KU LEUVEN

Visual Query of Soccer Actions

Shuo Sun

Thesis submitted for the degree of
Master of Science in Artificial
Intelligence, option Big Data
Analytics

Thesis supervisor:
Dr.Jesse Davis

Assessors:
Dr.Clément Gautrais
Dr.Robin De Croon

Mentors:
Dr.Clément Gautrais

Academic year 2020 – 2021

Preface

I would like to thank everybody who kept me busy the last year, especially my promoter and my daily advisors. I would also like to thank the jury for reading the text. My sincere gratitude also goes to my friends and the rest of my family.

Shuo Sun

Contents

Abstract

Sports analytics is transforming both how sports is played and how we interact with it. Various analytical tools have been created for "The Beautiful Game" soccer. The ability to collect an unprecedented amount of soccer data posed unprecedented challenges to soccer analytics. We present a web application for users to query a soccer action trajectory of interest by drawing on an interactive web interface. In order to develop the searching algorithm behind the web application. We first conduct experiment on accuracy in trajectories retrieval for different distance metrics. Then, we conduct another experiment to optimize the searching algorithm in trajectory retrieval. The result is an web application that can search for a similar trajectory effectively and efficiently.

List of Figures and Tables

List of Figures

List of Tables

Chapter 1

Introduction

Technological advancement in sports, especially wearable sensing devices and GPS location tracking systems, have made it possible to collect vast amounts of data at an unprecedented level. Sports players' spatial temporal information is being recorded in millisecond interval and centimeter precision. Those data are the oil to fuel sports' analytical engine. Sports analytics produces analytical tools of different complexity that discover various kinds of knowledge and insights. Those tools along with the knowledge and insights they generated are reshaping the landscape of the sports world.

Soccer, "The Beautiful Game", is no exception. Soccer analytics is transforming the experience of many stakeholders in the soccer industry. Soccer coaches use historical game data to find tactics and improve player performances. Soccer scouts use player ratings to find potential players. Soccer fans read visualizations such as info-graphics to keep track of their favorite team and players. Their interactions have been enriched by a growing number of soccer analytics apparatuses.

We develop a web application to add another dimension to how various stakeholders experience soccer games. With the web application, we present in an intuitive way to query soccer trajectory data. Users can draw a trajectory of interest in our interactive interface that is inspired by the coach tactical board. Our application will quickly return the most similar trajectories from the database. This easy and novel way to query soccer trajectory data can be an ideal ground for tackling a wide range of soccer analytics problems and benefit a variety of stakeholders in the soccer industry.

Structure of the master thesis

Chapter 2 gives an review of the related work. It stared from a emerging and board field called visual movement analystics. Then lead to an introduction of Sports Analytics and followed by an overview of soccer analystics. Finally, it narrowed down to the topic of soccer trajectories searching. Our work starts follow the footstep of previous researchers and starts from here.

Chapter 3 introduce the rational behind the selection of the data set and the challenges we face analyzing it.

In chapter 4 and 5, we explained the mathematical tools and algorithmic apparatus we used to tackle the various challenges in our study.

Chapter 6 explained the two experiments we did. The first experiment is to find the best distance metrics in order to accurately retrieve trajectories. The second experiment aim to reduce the run-time as much as possible by using optimization strategies.

Chapter 7 introduces the web application we made and the architecture behind it.

In Chapter 8, we discuss some of the limitations of our study, and foreach limitation, we propose a way to address and improve our work in future research.

In the last chapter, chapter 9, we concludes the thesis. We summarize the core result of the thesis.

Chapter 2

Related Work

There is a large body of scholarly work related to visual query of soccer actions data and its application. In this chapter, we first briefly review work on movement analytics, followed by a discussion on general soccer analytics. Afterwards, we detail methods on soccer trajectories searching. Finally, we introduce the novel aspects of our approach compared to other work in the area.

2.1 Visual Movement Analytics

Visual movement analytics is an evolving research area focusing on the discovery and visualization of knowledge and information by the analysis of movement data. It is a relatively recent trans-disciplinary research area that emerged from the fields of cartography, time-space geography and information visualization. It leverages the computational power of modern computer technologies and the visual and creative analytical thinking skills of human analysts.

The movement data comes in all shapes and sizes: some are discrete objects whose shape and size are fixed, for example, [4] represent multiple trajectories of vessel transportation. Others are continuous objects that can change their shapes and sizes, such as the ocean currents underneath the vessel.

Visual movement analytics gained popularity due to the rapid growth of information communication technology (ICT). Technological advancement in mobile computing and GPS devices made it possible to acquire detailed movement data with millisecond precision. By using visual analytics tools and data mining algorithms, interesting patterns of movement data can be found and acted upon to improve the goal of movement. For example, visual movement analysis is being used in traffic jams, urban planning, animal migration and sports analytics [2] [3].

2.2 Sports Analytics

Sports analytics provides new and exciting opportunities to facilitate the exploration, understanding and communication of sports data. Not only successful sports analytics often result in the improvement of performance of a particular sport team, but it also leads to a significant economic outcome. Done appropriately, sports analytics can be a competitive advantage for the sports team to do better with less time, even on a tight budget.

Perhaps the most famous sports analytics case is the Oakland Athletics' usage of baseball data statistics to build a successful baseball team on a minimal budget in 2002. Dallas Mavericks was also cited to win the National Basketball Association Championship in 2011 by using data driven decision in professional competition.

Since then, sports analytics has evolved with an unprecedented level of sophistication in data collection, analysis methodology and especially the visualization of sport data. The New York Time, Sport Reference family of sites and Fangraphs are some of the best free resources on sports data info-graphics and interactive visualizations. Perin et al.[5] made a comprehensive survey on the state of the art of sports data visualization.

Any sports can benefit from utilizing sports analytics. There exists a soaring growth of sports analytics work being done in most, if not all, traditional sports.

Goldsberry [12] assembling both spatial and visual analytics to quantify, visualise and communicate NBA performance. They evaluated every NBA player on their spatially aware shot site performances and identified the NBA players with the most potent in spatial shooting.

Pingali et al. [11] developed a system called LucentVision to automatically extract accurate motion information of the tennis players and the trajectories of the tennis ball during a game in real-time. Their work offers rich visualization of player performance and gives new insight into the style and strategy of the players. Their tennis visualization system has been used extensively in broadcasts of international tennis tournaments since 1998.

SnapShot, a visualization system for ice hockey intelligence gathering process, designed by Pileggi et al. [9] provided an advantage to make strategy and improve performance for competitive ice hockey.Legg et al. [31] have built an application called MatchPad running on a tablet computer to visualize and analyze Ruby data in real time. Their system uses an interactive glyph-based visualization to help the Welsh Rugby Union examine actions and events in detail and help in the decision making during the Rugby World Cup 2011.

The Berlin 2016 marathon visualization, built by tkw is an example that individual users can see their progress with all their other marathon runners on a

single map. Conventionally, participants in long-distance cycling events keep a ride diary and use photographs as a way to record their performance and construct a narrative during a long event. Similarly, Jo Wood provided an alternative way to help long-distance cycling riders plan their strategies by developing an web application that automatically visualizes event data during a cycling event [8].

Soccer, one of the most popular sports in the world, is a popular sport for sports analytics. They will be reviewed in the next section.

2.3 Soccer Analytics

Soccer analytics is only as good as the soccer data that is available for analysis. The traditional data was collected by hand on a piece of paper.

The first of such notational analysis systems for soccer was created by Charles Reep in the early 1950 [19]. He also co-authored the first scientific paper to use statistics in soccer analysis in 1968. 2,194 soccer matches were annotated by hand from 1950s to 1990s [14].

Since then, soccer analytics has made numerous progress thanks to the advancement in sensing technology. Cameras were installed around the soccer pitch to track the movement of the players and balls, and a GPS tracking system was implemented to record the exact location information of every movement. This made a wide range of different soccer analytics possible, they can be classified into three main categories based on the complexity of underlying algorithms and level of knowledge gained: basic, medium and advanced.

Basic soccer analytics is simple to apply and easy to understand. It mainly consists of pure measurements and simple statistics. The most popular example is using match statistics to discriminate among winning, drawing and losing teams in soccer. Another popular case is using heat maps to visualize a player's location information during a game or overlay location information of many games together to have an overview of a player's location preferences.

Medium soccer analytics requires high level statistics analysis technique or aggregation of data beyond a single player. Soccer pass graphs made by the visualization interface named "SoccerStories" built by Perin et al. [18] is an example in the medium category. SoccerStories provides a series of soccer passes visualization and an overview of game phases interface. Another example is the network pass graph made by Pena et al. [32], their network analysis on soccer passes graphs reveal the relative importance of each player in a soccer game. It is very interesting to soccer coaches as it shows the effect of removing players from a soccer game.

The most complex algorithm and highest information gain resides in the advanced category. The DTAI lab in KU Leuven has made exciting work in high level soccer analytics. This includes the algorithm made by Decroos et al. [33] that can automatically find the tactics of soccer players in spatio-temporal soccer match data. In another work [16], they vectorized soccer players using machine learning systems to automatically characterize the player style of a player that can be interpreted by human experist. These work represent advanced movement analysis in soccer matches that can automatically generate knowledge by complex algorithms and provide insight to the human viewer.

In the next section, we are going to review another set of research that also belongs to the advanced category: soccer trajectory searching.

2.4 Soccer Trajectories Searching

Soccer trajectories searching's goal is to search for similar soccer trajectories from all the trajectories from the database. There are generally two scenarios in this area of research.

The most popular approach is to provide a pre-selected soccer movement pattern, usually represented by soccer trajectories, as input and ask the searching algorithm to find the most similar trajectory. Beernaerts et al.[24] has developed a novel spatio-temporal qualitative calculus called qualitative trajectory calculus (QTC) as the distance metric that takes into account the speed information when comparing similar movements. This is very useful especially when the trajectories have different temporal lengths. Stein et al[27] went beyond the spatio-temporal features like the shape of the target trajectories, they take into account the information of players, events as well as high level context such as pressure into the process of similarity search. Their approach was tested on a data set containing around 60 matches over a complete season from two teams. Their approach is limited to searching for the same two teams. Therefore, there might be performance problems when scale up to more matches and teams.

An alternative approach is to use clustering algorithms to automatically identify repetitive patterns of players' movement data. The work of Yuan et al.[29] has utilized the clustering algorithms to not only discover movement patterns in individual players but also movement patterns in a group of players. Their experiment was being done in a data set containing 20 complete games.

