VRIJE UNIVERSITEIT AMSTERDAM

Tracking soccer players using computer vision

Connecting tracks

Cathy Tol, 2567806

A thesis presented for the degree of
Master Business Analytics

Supervisor: prof. dr. Sandjai Bhulai
Second reader: dr. Mark Hoogendoorn
Host organisation: TNO
External supervisor: Dr. ir. Wyke Pereboom - Huizinga

Tracking soccer players using computer vision

Cathy Tol, 2567806

Internship report

Vrije Universiteit Amsterdam
Faculty of Science
Business Analytics
De Boelelaan 1111
1081 HV Amsterdam
The Netherlands

Host organization:
TNO Locatie Den Haag - Oude Waalsdorperweg
Oude Waalsdorperweg 63
2597 AK Den Haag

June 2020

PREFACE

The last 6 months I worked as a graduate intern at TNO to finish my master Business Analytics by writing a thesis. The department I was working at, is the Intelligent Imaging group. They focus on all sorts of research regarding image processing and computer vision. My research in particular was focused on tracking soccer players by using computer vision. Or, more precise, connecting already existing tracks.

I would like to thank my supervisors. Firstly, I would like to thank my supervisor from TNO, Wyke Pereboom-Huizinga for the opportunity to graduate at TNO's Intelligent Imaging team, and for her advice and guidance throughout the process. Secondly, I would like to thank my supervisor from the VU, Sandjai Bhulai for his feedback during this research. Also, a warm thank you to all members of the Intelligent Imaging team at TNO for their support, time and willingness to always help and answer questions.

SUMMARY

To analyze a soccer match, it is necessary to track the position of each player in the match. The intelligent imaging group of TNO is already able to detect players and to a certain extent track them. However, the tracks in the current system contain many track breaks that need to be repaired in order to know which tracks belong to which player. This will be the focus of this report. The main aim is to connect the current set of tracks correctly, resulting in fewer and longer tracks. The problem can be stated as follows: "How can different tracks be connected such that a player's position is being tracked throughout the match using the available video feeds?"

In this research, a fully working pipeline is created to 1) combine different track sets of soccer players that resulted from different cameras recording a soccer match from different points of view and 2) to correctly connect the already existing tracks using the video data.

To combine multiple track sets into a single track set and to connect the tracks in this single track set, multiple steps needed to be taken. Figure 0.1 provides a schematic overview of the proposed method. All steps in the method are described in detail in sections 4.1 - 4.6.

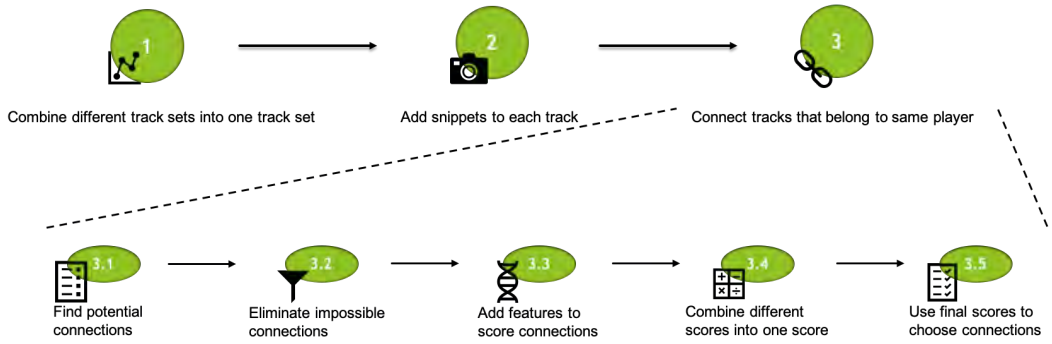

Figure 0.1: Overview proposed method

First, the two track sets needed to be compared and combined into a single track set. This is done based on the overlap of two tracks. To make use of the video feeds, snippets are added to each track. A snippet is the part of a certain frame in which the player is seen.

Then, the tracks that belong to a single player needed to be connected. To find these connections, first potential connections are found by making a potential connection between all ends of tracks and all beginnings of tracks, where the end of the one track needs to be earlier in time than the beginning of the other track.

From the potential connections, the impossible connections are eliminated using spatial-temporal reasoning. Then, using several features based on both track information as well as video data, different scores are given to each connection indicating how well the tracks connect based on these features. The features designed for this research based on the track information are speed, euclidean distance, delta in time. The features based on video data are a deep re-identification feature vector, a color histogram, the detected and classified player number and the skin color of the player.

Those different scoring values are then combined into a single score which is used to choose the winning connections. Multiple ways of combining these scores are researched and the results are compared. First, a simple sum of those scores is used. Second, a weighted sum of those scores is used where the weights are either learned by fitting linear regression model or they are given manually. Third, more sophisticated methods like a multi-layer perceptron classifier or a random forest classifier are used to determine the score. The predicted probability that a connection is true is then used as final score.

Having one final score for each connection, the true connections are chosen. This is again done in different ways. First, a random method is used. Second, a greedy method is used. Third, the true connections are determined using a max flow min cost graph. Last, a global clustering method is used.

Experiments are performed on a manually annotated 1-minute video sequence of a soccer match. After evaluating the different methods to score and choose the winning connections using different sets of features, it can be concluded that using the features based on time, distance, speed, a color histogram and a re-identification deep feature vector yields the best results. It is also found that the greedy and the min-cost-flow approach to choose winning connections with are best to use. Furthermore, the global clustering approach yields better results than the random approach. For the scoring methods it is hard to tell which methods are best since they yield similar result.

It can be concluded that, using a combination of both track-based features as well as video data-based features, different tracks can be connected such that a player's position is being tracked throughout the match.

CONTENTS

1 Introduction

1.1 Problem statement and Objective

A lot of money is going on in the soccer world. Players are getting more expensive every year, e.g. [5] shows an increase of 14.8 bln between 2006 and 2018 for the European market only. Therefore, it is desirable to obtain knowledge and information about each player and the way a team plays. This is done by analyzing the match. For this, it is necessary to track the position of each player in the match. Currently, the players are being tracked manually, for example, by OPTAsports [1]. During and after a match, an annotator will exactly point out which player was where and what is going on in the match. This is very time consuming and therefore the need rises for automating this process. One way to automate tracking is to use sensors [4]. A limitation of this is that then all players need to have sensors taped on their bodies which can limit their performance in the match. An other limitation is the price of such sensors which makes it not a suitable option for amateur matches.

