for detecting the ball's impact location on the club face. In very short terms we estimate the trajectory of the club and ball and compare these at the time of impact. An outline of the method is presented in Table 4.1. Our method has been implemented in MATLAB (2016).

Table 4.1: Outline of our entire method.

Given a video of a golf stroke recorded at a high frame rate behind the golf player, we do the following.

1. Track the ball's flight path.
2. Determine the location of the initial ball position.
3. Use flight path and initial position to compute the impact time.
4. Determine the point on the shaft where the shaft ends and the head begins. This point is referred to as the anchor point.
   a) Detect the club shaft.
   b) Segment the club head.
   c) Determine the anchor point.
5. Fit a model to the anchor points detected in each frame of the swing.
6. Interpolate the anchor point at impact time.
7. Estimate a homography between the club face in the image plane and the club face coordinate system at impact. The club face coordinate system has origin at the impact point.
8. Use the homography to map the anchor point to the club face coordinate system.
9. Given the coordinate of the anchor point in the standard club face coordinate system with origin in the middle of the club, we can compute the impact point by subtracting the vectors from each other.

4.1 Preliminaries

Ball flight tracking

To track the ball's flight path in our videos we use a tracker developed by TrackMan A/S. In very short terms it uses differences in neighboring frames to detect changed objects. From this, a number of tracks which are consistent with a physical model of a flying object only affected by gravity are generated.

Selection of ball flight track

Because the ball tracker outputs multiple tracks, there is need for a method to select the one corresponding to the ball. Other tracks may include flying dirt, the tee and noise and so on. The radar data contains a function $\mathbf{f}_{ball} : \mathbb{R} \to \mathbb{R}^3$ that describes the flight path of the ball. This is not accurate enough to select the correct track by least squares, but it still contains useful information. We project the ball flight to our camera

$$\mathbf{p}_{ball}(t) = \mathbf{P} \begin{bmatrix} \mathbf{f}_{ball}(t) \\ 1 \end{bmatrix}$$

Using this projection we can compute the velocity of the ball in the y-direction of the video as a function of time

$$v_y(t) = p'_{ball,y}(t) \approx \text{FPS} \left(p_{ball,y}(t) - p_{ball,y}\left(t - \frac{1}{\text{FPS}}\right)\right).$$

Let ${p_{ball,y}}^{-1}(y)$ be the inverse function of $p_{ball,y}(t)$. If we plug this in to $v_y(t)$ we obtain a function which yields the pixel velocity in the y-direction as a function of the position in the y-direction

$$V_y(y) = v_y({p_{ball,y}}^{-1}(y))$$

The track corresponding to the ball flight can then be selected as the track that has the most similar pixel velocity for each y in the track. The result of this process is shown in Figure 4.2 on the next page.

Camera calibration

The goal of this section is to achieve a camera calibration w.r.t. the radar's coordinate system, so we can project radar coordinates to the camera. Additionally, we also use the internal camera parameters to correct for radial optical distortion and we apply our method only on undistorted images.

Dataset 1

We have calibrated the internal parameters using a similar iPhone and a checkerboard. The radar has tracked the ball flight in $\mathbb{R}^3$. Given a set of external parameters

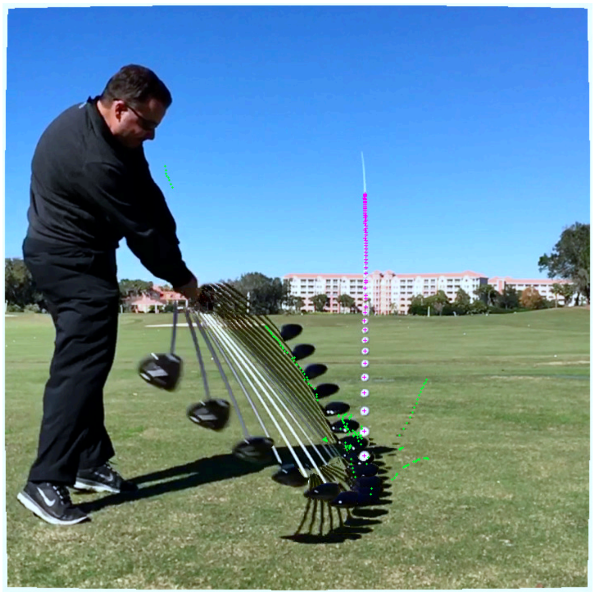


Figure 4.2: Example of all tracks generated by the ball tracker. The pink one is the track selected by our selection process, which corresponds to the ball track.

for the camera, we can project all ball flights to the camera and compare them with the tracked ball paths. We use numerical optimization to estimate the external parameters that minimize this reprojection error.

Dataset 2

We have calibrated the internal parameters using a checkerboard on location. The internal camera in the radar has a factory calibration w.r.t the radar's coordinate system. We use this fact so we only have to estimate the relative pose of the iPhones w.r.t. the internal camera. In order to estimate this, we have selected four frames with identically placed checkerboards in both iPhones and the internal camera in the radar. The relative pose can then be estimated by minimizing the reprojection error of the corner points of the checkerboards, as is commonly done in stereo.

Difference image

In order to analyze a video meaningfully, it is relevant to use information from the neighboring frames. We calculate an image that describes what parts of the image have changed and by how much. We will refer to this as a difference image.

The difference image is computed as follows: Let $\mathbf{V}(f)$ be the f'th frame from a video, as in Figure 4.3(a) on the facing page. We can compute the signed difference $\mathbf{S}$ between two frames, f_1, f_2:

$$\mathbf{S}(f_1, f_2) = \mathbf{V}(f_1) - \mathbf{V}(f_2). \tag{4.1}$$

Now we can compute the difference image D for a specific step size s as

$$\mathbf{D}(f, s) = \sqrt{\max(0, \mathbf{S}(f, f - s) \circ \mathbf{S}(f, f + s))}, \tag{4.2}$$

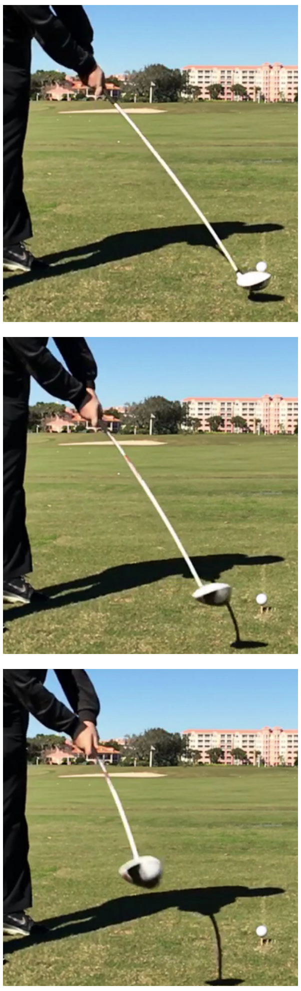
where $\circ$ denotes the Hadamard product. The reasoning behind Equation (4.2) is as follows. If the motion is fast enough and the step size is not too small, both $\mathbf{V}(f - s)$ and $\mathbf{V}(f + s)$ will contain background pixels, where the motion is in $\mathbf{V}(f)$. In the signed difference images Figure 4.3(b) on the facing page there is both positive and negative values which make them hard to interpret directly. The moving object is either brighter or darker than the background, but the difference will have the same sign for both forward and backward differences if it is a pixel that has changed in frame f. This implies that the product of differences will always yield a positive value where the pixel has changed in f with respect to both images. Because of this, the negative values are removed with a max. To compensate for the values being almost squared by the product, a square root is taken.

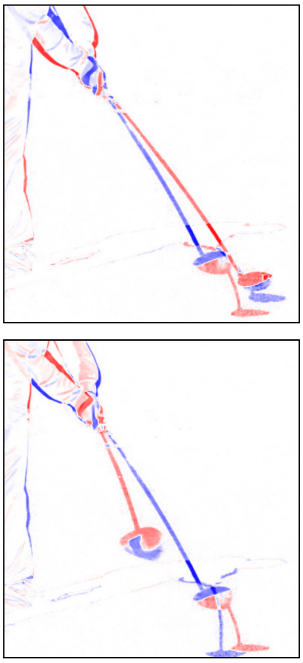
Background image

In frames, during the swing, we will need to know a background image, i.e. an image with the moving parts removed. For frame f in the video, we calculate this as

$$\mathbf{B}(f) = \underset{s \in \{-4, -2, 0, 2, 4\}}{\operatorname{median}} \mathbf{V}(f + s),$$

where $\mathbf{V}(f)$ is the f'th frame in the video. An example of a background image is shown in Figure 4.4 on page 28.


(a) Video frames: $\mathbf{V}(f-2)$, $\mathbf{V}(f)$, $\mathbf{V}(f+2)$


(b) Frame differences: $\mathbf{S}(f, f-2)$, $\mathbf{S}(f, f+2)$. Blue is positive values, red is negative.

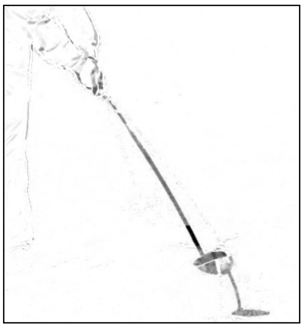
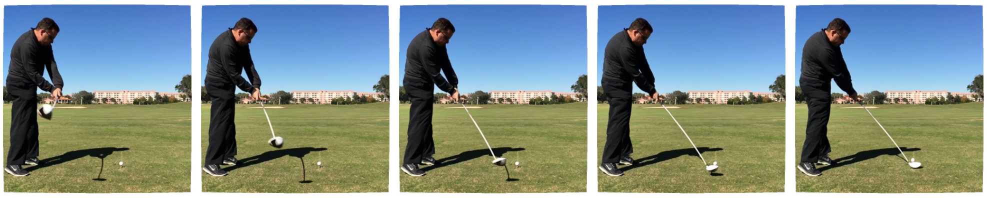

(c) Difference image: $\mathbf{D}(f, 2)$

Figure 4.3: Calculation of a difference image with step size 2. All images shown are a cropped version of the full frame for clarity.


(a) $\mathbf{V}(f-4)$, $\mathbf{V}(f-2)$, $\mathbf{V}(f)$, $\mathbf{V}(f+2)$, $\mathbf{V}(f+4)$.

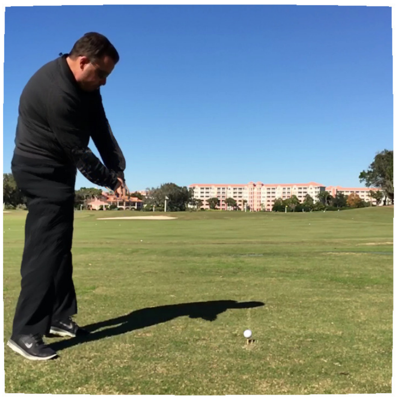

(b) Background image $\mathbf{B}(f)$.

Figure 4.4: Example of a computed background image $\mathbf{B}(f)$ and the video frames that have been used to compute it.

4.2 Initial ball location

In this section we describe how we locate the initial ball position, i.e. where the ball lies still before it is hit.

Region of interest (ROI)

The radar gives a position $(x_0, y_0, z_0) \in \mathbb{R}^3$, which can be projected to the camera in homogeneous coordinates via

$$\mathbf{p}_0 = s \begin{bmatrix} p_{0,x} \\ p_{0,y} \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix},$$

where $\mathbf{P}$ is the projection matrix from the radar coordinate system to the image plane and $(p_{0,x}, p_{0,y}) \in \mathbb{R}^2$ are the pixel coordinates. These coordinates will contain some error due to the fact that the radar position is extrapolated back to the time of impact from the measured radar data and errors in the camera calibration. Therefore we determine a ROI around $(p_{0,x}, p_{0,y})$, where we try to locate the ball. We can calculate an approximate radius of the ball in pixels with the formula

$$r_{px,approx} = \frac{r_{mm} \cdot f}{z_{0_{mm}}}, \tag{4.3}$$

where $r_{px,approx}$ is the ball radius in pixels, r_{mm} is the ball radius in mm, f is the focal length of the camera, and $z_{0_{mm}}$ is the depth coordinate of the ball in the radar coordinate system. Since the camera is placed at nearly the same depth as the radar this is a decent approximation. The ROI is then chosen as

$$\text{ROI} = [p_{0,x} - 6r_{px,approx}, p_{0,x} + 6r_{px,approx}] \times [p_{0,y} - 6r_{px,approx}, p_{0,y} + 6r_{px,approx}],$$

i.e. a square around $(p_{0,x}, p_{0,y})$ with side length $12r_{px,approx}$. An example of a chosen ROI can be seen in Figure 4.5 on the following page.

Difference image

We locate the ball by looking for differences in the frames before and after the ball has been hit. The radar gives a rough estimate $t_{0_{radar}}$ of this impact time. Relative to this impact time we choose 8 frames before, and 8 frames after. It is important that the first 8 frames are chosen when the golf ball is visible. We extract frames at the following times relative to $t_{0_{radar}}$

$$t_{pre} = \{\ -.5 \text{ s},\ -.4375 \text{ s},\ -.375 \text{ s},\ -.3125 \text{ s},\ -.25 \text{ s},\ -.1875 \text{ s},\ -.125 \text{ s},\ -.0625 \text{ s}\},$$
$$t_{post} = \{.25 \text{ s},\ .375 \text{ s},\ .5 \text{ s},\ .625 \text{ s},\ .75 \text{ s},\ .8125 \text{ s},\ .875 \text{ s},\ .9375 \text{ s}\}.$$

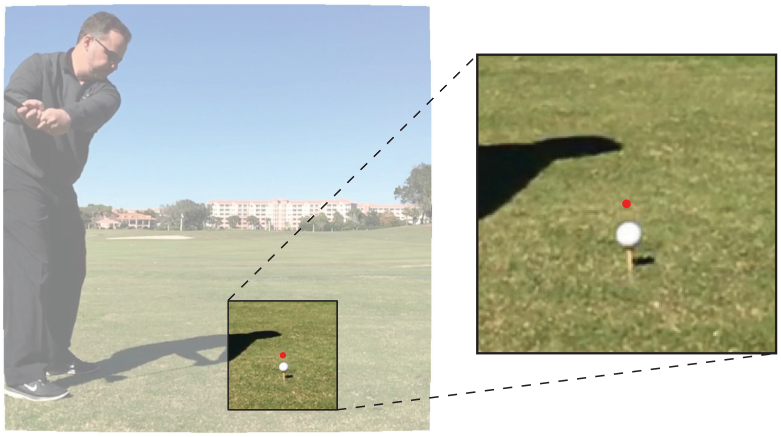


Figure 4.5: Example of the ROI we use to find the ball. The red dot is $(p_{0,x}, p_{0,y})$ and the enlarged square is the ROI.

These times have been chosen empirically. At 240 FPS, which our videos are recorded at, the impact radar frame is $f_{0_{radar}} = \lfloor t_{0_{radar}} \cdot 240 \rfloor$, and the frames used relative to this are

$$\begin{aligned} f_{pre} &= \{240\tfrac{1}{\mathrm{s}} \cdot t\}_{t \in t_{pre}} \\ &= \{-120, -105, -90, -75, -60, -45, -30, -15\}, \\ f_{post} &= \{240\tfrac{1}{\mathrm{s}} \cdot t\}_{t \in t_{post}} \\ &= \{60, 90, 120, 150, 195, 210, 225\}. \end{aligned}$$

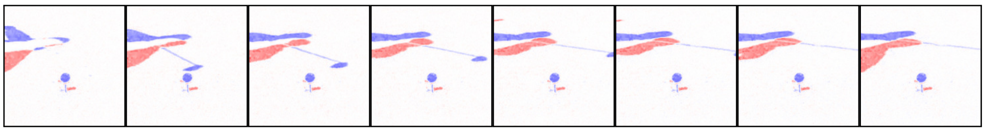
Using these sixteen frames we calculate a difference image. This is done slightly differently than described in Section 4.1 on page 25. We start by computing eight signed difference images using Equation (4.1) on page 26

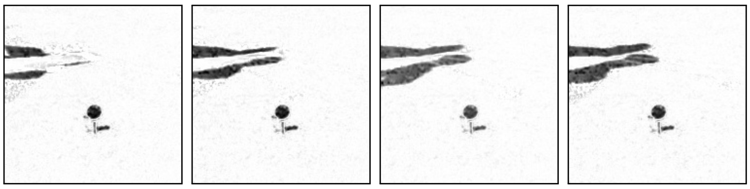
$$\mathbf{S}_i = \mathbf{S}(f_{0_{radar}} + f_{pre,i}, f_{0_{radar}} + f_{post,i}), \quad i \in \{1, 2, \dots, 8\},$$

where $f_{pre,i}$ is the i'th element of f_{pre} and similarly for $f_{post,i}$. Then we construct our difference image D as

$$\mathbf{D} = \sqrt{\min_{i \in \{1,2,3,4\}} \max\left(0, \mathbf{S}_i \circ \mathbf{S}_{i+4}\right)}.$$

Note that this is inherently different from other difference images that we use in that this finds differences in the frames before and after a given time, where other difference images describe which parts are moving at a specific time. The idea behind this formula is similar to the reasoning in Section 4.1 on page 25, with the difference that we do not have a central image that all the signed distances are computed with respect to. The ball will have high absolute values in all of them with the same sign, and high values in other pixels, primarily shadows, may only occur in one or few of them. The difference image and its constituents can be seen in Figure 4.6


(a) The eight signed difference images $\{\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_8\}$.


(b) The four $\left\{\sqrt{\max\left(0, \mathbf{S}_i \circ \mathbf{S}_{i+4}\right)}\right\}_{i \in \{1,2,3,4\}}$ images.

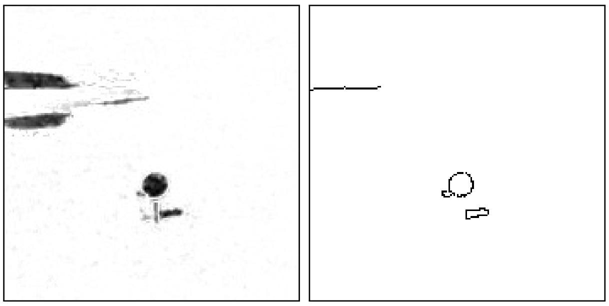

(c) From left to right: **D** and **E**

Figure 4.6: Illustration of images used to compute **E**.

Detecting ball as a circle

We use the difference image **D** to detect the ball as a circle. We do this in two different ways; using the Hough transform and using RANSAC. For both of these, we need to extract pixels that lie on edges in the image. We use the Canny edge detector to obtain the binary image **E** of edges.

Hough transform

We apply the Hough transform on **D**. The implementation we use computes the circles a little different from the standard Hough transform. First, it detects possible circle centers by using the gradient directions in **D**. Then it merges close centers together. For each of these a radius is computed (Peng 2005). To extract the circle that corresponds to the ball we weigh the circles detected by the sum of pixels from **D** that are inside the circles. After this weighing, the most dominant circle is chosen to as the ball.

RANSAC

We can also detect circles in **D** using RANSAC. The model used is

$$M(x_c, y_c, r): \ (x - x_c)^2 + (y - y_c)^2 = r^2,$$

where $(x_c, y_c) \in \mathbb{R}^2$ is the center of the circle and r is the radius. Thus, in each RANSAC iteration three points are needed. We solve the system of equations for the three points using a method based on computing determinants (Anton and Rorres 2010, Chapter 10.1).

In this application, we are only interested in circles of a relatively small radius. This gives rise to an efficient way of picking the three points in each RANSAC iteration. Specifically, because we have an approximation of the radius $r_{px,approx}$ from Equation (4.3) on page 29 we are only interested in circles with radius less than $r_{max} = 1.2 \cdot r_{px,approx}$. If any of the distances between the three points is bigger than $2 \cdot r_{max}$, we can know for sure that the circumcircle will have a radius greater than r_{max}. This is computationally efficient to check before we fit the circle, and avoids fitting circles we are certain will be too large.