Lin Shao et al [26] went one step further, they have developed a system that can search for trajectories using an interactive search interface. Their system allows the user to sketch a trajectory of interest and search for a movement pattern that is similar to the given sketch. However, their approach was limited to only a single match which limited the large scale usage for a wider audience.

Unprecedented amount of soccer data presents the exciting opportunity to search soccer trajectories on a large scale. It also comes with unprecedented challenge to tackle the searching problem effectively and efficiently on big soccer databases.

Our current study is an attempt to tackle this unprecedented challenge. Our study focuses on visual interactive searching for soccer trajectories within the area of visual movement analytics and sports analytics. We develop a new trajectory searching system that effectively and efficiently search on a large database. It is based on the legacy of existing work of soccer analytics described above. We introduce the database we use and the challenges we face in the next chapter.

Chapter 3

Data and Challenges

In this chapter, we introduce the rational behind the selection of the data set and the challenges we face analyzing it.

3.1 Data set

Soccer competitions data comes in different shapes and sizes. We choose the event stream data as the source for this study as it provides the both spatial and temporal of specific soccer actions (e.g., shots, passes, dribbles) that happened during a soccer competition. Event Stream data is being used to store soccer competitions data from many commercial data providers, such as Opta,StatsBomb and Wyscount. We have used a data set that is freely available from StatsBomb.Our data set consists of a total 628 games in five different soccer competitions in around the world: the FIFA World Cup, the Women's World Cup, the FA Women's Super League, Spanish La Liga and National Women's Soccer League from the United States.

The event stream data has been converted into the SPADL(Soccer Player Action Description Language) format[17]. SPADL is a simple and human-interpretable data structure for existing event stream data. We choose SPADL as the format of choose because it stripped away all irrelevant information and optional information snippets. It focuses on accurately describing soccer actions during the game that facilitates data analysis.

	game_id	period_id	time_seconds	timestamp	team_id	player_id	start_x	start_y	end_x	end_y	type_id	result_id	bodypart_id	action_i
0	69139	1	3.0	00:00:03.288	214	26105	52.058824	34.430380	51.617647	32.622785	0	1	0	
1	69139	1	3.0	00:00:03.655	214	26243	51.617647	32.622785	54.176471	34.602532	21	1	0	
2	69139	1	3.0	00:00:03.775	214	26243	54.176471	34.602532	72.352941	23.843038	0	1	0	
3	69139	1	5.0	00:00:05.358	214	26110	72.352941	23.843038	73.058824	26.769620	21	1	0	
4	69139	1	6.0	00:00:06.567	214	26110	73.058824	26.769620	21.529412	0.516456	0	0	0	

FIGURE 3.1: SPADL data set

An example of the data in our data set can be found in figure 3.1. Each row contains (among others) the following 7 attributes:

1. **Game ID**: the ID of the game,
2. **Team ID**: the ID of the team,
3. **Player ID**: the soccer player who performed the action on the ball,
4. **Start X**: the x coordinate of the start location,
5. **Start Y**: the y coordinate of the start location,
6. **End X**: the x coordinate of the end location,
7. **End Y**: the y coordinate of the end location,

3.2 Challenges

Querying a soccer trajectory is a searching problem. Formally, our task can be defined as:

Given: (1) every position (x, y) of the soccer ball in the field during a passing l in all 628 games and (2) a drawing made by the user represent the trajectory he or she wants to search for.

Goal: quickly find the most similar trajectory match of all trajectories compared with the user draw trajectory.

Both the magnitude of the data set and the complexity of the task pose some challenges to our study:

1. **Trajectory definition**. What constitute a trajectory? This is the first and foundation to build a trajectory searching algorithm. It is an algorithmic challenge to construct every possible soccer trajectory and efficiently calculate the distance between all of them and the user input.
2. **Similarity metrics**. Various similarity metrics exist and each has its own advantages and disadvantages. Each similarity metric needs to be examined for suitability.
3. **Optimization strategies**. How can we optimize the searching algorithm in order to make it a fixable web application, especially to reduce the runtime.
4. **User experience**.The user experience should be smooth. This includes having a user-friendly, intuitive interface for drawing and a responsive interface with a low processing time.

We introduce how we tackle each of the challenges in the following chapters

Chapter 4

Trajectory and Its Similarity Metrics

In this Chapter, we first define what is a trajectory in our study and introduce four distance metrics as measurements we use for comparing similarity between two trajectories.

4.1 Trajectory Definition

One of the key challenges of our study is to properly define what constitutes a trajectory in a soccer game with thousands of location data points. Here we define one trajectory as the movement of the soccer ball from one location to another location under the control of players from the same team. In other words, all the actions that are performed on the soccer ball during the course of a trajectory should be from players of the same team. It can be that one player is dribbling the ball until an opponent from another team catches the ball or multiple players are passing the ball to each other but all the players that touched the ball need to be from the same team in order to constitute a trajectory.

We define the longest contiguous trajectory as the longest trajectory of the soccer ball, so under the continuous control of the same team. Let K_m^n denote the n-th action done by player m of team K in the longest contiguous trajectory L. The course of actions done by player 1 from team A dribbling the ball until an opponent from another team catches the ball can be denoted as $L =< A_1^1, A_2^1, A_3^1, \ldots, A_n^1 >$. The course of actions done by multiple players passing the ball to each other from team B can be denoted as $L =< B_1^1, B_2^2, B_3^3, \ldots, B_n^m >$. Any contiguous subsequence of the longest contiguous trajectory also qualifies as a trajectory. A contiguous subsequence is the subsequence such that points in the subsequence are consecutive in the longest contiguous trajectory. For example in a trajectory $L =< A_1^1, A_2^2, A_3^3, A_4^4 >$, the subsequence $L_i =< A_2^2, A_3^3 >$ is a valid trajectory but subsequence $L_j =< A_1^1, A_3^3, A_4^4 >$ is not a valid trajectory as the points in this subsequence are not consecutive in the

original trajectory L. A typical example of movements of soccer ball in soccer pitch

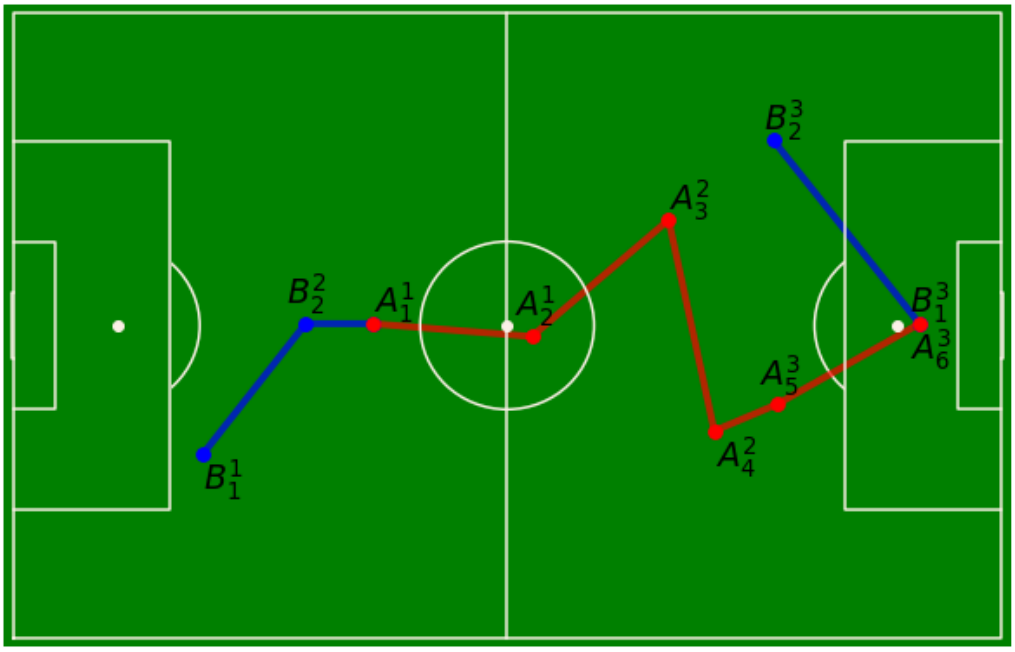

Figure 4.1: Trajectory definition. The red trajectory represents the trajectory of the soccer ball under the control of team A. The blue trajectory represents the trajectory of the soccer ball under the control of team B. The entire red trajectory, that is trajectory $L =< A_1^1, A_2^1, A_3^2, A_4^2, A_5^3, A_6^3 >$, is the longest contiguous trajectory of team A. Any contiguous subsequence of $< A_1^1, A_2^1, A_3^2, \ldots, A_6^3 >$ is a valid trajectory. In total, there are 15 trajectories out of trajectory L in this example.

can be found in figure 4.1. The red trajectory represents the trajectory of the soccer ball under the control of team A. The blue trajectory represents the trajectory of the soccer ball under the control of team B. The movement of the soccer ball was under the control of team B at the beginning, then changed to team A, and team A lost control of the ball to team B in the end. The entire red trajectory, that is trajectory $L =< A_1^1, A_2^1, A_3^2, A_4^2, A_5^3, A_6^3 >$, is the longest contiguous trajectory of team A. The superscripts $1, 2, 3$ represents three different soccer players from team A. The subscripts $1, 2, 3, 4, 5, 6$ represent the ordering of the actions within the longest contiguous trajectory. Any contiguous subsequence of $L =< A_1^1, A_2^1, A_3^2, A_4^2, A_5^3, A_6^3 >$ is a valid trajectory. In total, there are 15 trajectories out of trajectory L in this example. For example, $< A_2^1, A_3^2 >$ and $< A_2^1, A_3^2, A_4^2 >$ are both valid trajectories.

4.2 Similarity Metrics

In order to find a trajectory that is similar to a user drawing, we need to define what the "similarity between 2 trajectories" entails. It is clear that any suitable similarity metric must meet the following requirements:

1. The metric must quantify the similarity between two trajectories as a single real number. The metric must be such that a greater number means that the trajectories are "less similar" to each other. Therefore, the metric is often called a distance.

2. The metric should not only account for spatial, but also temporal differences. Indeed, trajectories are ordered sequences of events (i.e. soccer ball passings). The order is a crucial part of the information and the metric should account for that.