Another approach to automatically track the player's positions is using computer vision. The Intelligent Imaging group of TNO is already able to detect players and to a certain extent track them. However, there are parts in the videos in which that is difficult. For example, when multiple players are very close together or standing behind each other, it is very difficult to decide which player went in which direction. The tracks in the current system contain many track breaks that need to be repaired in order to know which tracks belong to which player. This will be the focus of this report. The main aim is to connect the current set of tracks correctly, resulting in fewer and longer tracks. The problem can be stated as follows: "How can different tracks be connected such that a player's position is being tracked throughout the match using the available video feeds?"

1.2 Relevant information about the host organization

TNO is the Dutch organization for applied scientific research. It is an independent research organization that focuses on applied science. TNO fulfills the role of innovator on behalf of the Ministry of Defence, the Ministry of Social Affairs and Employment and the Geological Survey of the Netherlands. The work of TNO is focused on nine domains which are in line with the challenges and goals of the national economic policy, based on so-called Top Sectors, and with social issues relevant to the Netherlands and Europe.
One of these nine domains is Defence, Safety & Security. In this domain is the expertise group Intelligent Imaging, where this research was conducted. The Intelligent Imaging research group consists of 40 professionals working on projects involving image processing, image enhancement, image analysis, visual pattern recognition and artificial intelligence.

1.3 Structure of the report

The remainder of this report is structured as follows. First, in Chapter 2, the relevant literature is discussed, explaining what research has already been done with respect to the problem and how this relates to the problem. Then, in Chapter 3, the data used in this research is shortly discussed. In Chapter 4, the framework of this research will be discussed. This includes an explanation of all models, techniques and theory used throughout this study. In Chapter 5 the results will be given. The conclusion of this research, together with its discussion, limitations and recommendations are presented in Chapter 6, followed by the appendices and references in Chapter 7.

2 Discussion of the literature

A lot of research is performed on automatically detecting and tracking multiple persons in videos and it is still one of the main research interests in computer vision. Multi-person tracking can be applied to any field where tracking the motion of multiple persons is of interest. Most papers use a tracking-by-detection approach [24], [13], [3], [8]. These approaches treat tracking as a repeated detection problem: first a detection algorithm is applied on individual frames and then these detections are connected across frames. To obtain detections in a single frame, many detection methods have been developed. In [24], aggregate channel features (ACF) are used to train the detectors and a target specific particle filter is used as motion tracker. In [13], a DPM detector is used after which detections are classified by team. Tracking is then performed frame by frame by assigning detections to existing tracks. This is done by using bi-partite matching where the matching cost is the Euclidean distance between centers of detections and predicted locations of tracks. In [3], a first-order Markov model is implemented, considering only information from the current and the last time step. Different detectors are used after which high-confidence detections are selected using target-specific classifiers trained during run-time for each person. Then tracking is performed using particle filtering.

Besides tracking-by-detection approaches, different approaches have been proposed in literature. With the advent of deep learning, end-to-end tracking using neural networks have emerged [16], [18], [25]. In [16] a recurrent neural network is used. This network is capable of performing all multi-target tracking tasks within a unified network structure. In [18] also a recurrent neural network is used. They used it for tracking and classifying a robot's surroundings in complex, dynamic and only partially observable real-world environments. In [25] The Correlation Filter is used, which is an algorithm that trains a linear template to discriminate between images and their translations. Here, The Correlation Filter learner is interpreted as a differentiable layer in a deep neural network.

Another paper that uses deep learning is [8]. This paper uses a tracking-by-detection approach with improved object detection using a deep learning-based Faster region convolutional neural network (Faster R-CNN) algorithm. This is followed by using appearance and improved motion features to track objects.

The aforementioned papers all use either pre-trained person detectors or other pre-trained models. The main challenge of using them is that, because of their pre-training on specific datasets, the detection and tracking is not generalizable to other datasets. Directly applying these methods may therefore result in false positives or missed detections and / or tracks. Usage of any of these methods will most certainly result in broken tracks, that need to be connected before the video can be analyzed.

Connecting a set of broken tracks can be viewed as a flow network problem through which a certain amount of flow must be sent in the cheapest possible way. These type of problems are called min-cost flow problems [23], and are solved by minimizing an appropriate cost function. The min cost flow optimization is used in different problems [9], [15], [6]. In [9],

they use min cost flow optimization to find optimal cargo transport of N types of containers with limited ship capacity, while minimizing the transport costs. In [15] the planning problem related to finding the allocation of vehicles to origin-destination pairs is solved by using min cost flow optimization. The origin-destination pairs have stochastic demand. In [6] min cost flow optimization is used for dynamic assignment procedures for networks with storage devices over time. In [2], [21] and [11] min cost flow optimization is used to connect detections. This, of course, only works with an appropriate cost function. In [22] the cost function is based on distance, in which a greedy algorithm is used to sequentially connect detections to tracks using shortest path computations on a flow network. In [10] the cost-function is based on learned models, where a learned dictionary of interaction features is used.

Besides a connection problem, connecting tracks of players can also be seen as a re-identification (ReID) problem, since re-identifying the soccer players using the detections allows for connecting the tracks correctly. Previously proposed methods have utilized deep learning to re-identify persons [26], [27], [28]. Re-identification is a commonly used approach to add detections to tracks. Literature discussed here is limited to the re-identification of persons in video streams. In [26], a spatial-temporal person ReID framework is proposed that uses both visual semantic information that captures information about the looks of a person and spatial-temporal information that captures information about the time and position of a person. In [27] a pipeline for learning deep feature representations from multiple domains with Convolutional Neural Networks (CNNs) is presented. They use a Domain Guided Dropout algorithm which they have shown improves the feature learning procedure. In [28] a survey is conducted about Deep Learning for Person Re-identification. An overview and analysis of existing methods is given.