Let $\mathrm{d}(x_1, x_2)$ be the Euclidean distance between x_1 and x_2, and p be points representing all edge pixels in **E**. Then we pick the three points in each RANSAC iteration as follows:

1. Pick a point p_1 at random.
2. Compute the distances $\mathrm{d}(p_1, p)$ to all other points p.
3. Choose two random points p_2 and p_3 for which $\mathrm{d}(p_1, p_2) < 2 \cdot r_{max}$ and $\mathrm{d}(p_1, p_3) < 2 \cdot r_{max}$
4. Repeat 3. until $\mathrm{d}(p_2, p_3) < 2 \cdot r_{max}$.

This procedure ensures that we fit fewer circles with a radius larger than r_{max}, compared to naively sampling three points. In order to count inliers in a manner that does not favor larger circles, we normalize the number of inliers in any iteration by dividing it with the radius r.

4.3 Computing impact time

Impact time with linear flight assumption

Let (x, y, z) be a point in $\mathbb{R}^3$. In homogeneous coordinates, we can map this into pixel-coordinates (p_x, p_y) with the 3×4 projection matrix $\mathbf{P}$

$$s \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{4.4}$$

where s is a scaling factor. For a short time after impact, we may assume that a golf ball travels in a straight line in $\mathbb{R}^3$. Hence, its position as a function of time t can be described by

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + \begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} \cdot t = \begin{bmatrix} x_0 & x_v \\ y_0 & y_v \\ z_0 & z_v \end{bmatrix} \begin{bmatrix} 1 \\ t \end{bmatrix},$$

where (x_0, y_0, z_0) is the starting point and (x_v, y_v, z_v) is a velocity vector. We can write this in homogeneous coordinates as

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ 1 \end{bmatrix} = \begin{bmatrix} x_0 + x_v \cdot t \\ y_0 + y_v \cdot t \\ z_0 + z_v \cdot t \\ 1 \end{bmatrix} = \begin{bmatrix} x_0 & x_v \\ y_0 & y_v \\ z_0 & z_v \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ t \end{bmatrix} \tag{4.5}$$

If we can combine Equations (4.4) and (4.5) we get the pixel-coordinates $p_x(t)$ and $p_y(t)$ of the ball as a function of time t

$$\begin{aligned} s \begin{bmatrix} p_x(t) \\ p_y(t) \\ 1 \end{bmatrix} &= \mathbf{P} \begin{bmatrix} x_0 & x_v \\ y_0 & y_v \\ z_0 & z_v \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ t \end{bmatrix} \\ &= \begin{bmatrix} Q_{1,1} & Q_{1,2} \\ Q_{2,1} & Q_{2,2} \\ Q_{3,1} & Q_{3,2} \end{bmatrix} \begin{bmatrix} 1 \\ t \end{bmatrix} \end{aligned}$$

Under the assumption that the ball moves on a straight line in $\mathbb{R}^3$ it will also move on a straight line in the image plane (Hartley and Zisserman 2004, pp. 196-197). We change the coordinate system such that the x-axis corresponds to this line with the origin at the initial ball position. This is illustrated in Figure 4.7 on page 35. We denote the new axes by p'_x and p'_y. The change of coordinate system is done through a rotation and translation. If $p_{x0} = p_x(0)$ and $p_{y0} = p_y(0)$ i.e. the initial ball position and $-\theta$ is the angle that we need to rotate the coordinate system with we find that

$$\begin{aligned}
s\begin{bmatrix} p'_x(t) \\ p'_y(t) \\ 1 \end{bmatrix} &= \mathbf{R}_z(-\theta)\left(\begin{bmatrix} s\,(p_x(t)-p_{x0}) \\ s\,(p_y(t)-p_{y0}) \\ s \end{bmatrix}\right) \\
&= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\left(\begin{bmatrix} sp_x(t) \\ sp_y(t) \\ s \end{bmatrix} - \begin{bmatrix} sp_{x0} \\ sp_{y0} \\ 0 \end{bmatrix}\right) \\
&= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} Q_{1,1} & Q_{1,2} \\ Q_{2,1} & Q_{2,2} \\ Q_{3,1} & Q_{3,2} \end{bmatrix}\begin{bmatrix} 1 \\ t \end{bmatrix} - s\underbrace{\begin{bmatrix} \cos\theta\cdot p_{x0} - \sin\theta\cdot p_{y0} \\ \sin\cdot p_{x0} + \cos\theta\cdot p_{y0} \\ 0 \end{bmatrix}}_{\mathbf{b}} \\
&= \underbrace{\begin{bmatrix} \cos\theta\cdot Q_{1,1} - \sin\theta\cdot Q_{2,1} & \cos\theta\cdot Q_{1,2} - \sin\theta\cdot Q_{2,2} \\ \sin\theta\cdot Q_{1,1} + \cos\theta\cdot Q_{2,1} & \sin\theta\cdot Q_{1,2} + \cos\theta\cdot Q_{2,2} \\ Q_{3,1} & Q_{3,2} \end{bmatrix}}_{\widetilde{\mathbf{Q}}}\begin{bmatrix} 1 \\ t \end{bmatrix} - s\begin{bmatrix} b_1 \\ b_2 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \widetilde{Q}_{1,1} & \widetilde{Q}_{1,2} \\ \widetilde{Q}_{2,1} & \widetilde{Q}_{2,2} \\ \widetilde{Q}_{3,1} & \widetilde{Q}_{3,2} \end{bmatrix}\begin{bmatrix} 1 \\ t \end{bmatrix} - s\begin{bmatrix} b_1 \\ b_2 \\ 0 \end{bmatrix}
\end{aligned}$$

We note that $p'_y(t) = 0$ for all $t \in \mathbb{R}$, and only use that

$$\begin{bmatrix} sp'_x(t) \\ s \end{bmatrix} = \begin{bmatrix} \widetilde{Q}_{1,1} & \widetilde{Q}_{1,2} \\ \widetilde{Q}_{3,1} & \widetilde{Q}_{3,2} \end{bmatrix}\begin{bmatrix} 1 \\ t \end{bmatrix} - s\begin{bmatrix} b_1 \\ 0 \end{bmatrix} \tag{4.6}$$

We denote the distance in pixels at time t to the initial position of the ball by $d(t)$. Since the points lie on the p'_x axis, we see that $d(t) = p'_x(t)$. We thus have that

$$\begin{aligned}
d(t) = p'_x(t) &= \frac{\widetilde{Q}_{1,1} + \widetilde{Q}_{1,2}\cdot t - \left(\widetilde{Q}_{3,1} + \widetilde{Q}_{3,2}\cdot t\right)b_1}{\widetilde{Q}_{3,1} + \widetilde{Q}_{3,2}\cdot t} \\
&= \frac{\left(\widetilde{Q}_{1,1} - \widetilde{Q}_{3,1}b_1\right) + \left(\widetilde{Q}_{1,2} - \widetilde{Q}_{3,2}b_1\right)t}{\widetilde{Q}_{3,1} + \widetilde{Q}_{3,2}\cdot t} \\
&= \frac{c_1 + c_2\cdot t}{c_3 + c_4\cdot t}
\end{aligned}$$

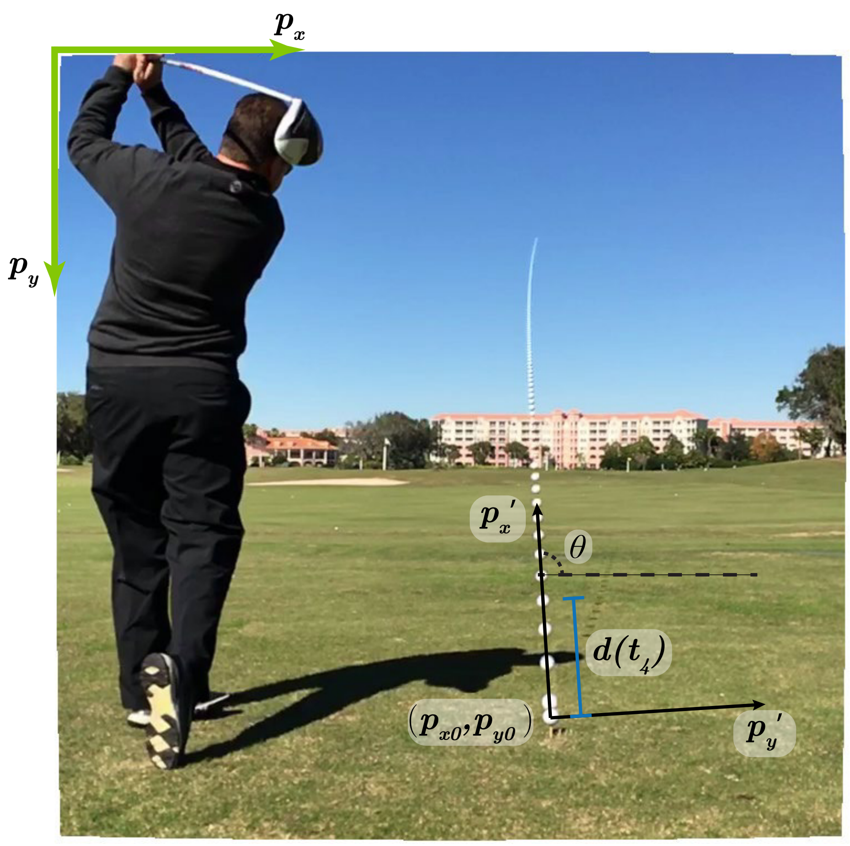


Figure 4.7: Illustration of the coordinate change. The (p_x, p_y)-axes are the original image coordinate system and the (p_x', p_y')-axes are our new coordinate system with origin in the initial ball position (p_{x0}, p_{y0}). θ is the angle that the original coordinate system is rotated with. This also illustrates that $p_x'(t) = d(t)$.

Now let $\mathbf{c} = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 \end{bmatrix}^T$ then

$$\begin{aligned} d(t) &= \frac{c_1 + c_2 \cdot t}{c_3 + c_4 \cdot t} \Rightarrow \\ d(t) \cdot (c_3 + c_4 \cdot t) &= c_1 + c_2 \cdot t \Rightarrow \\ d(t) \cdot (c_3 + c_4 \cdot t) - (c_1 + c_2 \cdot t) &= 0 \Rightarrow \\ \begin{bmatrix} -1 & -t & d(t) & d(t) \cdot t \end{bmatrix} \mathbf{c} &= 0 \end{aligned}$$

If we track the moving golf ball through n video-frames after impact with known times $t_1, t_2, \ldots, t_n$, and measure the distances $d(t_1), d(t_2), \ldots, d(t_n)$ in pixels to the initial position we can combine all these into a matrix

$$\mathbf{B} = \begin{bmatrix} -1 & -t_1 & d(t_1) & d(t_1)\cdot t_1 \\ -1 & -t_2 & d(t_2) & d(t_2)\cdot t_2 \\ & & \vdots & \\ -1 & -t_n & d(t_n) & d(t_n)\cdot t_n \end{bmatrix},$$

and then solve the linear system

$$\mathbf{Bc} = \mathbf{0}, \tag{4.7}$$

to find $\mathbf{c}$. We need the constraint $\|\mathbf{c}\| \neq 0$, since the zero-solution is trivial and not useful in any way. We ensure this by requiring that $\|\mathbf{c}\| = 1$. Since measurements will include noise Equation (4.7) will not hold perfectly so we instead solve the minimization problem

$$\min_{\mathbf{c}} \|\mathbf{Bc}\|_2 \quad \text{s.t. } \|\mathbf{c}\|_2 = 1.$$

This is a least squares problem which we can solve using SVD (Björck 1996, p. 185).

When $\mathbf{c}$ is found, we can compute the time of impact. At impact time t_0 we know that $d(t_0) = 0$ hence

$$\frac{c_1 + c_2 \cdot t_0}{c_3 + c_4 \cdot t_0} = 0 \Rightarrow t_0 = \frac{-c_1}{c_2}.$$

Non-linear minimization

The error that we minimize in the previous section is not statistically meaningful. It is of interest to model the error only on the observations we expect to contain errors i.e.

$$d(t) = \frac{c_1 + c_2 \cdot t}{c_3 + c_4 \cdot t} + \epsilon_t,$$

where ϵ_t is the noise. Moving the terms around we get

$$\begin{bmatrix} -1 & -t & d(t) & d(t)\cdot t \end{bmatrix} \mathbf{c} = (c_3 + c_4 t)\epsilon_t,$$

thus the minimization becomes

$$\begin{aligned} \min_{\mathbf{c}} \|\mathbf{Bc}\|_2^2 &= \min_{\mathbf{c}} \sum_{i=1}^{n} \left(\begin{bmatrix} -1 & -t_i & d(t_i) & d(t_i)\cdot t_i \end{bmatrix} \mathbf{c} \right)^2 \\ &= \min_{\mathbf{c}} \sum_{i=1}^{n} ((c_3 + c_4 t_i)\epsilon_{t_i})^2. \end{aligned}$$

We see that points are not weighted equally but with the value of the function in the denominator. A more meaningful minimization would be

$$\min_{\mathbf{c}} \sum_{i=1}^{n} \epsilon_{t_i}^2 = \min_{\mathbf{c}} \sum_{i=1}^{n} \left(d(t) - \frac{c_1 + c_2 t}{c_3 + c_4 t} \right)^2 ,$$

We solve this with a nonlinear minimization algorithm and use the linear estimate as a starting guess.

Impact time with gravity included

If we want to incorporate gravity in our model and drop the assumption that the flight path is linear we need to add a quadratic term to Equation (4.5) on page 33 to get

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ 1 \end{bmatrix} = \begin{bmatrix} x_0 + x_v \cdot t \\ y_0 + y_v \cdot t + y_a \cdot t^2 \\ z_0 + z_v \cdot t \\ 1 \end{bmatrix} = \begin{bmatrix} x_0 & x_v & 0 \\ y_0 & y_v & y_a \\ z_0 & z_v & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix}$$

If we assume that the camera is level with vertical and horizontal, i.e. that $\theta_{roll} = \theta_{tilt} = 0$ we see from Equation (2.3) on page 17 that $P_{1,2} = P_{3,2} = 0$ in the projection matrix $\mathbf{P}$ hence we get the pixel coordinates.

$$\begin{aligned} s \begin{bmatrix} p_x(t) \\ p_y(t) \\ 1 \end{bmatrix} &= \begin{bmatrix} P_{1,1} & 0 & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & 0 & P_{3,3} & P_{3,4} \end{bmatrix} \begin{bmatrix} x_0 & x_v & 0 \\ y_0 & y_v & y_a \\ z_0 & z_v & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix} \\ &= \begin{bmatrix} Q_{1,1} & Q_{1,2} & 0 \\ Q_{2,1} & Q_{2,2} & Q_{2,3} \\ Q_{3,1} & Q_{3,2} & 0 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix} . \end{aligned}$$

Note that the assumption $\theta_{roll} = \theta_{tilt} = 0$ will not hold exactly as it is difficult to set up the camera at specific angles. If the assumption is dropped $p_x(t)$ and s will also contain quadratic terms. With the assumption we obtain obtain the following models

$$p_x(t) = \frac{Q_{1,1} + Q_{1,2} \cdot t}{Q_{3,1} + Q_{3,2} \cdot t}, \tag{4.8}$$

$$p_y(t) = \frac{Q_{2,1} + Q_{2,2} \cdot t + Q_{2,3} \cdot t^2}{Q_{3,1} + Q_{3,2} \cdot t}. \tag{4.9}$$

If we let $\mathbf{q} = \begin{bmatrix} Q_{1,1} & Q_{1,2} & Q_{2,1} & Q_{2,2} & Q_{2,3} & Q_{3,1} & Q_{3,2} \end{bmatrix}^T$ then Equations (4.8) and (4.9) at time t_i can be rewritten as

$$\underbrace{\begin{bmatrix} -1 & -t_i & 0 & 0 & 0 & p_x(t_i) & t_i \cdot p_x(t_i) \\ 0 & 0 & -1 & -t_i & -t_i^2 & p_y(t_i) & t_i \cdot p_y(t_i) \end{bmatrix}}_{\mathbf{B}_i} \mathbf{q} = 0$$

If we have tracked the ball at times $t_1, t_2, \ldots, t_n$ with corresponding pixel coordinates $(p_x(t_1), p_y(t_1)), (p_x(t_2), p_y(t_2)), \ldots, (p_x(t_n), p_y(t_n))$ we can stack the $\mathbf{B}_i$s into the matrix

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \vdots \\ \mathbf{B}_n \end{bmatrix}$$

and then solve $\mathbf{Bq} = \mathbf{0}$. Again due to noise we instead solve the least squares minimization problem

$$\min_{\mathbf{q}} \|\mathbf{Bq}\|_2 \text{ s.t. } \|\mathbf{q}\|_2 = 1,$$

using SVD. Like in the linear case the error that is minimized is not meaningful and we can refine our estimate with non-linear minimization.

When $p_x(t)$ and $p_y(t)$ are determined we compute the impact time t_0 as

$$t_0 = \arg\min_t \left((p_{x0} - p_x(t))^2 + (p_{y0} - p_y(t))^2 \right),$$

where (p_{x0}, p_{y0}) is the position of the initial ball. Note that this has an analytic solution but we solve it numerically.

4.4 Anchor point detection

This section describes how we detect the anchor point. An outline is in Table 4.8 on the facing page.

Difference image

In order to compute a difference image suitable for club detection, it is desirable to have as few artifacts as possible. In Figure 4.3 on page 27, small regions of the club head are not visible in the difference image because a shadow from the previous image has the same color as the club. To compensate for this we use many step sizes.

$$steps = \{1, 2, 3, 5, 15\}.$$

Each of these will have their own artifacts, but if the median of all of these images is taken, the resulting image will have very few artifacts as shown in Figure 4.9 on page 40.

$$\mathbf{D}_{med}(f) = \underset{s \in steps}{\text{median}} (\mathbf{D}(f, s)) = \sqrt{\underset{s \in steps}{\text{median}}(\max(0, \mathbf{S}(f, f-s) \circ \mathbf{S}(f, f+s)))}$$

$\mathbf{D}$ refers to Equation (4.2) on page 26. Because we take the median of an odd number of elements, the result will be a single element from the set. This implies that we can take the square root outside the median. The median difference image $\mathbf{D}_{med}(f)$ will from now on be referred to simply as a difference image.

Table 4.8: Outline of the anchor point detection method.

1. Compute a difference image for the current frame.
2. Detect most dominant straight line in edges of difference image.
3. Find path with maximal sum of pixels along straight line.
4. Greedily search along straight line for path with high pixel sum.
5. Robustly fit quadratic to the two found paths.
6. Locate approximate center of club head.
7. Segment the club head.
8. Determine the anchor point by tracing the perimeter of the segmentation.

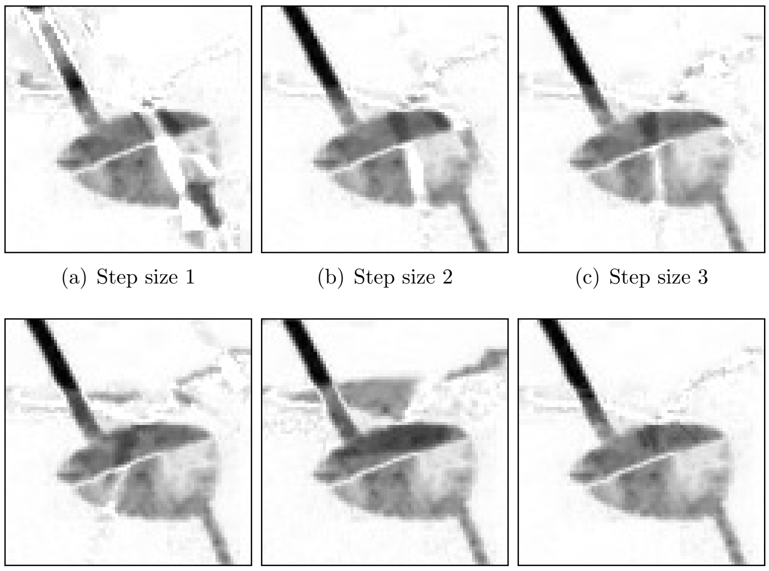
Initial club shaft detection

Using the difference image introduced in Section 4.4 on page 38, we can now determine an approximate location of the shaft. Edge detection is performed on the difference image using the Canny edge detector (Canny 1986). Afterwards, we fit a straight line to the edge pixels with RANSAC, using the straight line model

$$M(\theta, r): \; x\cos(\theta) + y\sin(\theta) = r.$$

Club shaft refinement

Using the straight line detected with RANSAC, we can sample a new image in a small region around this line, rotated such that the straight line is parallel with the x-axis, see Figure 4.11 on page 42. Figure 4.11(b) on page 42 shows the edge pixels that are inliers w.r.t. the straight line projected onto it. Using dynamic programming we compute the path from one side to the other of the rotated difference image that has the biggest sum. We perform the dynamic programming on a smoothed image to make the found solution smooth and centered on the club shaft. The dynamic programming solution generally does very well at finding the shaft. In some cases, however, it diverges from the shaft near the club head if e.g. a cast shadow has a higher difference value. This causes the dynamic programming to not trace the shaft all the way because the path with maximal sum will trace part of the shadow as Figure 4.11(d) on page 42 is an example of. Therefore, we incorporate a greedy search that is not able to make sudden jumps but will stay on a local "ridge" in the difference image, such as the shaft. Using the x-coordinate of the median of projected


(a) Step size 1 (b) Step size 2 (c) Step size 3

(d) Step size 5 (e) Step size 15 (f) Median of all step sizes

Figure 4.9: A close-up of the club head in $\mathbf{D}(f,s)$ for different step sizes s, and the median of images for all step sizes.

inliers, and sampling the corresponding y-value of the dynamic programming at this x, we have a point that most likely is located in the middle of the shaft. From this point, we initiate a greedy search towards both sides of the image. In this context greedy search means to choose the neighboring pixel with the highest value when we only are allowed to move either one pixel to the side or diagonally up and down. The dynamic programming solution together with the greedy search usually covers the entire club shaft with points.