The similarity or distance between two distinguished data sequences must be compared in order to give a distance score. With the complexity of the raw data, the measurements are diverse. Base on the review of literature on trajectory similarity [28][29]. Four candidate similarity measurements have been selected for experimenting: Euclidean distance, Hausdorff distance, Fréchet distance and dynamic time warping (DTW) distance. We discuss each of them here.

4.2.1 Euclidean distance

Euclidean distance is one of the most widely used distance metrics in data mining and can be used for our soccer trajectory analysis. The similarity between two trajectories measured by Euclidean distance is simple. The time complexity of the Euclidean distance algorithm is $O(n)$, where n is the total number of points in the given trajectories. Its linear computational complexity makes it a very efficient tool for comparing a large number of trajectories. The Euclidean distance is sensitive to outliers. The Euclidean distance between two trajectory L_i and L_j with each the same length n in a p-dimensional space can be denoted as $D_{Euclidean}$.

$$D_{Euclidean}(L_i, L_j) = \frac{1}{n} \sum_{k=1\ldots n} \sqrt{\sum_{m=1\ldots p} (a_k^m - b_k^m)^2} \tag{4.1}$$

Since we consider trajectories in 2-dimensional space, the formula can be simplified as follows, where $x_{i,k}$ refers to the x-coordinate of the k-th point of trajectory L_i.

$$D_{Euclidean}(L_i, L_j) = \frac{1}{n} \sum_{k=1\ldots n} \sqrt{(x_{i,k} - x_{j,k})^2 + (y_{i,k} - y_{j,k})^2} \tag{4.2}$$

The disadvantage of Euclidean distance is that it requires the two trajectories to be the same number of data points. However in reality, two trajectories may not have the same number of points, which limits the use of this classic distance metric. In order to overcome the shortages of Euclidean distance, several other distance metrics are proposed.

4.2.2 Hausdorff distance

Hausdorff distance is a distance algorithm that measures how far two trajectories are from each other. This distance algorithm measures the distance of every point from either set to every point of the other set and finds the maximum value among all the distances. In other words, two trajectories are considered similar to each other by Hausdorff distance algorithm when every point of either trajectory is close to all points from the other trajectory. The Hausdorff distance of two trajectory L_1 and L_2 with can be denoted as $D_{Hausdorff}$ where $h(L_1, L_2)$ is the Hausdorff distance between L_1 and L_2, and $dist(a, b)$ represents the Euclidean distance between point a and point b from L_1 and L_2 respectively:

$$\begin{cases} D_{Hausdorff}(L_i, L_j) = max(h(L_i, L_j), h(L_j, L_i)) \\ where: h(L_i, L_j) = max_{a \in L_i}(min_{b \in L_j}(dist(a, b))) \end{cases} \tag{4.3}$$

The two distances $h(L_1, L_2)$ and $h(L_2, L_1)$ are the bidirectional Hausdorff distances between two trajectories. The Hausdorff distance is sensitive to outliers and also parameter free. The time complexity of the Hausdorff distance algorithm is $O(m * n)$ where m and n are the total number of points in both trajectories, respectively.

4.2.3 Fréchet distance

Fréchet distance is a distance metric that takes both the location and the sequential relationship of the points from two trajectories when calculating how far away they are from each other. This algorithm calculates the Euclidean distance of the points from two trajectories one by one. The Fréchet distance is the maximum value of the Euclidean distance among all the pairs from two trajectories.

A common intuitive definition of the Fréchet distance is the "person walking a dog on a leash" analogy. In this analogy, a person walks a dog on a leash and both the person and the dog having their own distinct trajectory. The Fréchet distance is then the shortest possible leash length that is sufficient to traverse the coupled trajectories from start to finish. For example, the Fréchet distance between two concentric circles of radius r_1 and r_2 is $|r_1 - r_2|$ because the shortest possible leash length in this example is reached when both owner and dog walk at a constant angular velocity around the circle with starting points at a Euclidean distance of $|r_1 - r_2|$ away from each other.

The Fréchet distance between two trajectories L_i and L_j can be denoted as $D_{Fréchet}$ and is defined as follows.

$$\begin{cases} D_{Fréchet}(L_i, L_j) = min||C|| \\ where: ||C|| = max_{k=1...K} dist(a_i^k, b_j^k) \end{cases} \tag{4.4}$$

Fréchet distance can deal with trajectories with different numbers of data points. Let us assume the number of points of L_1 is m and the number of points of L_2 is

n, then $K = min(m, n)$ and a_i^k and b_i^k are the k-th points from trajectory L_1 and L_2 respectively. $dist(a_i^k, b_i^k)$ is the Euclidean distance between the two points. Fréchet distance is sensitive to outliers as it only considers the maximum distance among all the distance pairs. The time complexity of the Fréchet distance algorithm is $O(m * n)$.

4.2.4 Dynamic time warping distance

Dynamic time warping (DTW) is a popular distance measure for comparing the similarity between two temporal sequences, even if they vary in speed. It does not require the two trajectories to have the same length, and aligns or "warps" every point from one trajectory to one or more points from the other trajectory in a non-linear way. It finds an optimal alignment between two trajectories with the minimum

"warping cost". The DTW distance of two trajectories L_1 and L_2 can be denoted as D_{DTW} and is defined as follows.

$$D_{DTW}(L_i, L_j) = \begin{cases} if : m = n = 0 \longrightarrow 0 \\ if : m = 0 || n = 0 \longrightarrow \infty \\ else : dist(a_i^k, b_j^k) + min \begin{cases} D_D(Rest(L_i), Rest(L_j)) \\ D_D(Rest(L_i), L_j) D_D(L_i, Rest(L_j)) \end{cases} \end{cases} \tag{4.5}$$

Here, m and n are the number of points of L_1 and L_2, respectively. $Dist(a_i, b_i)$ is the Euclidean distance between two points a_i and b_i. $Rest(L_1)$ and $Rest(L_2)$ represent the remaining trajectory segments after removing the start point of L_1 and L_2. The DTW distance is thus defined in a recursive way.

From the definition, we see that every point of the trajectory L_1 is matched with one or multiple points of the trajectory L_2 and the first and last point of L_1 must be matched with the first and last point of L_2. Additionally, the mapping of points between both trajectories must be monotonically increasing.The time complexity of the DTW algorithm is $O(m * n)$.

Due to its performance of finding the optimal alignments between two sequences, DTW has a wide range of implementations in any data that can be transformed into sequential data such as video, audio to graphics data. Applications utilizing DTW including automatic speech recognition, signature recognition and shape matching.

Chapter 5

Methodology

In last chapter, we defined our trajectories and how to compare them, In this chapter, we introduce the searching algorithm and the optimization to reduce its run-time.

5.1 Trajectory Searching

We search all trajectories of a longest contiguous trajectory by enumerating all possible contiguous subsequences with a sliding window of size two points, that is it only involves two adjacent points, from the beginning of the longest contiguous trajectory moving one point at a time until it reaches the end point. We increment the size of the sliding window by one each time it reaches the end point and repeat the process from the beginning again. The maximum size of the sliding window is the number of points of the longest contiguous trajectory. After the size reaches its maximum value, the algorithm moves to the next longest contiguous trajectory and repeats the process again. The whole process finished when it enumerated all of the 628 games in the database and reached the end of the database. The pseudo-code of the algorithm can be found in algorithm 1. By applying this algorithm to enumerate all 628 games, we identified 8 392 255 trajectories in total.

Algorithm 1 Search all trajectories.

```
LCT = None
number_of_games = n
min_length=2
for game in number_of_games do
    if len(LCT) < min_length then
        continue
    end
    for window_length in range(2, len(LCT) + 1 do
        for i in range(len(LCT)-window_length+1) do
            window = l[i:i+window_length]
            distance = Compute the distance here
        end
    end
end
return distance
```

5.2 Generate User Draw Approximation

The first step towards finding the best match trajectory in respect with the user trajectory draw is to simulate a user draw trajectory by generating a user draw approximation.

The user draw trajectory needs to fulfil three criteria: first, it needs to have a corresponding target trajectory in the database to search and compare for as the ground truth. Second, it needs to be automatically generated to repeat the process a large number of times, this also helps to eliminate individual bias in the process. Third, the generation process needs to be reproducible so the entire process can be reproduced and tested by other people independently. After carefully considering various options, we choose to approach this experiment by approximating the user draw trajectory using the existing trajectory from the database.

We randomly draw 1 trajectory from the database as the target trajectories that we want to search for. We then add Gaussian noise, with $mean = 0$ and $sigma = 1$, to each point (x, y) of the trajectory to mimic the corresponding trajectory drawn by the user, as can be seen in equation 5.1. We denote this Gaussian noise as $\mathcal{N}(0, 1)$.

$$(x_{generated}, y_{generated}) = (x + \mathcal{N}(0, 1), y + \mathcal{N}(0, 1)) \tag{5.1}$$

In addition, if the Euclidean distance of any two consecutive points in a single trajectory is less than 1 meter away, we randomly omit one of the two points that are adjacent to one another. The reasons for doing this are twofold. First, consecutive points that are within 1 meter away from each other have little significance in determining the overall characteristic of the trajectory, especially its spatial length and shape. Second, it reduce the number of computation to compare the distance

between two trajectories therefore reduce the run-time.

An example of this can be seen in figure 5.1, the first two points of the yellow trajectory are too close to each other to have significant influence on the spatial length and shape of the trajectory. Second, this helps to differ the number of points in the generated user draw with respect to the original target trajectory. As in practice, users won't necessarily have the information of the exact number of actions happened during the course of the trajectory, they are more interested in the spatial length and shape of the trajectory they are searching for. Therefore, we want to allow our application to be flexible enough to accommodate the situation that the number of points from user draw and target trajectory are different. Finally, it speeds up the calculation of the distance algorithm.

This process is repeated 100 times in the experiment, we also keep a list of sampled trajectory to check and make sure there's no duplicated trajectory.