3 DATA USED

The Intelligent Imaging department cooperates with a small enterprise that makes videos of soccer matches on various levels of play, of both training and matches. This enterprise provides the Intelligent Imaging department with the videos on request. For this research, videos of soccer matches from Ajax were available.

A single match is captured in video feeds from eight cameras. Figure 3.1 shows the positions of these eight cameras: four are recorded from the long side of the field and four are recorded from the short side of the field. The four video feeds from the same side are placed next to each other such that the entire field is visible.

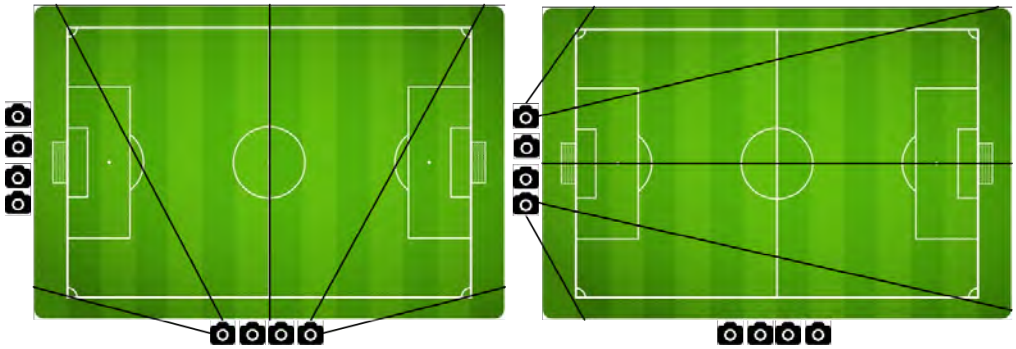

Figure 3.1: Representation of the setting

The delivered video feeds each have a frame rate of 25 frames per second of which, most of the time, 1/4 is processed. The actual frame rate is then 6.25 frames per second. Each frame has a resolution of 3840x2160 pixels.

The delivered video feeds are being processed by the already existing tracking software of the Intelligent Imaging department. This results in tracks of players for all eight video feeds. For the four videos from the same side, the tracks that exist on the borders of the video feeds, are being combined where possible. This eventually results in two sets of tracks: one set resulting from the video feeds recorded at the long side and one resulting from the video feeds recorded at the short side of the field. These track sets, however, contain many track breaks where it is not known which track belongs to which player.

4 Research method

This chapter elaborates on the used methods, the added features, how they are used to give a score to the connections and how those scores are used.

Figure 4.1 provides a schematic overview of the proposed method. All steps in the method are described in detail in sections 4.1 - 4.6.

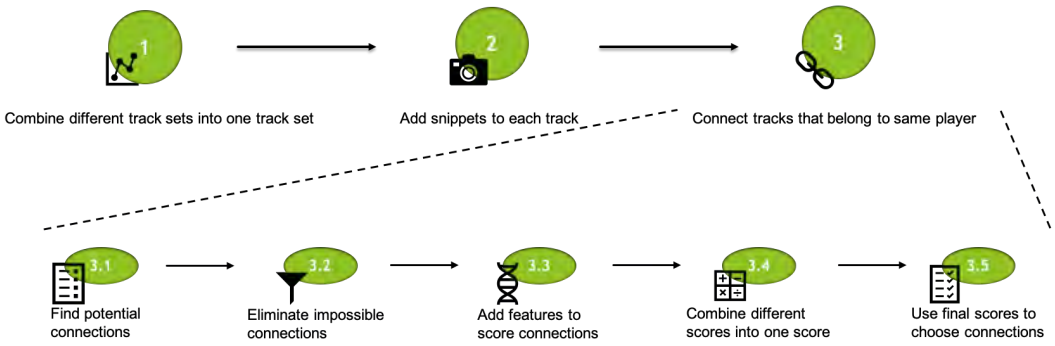

Figure 4.1: Overview proposed method

4.1 Combine tracks from different cameras

First, the two track sets that result from two sides of the field need to be compared and combined into a single track set. This is done based on the overlap of two tracks.

Tracks that could be the same are found using the spatial distance between the tracks. Only the points that exists at the same time for both tracks are considered. To compare the tracks against each other, three matrices are created where the rows represent the tracks from the first set and the columns represent the tracks from the second set. The first matrix consists of the mean spatial distance between the overlapping time points of two tracks. The second matrix consists of the number of times the distance between two points was less than a certain threshold. These numbers can be seen as the overlap of the two tracks. The third matrix consists of the values of the second matrix divided by the number of detections that existed at the same time point for both tracks. These values can be seen as the relative overlap of the two tracks.

All row-column indices where the mean in the mean-distance-matrix is less than a threshold, the overlap is at least 2 and the relative overlap is at least 50% are considered to be possibly the same.

The number of detections that existed at the same time for both tracks, divided by the length of the first track can be seen as the percentage of the first track that will be captured by the second track. If this percentage is higher than a user-defined threshold, the first track could be removed without losing information. In my method, the coverage threshold is set to 90%. Figure 4.2 shows an example of two tracks for which this percentage is calculated.