It is now of interest to locate which of the points found by the dynamic programming and the greedy search that are located on the shaft. The shaft curves slightly but it can be described well by a second-degree polynomial, which we fit to all the points using RANSAC. In order to count inliers with RANSAC, we use the geometric distance to the polynomial. For each x-coordinate, we discard the largest point-to-line distance, which leaves us with either the distance to the point from the dynamic programming solution or the greedy search. Then we count the number of inliers as the length of the longest section where every point has at least 20 points within a specified threshold in the 30 nearest points. The result of the fitting can be seen in Figures 4.11(f) and 4.12 on page 42.

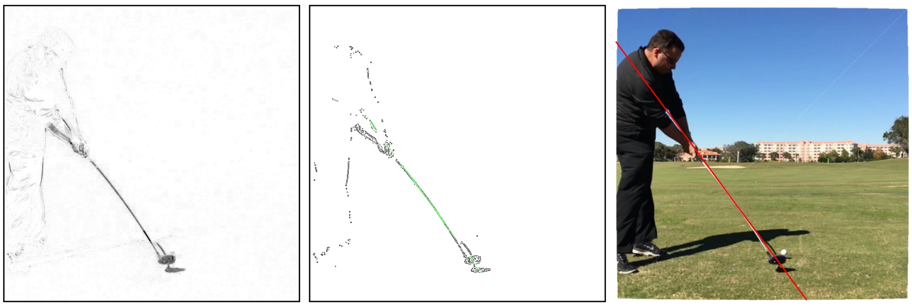

(a) Difference image (b) Edge pixels. Green are inliers (c) Detected line

Figure 4.10: Club shaft detection with RANSAC for straight lines.

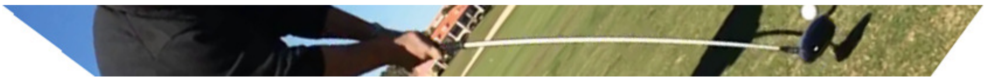
Club head center localization

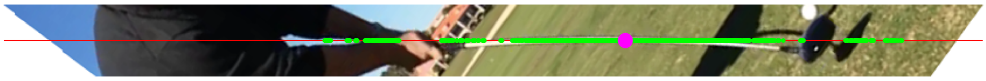
Looking at where the inliers of the second-degree polynomial end, we can determine an approximate location of the shaft end. A ROI of the rotated image around this point is extracted as shown in Figure 4.13(a) on page 43. We threshold the ROI image with a fixed threshold, and identify the biggest connected component. We take sums along columns of this component. The column containing the center of the club head is chosen to be where the smoothed sum achieves its maximum value. An illustration of this is in Figure 4.13(b) on page 43.

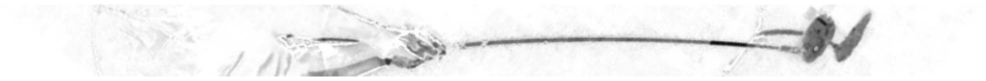
Club head segmentation

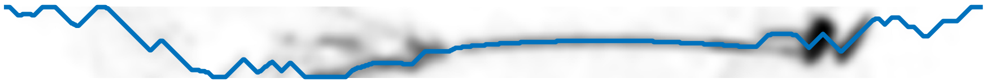
We segment the club head from the background using MRFs with a 4-neighborhood structure. To model these we utilize both knowledge of which parts are moving, from the difference image, and the color values. First, we use the determined club head x-coordinate to sample some points on the fitted polynomial near the club head and fit a straight line to these. The angle of this line with vertical is denoted by θ_{shaft}. Then we extract a ROI around the club head, rotated such that the y-axis is parallel with the fitted straight line. The line and ROI are shown as the yellow line and red box in Figure 4.16 on page 46. Since we know the fitted line lies on the club shaft we force it to belong to the foreground by setting the probability of this to a high value, and the probability of it being background to a very low value.

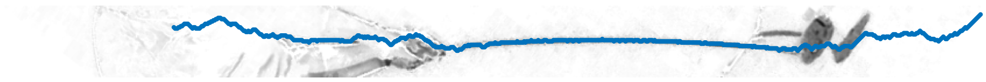
We threshold the difference image in the ROI with a fixed threshold. This yields an initial guess of a foreground and background segmentation, see Figure 4.15(d) on page 45.


(a) Rotated color image. Not used for computation, only illustration.


(b) Green points are inliers projected to the straight line. Magenta is median of projected inliers.


(c) Rotated difference image.


(d) Dynamic programming.


(e) Greedy search.

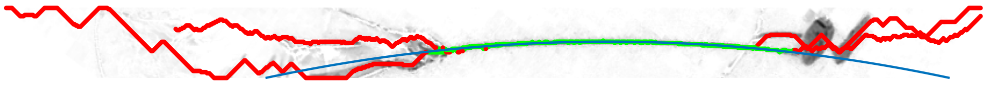

(f) Fitted second degree polynomial. Red is outliers, green is inliers.

Figure 4.11: Illustration of the club shaft refinement procedure.

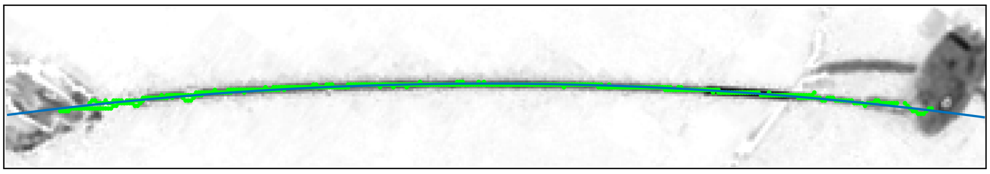
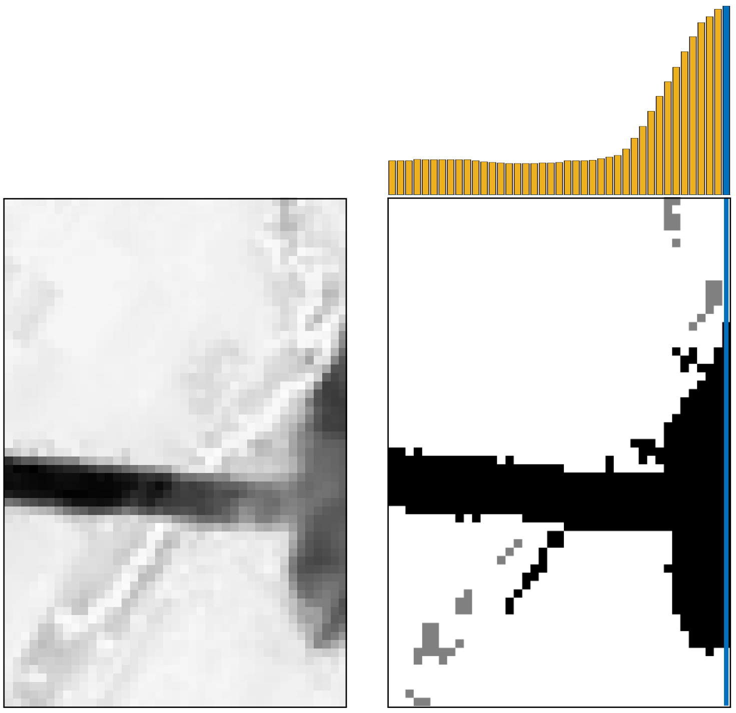


Figure 4.12: Close up of the fitted second degree polynomial.

Empirical probability density functions (PDFs) for difference images

We estimate empirical probability density function (PDF) $p_{d,f}(x_d)$ and $p_{d,b}(x_d)$ for the foreground and background in the difference images. We have done this by annotating the perimeter of the club head in ten difference images from DS1. These empirical PDFs are used in all segmentation tasks. Examples of difference images


(a) Extracted ROI.

(b) Thresholded ROI. Black indicates the biggest connected component and gray other thresholded areas. Blue is the selected club center. Smoothed column sum of the biggest connected component shown above.

Figure 4.13: Club center detection.

with corresponding color images and annotation masks are shown in Figure 4.14 on the next page.

Probability density functions (PDFs) for color images

We use GMMs to model the PDFs in the color images. To cope with the correlation between the red, green and blue color channels we perform principal component analysis (PCA) on the color image, see Figures 4.15(e) to 4.15(g) on page 45. We then extract the ROI in the computed background image (see Section 4.1 on page 26) and transform this to the PCA space. To this we fit a GMM using two three-dimensional Gaussian distributions to obtain the PDF $p_{c,b}(\mathbf{x}_{c,pca})$. With the initial segmentation we extract the foreground pixels and fit a PDF $p_{c,f}(\mathbf{x}_{c,pca})$ using a GMM with two three-dimensional Gaussian distributions.

Segmentation

Assuming independence we can combine the PDFs from the difference image and the color image to get the joint PDFs

$$
\begin{aligned}
p_f(x_d, \mathbf{x}_{c,pca}) &= p_{d,f}(x_d) \cdot p_{c,f}(\mathbf{x}_{c,pca}), \\
p_b(x_d, \mathbf{x}_{c,pca}) &= p_{d,b}(x_d) \cdot p_{c,b}(\mathbf{x}_{c,pca}).
\end{aligned}
$$

These are the PDFs that we use to model the 1-clique potentials in the MRF. To model the 2-clique potentials we set

$$
\beta_{ij} = \begin{cases} \beta_v & \text{if } i \text{ and } j \text{ are vertical neighbors} \\ \beta_h & \text{if } i \text{ and } j \text{ are horizontal neighbors,} \end{cases}
$$

where β_v and β_h are two predefined values chosen empirically.

We compute the MAP-MRF solution to obtain a new segmentation. This is done iteratively using the computed segmentation to estimate a new $p_{c,f}(\mathbf{x}_{c,pca})$ and thereby $p_f(x_d, \mathbf{x}_{c,pca})$. Note that through these iterations $p_{d,f}(x_d)$, $p_{d,b}(x_d)$ and $p_{c,b}(\mathbf{x}_{c,pca})$ are kept fixed. We stop after 10 iterations after which we have our final segmentation of the golf club.

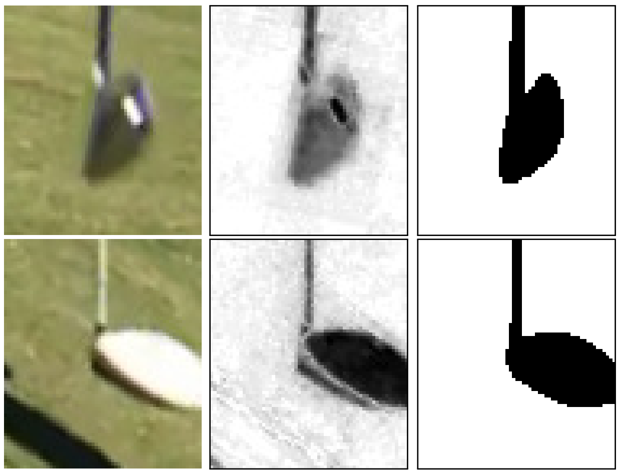
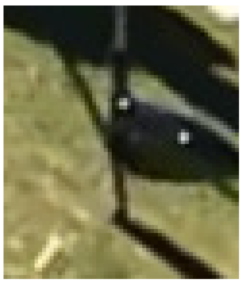
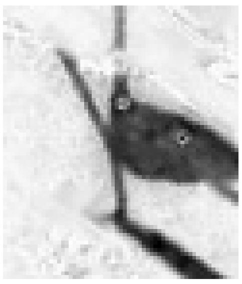
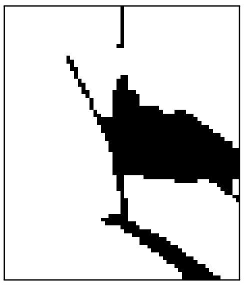


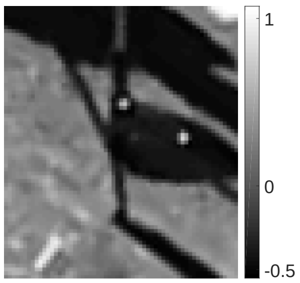
Figure 4.14: Examples of extracted ROI in color image, difference image and the annotated mask.


(a) Color image.

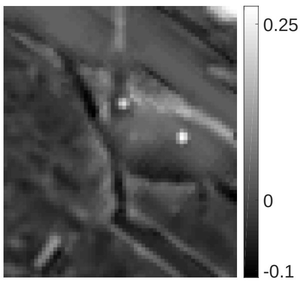

(b) Difference image.

(c) Background image.

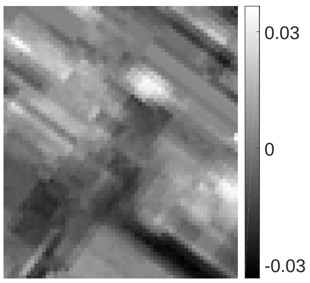

(d) Threshold segmentation.



(e) Color image projected to first PCA component.



(f) Color image projected to second PCA component.



(g) Color image projected to third PCA component.

Figure 4.15: Images used for segmenting the club head from the background.

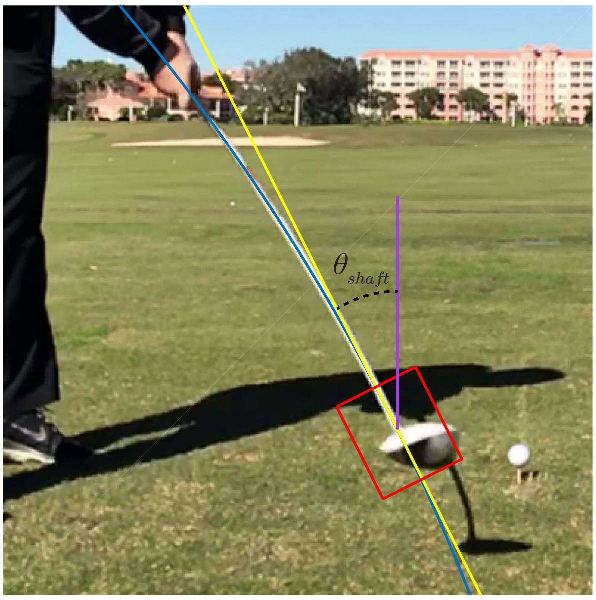


Figure 4.16: Blue line is fitted polynomial. Yellow line is the one fitted to the end of the inliers w.r.t. to the polynomial, red box is the extracted ROI for the club head segmentation and θ_{shaft} is the shaft angle.

Shaft-to-head transition detection

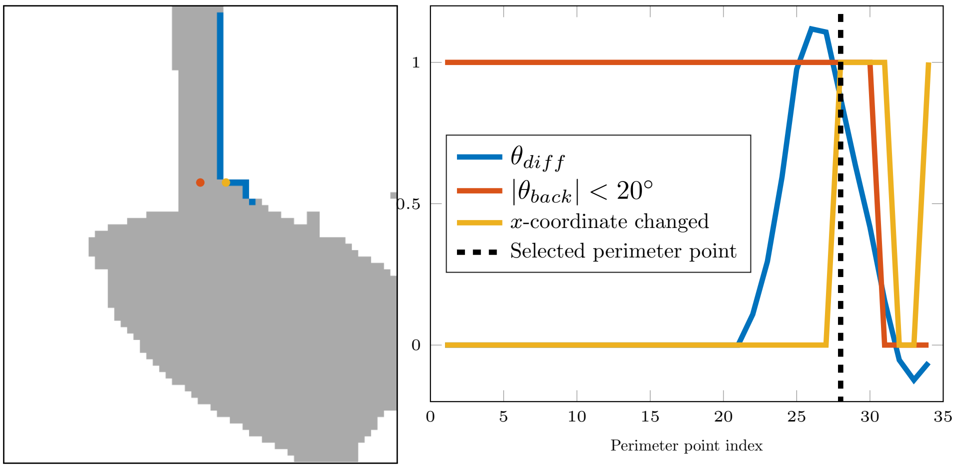
We trace the perimeter of the segmented club head and visit each perimeter point in clockwise order. At each point, we fit two lines, one to a few previous points, and one to the next few points. Using the angle of both lines, we compute the signed angle difference for this point

$$\theta_{diff} = \theta_{forward} - \theta_{back}.$$

The shaft-to-head transition point is then determined as the perimeter point where θ_{diff} is maximal and the following conditions are also satisfied.

- The x-coordinate is not larger than a specified threshold.
- The x-coordinate changes w.r.t. the previous point on the perimeter.
- $|\theta_{back}| < 20°$

This point is projected to the center of the shaft. The projected point is mapped back to the original image and used as our detected anchor point. The procedure is illustrated in Figure 4.17.



(a) Yellow is the selected perimeter point and red is projected to the shaft center. Blue points are perimeter points.

(b) Example of how the different criteria work together to find the transition point.

Figure 4.17: Shaft-to-head transition detection illustration. The perimeter points shown are the ones with x less than the threshold.

4.5 Anchor point interpolation

This section deals with different ways that we can use the detected anchor points in each frame to interpolate the position of the anchor point at the impact time. We consider interpolating in two fundamentally different ways. The first involves defining a plane in $\mathbb{R}^3$ on which a swing motion takes place. This is called the swing plane. The other approach is to consider it purely as an interpolation problem in $\mathbb{R}^2$. When we refer to a pendulum, we are describing a point moving around a pivot with a specified distance between them.

Pendulum with fixed length

During a golf swing, the club head position can be modeled as a circle or pendulum. We describe the angles in the circle with constant angular acceleration and a speed discontinuity at impact. The expression for θ thus becomes:

$$\theta(t) = \begin{cases} \frac{1}{2}\alpha_{pre} \cdot t^2 + \omega_{pre} \cdot t & t < 0 \\ \frac{1}{2}\alpha_{post} \cdot t^2 + \omega_{post} \cdot t & t \geq 0 \end{cases}.$$

This means that points on the circle will be given by

$$\mathbf{x}_{cam}(t) = \mathbf{K} \begin{bmatrix} \mathbf{R}_{sp} & \mathbf{t}_{sp} \end{bmatrix} \underbrace{\begin{bmatrix} r \cdot \sin(\theta(t)) \\ r \cdot \cos(\theta(t)) \\ 0 \\ 1 \end{bmatrix}}_{\mathbf{x}_{sp}(t)},$$

where $\mathbf{t}_{sp}$ is the center of the circle in $\mathbb{R}^3$ and $\mathbf{R}_{sp}$ is the rotation matrix describing the swing plane's orientation. The radar data includes a fitted model of this type, which gives us values for α and ω that we can use. Since we only care about the projection to the camera we can fix r to any value, as this will be equivalent to scaling $\mathbf{t}_{sp}$. If we choose $r = 1$ and can now compute $\mathbf{x}_{sp}(t)$ for all frames. Determining $\mathbf{R}_{sp}$ and $\mathbf{t}_{sp}$ is then reduced to solving the Perspective-n-Point (PnP) problem, which has many popular solutions with available implementations (Bradski 2000; Gao et al. 2003).