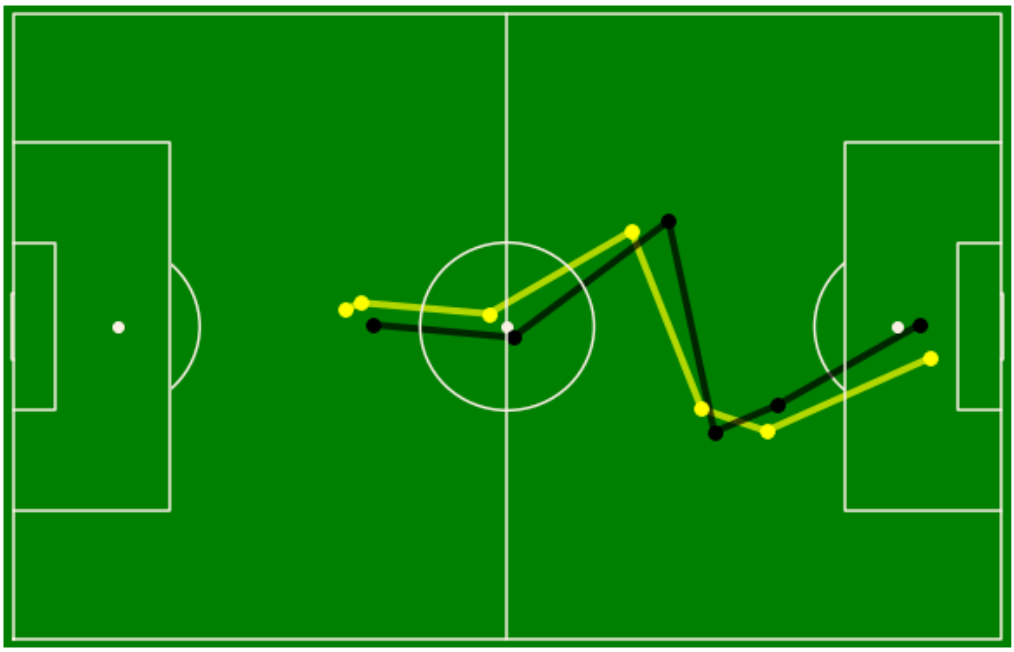

FIGURE 5.1: User draw approximation. This demonstrates how we add noise to the target trajectory and omit the point that is too close to its adjacent point. The yellow trajectory represents the target trajectory and the black trajectory represents the user draw. The user draw is approximated first by randomly omitting one of the two adjacent points that are too close to each other, which is measured by a Euclidean distance less than 1 meter. In this example, the first two points of the yellow trajectory are very close to each other, so one of them is omitted. Then Gaussian noise is added to each of the remaining points of the target trajectory to approximate the user draw.

5.3 Evaluation of the distance metrics

In our study, we use the "Top-k Accuracy" and choose 1 as the k number. It means we recognize the search result is correct only when the trajectories with smallest distance is trajectory trajectory we are looking for. The evaluation criterion of each of the distance metrics is how many of the target trajectories have been successfully found by the distance metric under test ($SuccessCount$), when evaluating a number of searches ($TotalCount$). We call the ration of both the $Accuracy$, which is defined as follows:

$$Accuracy = \frac{SuccessCount}{TotalCount} \tag{5.2}$$

5.4 Optimizations

In this section we propose five optimizations strategies to reduce the run-time of the distance calculations.

5.4.1 Adapt C code for distance calculation

Using Python code for the distance calculations is sub-optimal in terms of processing time because Python code has an inherent overhead associated with the usage of the Python interpreter. We could replace the Python implementation with a pure C implementation for example, and expect a much lower processing time. However, using Python code has several advantages compared to C code: we can easily interface with existing Python frameworks and data types (SPADL, Numpy, distance libraries...) which speeds up the experiments and prototyping. Therefore, we propose to only replace the core of the distance calculation by a C implementation. By doing so, we combine the best of both worlds: we save processing time because the costly distance calculation is accelerated when at the same time, we can keep the simplicity of the Python code. In our experiments, we integrated the Cython library called “traj_dist” to accelerate the distance calculation of Hausdorff, Fréchet and DTW. The library is available on GitHub.

5.4.2 Start and End Point Checking

In this optimization, before a candidate trajectory is used in the distance calculation, we first check if it is worth the effort and time to be computed by means of a filtering strategy. In this strategy, there are two filters: one for the start point of a trajectory and another for the end point of a trajectory. First, we check if the start point of a candidate trajectory is located in an adjacent area of the start point of the user draw trajectory. This is being done by comparing the x and y coordinate of the start point of the candidate trajectory to the user draw trajectory. Second, if the candidate trajectory passed the first filter, we check if the end point of a candidate trajectory is located in an adjacent area of the end point of the user draw trajectory. In order to find enough candidate trajectories but at the same time reduce the number of irrelevant candidate trajectories we choose to limit the size of the checking area to be

a square of 5 meters width on each side. An example of this strategy can be found in figure 5.2.

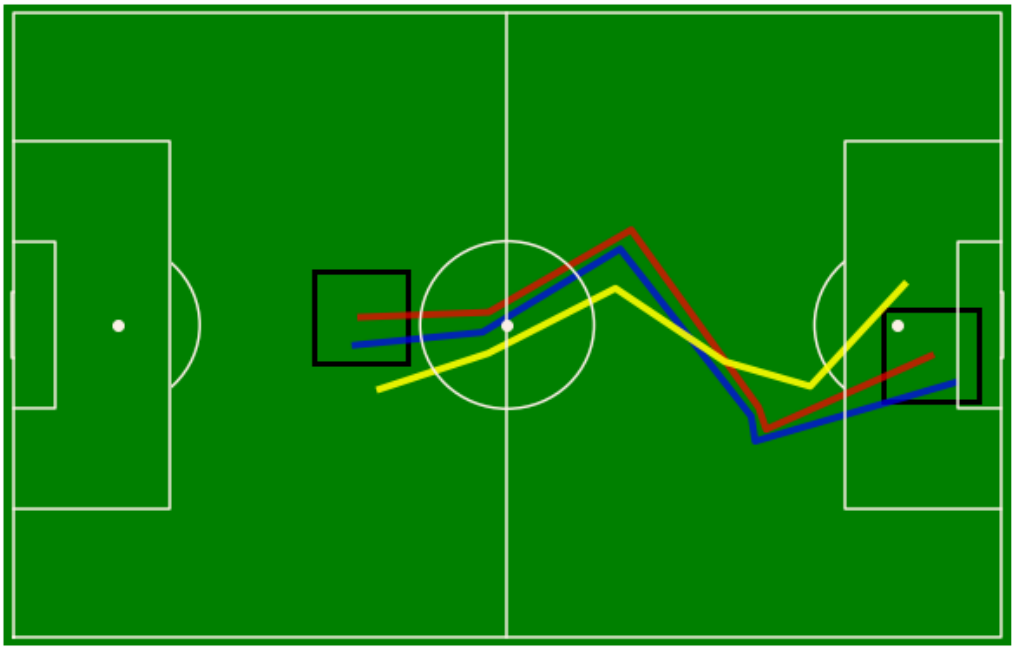

FIGURE 5.2: Start and end point checking. This demonstrates how we filter out the trajectories candidate by using a filter at the start point of the user draw and the end point of the user draw. The red trajectory represents the user draw, the blue and yellow trajectories represent two candidate trajectories from the database. In this example, The blue trajectory is kept for further analysis as both the start and end point of the blue trajectory are located within a small area of the user draw. While the yellow one is filtered out as both the start point and end point located outside the corresponding bounding box.

5.4.3 Trajectory sorting

Another approach to further optimize the trajectory searching algorithm and reduce the run-time of the application is to pre-process the trajectory database. In this strategy, we first read and enumerate the longest contiguous trajectories to find all 8 million candidate trajectories out of the database. These 8 million candidate trajectories are sorted based on the value of x coordinate value of the start point. If the x coordinate value of the first point is the same for two individual trajectories then we implement a secondary sort based on the value of y coordinate of the start point. This results in a sorted list of all trajectories saved into a variable called all_trajectories_sorted. This sorting step needs only to be done once before applying any distance metrics and is therefore a pre-processing step that does not negatively impact the final calculation time and user experience.

This strategy is designed to work with the start and end point checking. With the help of sorting, the searching algorithm can quickly identify a subset of all trajectories by only searching for the candidate trajectories that have a start point close to the start point of the user draw trajectory. In order to locate the start index and end index of the searching space in all_trajectories_sorted, the binary search algorithm is used.

5.4.4 Minimum bounding box

Minimum bounding box is a technique that limits the searching space for the most similar trajectory from the whole soccer pitch into the minimum rectangular box that encloses all points of the target trajectory with a certain margin. The minimum bounding box is constructed by considering all points of the user draw trajectory and considering the extreme points in x and y direction defined by the ordinates $(x_{min}, y_{min}, x_{max}, y_{max})$. A fixed margin value m is then added or subtracted to these ordinates, such that the minimum bounding box is defined by the following 4 points: $(x_{min} - m, y_{min} - m)$, $(x_{min} - m, y_{max} + m)$, $(x_{max} + m, y_{max} + m)$, $(x_{max} + m, y_{max} - m)$.

After constructing the bounding box, all trajectories will be checked to verify if they contain a location point that is located outside the minimum bounding box. All trajectories that contain such an out of bounding box point will be eliminated at this step and not considered further. An example of applying the bounding box technique to reduce the search space can be found in figure 5.3. The minimum bounding box works as a filter that uses the original geo-spatial position on the soccer pitch and has the advantage that the comparison calculation is very fast. The minimum bounding box filter quickly returns a selected set of trajectory for calculating the DTW distance. In order to find the most similar trajectory to the target trajectory we choose the margin m = 5 meters around the minimum bounding box.

This significantly reduces the trajectory that is needed for further calculation, therefore, reduces the run-time.