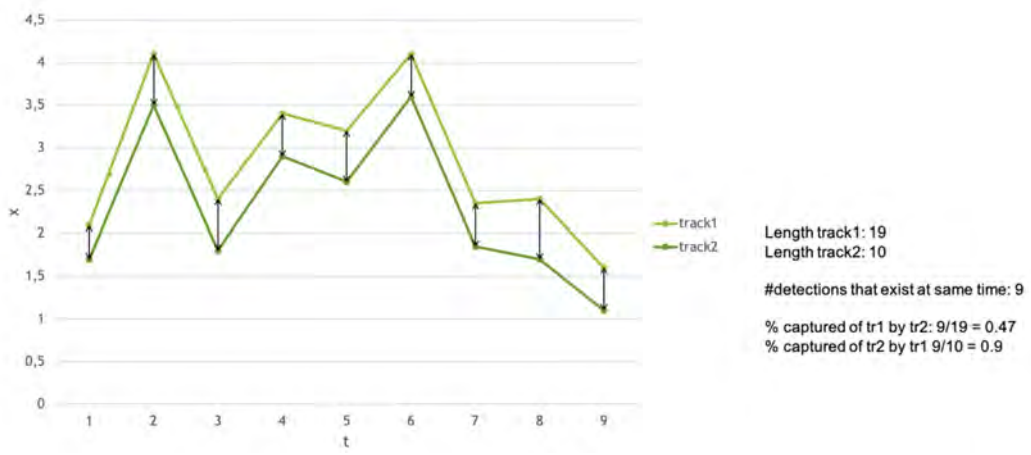

Figure 4.2: Percentage captured

The remaining tracks are clustered in such a way that a cluster consists of all tracks that possibly belong to a single player. Tracks that exist at the same time and originate from the same video feed cannot be in the same cluster. If this is the case, it means that the tracks are clustered incorrectly. This is caused by an incorrect possible merge of two tracks with a track from the other camera. An example is shown in figure 4.3.

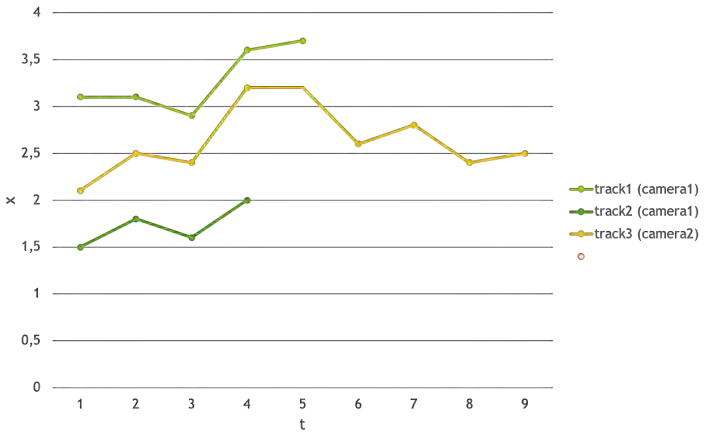

Figure 4.3: Tracks in cluster

To resolve this problem of incorrect clusters, a choice is made which track to keep, based on overlap and density with the other tracks that result from the same video feed. The density can be seen as a measure of goodness of the track, since when a track is dense, i.e. it contains many detections, the tracking algorithm is more reliable and hence, the resulting track is trustworthy. Consider two tracks: track1 and track2. The overlap of track1 with track2 is defined as the number of detections that existed at the same time point for both tracks,

divided by the length of track1. This overlap can be seen as the percentage of track1 that will be captured by track2. A combination of the overlap and the density is used to determine which track is redundant and will be removed from the cluster. After removal of the redundant tracks, the clustering is repeated. This results in new clusters which will replace the old cluster. This checking and adapting is done in a recursive way, until all clusters have been resolved, including the newly made clusters.

Now, all tracks that are in the same cluster belong to a single player and thus they should all be merged. The tracks are merged in such a way that a single track results. For the time points having multiple detections, the coordinates of the detections will be merged by taking the weighted average of the values for all fields. As weight, the confidence of the detection is used, which is a result from the used detection method. This results in a single set of tracks. Since tracks that are not considered for merging also exist, these are taken into account as well to make the set as complete as possible. However, they are only added when they have at least 10 detections, to only have tracks of certain reliability. It is assumed that these tracks are different from the already merged ones and the ones that were found by the other camera.

This track set combining method can easily be extended to multiple track sets when more cameras will be placed.

4.2 Add snippets to tracks

In order to use information from the video feeds for combining tracks, so-called snippets are added to each track for a limited number of detections. Snippets contain pixel information inside the bounding boxes of the detections, i.e. are visual representations of what was detected in time and space. Snippets of a maximum of 10 detections were added. This number is limited due to computation limitations.

To capture as much information as possible in the 10 snippets, snippets are desired where the player is facing different directions. This is done by randomly choosing 30 detections from the track. From those 30 detections, the 10 best representing are chosen. The representativeness of a detection is determined by for example using SIFT features as introduced by Lowe in [12] which are used to recognize parts of images that are the same. This, however, will only work if the object itself does not change much. The fact that soccer players are moving their bodies while playing, causing the person to change in appearance, makes this an inappropriate method. Another method is to use the direction in which the player is moving. The direction is measured by the atan2(dy, dx) where dy and dx are the differences in meters between the current time point and the previous time point. The 10 detections that were chosen are the ones whose directions are most spread. Snippets belonging to these detections are determined.

Because the two outer cameras are also placed in the middle of the line, the players are not projected straight up while the bounding boxes are. This causes the bounding box to not always capture the player entirely e.g. sometimes only the feet are captured. To account for this, the bounding box is rotated by the same angle the player is projected and a bigger bounding box is fitted around that rotated bounding box such that the player is inside the

bounding box. This bigger bounding box is then used to create snippets of players. Figure 4.4 shows an example of snippets of a track.

Figure 4.4: Example of snippets of a track

4.3 Find potential connections

After the second step, a single set of tracks is obtained, where each track belongs to a single player, and relevant video data corresponding with these tracks is added. The next challenge is to combine the tracks, such that a single player is tracked throughout the entire match with a single track. The problem can be seen as a connection problem with a set of players on time point t on the one hand and a set of players on time point t+delta on the other hand. To connect the tracks, first all potential connections need to be found. This is done by making a potential connection between all ends of tracks and all beginnings of tracks, where the end of the one track needs to be earlier in time than the beginning of the other track. Now, from these potential connections, the true connections need to be found. Figure 4.5 shows an example of tracks with the found potential connections.