Pendulum with quadratic length

The model in the previous section has certain limitations that we are interested in overcoming. A problem is that the observed anchor point is not physically consistent, because the club head overlaps the shaft which makes us detect a point that lies further up on the shaft. See Figure 4.3(a) on page 27 for an example of an overlapping club head. Because of this, it is a bad assumption to have a constant length of the pendulum. This is overcome by letting the radius be a second-degree polynomial.

$$\mathbf{x}_{cam}(t) = \mathbf{K}\left(\mathbf{t}_{sp} + \mathbf{R}_{sp}\begin{bmatrix}\left(a\cdot\theta(t)^2 + b\cdot\theta(t) + c\right)\cdot\sin(\theta(t))\\ \left(a\cdot\theta(t)^2 + b\cdot\theta(t) + c\right)\cdot\cos(\theta(t))\\ 0\end{bmatrix}\right)$$

The rotation and translation have six degrees of freedom, and the polynomial introduces three variables. This leaves the problem with more variables than degrees of freedom, because the polynomial can scale the circle. This is again equivalent to scaling $\mathbf{t}_{sp}$. One could fix $c = 1$ if there was interest in having the number of variables equal the degrees of freedom. In both cases, the model can be fitted numerically. We have utilized the Levenberg-Marquardt minimization (Levenberg 1944; Lourakis Jul. 2004; Marquardt 1963). Additionally, we apply RANSAC to make it robust w.r.t. outliers. Each RANSAC iteration requires four points.

Pendulum with quadratic length and fixed swing plane

The above model can risk producing bad fits in cases where there are many outliers present. In order to combat this, we utilize that the radar data contains an estimate of the normal of the swing plane. This reduces the problem to six degrees of freedom. Because we only use the normal vector of the swing plane, we need to estimate the rotation of the plane around the normal vector θ_0.

$$\mathbf{x}_{cam}(t) = \mathbf{K}\left(\mathbf{t}_{sp} + \mathbf{R}_{sp}\begin{bmatrix}\left(a\cdot\theta(t)^2 + b\cdot\theta(t) + c\right)\cdot\sin(\theta(t)+\theta_0)\\ \left(a\cdot\theta(t)^2 + b\cdot\theta(t) + c\right)\cdot\cos(\theta(t)+\theta_0)\\ 0\end{bmatrix}\right)$$

This can again be solved by Levenberg-Marquardt minimization, but only requires three points for each RANSAC iteration.

Smoothing spline

We can also interpolate the anchor point using a smoothing spline. This method does not have a minimum amount of points to fit, and it is thus difficult to use RANSAC to fit this model. Because of this we will use the inliers from our best performing method to fit this model. The smoothing spline has the property that it provides good fits locally which might be advantageous. It is fitted using the built-in Matlab function `fit`. The smoothing spline minimizes the expression

$$p\sum_i (y_i - s(x_i))^2 + (1+p)\int\left(\frac{d^2 s}{dx^2}\right)^2 dx,$$

where p is a smoothing parameter chosen automatically.

Interpolation of shaft angle

Using the inliers found using the former models we interpolate the shaft angle to the time of impact using a polynomial fitted to the computed θ_{shaft} in each frame.

4.6 Pose estimation at the time of impact

Our goal is to determine where the on the club face the ball is hit. In order to do this, it is important to know the pose of the club head at impact time. In this section, we describe how to determine this pose and how we utilize it. Throughout the section, we will continually use various golf terms. We refer the reader to Section 2.3 on page 4 for an overview of these terms.

Correction of ball center and radius

The center of the circle found in Section 4.2 is not the projection of the true center $\mathbf{p}_{true}$ of the golf ball but simply the center of the ball pixels, see Figure 4.18 on the next page. The angles $\theta_{true,x}$ and $\theta_{true,y}$ that $\mathbf{p}_{true}$ has with the principal point can be computed as the mean angles

$$\theta_{true,x} = \frac{\theta_{1,x} + \theta_{2,x}}{2}, \qquad \theta_{true,y} = \frac{\theta_{1,y} + \theta_{2,y}}{2},$$

where $\theta_{1,x}$ and $\theta_{2,x}$ are the angles between the projection of the left and right side of the ball with the principal point respectively, and similarly for $\theta_{1,y}$ and $\theta_{2,y}$. We calculate these as

$$\theta_{1,x} = \tan^{-1}\left(\frac{(p_x - r_{px}) - pp_x}{f}\right), \qquad \theta_{2,x} = \tan^{-1}\left(\frac{(p_x + r_{px}) - pp_x}{f}\right)$$
$$\theta_{1,y} = \tan^{-1}\left(\frac{(p_y - r_{px}) - pp_y}{f}\right), \qquad \theta_{2,y} = \tan^{-1}\left(\frac{(p_y + r_{px}) - pp_y}{f}\right),$$

where (p_x, p_y) and r_{px} are the detected center and radius of the circle, (pp_x, pp_y) is the principal point and f is the focal length. Using $\theta_{true,x}$ and $\theta_{true,y}$ we can construct the point

$$\mathbf{c} = \begin{bmatrix} f\tan(\theta_{c,x}) \\ f\tan(\theta_{c,y}) \\ f \end{bmatrix}$$

which reprojects to the same point $\mathbf{p}_{true}$ as the center of the golf ball. I.e.

$$\mathbf{p}_{true} = s\begin{bmatrix} p_{true,x} \\ p_{true,y} \\ 1 \end{bmatrix} = \mathbf{Kc}$$

where $\mathbf{K}$ is the camera matrix.

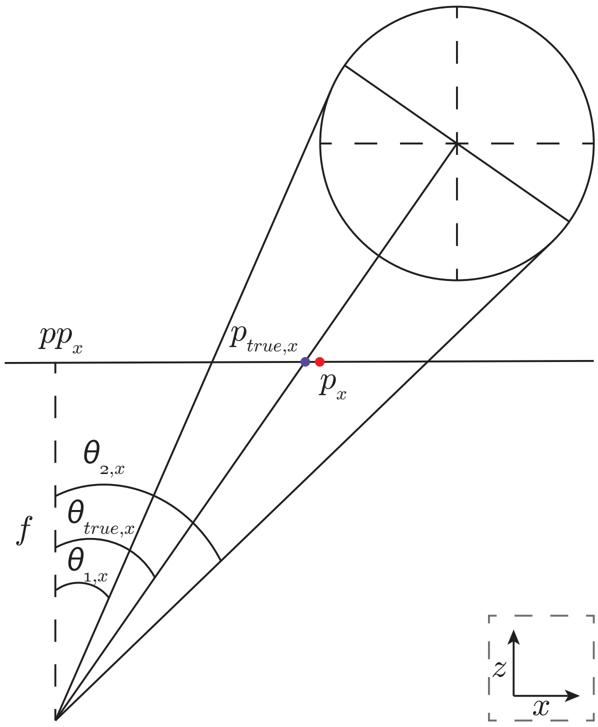


Figure 4.18: Overdrawn illustration of the top view of the projection of the golf ball to the image plane. The true ball center projects to $p_{true,x}$, p_x is the x-coordinate of the center of the detected circle, $\theta_{1,x}, \theta_{2,x}$ and $\theta_{true,x}$ is the angles between the the left side, right side and center of the ball, pp_x is the x-coordinate of the principal point and f is the focal length.

Ball radius

The detected radius of the golf ball is also skewed due to the camera perspective. If a ball is kept at the same depth but moved sideways the detected radius will become larger, but since the ball is moved further away from the camera the radius should, in fact, become smaller. Similarly, when the ball is moved vertically. If r_m is the measured radius in pixels and $d_x = p_x - p_{true_x}$, $d_y = p_y - p_{true_y}$ are x and y-coordinates of the distance between the detected ball center and the projection of the true ball center we can calculate a more accurate radius of the ball by

$$r_{px} = \frac{\cos(\theta_{true,x})(r_m + d_x) + \cos(\theta_{true,y})(r_m + d_y)}{2}.$$

Figure 4.19 on the following page illustrates the first term of the numerator. Note that this is just an approximation of the radius, and that the ball, in reality, should have two different radii in the vertical and horizontal direction, but it is a better approximation than the measured one.

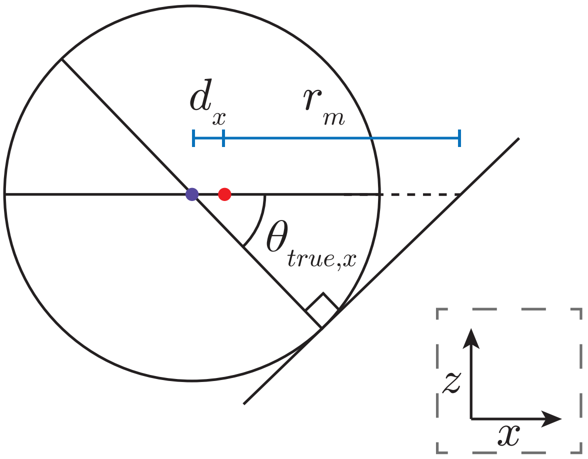


Figure 4.19: Illustration of the measured ball radius. The purple point is the center of the ball, the red point is the center of the projected circle, d_x is the distance between them, r_m is the measured radius and $\theta_{true,x}$ is the angle computed in the previous section.

True impact point

The golf ball is not hit at $\mathbf{p}_{true}$ but at the projection of the point where the club, which is assumed to be flat, is tangent to the ball. The radar data include a dynamic face angle and a dynamic loft angle. These are defined in the radars coordinate system, but can easily be transformed to the camera's coordinate system using our camera calibration. In spherical coordinates the dynamic loft angle $\theta_{loft,d}$ corresponds to the elevation and the dynamic lie angle $\theta_{lie,d}$ corresponds to the azimuth angle, hence they define a normal vector $\mathbf{n}_{club}$ to the hitting plane of the club at impact time. If we define

$$\begin{aligned}\mathbf{R}_{lf} &= \mathbf{R}_{cam}\mathbf{R}_y(\theta_{face,d})\mathbf{R}_x(\theta_{loft,d})\\ &= \mathbf{R}_{cam}\begin{bmatrix}\cos(\theta_{face,d}) & 0 & \sin(\theta_{face,d})\\ 0 & 1 & 0\\ -\sin(\theta_{face,d}) & 0 & \cos(\theta_{face,d})\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\theta_{loft,d}) & -\sin(\theta_{loft,d})\\ 0 & \sin(\theta_{loft,d}) & \cos(\theta_{loft,d})\end{bmatrix},\end{aligned}$$

then the normal vector is

$$\mathbf{n}_{club} = \mathbf{R}_{lf}\begin{bmatrix}0\\0\\1\end{bmatrix} = \mathbf{R}_{cam}\begin{bmatrix}\sin(\theta_{face,d})\cos(\theta_{loft,d}))\\ -\sin(\theta_{loft,d})\\ \cos(\theta_{face,d})\cos(\theta_{loft,d}))\end{bmatrix}.$$

The true impact point $\mathbf{p}_{impact}$ is the point on the ball which has normal $-\mathbf{n}_{club}$ projected to the camera hence

$$\mathbf{p}_{impact} = \mathbf{K}\left(\mathbf{c} - r_{px}\mathbf{n}_{club}\right).$$

A couple of examples are shown in Figure 4.20 on page 55.

Computing dynamic lie angle

To fully determine the pose of the golf club at impact time we need to estimate $\theta_{lie,d}$. This angle describes how much the club is rotated around $\mathbf{n}_{club}$. Using the known variables $\theta_{face,d}$, $\theta_{loft,d}$, $\theta_{loft,s}$, $\theta_{lie,s}$ and the interpolated shaft angle θ_{shaft} in the image plane we can set up an equation which we can solve for $\theta_{lie,d}$. We utilize that we can observe the club shaft which is fixed to the head. When the club is in neutral position, the direction vector of the shaft in $\mathbb{R}^3$ is given by

$$\mathbf{v}_{shaft,n} = \begin{bmatrix} -\cos(\theta_{lie,s}) \\ -\sin(\theta_{lie,s}) \\ 0 \end{bmatrix}.$$

If we rotate the club around the x-axis such that the club face has normal equal to the z-axis we get

$$\begin{aligned}
\mathbf{v}_{shaft,r} &= \mathbf{R}_x(-\theta_{loft,s})\mathbf{v}_{shaft,n} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_{loft,s}) & \sin(\theta_{loft,s}) \\ 0 & -\sin(\theta_{loft,s}) & \cos(\theta_{loft,s}) \end{bmatrix} \begin{bmatrix} -\cos(\theta_{lie,s}) \\ -\sin(\theta_{lie,s}) \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} -\cos(\theta_{lie,s}) \\ -\cos(\theta_{loft,s})\sin(\theta_{lie,s}) \\ \sin(\theta_{loft,s})\sin(\theta_{lie,s}) \end{bmatrix}.
\end{aligned}$$

If the interpolated anchor point is $(p_{ap,x}, p_{ap,y})$ then we can define a point in $\mathbb{R}^3$ that projects to our anchor point.

$$\mathbf{t}_{ap} = \begin{bmatrix} p_{ap,x} - pp_x \\ p_{ap,y} - pp_y \\ f \end{bmatrix}$$

Using the rotated direction vector of the club shaft $\mathbf{v}_{shaft,r}$ and $\mathbf{t}_{ap}$ we can compute what shaft angle a given dynamic lie will result in. We use that a lie angle is defined as a rotation around the normal vector of the club face, and that $\mathbf{R}_{lf}$ rotates the z-axis to the normal of the club face in camera coordinates.

$$\mathbf{p}_{shaft}(\theta_{lie,d}) = \mathbf{K}\begin{bmatrix}\mathbf{R}_{lf}\mathbf{R}_z(\theta_{lie,d}) & \mathbf{t}_{ap}\end{bmatrix}\begin{bmatrix}\mathbf{v}_{shaft,r} \\ 1\end{bmatrix}$$

We thus have a point on the shaft in the image plane as a function of the dynamic lie $\theta_{lie,d}$. From this, we obtain the equation

$$\theta_{shaft} = \tan^{-1}\left(\frac{p_{shaft,x}(\theta_{lie,d}) - p_{ap,x}}{p_{shaft,y}(\theta_{lie,d}) - p_{ap,y}}\right),$$

where θ_{shaft} is the interpolated shaft angle in the image plane. We solve this equation for $\theta_{lie,d}$ numerically. Note that this angle does not depend on the depth at which $\mathbf{t}_{ap}$ is created (see Appendix A.2 on page 87).

Hitting plane homography

The dynamic lie, loft, and face angles uniquely determine the pose of the golf club at impact time. We define the hitting plane as the plane which has normal $\mathbf{n}_{club}$ and is rotated around this normal with $\theta_{lie,d}$. Examples of different dynamic lies are shown in Figure 4.21 on page 56.

We consider the following points in $\mathbb{R}^3$,

$$\mathbf{b}_1 = \begin{bmatrix} r_{px} \\ r_{px} \\ -r_{px} \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} -r_{px} \\ r_{px} \\ -r_{px} \end{bmatrix}, \quad \mathbf{b}_3 = \begin{bmatrix} r_{px} \\ -r_{px} \\ -r_{px} \end{bmatrix}, \quad \mathbf{b}_4 = \begin{bmatrix} -r_{px} \\ -r_{px} \\ -r_{px} \end{bmatrix}.$$

We transform these to the hitting plane and project them to the camera by

$$\mathbf{p}_{px,i} = \mathbf{K}\begin{bmatrix}\mathbf{R}_{lf}\mathbf{R}_z(\theta_{lie,d}) & \mathbf{c}\end{bmatrix}\begin{bmatrix}\mathbf{b}_i \\ 1\end{bmatrix}, \quad i \in \{1,2,3,4\}$$

We make the four corresponding points

$$\mathbf{p}_{cm,1} = \begin{bmatrix} -r_{cm} \\ r_{cm} \\ 1 \end{bmatrix}, \quad \mathbf{p}_{cm,2} = \begin{bmatrix} r_{cm} \\ r_{cm} \\ 1 \end{bmatrix}, \quad \mathbf{p}_{cm,3} = \begin{bmatrix} -r_{cm} \\ -r_{cm} \\ 1 \end{bmatrix}, \quad \mathbf{p}_{cm,4} = \begin{bmatrix} r_{cm} \\ -r_{cm} \\ 1 \end{bmatrix}$$

Using these four point correspondences we compute a homography $\mathbf{H}$ such that

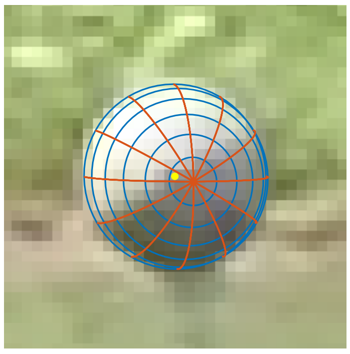
$$\mathbf{p}_{cm,i} = \mathbf{H}\mathbf{p}_{px,i}, \quad i \in \{1,2,3,4\}$$

With the assumption that the interpolated anchor point $\mathbf{p}_{ap}$ lies in the hitting plane we can now compute a vector in cm from the impact point to $\mathbf{p}_{ap}$ using the homography. We get this from

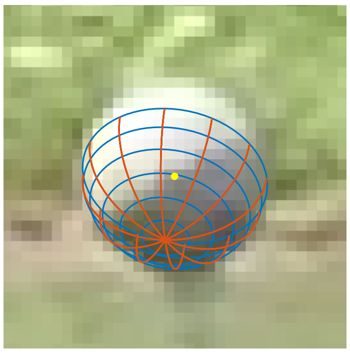
$$\mathbf{v}_{impact\to ap} = \mathbf{H}\mathbf{p}_{ap},$$

where we note that $\mathbf{v}_{impact\to ap}$ is in homogeneous coordinates.

(a) Example of pre-impact frame.

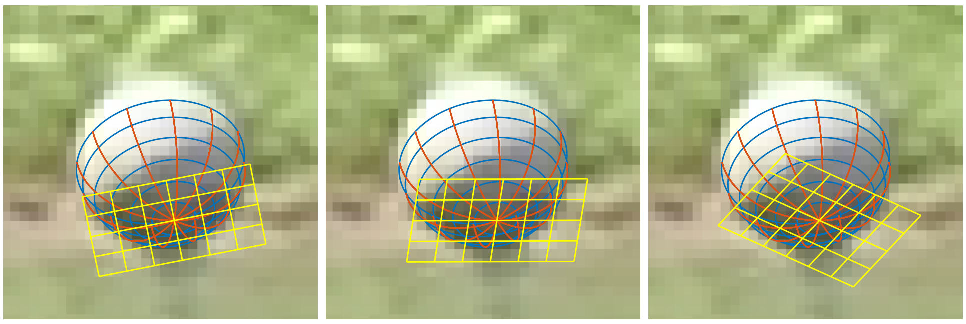


(b) $\theta_{face,d} = \theta_{loft,d} = 0°$.



(c) $\theta_{face,d} = 20°$, $\theta_{loft,d} = 40°$.

Figure 4.20: Two examples of the true impact point for different $\mathbf{n}_{club}$'s. The yellow point is the center of the detected circle and the intersection of lines is the true impact point. Note that these only coincide for $\theta_{face,d} = \theta_{loft,d} = 0°$ when the center is equal to the principal point due to the perspective of the camera.



(a) $\theta_{lie,d} = -15°$ (b) $\theta_{lie,d} = 0°$ (c) $\theta_{lie,d} = 30°$

Figure 4.21: Three examples of hitting planes with different dynamic lies. In all three examples $\theta_{loft,d} = 40°$, $\theta_{face,d} = 0°$. The width of the hitting planes are $2r_{px}$ and the height is $\frac{4}{3}r_{px}$.