5.4.5 Trajectory spatial length restriction

Trajectory spatial length restriction is a technique that limits the spatial length of candidate trajectories in the search space. In this step, all trajectories will be checked if its spatial length is within a certain range. If the range is smaller or bigger than the suggested range, the trajectory will be eliminated at this step. Similar to the minimum bounding box, trajectory spatial length restriction also works as a filter that uses the spatial length information of the trajectory and has the advantage that the comparison of length is very fast. The trajectory length restriction filter quickly returns a selected set of trajectory for calculating the DTW distance. In order to find the most similar trajectory to the target trajectory we use the minimum length

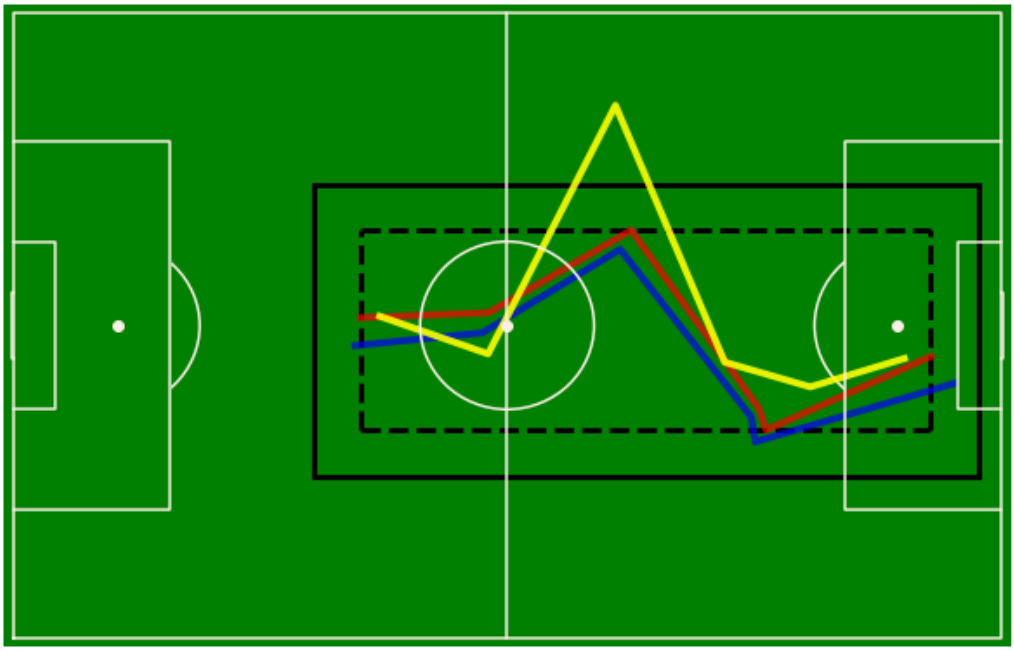

FIGURE 5.3: Bounding box. This demonstrates how we filter out the trajectories candidate by using a bounding box around the user draw. The red trajectory represents the user draw, the blue and yellow trajectories represent two candidate trajectories from the database. In this example, The blue trajectory is kept for further analysis as the entire blue trajectory is located within the bounding box of the user draw. While the yellow one is filtered out as both some of its middle part and end part are located outside the bounding box.

50% of the user draw trajectory and max length 150% of the user draw trajectory as the filtering range. An example applying the trajectory spatial length restriction to filter out candidate trajectories can be found in figure 5.4.

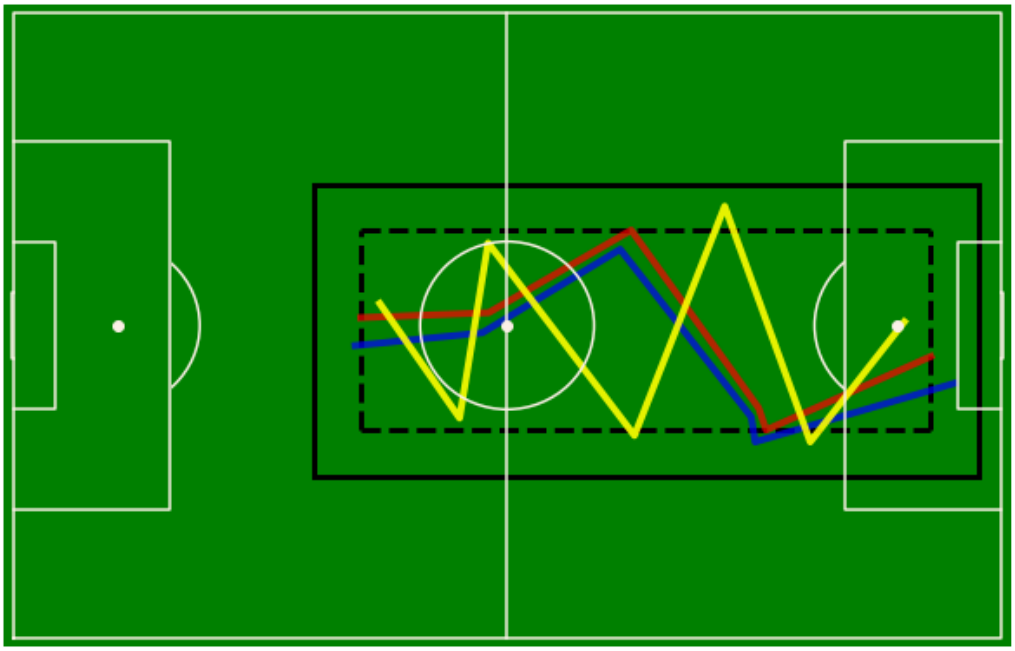

Figure 5.4: Trajectory lengths restriction. This demonstrates how we filter out the trajectories candidate by comparing the spatial length of the user draw with the candidate trajectory. The red trajectory represents the user draw, the blue and yellow trajectories represent two candidate trajectories from the database. In this example, The blue trajectory is kept for further analysis as the spatial length of blue trajectory is similar to the spatial length of the user draw trajectory. While the yellow one is filtered out as its spatial length is significantly longer than the user draw trajectory.

Chapter 6

Experiments

The goal of the study is to find the best match trajectory measured on each distance metric as fast as possible. Therefore, our study consists of two main tasks: (1) identifying the similarity measurement that can find the most similarity match with the input drawn by the user. (2) implementing different optimization techniques to reduce the run time as much as possible. We set up each experiment for each task.

6.1 Accuracy in trajectory retrieval for different distance metrics

The first experiment aims to identify the similarity metric that can find the best match trajectory with respect to the user draw measured by distance.

Each approximated user draw trajectory is used as search input to search for the corresponding target trajectories in the database. Four candidate distance metrics are used in the experiments. These metrics were introduced already in chapter 4: Euclidean distance, Hausdorff distance, Fréchet distance and Dynamic Time Warping (DTW) distance.

We start with the Euclidean distance metric as an example to explain the process. A schematic diagram of the first experiment for Euclidean Distance can be found in figure 6.1.

The first experiment starts from the database containing all 628 soccer games. We randomly sample 100 trajectories using the search algorithm introduced earlier. We take the first target trajectory and add Gaussian noise and omit the close points as described in chapter 5. The resulting trajectory is the approximated user draw trajectory.

Next, this user draw trajectory is input into the Euclidean distance algorithm along with each of the all 8 million trajectories enumerated from the database. The output of each computation is the Euclidean distance between user draw trajectory

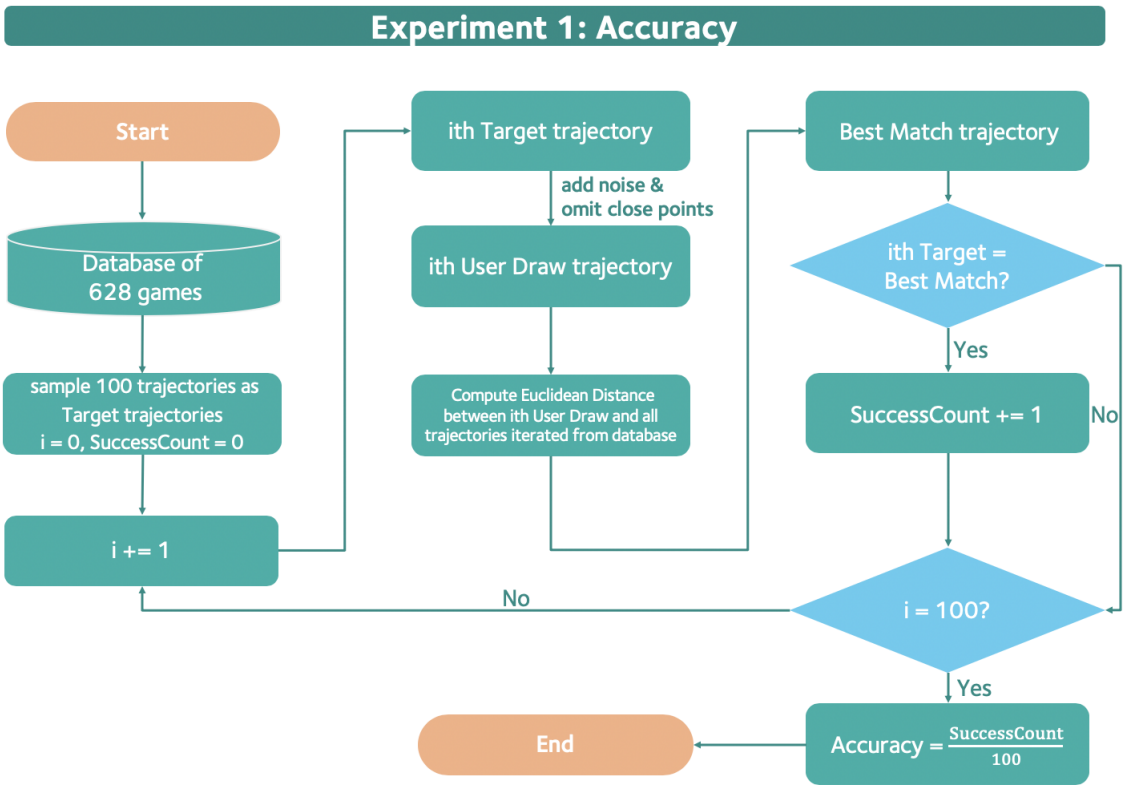

Figure 6.1: Schematic diagram of the first experiment for Euclidean Distance. Same process is repeated for Hausdorff distance, Fréchet distance and Dynamic Time Warping(DTW) distance.

and one of the all trajectories. Since the algorithm on each of the 8 million computations is on a sequential manner, that is the Euclidean distance is computed one by one, we keep the first output of the Euclidean distance and its corresponding trajectory in a variable named min_distance_trajectory. We compare the distance in this variable to the next Euclidean distance and update the distance and its corresponding trajectory when the next distance is smaller than the one in the variable. This minimum distance updating operation is repeated on the result of all the distance computation.

In the end, only the trajectory with the smallest Euclidean distance is saved. The corresponding trajectory of this smallest distance is called Best Match trajectory.