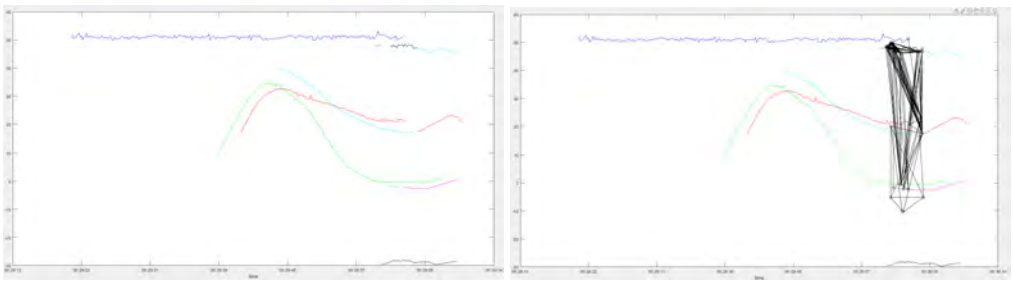

Figure 4.5: Example of tracks with potential connections

4.4 Eliminate connections

From the potential connections, the impossible connections can be eliminated using spatial-temporal reasoning. If there is, for example, a connection between two tracks where the end of the first track is at the one side of the field and the beginning of the other track is at the other side of the field and the difference in time is very small, this connection is not possible. To eliminate connections, the speed and a combination of the work and power that would be needed for this connection to exist are considered. Using some thresholds, impossible connections are eliminated. The threshold used to eliminate the connections based speed is based on top speeds obtained at the 100m sprint at the Olympics running contests.

At the 100 meter sprint at the Olympics, the top speed reachable for man is 12 m/s and for woman 10.5 m/s. In this research it is assumed that the soccer players will not exceed a speed of 10 m/s. Connections where the speed should have been higher than 10 m/s are eliminated. Since the locations of the players are determined using a detection model on the video feeds, the coordinates can deviate slightly from the true coordinates. To correct for an eventual deviance, for the speed the following formula is used: (dr-3)/(dt+0.5), where dr is the Euclidean distance and dt is the difference in time. Here, -3 in the numerator is to account for a deviance and +0.5 in the denominator is to account for the fact that the difference in time can be 0 and dividing by 0 is not allowed. This adaptation, makes sure that only connections which are really impossible, are eliminated.

The work and power features represent the needed work and power a player has to do to get from the end of the one track to the beginning of the other track. If these values are higher than is possible for a human being, the connection is not possible and can thus be eliminated. To eliminate connections, a combination of the two is used: work/sqrt(power). It has been empirically shown by TNO that this measure works well for connecting tracks. Since it is hard to tell at what threshold a connection is not possible anymore, this threshold is set at 200 after empirically testing different threshold values. The value is chosen to be high to be sure that only really impossible connections are removed.

4.5 Score connections

To find the correct connection using some optimization, a cost function needs to be defined, that is low for the optimal connection and high for the least optimal connection. This cost function is based on various features. Each feature assigns a score to a possible connection. The created features and the belonging scores are explained in this section.

4.5.1 Euclidean distance

Tracks where the end of the first track is near the beginning of the other track are more probable to belong to each other. Therefore, the Euclidean distance is used as a score function. Let track1 and track2 be two candidates for connecting, where track2 appears later in time than track1. The Euclidean distance that is used is the Euclidean distance between the detection at the last time point of track1 and the detection of the first time point of track2.

4.5.2 Difference in time

Since the first true connection is the real true connection, the difference in time is used as a score function. Let track1, track2 and track3 be three candidates for connecting, where track1, track2 and track3 appear after each other in time. The difference in time between track1 and track2 that is used is the difference in time between the detection at the last time point of track1 and the detection of the first time point of track2. The difference in time between track1 and track3 that is used is the difference in time between the detection at the last time point of track1 and the detection of the first time point of track3. The difference in time between track1 and track2 is smaller than the difference in time between track1 and track3. The connection between track1 and track2 should thus score higher than the connection between track1 and track3.

4.5.3 Deep feature vector RE-ID

For the deep feature vector, a partly pre-trained human RE-ID feature descriptor model is used. The first layers of this model are the pre-trained layers of the ResNet50-architecture. The last layers are re-trained using annotated snippets of a soccer match of Ajax. These snippets are annotated by team. For this, the data is divided into four classes: 'team1', 'team2', 'keeper' and 'rest'. These layers are then trained with the batch hard triplet loss as introduced in [7]. This ideally causes the model to make embeddings of snippets that are close to each other in the embedding space when the players belong to the same team. The reason for annotating on team level instead of person level is that the difference between players of the same team are not descriptive enough to learn re-identification features for classification on person level.
The Euclidean distance between the deep feature vectors of two tracks is used as score function.

4.5.4 Color histogram

Since the two teams must wear different coloring shirts, the color histograms of the snippets can be very useful to find out to what team a player belongs. For each snippet, a color histogram is made using the non-green pixels of the snippet. The pixel values of a snippet can range from 0 to 255. Computations on the 256 histogram bin color histogram is too expensive, which is why the histogram is reduced to 64 bins. These color histograms can be used in different ways.

First, they are used to classify the snippet at team level. This is done by manually choosing snippets that represent a class and comparing those histograms to the histograms of the snippets that need to be classified. The class of the class-snippet that has the most similar histogram wins. This results in a winning class for each snippet of each track. For each track, the mean value of the winning classes is used as final team classification. However, the performance is dependent on the chosen snippets. To overcome this limitation, the histograms are clustered into k clusters. The histograms of the snippets that need to be classified are then

compared to the cluster means of the k clusters. And again, the class of the cluster mean that has the most similar histogram wins. As k, 4 is chosen. This because of the fact that 4 clusters are expected: team1, team2, keeper and the rest. This also results in a winning class for each snippet of each track. And again, for each track, the mean value of the winning classes is used as final team classification.
Then, for each connection, the classified teams of the two tracks are compared for both methods. This results in two Boolean variables indicating whether it is the same team or not according to each of the two methods. If for a track more than 4 times no winner was found, it can be said that the track is not reliable. Instead of using a Boolean variable, the value 0.3 is given so that it will average out. The sum of the two Boolean variables resulting from the two methods is used as score function.

Second, the distance between the color histograms of two tracks is used as score function.