4.7 Estimating impact point

From the previous section we obtain the vector from the impact point to the anchor point $\mathbf{v}_{impact \to ap}$. The vector $\mathbf{v}_{impact}$ from a specified origin on the golf club to the true impact point has been annotated. These are shown as the green and blue vectors in Figure 4.22. We can add them to obtain

$$\mathbf{v}_{ap} = \mathbf{v}_{impact} + \mathbf{v}_{impact \to ap},$$

which is vector going from the origin on the golf club to the anchor point. This vector is shown as the red vector in Figure 4.22. If we can compute $\mathbf{v}_{impact \to ap}$ without error, $\mathbf{v}_{ap}$ should be constant for a given golf club. We can however not compute $\mathbf{v}_{impact \to ap}$ without error and $\mathbf{v}_{ap}$ should have an almost identical error since $\mathbf{v}_{impact}$ is expected to be accurate. Variations in $\mathbf{v}_{ap}$ over all strokes for a given club thus describe how accurately we are able to determine $\mathbf{v}_{impact \to ap}$.

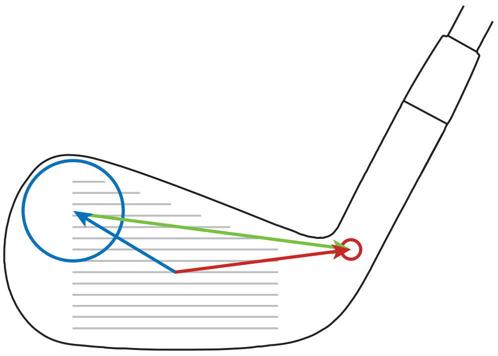


Figure 4.22: Illustration showing the vectors $\mathbf{v}_{impact \to ap}$, $\mathbf{v}_{impact}$ and $\mathbf{v}_{ap}$ as the green, blue and red vectors respectively. The blue circle is an outline of the impact of the ball, and the red circle is the anchor point.

We assume that each computed $\mathbf{v}_{ap,i}$ is the sum of a true anchor point $\mathbf{v}_{ap,true}$ for the given club, a potential bias term $\boldsymbol{\beta}$ and a Gaussian distributed error $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \boldsymbol{\sigma})$. I.e.

$$\begin{aligned} \mathbf{v}_{ap,i} &= \mathbf{v}_{ap,true} + \boldsymbol{\beta} + \boldsymbol{\epsilon_i} \\ &= \mathbf{v}_{ap,observed} + \boldsymbol{\epsilon_i}. \end{aligned}$$

Hence, $\mathbf{v}_{ap,i} \sim \mathcal{N}(\mathbf{v}_{ap,observed}, \boldsymbol{\sigma})$. Given enough strokes, we can compute the maximum likelihood estimate of $\mathbf{v}_{ap,observed}$ as

$$\widetilde{\mathbf{v}}_{ap,observed} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{v}_{ap,i}.$$

Finally, the computed impact position is calculated as

$$\mathbf{v}_{impact} = \widetilde{\mathbf{v}}_{ap,observed} - \mathbf{v}_{impact \to ap}.$$

Since $\widetilde{\mathbf{v}}_{ap,observed}$ is a constant for a specific club, we see that the error distribution of $\mathbf{v}_{impact}$ is the same as for $-\mathbf{v}_{impact \to ap}$.

CHAPTER 5

Results

5.1 Camera calibration

Dataset 1

The root mean square error (RMSE) between the projected ball positions from the radar and the detected positions is 7.82 pixels.

Dataset 2

The reprojection using the estimated internal and external parameters for one of the four frames can be seen in Figure 5.1. RMSE of all the reprojected points in the four image sets is 0.38 pixels.

(a) Radar internal camera. (b) DS2n (c) DS2z

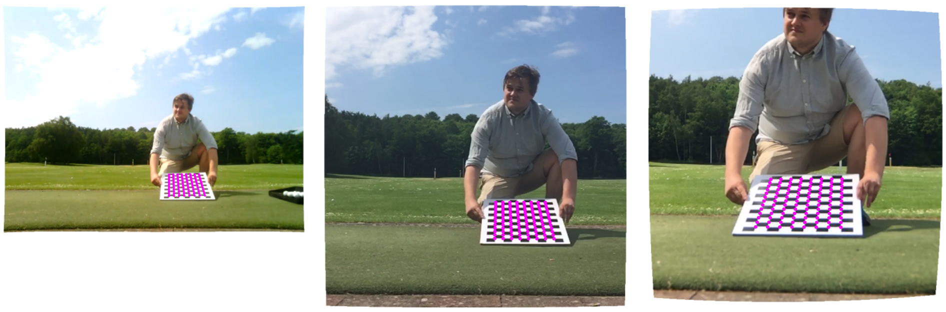


Figure 5.1: Reprojection of checkerboard points in DS2 with final camera calibration.

5.2 Ball position

In Table 5.2 on the next page, we present how well our two different methods for initial ball detection perform, compared to manual annotations of the initial ball. We have not used stroke 40 from DS1 as the radar data for this stroke are imprecise. In order to understand the scale of one pixel, Table 5.4 on page 61 gives an overview of how big the balls in each dataset approximately are. Furthermore, Figure 5.3 on the next page shows visually how the detected positions with the Hough method differ from the annotations.

Table 5.2: Initial ball detection compared with annotations. Unit is pixels, μ is mean and σ is standard deviation. The inlier threshold for the RANSAC method is 0.5 pixels.

	Position					Radius		
Dataset	RMSE	$\mu_{\Delta x}$	$\mu_{\Delta y}$	$\sigma_{\Delta x}$	$\sigma_{\Delta y}$	RMSE	$\mu_{\Delta r}$	$\sigma_{\Delta r}$
Hough								
DS1[†]	0.46	-0.17	-0.16	0.26	0.31	0.74	-0.67	0.32
DS2n	0.76	0.36	0.05	0.40	0.54	0.51	-0.10	0.50
DS2z	0.81	0.26	0.09	0.36	0.68	0.54	0.10	0.54
RANSAC								
DS1[†]	1.13	-0.24	-0.25	0.62	0.89	1.06	-0.93	0.52
DS2n	0.84	0.13	-0.17	0.39	0.72	0.67	-0.39	0.55
DS2z	1.04	0.17	0.03	0.38	0.96	0.63	-0.11	0.62

[†]Excluding stroke 40.

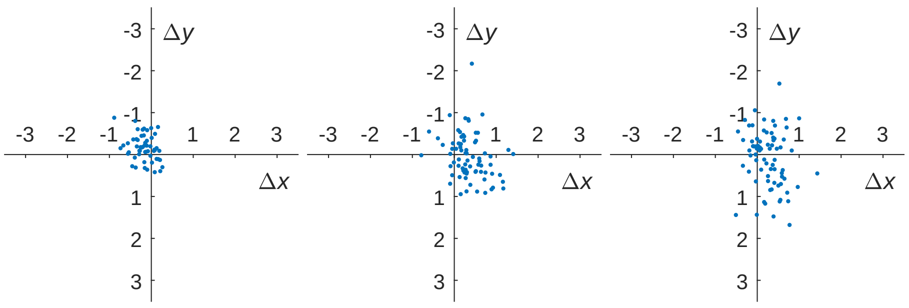


Figure 5.3: Differences of the initial ball position between annotations and Hough result.

Inlier threshold for RANSAC

The RANSAC algorithm requires a threshold to determine which points are inliers and outliers for a given model. We show the RMSE of the detected ball position over all three datasets as a function of the threshold in Figure 5.5 on the facing page.

Table 5.4: Average ball radius in pixels from annotations.

	DS1	DS2n	DS2z
μ_r	7.4	9.3	13.3

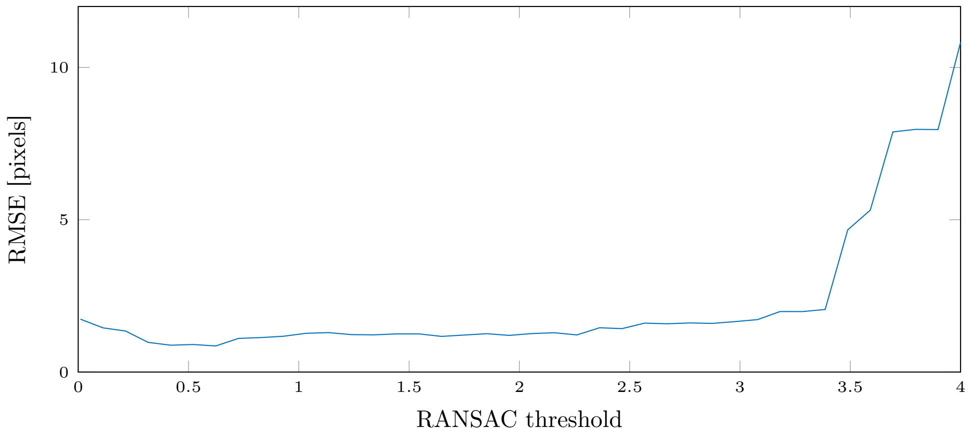


Figure 5.5: RMSE between annotations and detected ball positions as a function of the inlier threshold used for RANSAC.

5.3 Impact time

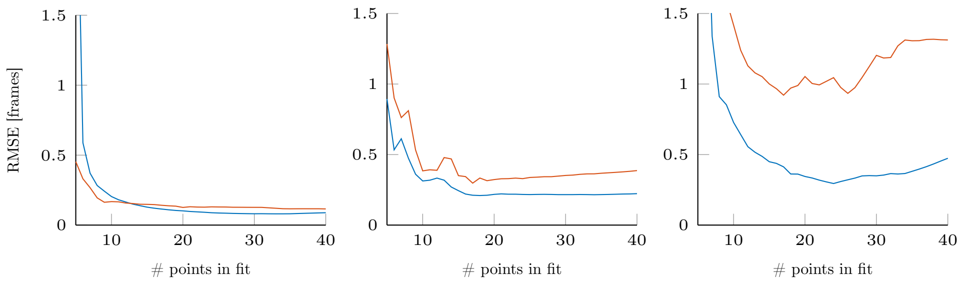
Using the annotation of the first visible ball after impact we can estimate how well our impact time model works. This is done by estimating the frame of the annotation from its position and the ball track. If the annotated frame is also in the track, the track in this frame is omitted. Figure 5.7(a) on page 63 shows the RMSEs of the different methods for each dataset. These numbers are very high because of a few outliers as illustrated in Figure 5.7(b) on page 63. Because of this, we also present a combined RMSE for all datasets, where strokes that have an absolute error of more than one frame at 20 fitted points with either method have been omitted. These strokes are shown in Table 5.6. The result is shown in Figure 5.7(c) on page 63.

Table 5.6: Strokes that have an absolute error of more than one frame at 20 fitted points with either linear or quadratic, both nonlinearly optimized.

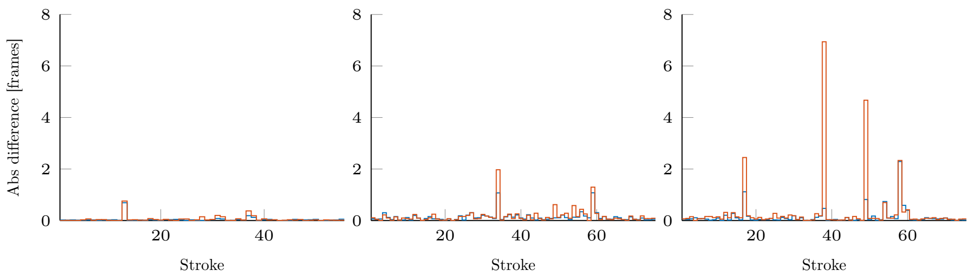
Dataset	Strokes
DS1	None
DS2n	34, 59
DS2z	17, 38, 49, 58

Impact of missing frames in ball tracks

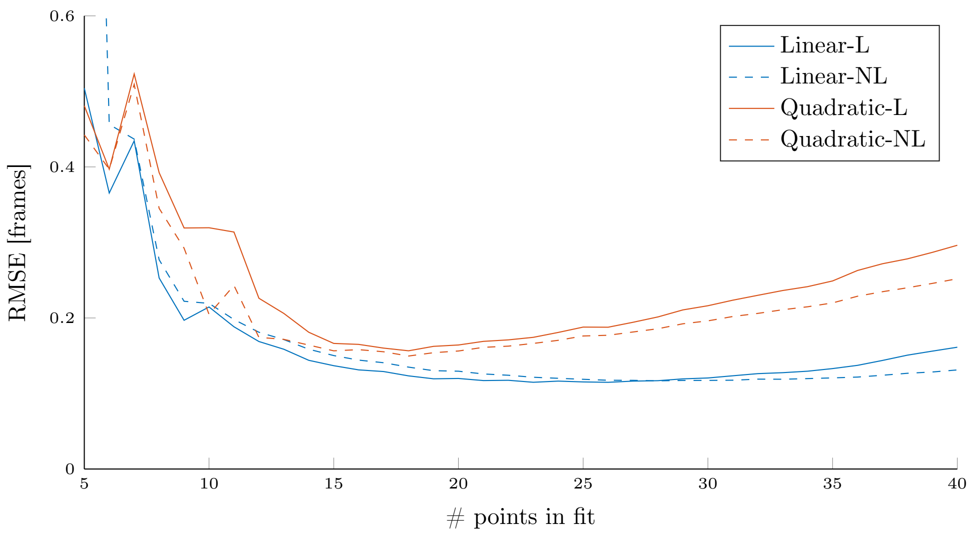
The ball is not always tracked immediately due to various causes e.g. the club being in front of it. For each stroke in the three datasets, we have compared the frame numbers at which the ball tracks start with the impact frame rounded up. Figure 5.8(a) on page 64 shows the distributions of these differences for the three datasets. Using only the strokes that have zero or one frame difference we calculate how the RMSE worsens when we remove points from the tracks. We remove points from the beginning of the tracks and compute the frame number of the annotated first visible post-impact ball using the linear model with non-linearly optimized parameters. Comparing with the true frame number we can compute the RMSE of the used strokes. Figure 5.8(b) on page 64 shows the RMSEs as a function of how many points are removed for the three datasets.



(a) RMSE of linear (blue) and quadratic (red) impact time methods. Both methods are nonlinearly optimized. From left DS1, DS2n and DS2z.

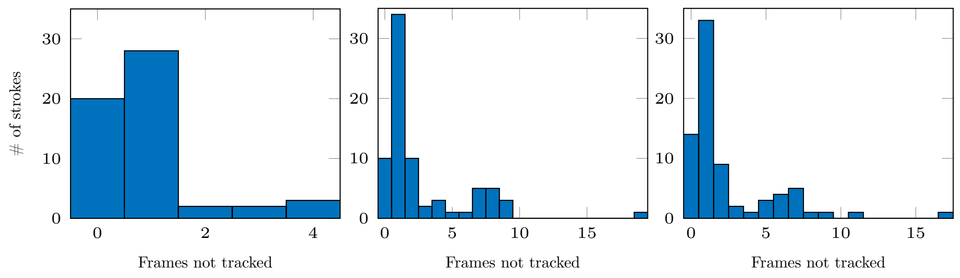


(b) Absolute frame difference for each stroke with 20 fitted points for linear (blue) and quadratic (red). Both methods are nonlinearly optimized. From left DS1, DS2n and DS2z.

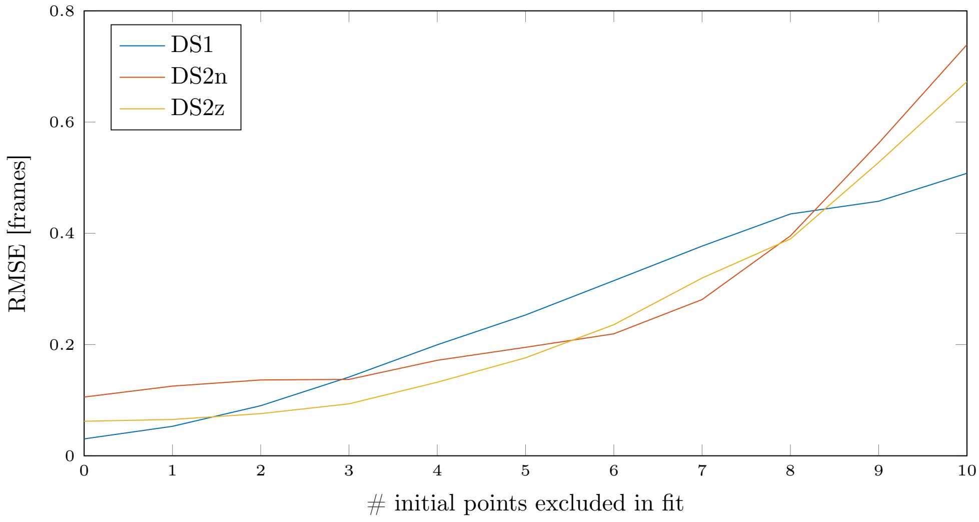


(c) RMSE on all strokes except the ones in Table 5.6 on page 61. L means Linear and NL nonlinear.

Figure 5.7: Results on impact time computations.



(a) Histograms showing the distributions of how many frames immediately after impact that has not been tracked for each of the three datasets.



(b) RMSE as a function of number of points excluded from the ball tracks. Only those strokes that have tracked the second frame immediately after impact have been used in this graph. The RMSEs are calculated with respect to the first frame after impact, where the ball has been annotated.

Figure 5.8: Results that relate missing frames in ball tracks to the error in impact time.

5.4 Anchor point detection

Figure 5.9 shows the result of our club head segmentation and anchor point detection in five randomly selected cases.

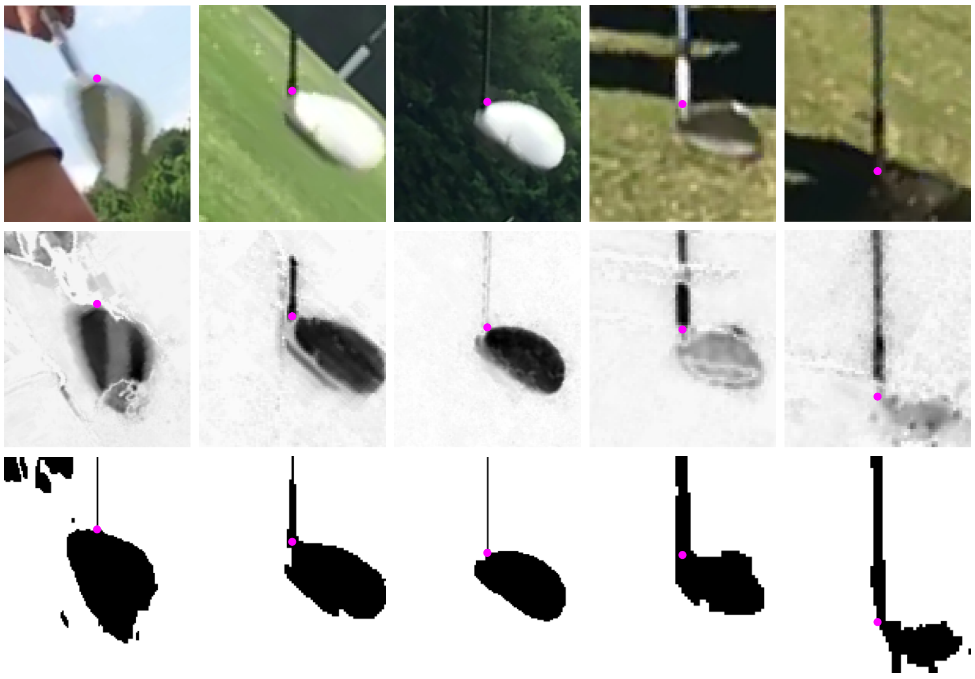


Figure 5.9: Images showing the result of our club head segmentation with their respective detected anchor points. Top row: color image, middle row: difference image, bottom row: segmentation.

Empirical probability density functions for difference images

The empirical densities computed from ten annotated difference images, that have been used for the modeling of MRFs are shown in Figure 5.10 on the next page.