We then compare this Best Match trajectory to the target trajectory to check if they are the same. If so, then we record that the Euclidean distance algorithm found the target trajectory of the first user drawn trajectory by incrementing a *SuccessCount* counter by 1. We also keep a record of the run-time the algorithm

Distance metric	SuccessCount	Average trajectories count	Average Runtime
Euclidean distance	3	580,774	32.6s
Frechet distance	4	8,392,255	1250.65s
Hourdoff distance	4	8,392,255	1647.82s
DTW distance	5	8,392,255	1835.15s

TABLE 6.1: Table showing results of experiment.

used to return each result.

After the first user draw trajectory has been searched for and verified with the target trajectory, we iterate this process for the other 99 user draw trajectories we sampled earlier.

After all 100 user draw trajectories have been searched for, we compute the accuracy of the algorithm using formula 5.2. After the experiment on Euclidean distance is done, we then repeat this process for each of the other three similar measurements: Hausdorff distance, Fréchet distance and Dynamic Time Warping (DTW) distance.

Each of the four different optimization strategies have been tested on the searching algorithm with DTW distance metric to further reduce the run-time so it is suitable for a web application. The result of each of for individual tests can be found in table 6.3.

A pilot experiment on five user draw trajectories of various length is done to evaluate the feasibility, run-time and accuracy of each distance metric. The result shows that average run-time over five runs is 32.6s for Euclidean distance, 1250.65s for Fréchet distance, 1647.82s for Haussdorff distance and 1835.15s for DTW distance. Except for Euclidean distance, all other run-times are too long to scale the experiment to 100 user draw trajectories, as that will take up to 132 hours to complete the experiment one.

Therefore, we identify that the distance calculation is the part of the code that is most time consuming. The experimental code was written in Python, and it is sub-optimal as it is an interpreted language. We then replace the distance calculation in Python by a C implementation to speed up the experiment. We integrated the Cython library called “traj_dist” and to accelerate the calculation of Hausdorff distance, Fréchet distance, and a Cython library named “dtaidistance” to speed up DTW distance calculation.

The result of the experiment 1 can be found in table 6.2. As shown from the result, DTW distance achieved an accuracy of 96% and is the best performance among

Distance Metric	Accuracy	Average Trajectories Count	Average Runtime
Euclidean distance	61%	580,774	30.21s
Frechet distance	92%	8,392,255	157.74s
Hausdorff distance	88%	8,392,255	291.57s
DTW distance	96%	8,392,255	162.15s

Table 6.2: Results of experiment 1.

all four distance metrics. The Fréchet distance with an accuracy of 92% comes the second, Hausdorff distance and Euclidean distance with an accuracy of 88% and 61% fill the rest of the list.

The low performance of the classic Euclidean distance can be explained that it can only compute the distance of two trajectories that have the same number of points. However, many of the users draw trajectories and target trajectories contain different number of points. Euclidean distance does not have the flexibility to allow the user draw and target to have different number of location points.

The average trajectories count adds proof to this explanation as Euclidean distance computed 580 774 trajectories on average. All the other three distance metrics can compute distances of two trajectories with different number of points, therefore, they all computed all 8 392 255 trajectories from the database. As a result, Euclidean distance computed 93.08% less trajectories than the other three distance metrics have computed. Similar pattern can be found in the average run-time of four distance metrics. The Euclidean distance cost significantly less time as a result of less computation. DTW distance is not the most time efficient metric but there is only a negligible few seconds of difference compared to the most time efficient distance metric Fréchet distance.

Since we prioritize accuracy over run-time in this experiment, we choose DTW as the best metric in searching for user draw trajectories. Best as it is, however, 162.15s is still a bit of a pain to wait for a user every time he or she has to search for a specific trajectory when using our web application. Therefore, in the next experiment, we implement different optimization techniques in order to reduce the run-time of searching algorithm as much as possible.

6.2 Optimization strategies in trajectory retrieval

The second experiment aims to optimize the dynamic time warping algorithm to reduce the run time so it is suitable for a web application. A schematic diagram of the second experiment can be found in figure 6.2. Four optimization strategies have been tested: (1) start and end point checking (2) trajectories sorting, (3) apply minimum bounding box with margin, (4) implement trajectory spatial length restriction.

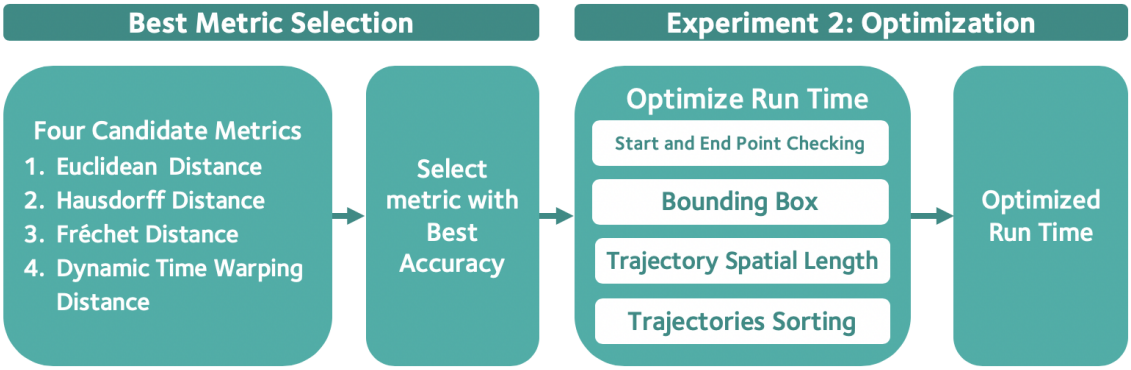

FIGURE 6.2: Schematic diagram of the second experiment.

Optimization	**Acc**	**Avg trajectories (count)**	**Impr (%)**	**Avg runtime (s)**	**Impr (%)**
None	96%	8,392,255	NA	157	NA
Start + End Checking	96%	3,362	99.96	2.26	98.56
Bounding Box	96%	768,810	90.84	53.50	65.92
Trajectory length	96%	4,431,281	47.20	117.11	25.41
Start + End Checking & Trajectory Sorting	96%	3362	99.96	1.55	99.01
All combined	96%	589	99.99	1.46	99.08

TABLE 6.3: Results of the experiment 2. Note that "Impr" stands for "Improvement" and "Acc" stands for "Accuracy".

The first optimization strategy is start and end point checking, this strategy reduced the searching space from over 8 million trajectories to an average only 3362 trajectories, an reduction of 99.96%, also reduced the run-time from 157s to 5.3s, a reduction of 98.56%, and 29.62 times faster than the non-optimized version. This strategy drastically reduce the search space by a factor of 2,496.2. The requirement of having the start and end point of candidate trajectories closely located the set margin is quite strict.

The second optimization strategy is a minimum bounding box with a margin. This strategy reduced the searching space from over 8 million trajectories to an average 768 810 trajectories, a reduction of 97.32%, also reduced the run-time from 157s to 53.50s, a reduction of 65.92%. This results in the searching algorithm is 2.93 times faster.

The third optimization strategy is trajectory length restriction. In order to im-

plement this optimization strategy. We have already pre-computed all of the length of trajectory for quick comparison. since, a pilot experiment has shown that computing the length of over 8 millions trajectories on the go cost an average of 538.20s for each trajectories. Therefore, we choose to pre-compute them for the searching algorithm to work on. This strategy reduced the searching space from over 8 million trajectories to an average 4 431 281 trajectories, a reduction of 47.20%, also reduced the run-time from 157s to 117.11s, a reduction of 25.41%. This is not a drastic changes in rum-time because the extra step in comparing the length of the trajectory cause time. More importantly, over 4 million trajectories pass the filter and need to be compared with the user draw trajectories one by one using the distance metric.

The last optimization strategy is to combine start and end point checking with trajectory sorting. The searching space reduction is the same as start and end point checking without trajectory sorting. But it reduced the average run-time to 1.55s, a 99.01% reduction from the original 157s. This is an 101.29 times improvement in terms of run-time. Sorting all trajectories makes it easy to reduce search space quickly. Two binary searches are performed for the searching algorithm to locate the bounding index of the start point and end point.

As the optimizations are independent, we can combine them together to combine their optimizing power to further reduce the searching space and the run-time. The result is when combine them together in the order of: (1) start and end point checking & trajectory sorting, (2) bounding box and (3) trajectory length together, the searching space is further reduced to as little as a subset containing only 589 trajectories, a 99.99% reduction. And the run-time is reduced to 1.46s, a reduction of 99.01%.

The result shows that the effect of all optimizations are cumulative. They can work independently when used alone. When combined together, the optimising power also combined to achieve a runtime 107.53 times faster compare with the original non-optimized one. We believe an average runtime of 1.46s is suitable for a trajectories searching application to provide smooth user experience.

Chapter 7

Web Application

In this chapter, we briefly introduce our web application we developed based on the results of the experiments in the last chapter.

7.1 Interface

The name of our web application is called "Visual Query of Soccer Actions". The interface of the web application has a title, an instruction and a soccer pitch as a scratch area and a search button. A screenshot of the interface can be found in figure 7.1.

The user is instructed to draw a soccer trajectory on the soccer pitch and after they finished drawing, they need to click the Search button. The interface will send the user draw to the server and run the optimized searching algorithm in Python code. Upon finishing the searching algorithm, the Best Match is returned to the front end interface and displayed on another soccer pitch below the user draw scratch area.

An example of a user query of a soccer trajectory can be found in figure 7.2. In this example, the user drew a trajectory he or she wants to search for in the database. After clicking the search button, the web application returns the best match it found from the database and displays the best match below the user draw and also displays relevant information about the trajectory. The DTW distance between user draw trajectory above and the best match returned, the name of both home team and away team. The name of the competition and the date of the soccer match. Those information can help users to identify which game the best match trajectory belongs to.