4.5.5 Detected and classified player number

The most descriptive element of a soccer player is probably the number on his back. This number is detected and classified in each of the 10 snippets. It can be the case that a number is not detected or that the detected number is misclassified in some snippets. For each track, all detected and classified numbers are concatenated into one list. Only a part of this list is kept based on some decisions explained further. The number of intersecting numbers of the two tracks is then used as score.

To decide which numbers to keep, four methods have been researched This resulted in four features. First, all numbers are kept. The problem with this is that numbers that occur a few times, are not that probable to be correct and will still be in the result. Second, only a percentage of the numbers is kept, giving priority to numbers that occur more often. The problem with this is that equally frequent occurring numbers could not get in the result. Third, only the numbers that occur most are kept. The problem with this is that numbers that do not occur frequently can also get in the result if there are not so many different numbers found. Fourth, only numbers that account for at least a certain percentage of the total length of the list are kept. The problem with this is that if many different numbers are found, the result would be an empty list.

To detect and classify numbers, two classifiers are trained on the Street View House Numbers (SVHN) dataset introduced by [17]. This is a dataset obtained from house numbers in Google Street View images. The classifiers are trained according to [20]. The first classifier is trained to determine whether or not a number is present. This results in two classes: 0 if no number is present, 1 if a number is present. The second classifier is trained to determine what number is present. This results in 10 classes: a probability for each possible number between 0 and 9.

To detect numbers, candidate regions are found using the Maximally Stable Extremal Regions (MSER) algorithm. The MSER algorithm is presented in [14] where it is used to find correspondences between a pair of images taken from different view points. The first classifier determines whether or not a number is present in the proposed region. Then, for all

proposed regions which are classified as numbers, the second classifier determines the number. The two classifiers have the same network architectural backbone. The only difference is the number of classes.

Figure 4.6 shows two examples of snippets with their found numbers. The left snippet shows that one number is detected which is classified as 4 with a confidence of 1.00. The right snippet shows that two numbers are detected. The first number is detected as 2 with a confidence of 1.00 and the second number is detected as 1 with a confidence of 0.96.

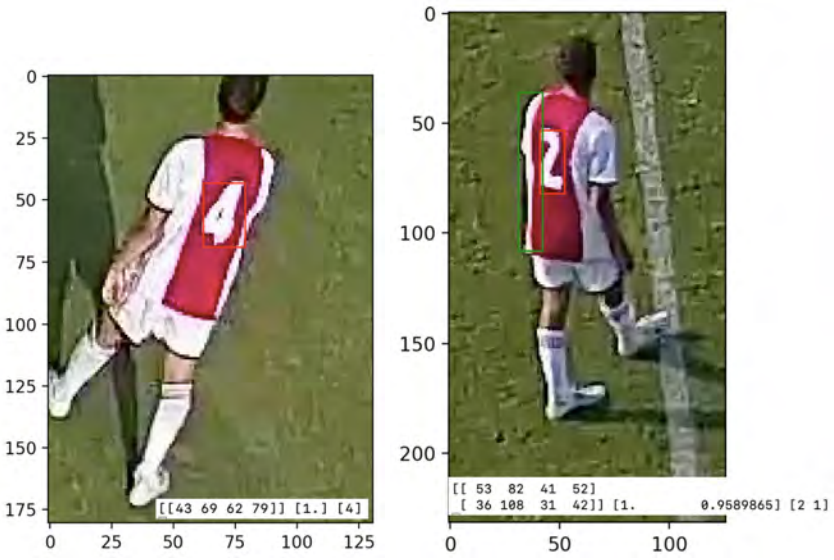

Figure 4.6: Examples of found numbers
Bouding box coordinates of found numbers and corresponding detected numbers with confidence levels are shown at the bottom of the snippets.

If a player walks away from the camera, the camera is pointed to his back and the number should thus be visible. However, if the player walks towards the camera, the camera is pointed to his front and the number is thus not visible. To filter out some false positive number detections, the direction in which the player walks is used to determine whether the number can be visible in a certain snippet. Only the snippets of which the number can be visible based on the direction, are passed to the method to detect and recognize numbers.

For snippets taken by a camera on the long side, a number can be visible if the direction is between 0 and 180 degrees. For snippets that are taken by a camera on the short side, a number can be visible if the direction is between -90 and 90 degrees.

To determine the direction, first the track is smoothened by interpolation. This is necessary because not smoothened tracks can result in incorrect directions since a player can be moving very active and coordinates of detections can deviate. This makes it safer to use smoothed tracks. The smoothing degree is based on the length of the track. Then using the atan2 function, the direction is found. The final direction that is used for a point, is the mean value of the direction towards that point and the direction from that point.

First, the two classifiers where trained on the SVHN dataset only. This dataset consists of snippets of street view house numbers. The numbers of these snippets can look different from the numbers that are on the back of a player. To improve classification accuracy on the back number, the final layers of the classification networks are re-trained on snippets from the used soccer dataset. This caused the test accuracy on the first classifier, the detector, to go from 0.953 to 0.987 and the test accuracy on the second classifier, the recognizer, to go from 0.277 to 0.504.

4.5.6 SKIN COLOR CLASSIFICATION

To determine the skin color, first the skin pixels are segmented from the snippet by filtering on a defined threshold. Then, the skin-pixels are clustered into 4 classes based on their RGB values. The class means then represent the dominant colors of the skin. The class mean, weighted by the cluster size, is then used to represent the mean dominant skin color. This is done for all snippets of all tracks. A visualization of the process of determining the mean dominant skin color can be seen in figure 4.7.

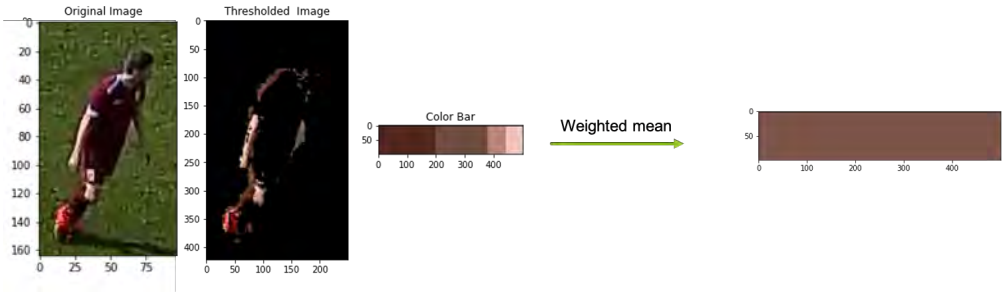

Figure 4.7: Skin color extraction

Those mean values of some snippets are then clustered into 2 classes whose cluster means represent the final skin colors. All snippets are then classified as skin color 0 or skin color 1 by comparing the mean dominant skin color to the cluster means. The class whose cluster mean is closest to the mean dominant skin color wins.