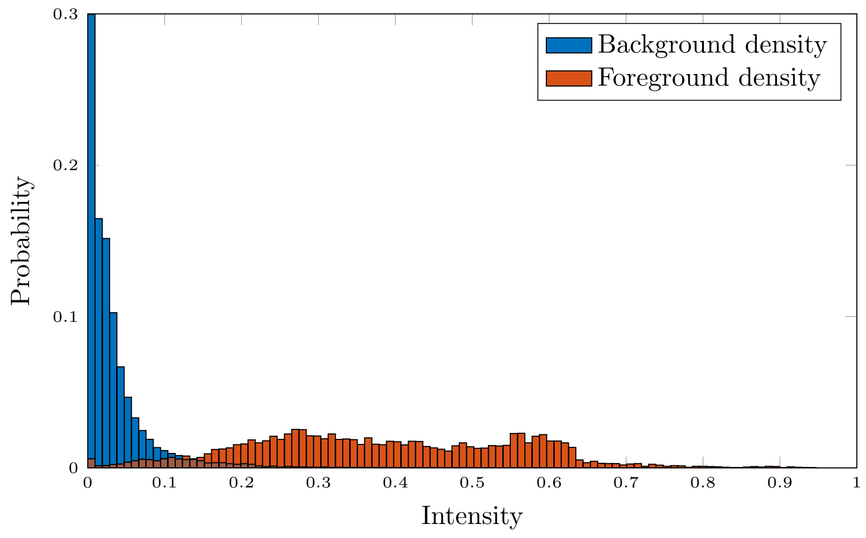


Figure 5.10: Empirical foreground and background PDFs from annotated difference images.

5.5 Anchor point interpolation

In our datasets, there are various elements of missing data that render strokes unsuitable for evaluating our method on. Table 5.11 shows these strokes. Additionally, in the video of stroke 20 in DS2z, there is a duplicated frame near impact. Because of this, we have not used that video.

Table 5.11: Strokes with missing data. Total of thirteen strokes from DS1 and five strokes from DS2.

Missing	DS1	DS2
Ground truth	28, 40, 45, 54	46
Club radar data	13, 40	24, 32, 45
Club radar post impact speed	43, 44, 45, 46, 47, 48, 49, 52, 54, 55	9

When fitting the swing models, we have used the eight frames before impact and the ten after, yielding a total of eighteen data points.

In order to evaluate the anchor point interpolation method, we use the annotated anchor point before impact and fit the models without the anchor point detected in this frame. Figure 5.12 on the next page shows the RMSE for each of our interpolation methods for varying RANSAC thresholds. Because the best performing model is the pendulum with quadratic length, we have used the inliers from this to fit the smoothing spline.

We have also attempted fitting to fewer frames than eighteen by removing n frames from both ends of the interval. The result of this is shown in Figure 5.14 on page 69.

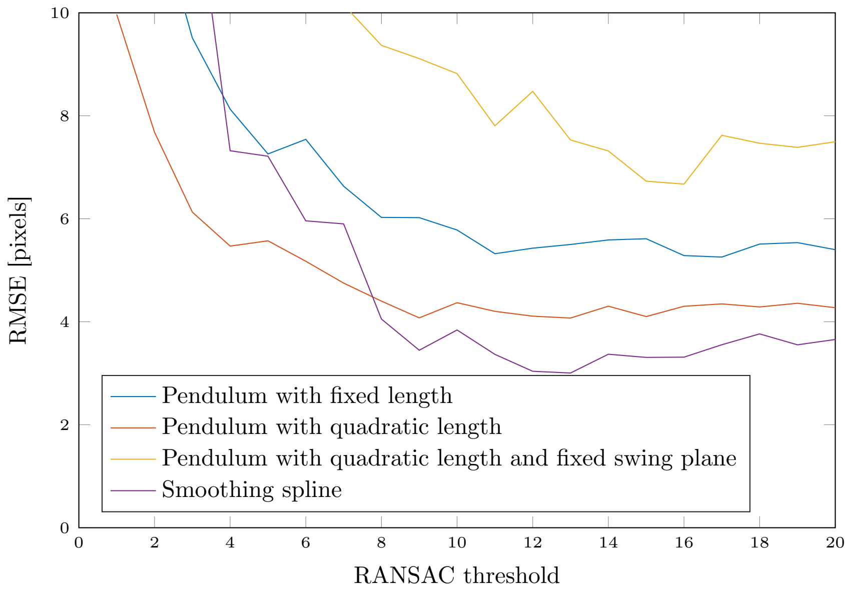


Figure 5.12: RMSEs for interpolated anchor point compared to annotation as a function of RANSAC threshold.

An example of a model fitted to detected anchor points is shown in Figure 5.13 on the next page

Interpolation of shaft angle

Using the inliers of the pendulum with quadratic length at a threshold of twelve pixels we can estimate the value of θ_{shaft} in the last frame before impact. We compare the fitted value against the computed value in the frame. Because the fit is computed using the same inliers, naturally the value of θ_{shaft} in the frame predicted is not used. The result of predicting θ_{shaft} for different degrees of the polynomial can be seen in Figure 5.15 on page 69

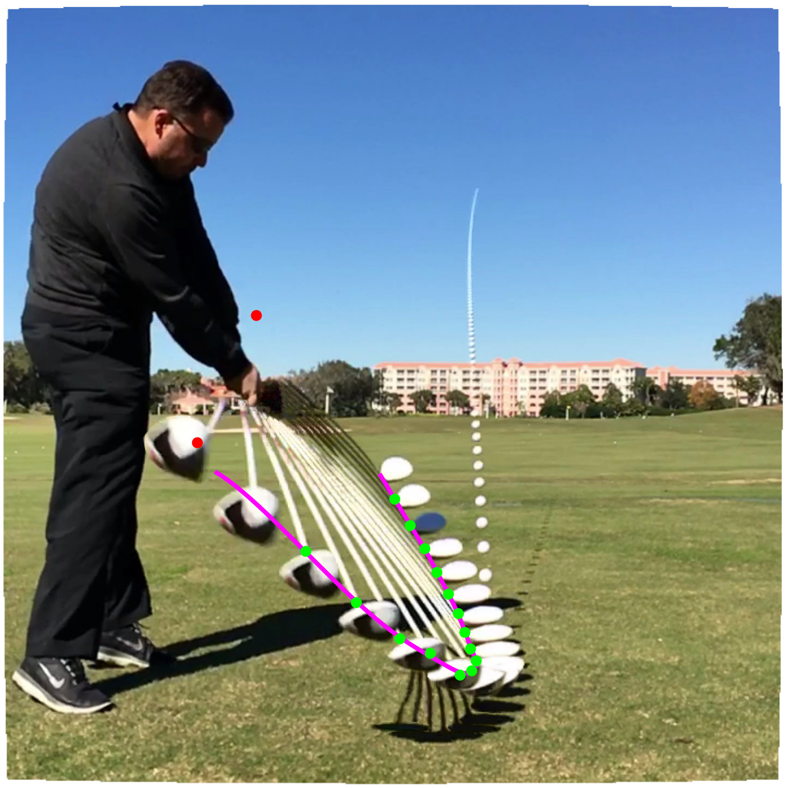


Figure 5.13: Example of a swing. Detected anchor points are shown as dots. Inliers w.r.t. the fitted pendulum with quadratic length are shown as green points, while outliers are red. The pink line is the fitted smoothing spline.

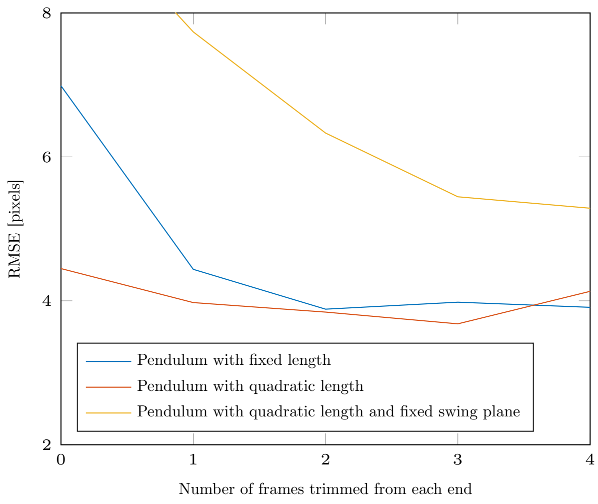


Figure 5.14: RMSE as a function of number of frame pairs not used out of the eighteen frames.

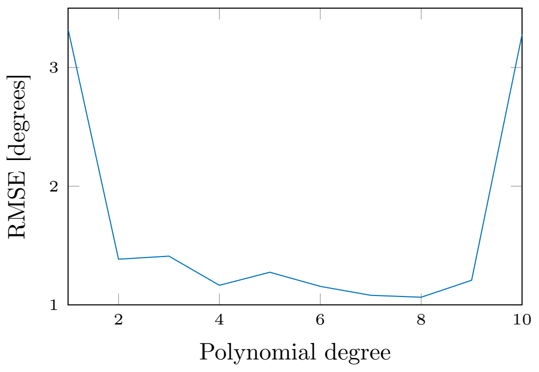


Figure 5.15: Degree of polynomial to predict θ_{shaft}.

5.6 Anchor point vectors

Table 5.16 shows standard deviations in the x- and y-directions for $\mathbf{v}_{ap}$ over all possible strokes. The mean has been subtracted for each club because we expect different clubs to have different anchor points. When we refer to "each club" it is in the context of both club and camera angle, which implies that we treat the videos from DS2n and DS2z as separate observations on different clubs. The results are shown for the two best performing methods for anchor point interpolation with 12 pixels as the RANSAC threshold, the linear impact time method with non-linearly optimized parameters and the Hough method for locating the initial ball. The check-marks indicate whether an angle has been taken into account in computing the hitting plane homography. If not used, the value is set to 0°. A stroke has been removed since $\mathbf{v}_{impact \to ap}$ was longer than 20 cm, which is much longer than the diagonal of the golf clubs used. The rest of the vectors are shorter than 10 cm. Tables 5.17 to 5.19 on this page and on the next page show the errors for each club in each dataset with the smoothing spline method using only lie angle for the hitting plane homography.

Table 5.16: Standard deviations on $\mathbf{v}_{ap}$ for the two best-performing anchor point interpolation methods, with different angles considered for the hitting plane homography. Means have been subtracted on a for each club. The unit is mm.

	M1	M2	M3	M4	M5
Loft angle		✓		✓	✓
Face angle					✓
Lie angle			✓	✓	✓
Pendulum with quadratic length					
σ_x	5.19	5.22	5.07	5.14	4.94
σ_y	5.73	5.87	5.56	5.88	5.79
Smoothing spline					
σ_x	4.55	4.63	4.18	4.26	4.30
σ_y	5.34	5.53	5.20	5.66	5.51

Table 5.17: Standard deviations on $\mathbf{v}_{ap}$ for each club in DS1 using M3. The unit is mm.

	All	Driver 1	Driver 2	Iron 1	Wedge 1[†]
σ_x	2.59	2.61	2.59	2.08	3.77
σ_y	4.35	3.30	2.79	4.02	9.73

[†]Computed from only three strokes due to missing radar data.

Table 5.18: Standard deviations on $\mathbf{v}_{ap}$ for each club in DS2n using M3. The unit is mm.

	All	Driver 1	Driver 3	Iron 2	Wedge 2
σ_x	5.38	2.83	9.08	1.67	4.14
σ_y	7.21	4.29	13.52	3.35	4.14

Table 5.19: Standard deviations on $\mathbf{v}_{ap}$ for each club in DS2z using M3. The unit is mm.

	All	Driver 1	Driver 3	Iron 2	Wedge 2
σ_x	5.24	6.98	4.95	1.81	3.53
σ_y	4.58	4.94	5.39	4.24	4.06

Anchor point clouds

For each stroke, we perform our method on we get a sample of $\mathbf{v}_{ap}$ as explained in Section 4.7 on page 57. These anchor points can be plotted in a scatter plot, which is what we mean when we refer to the anchor point cloud. In Figure 5.20 on page 73 computed $\mathbf{v}_{ap,i}$s are shown on top of an image corresponding to their respective club types. The method used is the smoothing spline method and M3 from Table 5.16 on page 70. Note that the chosen images are randomly chosen examples (among the images that are focused) and the location of the impact ball has nothing to do with the anchor point cloud. Similar plots for other clubs and used angles can be seen in Appendix B on page 89.

Figure 5.21 on page 74 shows the $\mathbf{v}_{ap,i}$s for a wedge if we include the loft angle, i.e. M4. Table 5.22 shows the standard deviations for the wedges in the datasets if M4 is used.

Table 5.22: Standard deviations on $\mathbf{v}_{ap}$ for the wedges in the datasets using M4, i.e. including the loft angle. The unit is mm.

	DS1 Wedge 1	DS2n Wedge 2	DS2z Wedge 2
σ_x	5.03	4.34	3.81
σ_y	9.69	6.73	5.69

Using only every second anchor point

We have also computed results where we have only used every second anchor point. These can be seen in Table 5.23 on the following page. We have removed the same stroke as in the previous section, and two additional strokes that produced $\mathbf{v}_{impact\to ap}$ vectors longer than 10 cm.

Table 5.23: Standard deviations on $\mathbf{v}_{ap}$ for the smoothing spline, where only every second anchor point has been used. Means have been subtracted for each club. Methods are as in Table 5.16 on page 70.

	M1	M2	M3	M4	M5
Smoothing spline					
σ_x	5.47	5.53	5.13	5.24	5.22
σ_y	6.37	6.57	6.40	6.86	6.79

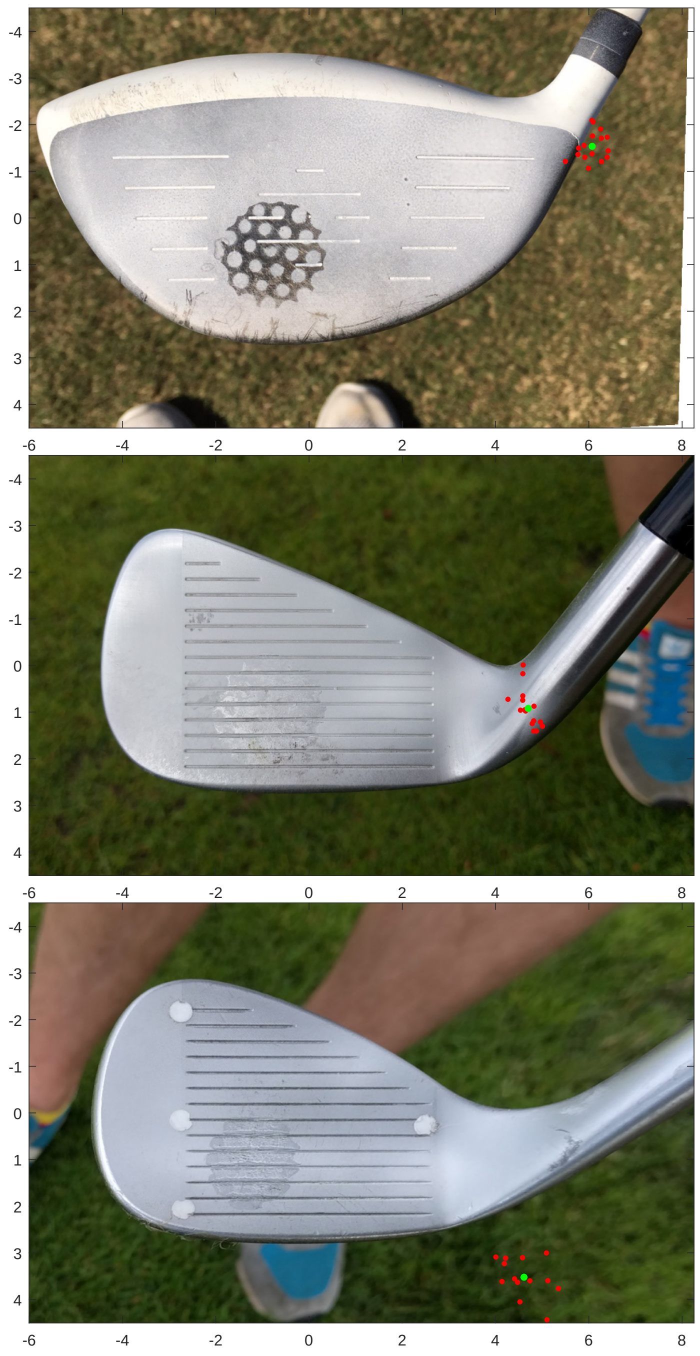


Figure 5.20: Computed $\mathbf{v}_{ap,i}$s shown on top of the corresponding club. The red points are the anchor point clouds, while the green points are the means. From top to bottom: DS1 Driver 1, DS2z Iron 2 and DS2n Wedge 2. The method is smoothing spline and M3. Units are cm.

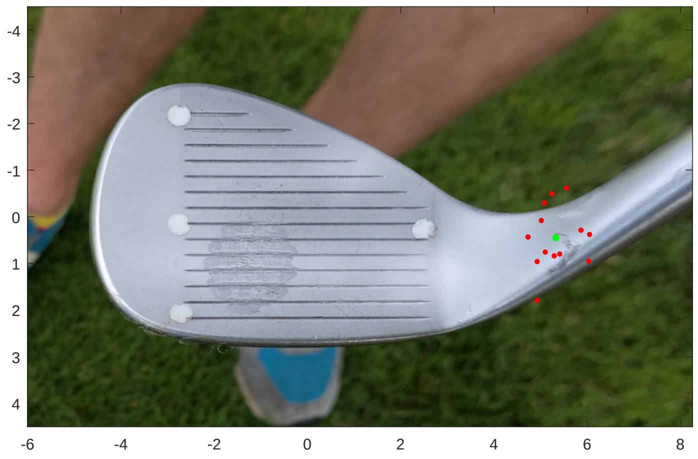


Figure 5.21: Computed $\mathbf{v}_{ap,i}$s shown on top DS2n Wedge 2 in red. The green point is the mean. Units are cm.

5.7 Impact position

Figure 5.24 on the next page shows predicted impact positions $\mathbf{v}_{impact}$ on eighteen randomly selected strokes from our datasets. The anchor point used is $\widetilde{\mathbf{v}}_{ap,observed}$ computed for the specific club and specific camera. The anchor point interpolation is done with the smoothing spline method. For drivers and irons, we have only used the dynamic lie angle, and for wedges, we have also included the dynamic loft angle in the hitting plane homography. As described in Section 4.7 on page 57 the horizontal and vertical errors are the same as the errors for $\mathbf{v}_{impact \to ap}$, which are found in Section 5.6 on page 70.

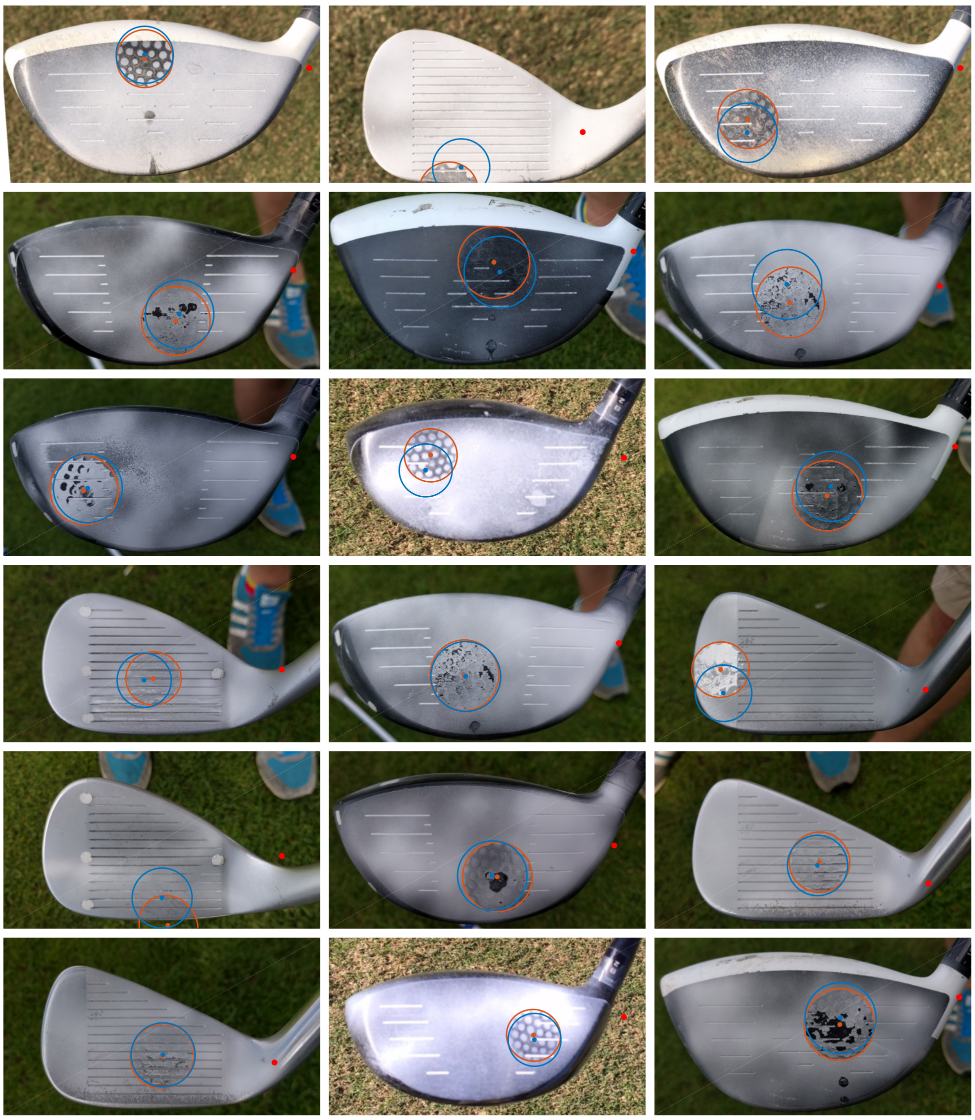


Figure 5.24: Random selection of predicted impact locations. Red circles are the annotated positions and blue circles are the predicted ones. The red dot is the used anchor point.