7.2 Architecture

The architecture of the web application can be found in figure 7.3. on the front end side is the web browser, the web page is written in JavaScript, HTML and CSS. We

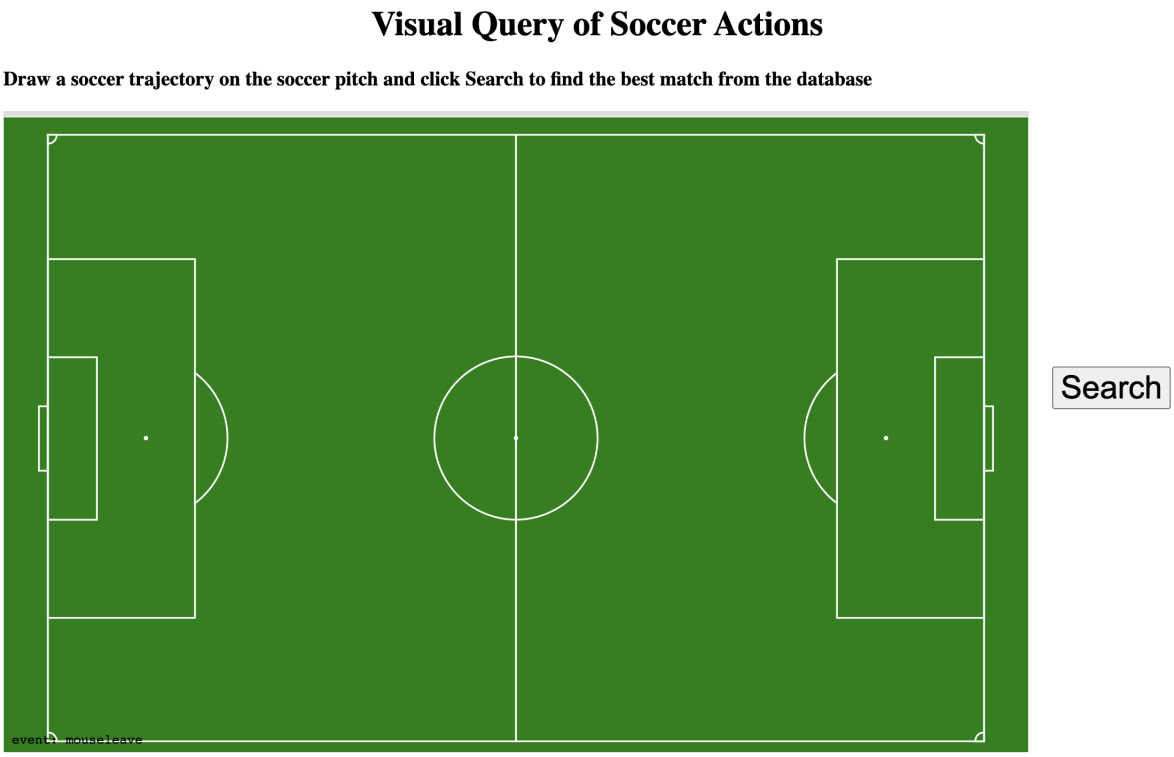

Figure 7.1: The interface of the Visual Query of Soccer Actions Web Application.

have used the D3.js library to achieve a smoother user experience when using our web application. From the web browser, the user can draw a trajectory of interest and the data will be sending to the back end. On the back end side is the web server. It is built on the Flask framework using Python code. After receiving the user draw, it will make a request to read the data from the data set and analysis the data upon successfully reading it.

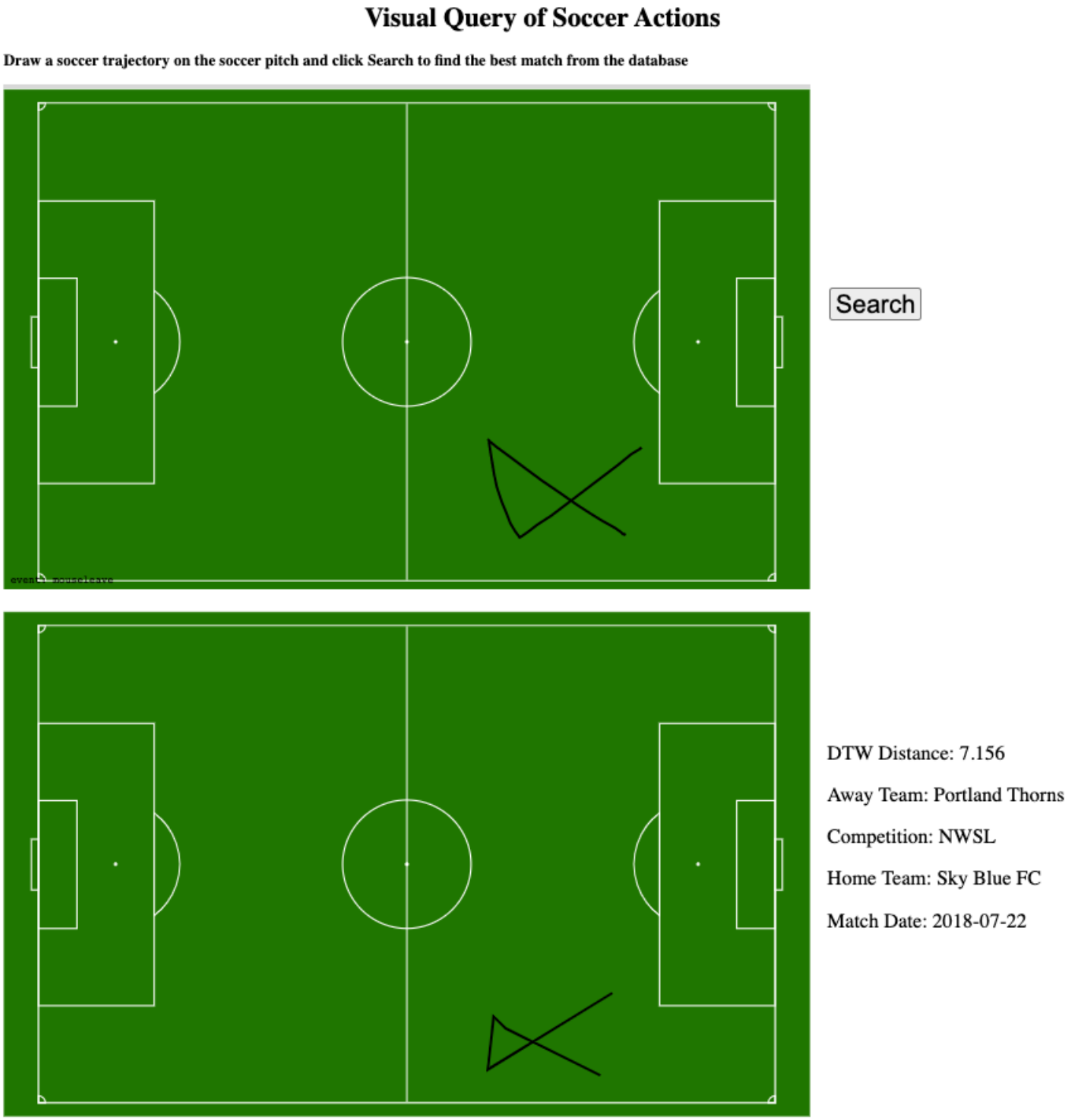

FIGURE 7.2: An example of a user query of a soccer trajectory.

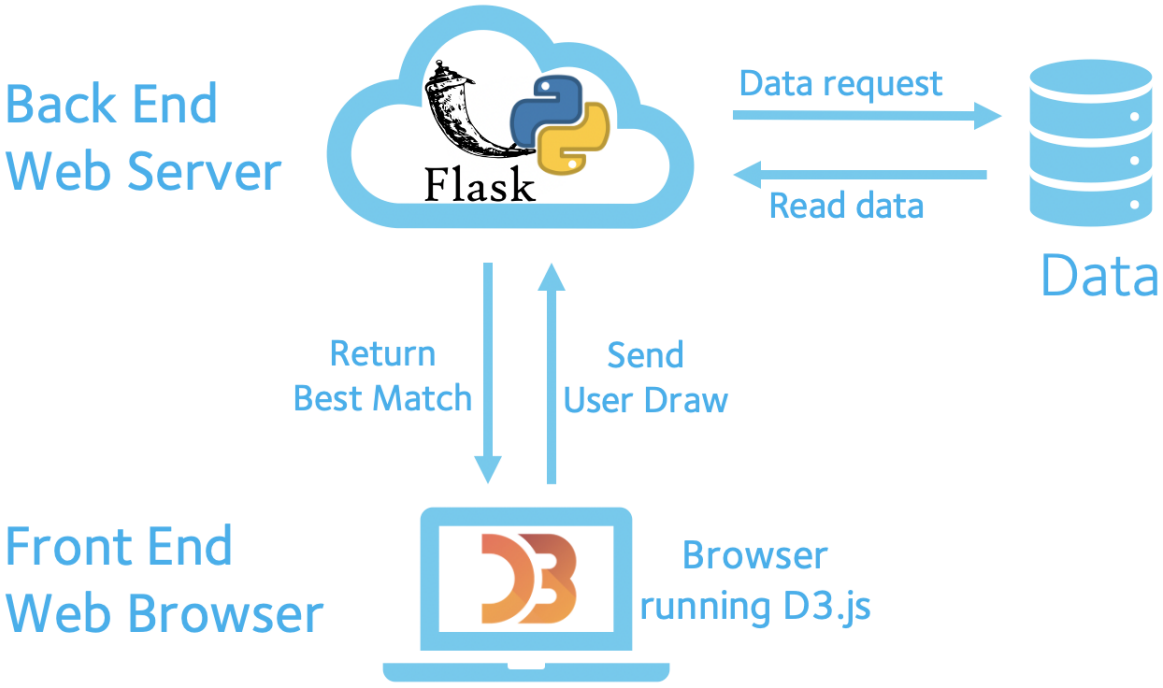

Figure 7.3: The architecture of the web application.

Chapter 8

Discussion

In the discussion chapter, we discuss some of the limitations of our study, and for each limitation, we propose a way to address and improve our work in future research.

- A database of 628 soccer games is big, but it can definitely grow bigger as more data are collected. Any effect to extend our existing work will face the inevitable reality that too much data will increase the run-time. For that, we propose to integrate our trajectory searching web application with a distributed system such as Hadoop and Spark. The integration of parallel computing will keep the run-time at a minimal level. A distributed system also requires more advanced infrastructure. Alternatively, there is a more economical approach available: adding a function to allow users to provide a set of information related to the trajectory of interest. For example, we can ask users to input the name of the team and a range of time-span in terms of year that they want to search for. Those information can vastly reduce the search space to a small subset of the entire database. Hence reduce the run-time.

- The system is designed to search a trajectory that is most similar with a user draw trajectory in the same spatial location. It is possible that users might only be interested in the shape of the trajectory but less so with its spatial location. Therefore, a searching function can be added to accommodate this need by only searching for the relative location of the trajectory. This function can be achieved by normalizing the x and y coordinates of all the points in the trajectories so the exact location information is stripped away. As a result, only the relative spatial information of each point remains. The spatial information of each point preserves the shape information of the trajectory.