A Boolean variable indicating whether the skin colors of two tracks are the same is used as score function.

4.5.7 Resulting features

The features that result are: **dt**, which is the difference in time between the end of the one track and the beginning of the other track, **eucl_distance**, which is the euclidean distance between the end of the one track and the beginning of the other track, **speed**, which is the speed that would be needed to go from the end of the one track to the beginning of the other track, **wp**, which is a combination of the work and power that would be needed to go from the end of the one track to the beginning of the other track, **average_hist_dist**, which is the average distance between the histograms of the one track and the histograms of the other track, **average_feature_vec_dist**, which is the average distance between the feature vectors of the one track and the histograms of the other track, **same_team_hist**, which tells whether the players of the two tracks are classified as being of the same team using the classification model where manually chosen snippets are used to represent a class, **same_team_cluster**, which tells whether the players of the two tracks are classified as being of the same team using the classification model where all color histograms are clustered first and the cluster means represent a class, **same_team_sum**, which is the sum of *same_team_hist* and *same_team_cluster*, **intersection_numbers_found**, **intersection_numbers_found_x_percent_most**, **intersection_numbers_found_x_most** and **intersection_numbers_found_x_percent**, which all four tell the intersection of the found numbers of the two tracks by using different numbers of the lists (as explained in section 4.5.5) and **same_skin_cluster**, which tells whether the players of the two tracks are classified as being of the same color.

4.6 Combine features scores into one score

All connections now have multiple scoring values that can be used to determine what connections may be true. In order to use these scores, the scores need to be combined to a single score. To enable the combining of the scores, all scores are normalized to a value between 0 and 1.

First, a simple sum of those scores can be used. To take into account that some features are more important than others, I propose to use a weighted sum. The weights will be learned by fitting a linear regression model or they will just be given manually.

Also, more sophisticated methods like a multi-layer perceptron classifier and a random forest classifier are used to determine the score. These models are trained to predict whether a connection is true or false, using annotated connections and the corresponding scores from each feature. The predicted probability that a connection is true is then used as final score.

Multiple track breaks can occur, e.g. at time points t, t+1 and t+2. To prevent that a track at time point t is connected to a track at time point t+2 instead of the track at t+1, a connection is only seen as true if the two tracks are of the same person AND if the second track is the first one to occur after the first track.

4.7 Choose winning connections

Having one final score for each connection, the true connections can be chosen. This can be done in different ways. The used methods are explained in the next sub-sections.

4.7.1 Random

To have a first baseline, the winning connections are chosen randomly. Simply choosing some connections will lead to an incorrect solution. To prevent that multiple tracks will be connected to a single track, connections are chosen iteratively in a random way while removing all connections that are not possible anymore by choosing this connection.

4.7.2 Greedy

Since connections that have the highest scores are most probable to be true, a greedy approach can be used. In the greedy approach, the tracks are connected using this score: the connection having the maximum score of available connections is true. Like with random connecting, the connections are iteratively chosen while removing all connections that are not possible anymore by choosing this connection, to prevent connecting multiple tracks to a single other track.

4.7.3 Max flow with min cost

The greedy approach will search for the locally optimal choice at each stage. This will not necessary lead to the global optimum. By following the greedy approach, mistakes made in the beginning are propagated further. Since we are interested in the global optimum, an optimization approach is used to choose the winning connections. Since the problem can be seen as some sort of connection problem, this can be modeled as a minimum cost maximum flow problem which can be efficiently solved by the network simplex algorithm, first introduced by [19], in polynomial time. The graph of the original problem consists of a bipartite graph with nodes that represent idx1 on the one side and nodes that represent idx2 on the other side as seen in figure 4.8.

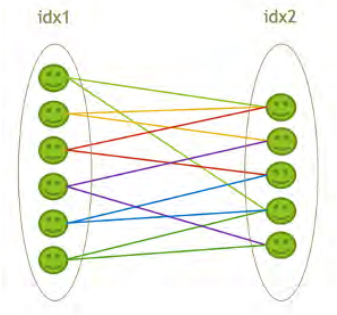

Figure 4.8: Graph of original problem

The edges represent the potential connections. All edges have a cost that is equal to the score of the connection multiplied by -1. The scores of the connections had to be multiplied by -1 to be used as costs, because the scores had to be maximized while the minimum cost maximum flow problem minimizes the costs. And, multiplying the scores of the connections by -1 and then minimizing the scores, gives the same result as maximizing the original scores of the connections.

The goal is to find the maximum number of connections with minimum cost. The idea is to reduce this problem to a network flow problem. This is done by adding a source node s and a sink node t. The source node is connected to all nodes of idx1 and all nodes of idx2 are connected to the sink node. The capacity of all edges is set to 1. The costs of the already existing edges stay the same. The costs of the newly added edges are set to 0. This new graph is depicted in figure 4.9.

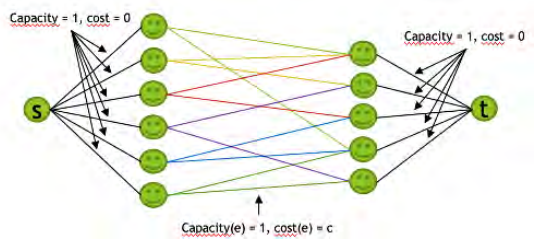

Figure 4.9: Graph of network flow problem

4.7.4 GLOBAL CLUSTERING

Connections at different timesteps do not depend on each other. They are equally important. Therefore, a greedy look-a-head, in which connections made earlier in time could be disregarded because another connection is encountered later in time, is not a suitable connection approach.