CHAPTER 6

Discussion

6.1 Project-wide considerations

Choice of camera

In this project, we have chosen to use an iPhone in a single camera setup. This has been done for the following reasons: Firstly it is an accessible camera that a lot of people already have in their pockets, and is still able to produce videos at 240 FPS at a reasonable resolution. Secondly, the radar can communicate with it, which made it easy to incorporate with the radar data. We could have chosen a less accessible high-FPS camera or a multiple camera setup. This would however not be a good solution when considering an end user. If the results of the project are used in a product, it would be more expensive and possibly more complicated to set up.

Globally fitted motion model

In our project, we have utilized the approach of detecting the anchor point individually in each frame and afterwards fitting a model to all of our detections. The alternative to this approach would be to detect the anchor point in one or more frames, and use these frames to guide the method in subsequent frames to where we expect to find the anchor point. This could be done with methods such as Kalman filtering.

Furthermore, having the majority of the computations begin independent on a frame-basis allows for easier parallelization in an implementation.

To improve the estimate of the club head trajectory we could recompute the anchor points classified as outliers using the fitted model as a guide.

Optical distortion

We have chosen to perform our method on distorted frames. This is more computationally intensive and would typically not be done in a commercial product. In such a case computations would be done on distorted frames, and only the results of the computation such as points and angles would be corrected for optical distortion. It is however not entirely trivial to undistort an angle or a radius, and the expected result is not clearly defined. The reason for choosing to execute our method on undistorted frames was that we wanted to see what was possible with the data under optimal circumstances and computational complexity has thus not been a big concern.

Assumption of flatness for drivers

In our methods, we have assumed that club heads are flat. For wedges and irons, this is a correct assumption, but it does not hold for drivers, as they are slightly curved as mentioned in Section 2.3 on page 4. This causes problems in our method where we use the angles of the club face measured by the radar. That is because the radar is only able to measure the normal of the club face at the point of impact and not the pose of the entire club face. I.e., if a ball is hit close to the toe, the face angle be pointing more towards the golf player than it should. In order to correct for these effects, we need to know the local normal on the club face where the ball has been hit. The latter can be approximated with our method, which in turn implies that this correction could be implemented with our existing method. This could be done by computing the impact location and refining the face angles iteratively.

Rolling shutter

We have not compensated for rolling shutter in our project. If the iPhone is oriented in portrait mode, the sensor will roll from right to left. This will only cause the impact time to be found with the constant bias from the column of the initial ball position as the ball flight is primarily in the y-direction. Assuming the model fitted to the anchor point describes the observed pixel locations correctly, this model will implicitly include the rolling shutter. When the fit is evaluated at impact time, the bias will be present in the impact time. However, the fit will have been fitted to points near impact that also have the same bias in their measured times. Thus, because the anchor point and initial ball position are close together in the x-direction, rolling shutter will have limited effect. We expect the worst case distance between these two to be 100 pixels, which if the rolling shutter takes 100% of a frame, the error committed will be approximately 0.6 milliseconds.

This will cause an error in the time that we input to the anchor point model fit.

This is disregarding that rolling shutter can also change the measured θ_{shaft}, but we expect this to be even less significant.

6.2 Camera calibration

Dataset 1

An RMSE 7.82 pixels is not bad in this case because we are comparing pixel data with radar data, which is not very accurate in the $x - y$ direction. Because of that, this error is reasonable.

Dataset 2

The RMSE of 0.38 pixels is low and very acceptable, which mean that the only error in our calibration w.r.t. the radar's coordinate system is the same as the internal camera has.

6.3 Ball position

From Table 5.2 on page 60, it is evident that we are able to locate the golf ball successfully using both the Hough transform and RANSAC. The Hough transform performs slightly better than RANSAC, particularly for DS1. We believe this slight difference is because of the Hough transform implementation which utilizes the gradient direction and merges several close detected centers into one, which the RANSAC method does not.

The balls that are we are trying to locate are small, as Table 5.4 on page 61 shows. This implies that the annotations are bound to include some errors. In this light we find these results very satisfactory.

Figure 5.5 on page 61 shows that the RANSAC method is not very sensitive to the choice of threshold with which we regard a point as inlier for a given fitted model. All values between 0.1 and 3, seems to perform equally well.

Influence on subsequent results

The initial position of the golf ball influences many of the later results, which is why it is important to detect it precisely.

Influence on impact time

The golf balls move approximately 35 pixels per frame right after impact. This means that a detection error of 1 pixel on the ball position will introduce an error in the computed impact of approximately 2.9% of a frame. At 240 FPS this corresponds to approximately 0.1 milliseconds. Compared with the contact time between club and ball which is approximately 0.5 milliseconds (Cochran and Stobbs 1968), the sub-pixel errors on the position that we have are negligible, in the context of impact time.

Errors in the detection will also influence the vector $\mathbf{v}_{impact \to ap}$. If r_{mm} and r_{px} is the radius of the ball in mm and pixels then a one-pixel difference in the detected radius will result in a

$$\frac{\frac{r_{mm}}{r_{px}} - \frac{r_{mm}}{r_{px}+1}}{\frac{r_{mm}}{r_{px}}} = 1 - \frac{r_{px}}{r_{px+1}},$$

change in the mm per pixel factor. For DS1 this percentage is 11.9%, for DS2n 9.7% and for DS2z 7.0% using the average ball radii found in Table 5.4 on page 61. Hence, if the camera is placed closer to the ball, whereby we obtain more pixels per mm, the errors in the detected radius become less influential. From Table 5.2 on page 60 we furthermore see that we predict the radius with greater accuracy for DS2n and DS2z. The higher RMSE for DS1 is primarily due to a bias, but that bias might just as well be coming from the annotations.

Influence on impact vector

Errors in the position of the ball also affect the $\mathbf{v}_{impact \to ap}$ vector. If the hitting plane at impact time is parallel with the image plane ($\theta_{face,d} = \theta_{loft,d} = 0$) then a one-pixel detection error on the position affects $\mathbf{v}_{impact \to ap}$ with

$$1 \text{ pixel} \cdot \frac{r_{mm}}{r_{px}},$$

if we do not consider the position of the camera relative to the ball. The golf ball has radius 21.35 mm and using the average radii we find that this error is 2.9 mm, 2.3 mm and 1.6 mm for the three datasets. The assumption that the hitting plane is parallel with the image plane rarely holds. Generally $\theta_{loft,d}$ is much larger than $\theta_{face,d}$ which is close to zero. A better size estimate of the error in the y-direction is

$$\frac{\text{error}_{y,mm}}{\cos(\theta_{loft,d})},$$

where $\text{error}_{y,mm}$ is the detection error in the y-direction. Suppose there was only error in the y-direction and $\theta_{loft,d} = 45°$, which is a typical value for the irons in our datasets, then adjusting the former found error estimates we get 4.1 mm, 3.3 mm and 2.3 mm for DS1, DS2n and DS2z respectively. These values show that detecting the correct position is very important as small errors correspond to relatively large errors in $\mathbf{v}_{impact \to ap}$. If the camera is closer to the ball we see that the error in the detected position has less influence on $\mathbf{v}_{impact \to ap}$. However, the position is also more error prone when the ball is larger.

6.4 Impact time

Figure 5.7 on page 63 shows the RMSEs on the impact time errors for each of the three datasets, as functions of the number of points in the track we have used to fit the model. We stress that this is not the true error since we have no ground truth, but only an estimate of the magnitude of the error, based on annotations in the frames after impact where we do know the true time.

On all three datasets, the method based on a linear flight assumption performs better than the one with a quadratic term included as seen in Figure 5.7(a) on page 63. For DS2z general performance of the quadratic model is greatly affected by a few errors as seen in Figure 5.7(b) on page 63. Using the approximate impact time from the radar data one could detect these as outliers and thereby render the stroke unusable. If we remove all strokes where the error is greater than 1 frame, which are the ones we expect to be able to detect with the radar data, we get the RMSEs shown in Figure 5.7(c) on page 63. These RMSEs are calculated over all three datasets. We expect that the result of this graph is how our methods generalize to new data given that outliers can be detected and removed using radar data.

For the linear model, there is not much gain by non-linearly optimizing the parameters. Only for 30 fitted points and above we get a small boost, and it even gets a little worse for fewer points, but we deem this to be a coincidence as the fit has an objectively lower error w.r.t. the points the model is fitted on. The linear model starts to become worse when we use more points in the fit, which was anticipated. For the quadratic model, the results are a slightly better with non-linearly optimized parameters, specially when few points are used to fit the model. From fifteen points and above the benefit is very small. The graph shows that the linear model gives better results than the quadratic model. We find this surprising.

Because we are looking in a very short time span, the effect that gravity has on the flight path is limited. Thus, the quadratic part of the model might also describe other factors than gravity. Since the tracked positions of the ball are bound to include some error, the quadratic term might make the model better at fitting to these errors, which will not generalize for points outside the track. We believe that this is part of the explanation as to why the quadratic model has worse performance. Furthermore, the linear model uses the distance to the initial ball instead of using $p_x(t)$ and $p_y(t)$ directly, as the quadratic model does. We believe that this makes the linear model more robust towards small errors in the detected ball positions.

The linear model with non-linearly optimized parameters has the lowest RMSE of 0.12 frames when using 28 points to fit the model, however, there is little difference in the range 20 to 40 points. The lowest RMSE for the quadratic model with non-linearly optimized parameters is 0.15 frames when using 18 points for the fit. This model is more sensitive to the number of points fitted than the linear model. At 240 FPS these values correspond to 0.5 milliseconds and 0.63 milliseconds. We find this acceptable when considering the approximate contact time between club and ball of 0.5 milliseconds.

Sensitivity to time before first tracked ball after impact

The computed impact time is sensitive to how fast we are able to start tracking the ball after impact. Figure 5.8(b) on page 64 shows how the impact frame is affected if we are not able to track the ball immediately after impact. Clearly, the computed impact time becomes less precise, if we have not tracked the ball in the beginning of the trajectory. From Figure 5.8(a) on page 64 we see that we are not always able to track the ball immediately after impact. Because of this, we can expect the impact time to be less accurate for these strokes. We see that in DS2n and DS2z tracks start later than in DS1. This is because the cameras than in DS2n and DS2z are placed closer to the ground than in DS1, as shown in Figure 3.2 on page 19, resulting in the golf club blocking the ball for multiple frames in some cases.

6.5 Anchor point detection

From Figure 5.9 on page 65 we see that our method to detect the anchor point works well in the presented cases. In column one and three, we can tell by the thinness of the shaft in the segmentation, that the shaft is only present because we have forced the expected shaft line to be segmented as club. Even though we have detected the orientation of the shaft poorly in the first case, we have still found a relatively satisfactory anchor point, but this could be a coincidence. The fifth case is a challenging case with the club head being difficult to see in both the color and difference image, but both the segmentation and anchor point detection have produced satisfactory results.

Empirical probability density functions for difference images

We expect that the empirical PDFs from the annotated difference images will generalize to new data, as difference values should have the same range. This is implicitly shown by them working well on our data, as the annotated difference images only come from DS1. The assumed independence between these and the GMMs for the color values is most likely not true. However, it is a simple method of combining information from both the difference data and color data, which seems to work well.

6.6 Anchor point interpolation

We have tried out various models that are able to describe the motion of the anchor point. We have omitted some strokes from the model fitting because of missing data. This is not a problem as we have not removed the strokes because of bad results, but only because our method was not able to run on these strokes.

From Figure 5.12 on page 67 we can see that the best performing physical model is the pendulum with quadratic length. Because of this, we use the inliers from this to fit the smoothing spline, which gives a better result. This is not entirely surprising as the smoothing spline is a good fit locally, so points far from the impact have little effect on it. The worst performing model is the pendulum with quadratic length and fixed swing plane. This could be caused by inaccuracies in the swing plane measured by the radar or a not sufficiently good camera calibration. We suspect it is a combination of both.

We started out by fitting the models to the eighteen frames as they cover the lower part of the swing, where we expect the models to fit well. The eighteen frames of a swing are shown in Figures 4.2 and 5.13 on page 25 and on page 68. When we lower the number of frames we're fitting to, the results for the pendulum with fixed length become drastically better as we are looking at a smaller interval. We expect this to be because the assumption of fixed length is better near impact, but primarily because the anchor point is more consistent with a physical point near impact.

Interpolation of shaft angle

Using different degrees of a polynomial to interpolate θ_{shaft} as seen in Figure 5.15 on page 69 we are able to achieve a reasonably low RMSE. The lowest RMSE of 1.06° is achieved at a degree of eight, however, we determine that a degree of four is better to interpolate with, as this will be more robust in the case of very few inliers and it has a similar RMSE.

In order to fit the polynomial, we use the inliers from the anchor point RANSAC model fit. We could have fitted both models simultaneously and used distances to both models to evaluate our inliers since both θ_{shaft} and the anchor point should agree with their fitted models for a given consensus set. We have however not done this because it is simpler to compute as it has only one threshold, and performs acceptably well as is illustrated by the RMSE of the interpolated angle.

It should be noted that this error is not the error between the actual θ_{club} and the interpolated value, but only a value of θ_{club} found by our method. Because of this, we expect the RMSE to be slightly higher than if we had compared with an annotated angle.

6.7 Anchor point vectors

As mentioned in Section 4.7 on page 57, variations in $\mathbf{v}_{impact \to ap}$ correspond to variations in $\mathbf{v}_{ap}$. From Table 5.16 on page 70 we see that these variations are small. The best method uses the smoothing spline for interpolation of the anchor point, and only considers the dynamic lie angle for the hitting plane homography. We find the latter surprising as the loft angles for irons and wedges are considerably different from 0°.

As mentioned one stroke is removed before computing these results, since the corresponding $\mathbf{v}_{impact \to ap}$ is more than 20 cm long. The cause of this is a very bad estimate of the impact time which is several frames off.

From Tables 5.17 to 5.19 on page 70 and on page 71 we see that the variations are smaller for the drivers in DS1, than the other two datasets. We suspect it stems from the fact the cameras in the two other datasets are placed lower which makes drivers, that have big club heads, obscure the view of the flying ball initially, which decreases the accuracy in the computed impact time as discussed in Section 6.4 on page 81. The tables show that DS2n and DS2z both are best on the iron.

We expected that the results from DS2n and DS2z would have been better than the results on DS1, since the cameras were placed closer to the golf strokes and the videos, therefore, has less mm per pixel. The results, however, do not support this expectation.

Anchor point clouds

The anchor point clouds shown in Figure 5.20 on page 73 tells us where $\widetilde{\mathbf{v}}_{ap,observed}$ is located when seeing the clubs from the front, and gives an idea about the magnitude and direction of the bias terms $\boldsymbol{\beta}$. For the irons, this corresponds very well with how we have defined the anchor points. For the wedge, there is a large bias in the y-direction. Wedges have the largest dynamic and static loft angles, and we believe the bias comes from the fact the loft angle is omitted in the hitting plane homography. Figure 5.21 on page 74 supports this statement, as we see that $\widetilde{\mathbf{v}}_{ap,observed}$ is pretty close to where we would expect, which in turn implies that $\boldsymbol{\beta}$ is small. However, the point cloud is more spread out for this model when comparing the numbers in Table 5.22 on page 71 with the corresponding numbers in Tables 5.17 to 5.19 on page 70 and on page 71.

For the driver, and other drivers, there seems to be a small positive bias in both directions. We cannot conclude why this is based on our datasets.

We observe that there is a general tendency for the $\widetilde{\mathbf{v}}_{ap,observed}$ to be located with the same offset, $\boldsymbol{\beta}$ from the true anchor point for each club type. This could indicate that the $\boldsymbol{\beta}$s could be compensated for by having computed approximate values of them for each type of club.

Using only every second anchor point

Table 5.23 on page 72 shows the performance of our method when we only use every second anchor point. This is done in order to simulate how our method would perform on a video with only 120 FPS. We are pleasantly surprised by how little the performance decreases. The errors committed are still in ranges where we expect them to be useful in an application. The only problem with our method is that we do not truly emulate lower frame rate video because we have still used all frames of the ball track to compute the impact time. We do, however, expect that the effect on the impact time from a lower FPS video will be relatively small.

6.8 Impact position

The predicted positions in Figure 5.24 on page 76 shows that it is possible with our method to predict the impact position on the club head with reasonable accuracy. Tables 5.17 to 5.19 on page 70 and on page 71 show that the results are more accurate in the horizontal direction than in the vertical. This is a nice attribute as a golfer is often more interested in the horizontal impact position than the vertical.

CHAPTER 7

Conclusion

We have analyzed videos of golf strokes recorded with an iPhone paired with data obtained using a Doppler radar.

We have devised and implemented methods for locating the initial ball position, determining the time of impact, determining the shaft angle and estimating the impact point between the club and ball.

We have shown that the impact time between club and ball can be estimated using the ball trajectory and given an estimate of the error. We have also shown that the initial ball position can be accurately detected.

The results show that it is feasible to estimate the impact point between ball and club with a useful accuracy without using markers, based on the data, by determining the club and ball trajectories. We expect the results of this thesis to be of great value to TrackMan A/S.

7.1 Recommendations for TrackMan A/S

Based on our analysis a reduction in FPS to 120 only worsens the results slightly. Depending on the desired accuracy 120 FPS could be a suitable frame rate. However, higher frame rates do yield better results.

We have not observed the expected benefit from having more pixels per mm, which leaves us unable to give any recommendations on pixels per mm required for a product.

We recommend investigating the height at which the camera is placed further, as we have seen considerable differences between datasets, we suspect are caused by this. A higher placed camera has a better view of the ball trajectory immediately after impact.

In general, we have not focused on the computational complexity or simplicity of our used methods. E.g. we suspect that the MRF could be replaced by a simple threshold and morphological operations.

APPENDIX A

Appendix A

A.1 Geometrical distance to second degree polynomial

Given a polynomial

$$f(x) = ax^2 + bx + c,$$

and a point (x_0, y_0) we want to calculate the perpendicular or geometrical distance between them. The squared distance is given by

$$\begin{aligned}\Delta^2 &= (x_p - x_0)^2 + (f(x_p) - y_0)^2 \\ &= (x_p - x_0)^2 + ((ax_p^2 + bx_p + c) - y_0)^2\end{aligned}$$

If we take the derivative with respect to x_p and set this equal to zero we get the equation

$$\begin{aligned}0 &= \frac{d\Delta^2}{dx_p} \\ &= 2(x_p - x_0) + 2((ax_p^2 + bx + c) - y_0)(2ax_p + b) \\ &= 4a^2x_p^3 + 6abx_p^2 + (2 + 4(c - y_0)a + 2b^2)x_p - (2x_0 - 2(x - y_0)b).\end{aligned}$$

The smallest real solution to this is the square of the geometrical distance. The equation can be solved either analytically or numerically.