- In our study, we used a trajectories sorting strategy to reduce the search space. But this strategy won't work when only the shape of the trajectories is of interest. Therefore, we can implement a clustering based pre-processing to cluster the all trajectories into different clusters and only search in a few clusters that is similar to the user draw trajectory.

- In our experiment, we set the margin to search for the start and end point and bounding box at 5 meters. We choose this 5 meters based on the fact that we added Gaussian noise with $mean = 0$ and $sigma = 1$. Thus we assume user draw will not be too far away from the target trajectories. And in theory, over 99% of the user trajectory in the experiment should be located within that five meter margin. However, we can not guarantee that is always the case in reality. So we can add a function to change the margin size based on how confident the user is with the accuracy of their drawing. If they are less confident then we should choose a bigger margin, such as a 10 meter margin. In short, the system can adjust the margin based on the precision of user drawing. This depends on the user as an soccer expert might draw more precise trajectories than an average user.
- Currently, our system only takes the spatial information of the trajectory into account. It will be interesting to also include more contextual information such as the kinds of action each player performed on the soccer ball and how much the pressure they face as well.

Chapter 9

Conclusion

In this master thesis, we presented a novel visual soccer querying of soccer trajectories web application on a big database of 628 complete games. Our web application allows users to draw a trajectory of interest on a computer screen or a tablet computer and to effectively and efficiently search the most similar trajectory from over 8 million candidate trajectories from the database. We conducted two experiments to develop this searching algorithm.

First, we repeated the 100 trajectories searching test on each of the four distance metrics, namely Euclidean distance, Fréchet distance, Hausdorff distance and DTW distance. Then we identified the DTW distance as the most effective one based on the top-1 accuracy. Next, we implemented four different optimization strategies on the searching algorithm to reduce the search space and run time for 99.99% and 99.08% each. This resulted in a searching algorithm that is 107.53 times faster than the original one and only costs an average of 1.44s to find the best match.

Our web application can be used as an expert tool to search for a trajectory that contains an important tactic and also used for sports fans to explore their favourite soccer movement pattern. However, our web application is not limited to soccer trajectory, but the searching algorithm can be useful in other sports like basketball and rugby and ice hockey. Moreover, it can be helpful to other trajectories searching for situations such as animal migration analysis. We hope our work can shed new light and serve as a starting point for trajectory searching in big databases.

Bibliography

[1] Guo, H., Wang, Z., Yu, B., Zhao, H., Yuan, X. "TripVista: Triple Perspective Visual Trajectory Analytics and its application on microscopic traffic data at a road intersection.", *IEEE Pacific Visualization Symposium 2011, PacificVis 2011*, March, 2011, 163–170.
https://doi.org/10.1109/PACIFICVIS.2011.5742386

[2] Andrienko, N., Andrienko, G. "Visual analytics of movement: An overview of methods, tools and procedures.", *Information Visualization*, 2013, 12(1), 3–24.
https://doi.org/10.1177/1473871612457601

[3] Andrienko, G., Andrienko, N., Bak, P., Keim, D., Wrobel, S. "Visual analytics of movement.", *Visual Analytics of Movement*, 2013.
https://doi.org/10.1007/978-3-642-37583-5

[4] Butkiewicz, T., Ware, C. "Multi-touch 3D exploratory analysis of ocean flow models.", *OCEANS'11 - MTS/IEEE Kona*, 2011, Program Book.
https://doi.org/10.23919/oceans.2011.6107079gg

[5] Perin, C., Vuillemot, R., Stolper, C. D., Stasko, J. T., Wood, J., Carpendale, S "State of the Art of Sports Data Visualization.", 2018, 37
https://doi.org/10.1111/cgf.13447

[6] Basole, R. C., Saupe, D. "Sports Data Visualization [Guest editors' introduction].", *IEEE Computer Graphics and Applications*, 2016, 36(5), 24–26.
https://doi.org/10.1109/MCG.2016.85

[7] Chung, D. H. S., Parry, M. L., Griffiths, I. W., Laramee, R. S. "Knowledge-Assisted Ranking : Sports Event Data.", 2016

[8] Wood, J. "Visualizing Personal Progress in Participatory Sports Cycling Events.", *IEEE Computer Graphics and Applications*, 2016, 35(4), 73–81.
https://doi.org/10.1109/MCG.2015.71

[9] Pileggi, H., Stolper, C. D., Boyle, J. M., Stasko, J. T. "SnapShot: Visualization to propel ice hockey analytics.", *IEEE Transactions on Visualization and Computer Graphics*, 2013, 18(12), 2819–2828.
https://doi.org/10.1109/TVCG.2012.263

[10] Gudmundsson, J., Horton, M. "Spatio-temporal analysis of team sports.", *ACM Computing Surveys*, 2017, 50(2).
https://doi.org/10.1145/3054132

[11] Pingali, G., Opalach, A., Jean, Y., Carlbom, I. "Visualization of sports using motion trajectories: Providing insights into performance, style, and strategy.", *Proceedings of the IEEE Visualization Conference*, 2001, (January), 75–82.
https://doi.org/10.1109/visual.2001.964496

[12] Goldsberry, K. "Courtvision: New visual and spatial analytics for the NBA.", *Proc. 6th Annual MIT Sloan Sports Analytics Conference*, 2012, 1-7.
http://www.sloansportsconference.com/wp-content/uploads/2012/02/Goldsberry_Sloan_Submis

[13] Sox, R "Moneyballs."

[14] Pappalardo, L., Cintia, P., Rossi, A., Massucco, E., Ferragina, P., Pedreschi, D., Giannotti, F. "A public data set of spatio-temporal match events in soccer competitions.", *Scientific Data*, 2019, 6(1), 236.
https://doi.org/10.1038/s41597-019-0247-7

[15] Bernard, J., Ritter, C., Sessler, D., Zeppelzauer, M., Kohlhammer, J., Fellner, D. "Visual-interactive similarity search for complex objects by example of soccer player analysis.", *VISIGRAPP 2017 - Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2017, 3, 75–87.
https://doi.org/10.13140/RG.2.2.14459.98088

[16] Decroos, T., Davis, J. "Player Vectors: Characterizing Soccer Players' Playing Style from Match Event Streams.", *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020, 11908 LNAI, 569–584.
https://doi.org/10.1007/978-3-030-46133-1_34

[17] Decroos, T., Van Haaren, J., Bransen, L., Davis, J. "Actions speak louder than goals: Valuing player actions in soccer.", *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, (1), 1851–1861.
https://doi.org/10.1145/3292500.3330758

[18] Perin, C., Vuillemot, R., Fekete, J. D. "SoccerStories: A kick-off for visual soccer analysis.", *IEEE Transactions on Visualization and Computer Graphics*, 2013, 19(12), 2506–2515.
https://doi.org/10.1109/TVCG.2013.192

[19] Pollard, R. "Charles reep (1904 – 2002): Pioneer of notational and performance analysis in football.", *Journal of Sports Sciences*, 2012, 20(10), 853–855.
https://doi.org/10.1080/026404102320675684

[20] Fritzen Machado, V. “Visual Soccer Match Analysis.”, 2016

[21] Bornn, L., Cervone, D., Fernandez, J. “Soccer analytics: Unravelling the complexity of “the beautiful game.””, *Significance*, 2018, 15(3), 26–29. https://doi.org/10.1111/j.1740-9713.2018.01146.x

[22] Sacha, D., Al-Masoudi, F., Stein, M., Schreck, T., Keim, D. A., Andrienko, G., Janetzko, H. “Dynamic Visual Abstraction of Soccer Movement”, *Computer Graphics Forum*, 2017, 36(3), 305–315. https://doi.org/10.1111/cgf.13189

[23] Sacha, D. “Knowledge Generation in Visual Analytics : Integrating Human and Machine Intelligence for Exploration of Big Data”, 2018, (April), 195. http://nbn-resolving.de/urn:nbn:de:bsz:352-2-163cdymlht22d9

[24] Beernaerts, J., de Baets, B., Lenoir, M., van de Weghe, N. “Spatial movement pattern recognition in soccer based on relative player movements.”, *PLoS ONE*, 2020, 15(1), 1–16. https://doi.org/10.1371/journal.pone.0227746

[25] Feuerhake, U. “Recognition of Repetitive Movement Patterns—The Case of Football Analysis.”, *ISPRS International Journal of Geo-Information*, 2016, 5(11), 208. https://doi.org/10.3390/ijgi5110208

[26] Shao, L., Sacha, D., Neldner, B., Stein, M., Schreck, T. “Interactive Search for Soccer Trajectories to Identify Interesting Game Situations.”, 2016, 1–10.

[27] Stein, M., Janetzko, H., Schreck, T., Keim, D. A. “Tackling Similarity Search for Soccer Match Analysis: Multimodal Distance Measure and Interactive Query Definition.”, *IEEE Computer Graphics and Applications*, 2019, 39(5), 60–71 https://doi.org/10.1109/MCG.2019.2922224

[28] Toohey, K., Duckham, M. “Trajectory similarity measures.”, *SIGSPATIAL Special*, 2015, 7(1), 43–50 https://doi.org/10.1145/2782759.2782767

[29] Yuan, G., Sun, P., Zhao, J., Li, D., Wang, C. “A review of moving object trajectory clustering algorithms.”, *Artificial Intelligence Review*, 2017, 47(1), 123–144. https://doi.org/10.1007/s10462-016-9477-7

[30] Woelfle, M.; Olliaro, P.; Todd, M. H. “Open science is a research accelerator”, *Nature Chemistry*, 2011, 3 (10): 745–748. https://doi.org/10.1038

[31] Legg, P. A., Chung, D. H. S., Parry, M. L., Jones, M. W., Long, R., Griffiths, I. W., Chen, M. "MatchPad: Interactive glyph-based visualization for real-time sports performance analysis.", *Computer Graphics Forum*, 2012, 31(3 PART 4), 1255–1264.
https://doi.org/10.1111/j.1467-8659.2012.03118.x

[32] Peña, J. L., Touchette, H. (2012). "A network theory analysis of football strategies, 1–6. Retrieved from", , 2012, 31(3 PART 4), 1255–1264.
http://arxiv.org/abs/1206.6904

[33] Decroos, T., Van Haaren, J., Davis, J. (2018, July). "Automatic discovery of tactics in spatio-temporal soccer match data.", *In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* ,pp. 223-232.