A.2 Projected shaft angle

Suppose we have two points $\mathbf{x}$ and $\mathbf{r}$ in $\mathbb{R}^3$, where $\mathbf{x}$ projects to the anchor point of the club and $\mathbf{r}$ is a direction vector of the shaft. We do not know the depth of $\mathbf{x}$, so we let it be the unknown scalar s. $\mathbf{x} + \mathbf{r}$ should project to a point on the shaft. If we project this to the image plane we get

$$\begin{aligned}\mathbf{K}(\mathbf{x} + \mathbf{r}) &= \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} sx_1 \\ sx_2 \\ s \end{bmatrix} + \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \right) \\ &= \begin{bmatrix} s(fx_1 + p_x) + fr_1 + p_x r_3 \\ s(fx_2 + p_y) + fr_2 + p_y r_3 \\ s + r_3 \end{bmatrix}\end{aligned}$$

And the Cartesian pixel coordinates are thus

$$\mathbf{p}_1 = \begin{bmatrix} \frac{s(fx_1+p_x)+fr_1+p_xr_3}{s+r_3} \\ \frac{s(fx_2+p_y)+fr_2+p_yr_3}{s+r_3} \end{bmatrix}$$

We know that $\mathbf{x}$ projects to the anchor point.

$$\begin{aligned} \mathbf{K}(\mathbf{x}) &= \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} sx_1 \\ sx_2 \\ s \end{bmatrix} \right) \\ &= \begin{bmatrix} s(fx_1+p_x) \\ s(fx_2+p_y) \\ s \end{bmatrix} \end{aligned}$$

The Cartesian coordinates are

$$\mathbf{p}_0 = \begin{bmatrix} fx_1+p_x \\ fx_2+p_y \end{bmatrix}$$

From these we can create a direction vector of the shaft in the image plane

$$\begin{aligned} \mathbf{p}_d &= \mathbf{p}_1 - \mathbf{p}_0 \\ &= \begin{bmatrix} \frac{s(fx_1+p_x)+fr_1+p_xr_3}{s+r_3} \\ \frac{s(fx_2+p_y)+fr_2+p_yr_3}{s+r_3} \end{bmatrix} - \begin{bmatrix} fx_1+p_x \\ fx_2+p_y \end{bmatrix} \\ &= \begin{bmatrix} \frac{f(r_1-r_3x_1)}{s+r_3} \\ \frac{f(r_2-r_3x_2)}{s+r_3} \end{bmatrix} \end{aligned}$$

The slope of this line is

$$\frac{p_{d,2}}{p_{d,1}} = \frac{r_2 - r_3x_2}{r_1 - r_3x_1},$$

where we see that this is independent of s, meaning that the depth at which $\mathbf{x}$ is created along its line of projection does not influence the slope of the direction vector in the image plane.

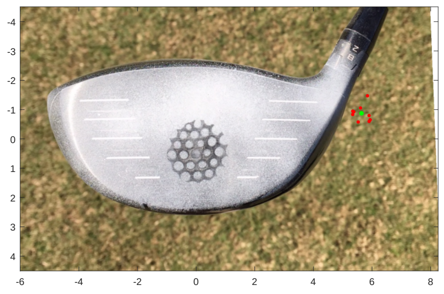
APPENDIX B

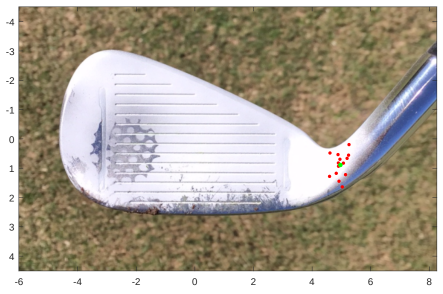
Appendix B

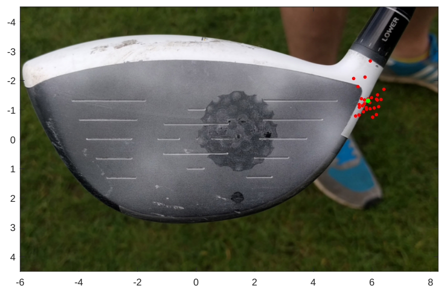
Images showing anchor point clouds on the clubs from the three datasets.

B.1 Anchor point clouds

Using only lie angle


(a) DS1 Driver 2


(b) DS1 Iron 1


(c) DS2n Driver 1

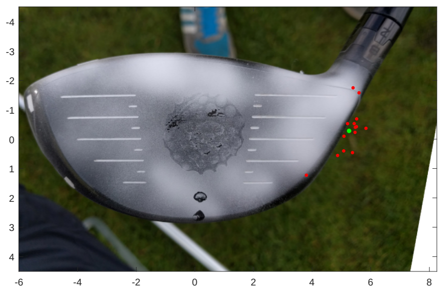
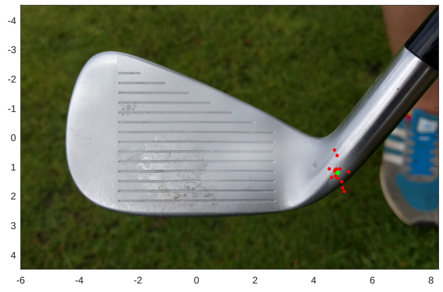
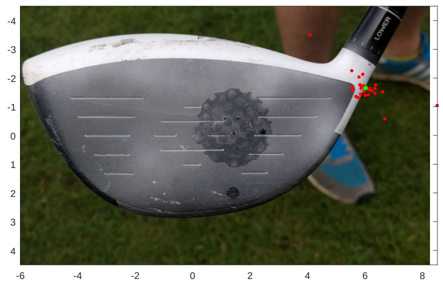
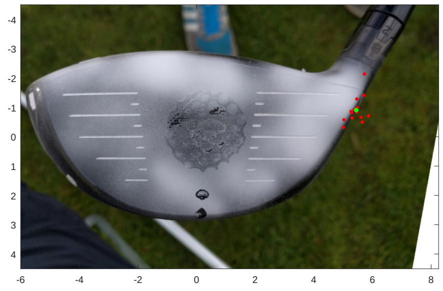

(d) DS2n Driver 3

Figure B.1: Only lie angle.


(a) DS2n Iron 2


(b) DS2z Driver 1


(c) DS2z Driver 3

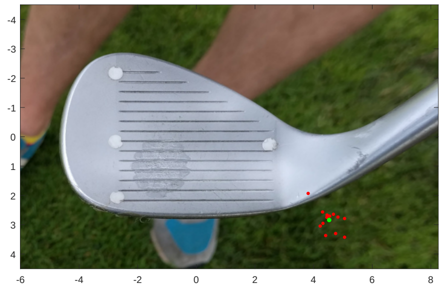
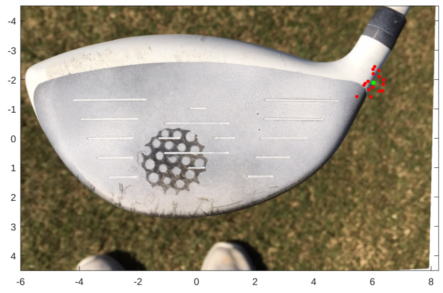

(d) DS2z Wedge 2

Figure B.2: Only lie angle.

Using loft and lie angle


(a) DS1 Driver 1

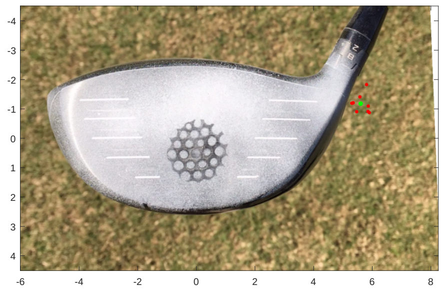
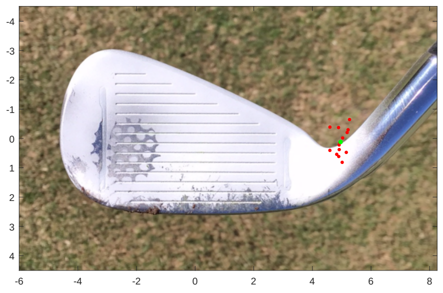
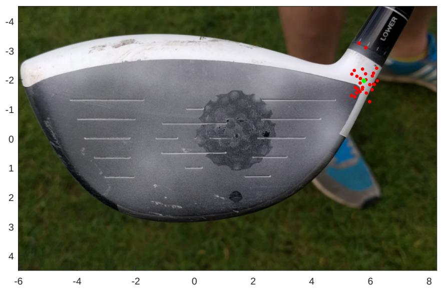
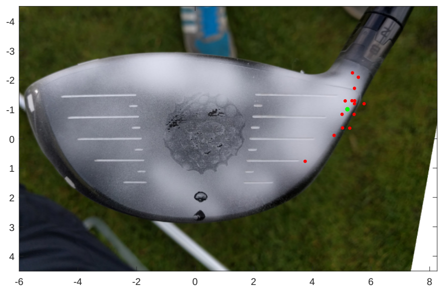
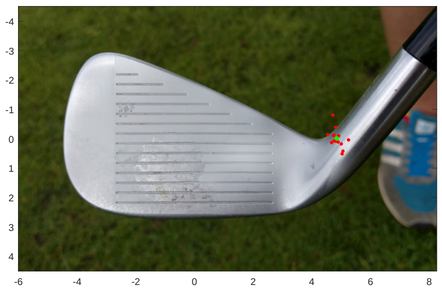

(b) DS1 Driver 2

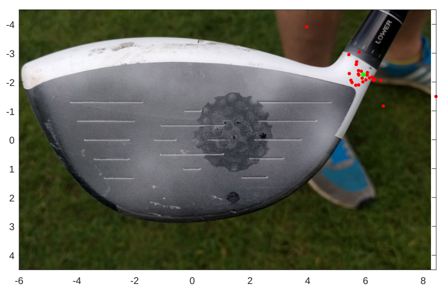
Figure B.3: Loft and lie angle.


(a) DS1 Iron 1


(b) DS2n Driver 1


(c) DS2n Driver 3


(d) DS2n Iron 2


(e) DS2z Driver 1

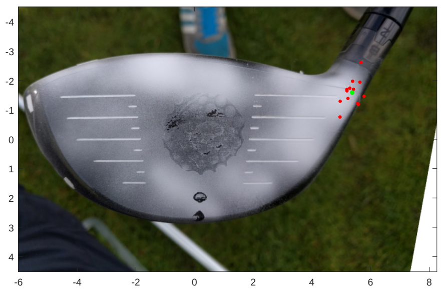

(f) DS2z Driver 3

Figure B.4: Loft and lie angle.

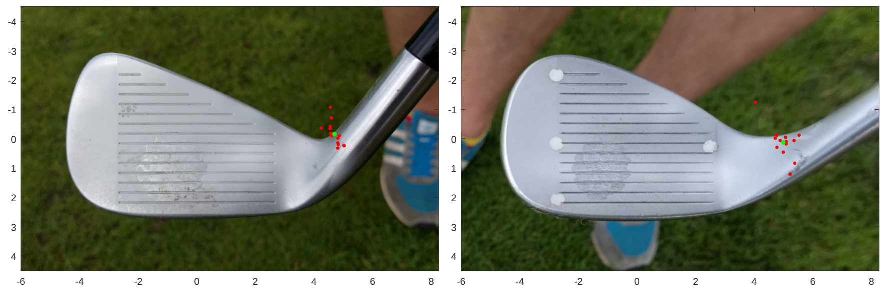
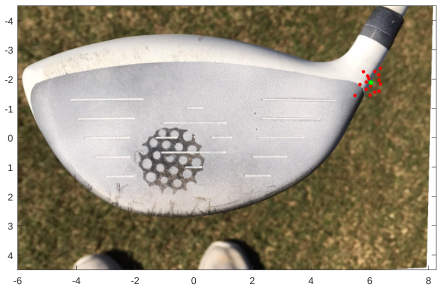
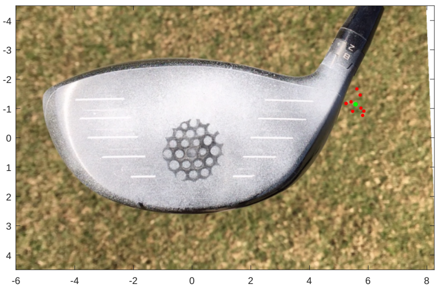
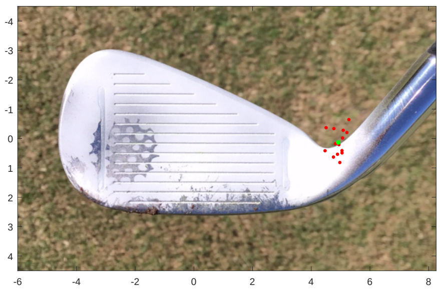

(a) DS2z Iron 2 (b) DS2z Wedge 2

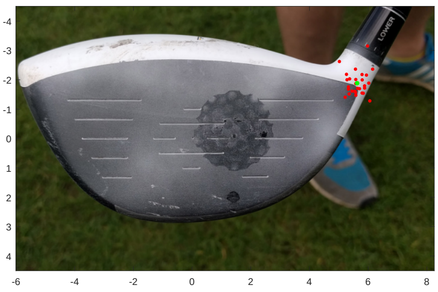
Figure B.5: Loft and lie angle.

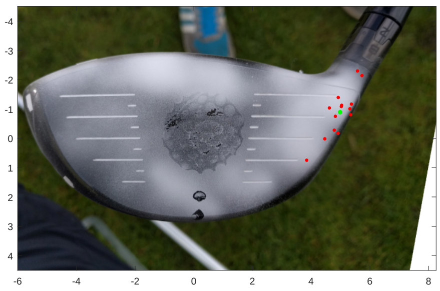
Using loft, face and lie angle


(a) DS1 Driver 1


(b) DS1 Driver 2


(c) DS1 Iron 1


(d) DS2n Driver 1


(e) DS2n Driver 3

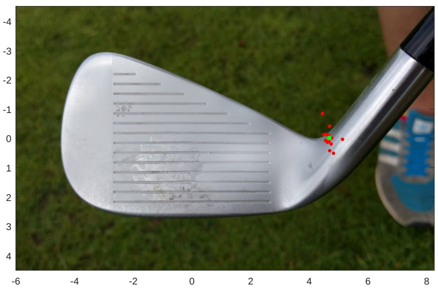
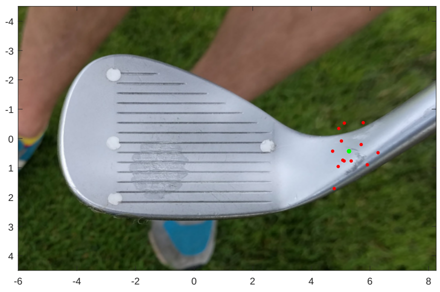
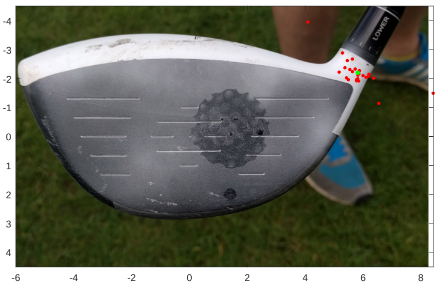
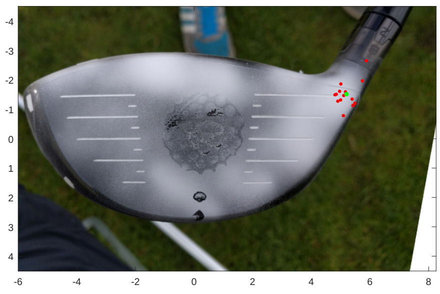
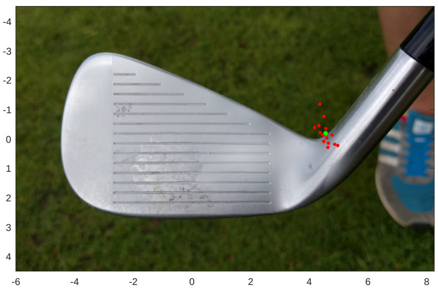

(f) DS2n Iron 2

Figure B.6: Loft, face and lie angle.


(a) DS2n Wedge 2


(b) DS2z Driver 1


(c) DS2z Driver 3


(d) DS2z Iron 2

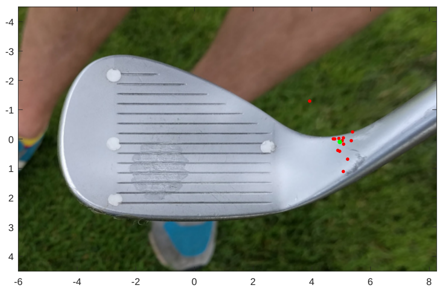

(e) DS2z Wedge 2

Figure B.7: Loft, face and lie angle.

Bibliography

Aanæs, Henrik (2015). *Lecture Notes on Computer Vision.* DTU Compute.

Anton, Howard and Chris Rorres (2010). *Elementary Linear Algebra: Applications Version.* John Wiley & Sons.

Björck, Åke (1996). *Numerical methods for least squares problems.* Siam, p. 408.

Bradski, G. (2000). “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools.*

Canny, John (1986). “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6, pp. 679–698.

Cochran, Alastair J. and John Stobbs (1968). *The search for the perfect swing.* Triumph books Chicago, Illinois.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the Royal Statistical Society, Series B* 39.1, pp. 1–38.

Duda, Richard O. and Peter E. Hart (1972). “Use of the Hough Transformation to Detect Lines and Curves in Pictures”. In: *Commun. ACM* 15.1, pp. 11–15.

Fischler, Martin A. and Robert C. Bolles (1981). “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6, pp. 381–395.

Foresight (2016). *Foresight HMT.* http://www.foresightsports.com/catalog/hmt-head-measurement. Accessed 10/08/2016.

Gao, Xiao-Shan, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng (2003). “Complete Solution Classification for the Perspective-Three-Point Problem”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 25.8, pp. 930–943.

Gears (2016). *Gears Golf.* http://gearssports.com/. Accessed 10/08/2016.

Gehrig, Nicolas, Vincent Lepetit, and Pascal Fua (2003). “Visual Golf Club Tracking for Enhanced Swing Analysis”. In: *British Machine Vision Conference, BMVC 2003, Norwich, UK, September, 2003. Proceedings*, pp. 1–10.

Groen, Pieter de (1996). “An Introduction to Total Least Squares”. In: *Nieuw Archief voor Wiskunde* 14.2, pp. 237–253. arXiv: `9805076 [math]`.

Hammersley, J. M. and P. E. Clifford (1971). “Markov random fields on finite graphs and lattices”. In: *Unpublished manuscript.*

Hartley, Richard and Andrew Zisserman (2004). *Multiple View Geometry in Computer Vision.* Second. Cambridge University Press.

Hough, Paul V. C. (1962). *Method and means for recognizing complex patterns.* US Patent 3,069,654.

Jorgensen, Theodore P. (1999). *The physics of golf.* Springer Science & Business Media.

Kolmogorov, Vladimir and Ramin Zabih (2004). “What energy functions can be minimized via graph cuts?” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2, pp. 65–81.

Levenberg, Kenneth (1944). “A method for the solution of certain non-linear problems in least squares”. In: *Quarterly Journal of Applied Mathmatics* II.2, pp. 164–168.

Li, Stan Z. (2009). *Markov Random Field Modeling in Image Analysis.* 3rd. Springer Publishing Company, Incorporated.

Lourakis, M.I.A. (Jul. 2004). *levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++.* [web page] http://www.ics.forth.gr/~lourakis/levmar/. [Accessed on 26 Jun. 2016.]

Marquardt, Donald W. (1963). “An algorithm for least-squares estimation of nonlinear parameters”. In: *SIAM Journal on Applied Mathematics* 11.2, pp. 431–441.

MATLAB (2016). *version 9.1.0.390522 (R2016b) Prerelease.* Natick, Massachusetts: The MathWorks Inc.

Peng, Tao (2005). *Detect circles with various radii in grayscale image via Hough Transform.* Matlab Central File Exchange. Accessed 11/08/2016.

Qualisys (2016). *Qualisys.* http://www.qualisys.com/applications/sports/golf/. Accessed 10/08/2016.

Tuxen, Fredrik (2014). *3D Club Model.* Unpublished.

Woodward, Alexander and Patrice Delmas (2005). “Computer Vision for Low Cost 3-D Golf Ball and Club Tracking”. In: *In Proceedings of the Image and Vision Computing New Zealand Conference (IVCNZ)*, pp. 293–298.