With the greedy approach, connections are iteratively chosen by choosing the highest scoring connection and removing the connections that are not possible anymore. This can cause a combination of connections to not be considered. To take more combinations into account, a global clustering method is performed.

With the global clustering method, all connections having a score higher than a threshold are taken as potential winners. Those connections are clustered in such a way that all tracks that belong to the same player are in the same cluster. For example, if track1 is connected to both track2 and track3, they are all in the same cluster. But, if track2 and track3 are not of the same player, one of the two connections should be incorrect. To correct for those mistakes, all those triplets are found and checked. In the afore mentioned example, if the score of the connection between track2 and track3 would be less than a threshold, the cluster is incorrect and thus either the connection between track1 and track2, or the connection between track1 and track3 is incorrect. The connection that has the highest score is taken to be the correct one. If it is not a triplet but multiple tracks with which one track has a connection, the

highest scoring connection is taken to be true. All connections between the winning track and the other tracks of which the score is lower than a threshold, are seen as incorrect and are removed from the cluster. This thus results in a cluster where all tracks are connected to each other.

5 RESULTS

To evaluate the chosen connections, a ground truth is needed. This is made manually by classifying each connection to be true or false. A connection is seen as true if the two tracks are of the same person AND if the second track is the first one to occur after the first track.

Using a variety of evaluation metrics, results of all combinations of the following scoring and choosing methods are evaluated: MLP, linear regression and random forest for combining the scores of each connection and global clustering, greedy, min-cost-flow and random for choosing the winning connections. The metrics that are used are: **accuracy**, which is the ratio of number of correct predictions to the total number of input samples, **precision**, which is the number of correct positive results divided by the total number of positive results predicted, **recall**, which is the number of correct positive results divided by the number of all real positive samples, **f1**, which is the Harmonic Mean between precision and recall:

$$2 \cdot \frac{precision \cdot recall}{precision + recall}$$

, **fbeta 0.5**, and **fbeta 2** which are the same as the f1 score but now weighted where the beta is the weight of recall and **AUC score** which is equal to the probability that a randomly chosen positive sample will be ranked higher than a randomly chosen negative sample. It is the area under the curve of a plot of the False Positive Rate vs the True positive Rate.

In order to compare the evaluation metrics with the ground truth, the number of positive predicted values and the number of true positives are also given.

Models are evaluated on a one-minute video sequence of a soccer match. In this video sequence, 678 different tracks are found. For those tracks, 78754 potential connections are found. Ten percent of the potential connections that where left after elimination are used to train the models and the rest is used to test the models. The test set consists of 70670 connections. In this test set are 561 manually annotated positives and thus 70109 negatives. Due to a high class imbalance in the dataset, i.e. the number of negative connections is much higher than the number of positive connections, accuracy is not a suitable metric. Since 99% of the samples are negative, predicting every sample as negative will already give 99% accuracy while we are actually interested in the positive samples. This gives thus a false sense of high accuracy.

Not all features add value to the model. Wrong connections may result from using features with low predictive value. To find the best combination of features, several combinations are tested. Evaluation is conducted using 10-fold cross-validation.

The features that can be used are listed and explained in section 4.5.7. Before going into a model, all values are normalized for each feature such that all values are between 0 and 1.

First, evaluation is conducted using all features. This is set 0. Table 7.15 of Appendix B shows the average results of the 10 test rounds using all features. The standard deviations (rounded to 5 decimal points) between the results of those 10 test rounds can be found in table 7.16 of Appendix B. The standard deviations between tests can be seen as some robustness value of the models.

Then, evaluation is conducted using each feature alone for different models. This is to see whether some features on their own already add value. Each feature is tested for one test round. The results of this can be found in Appendix A. It can be seen that when looking at the overall results, the features *dt_normalized*, *eucl_distance_normalized* and *speed_normalized* add the most value when used by themselves for the tested models.

As mentioned in section 4.6, among others, a linear regression model and a random forest model are fitted. By fitting a Linear Regression model, the weights given to each feature can be seen as some importance value for that feature. Figure 5.1 shows the weights for each feature sorted by its absolute value. A fitted Random Forest model also gives feature importance values for all features. These are shown in figure 5.2.

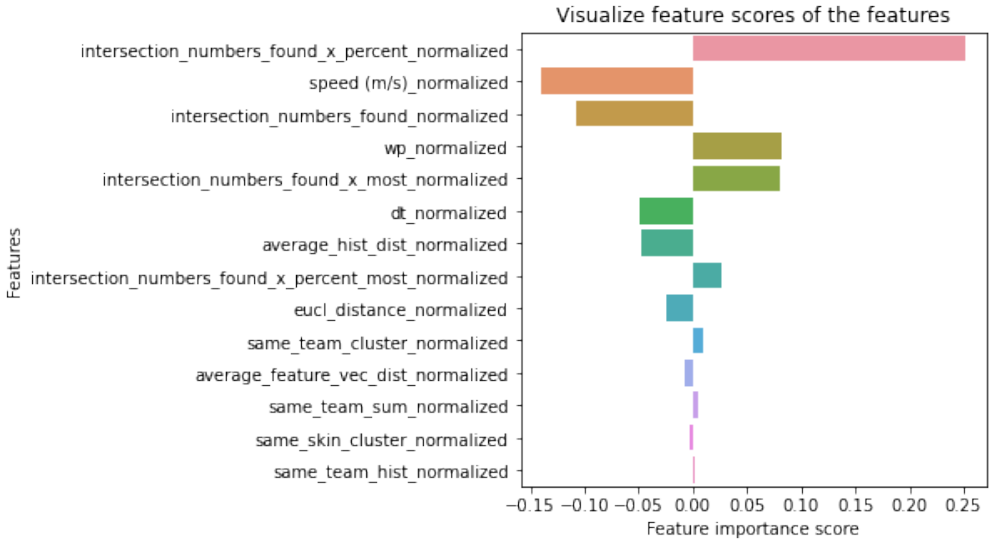

Figure 5.1: Feature importance by Linear Regression model

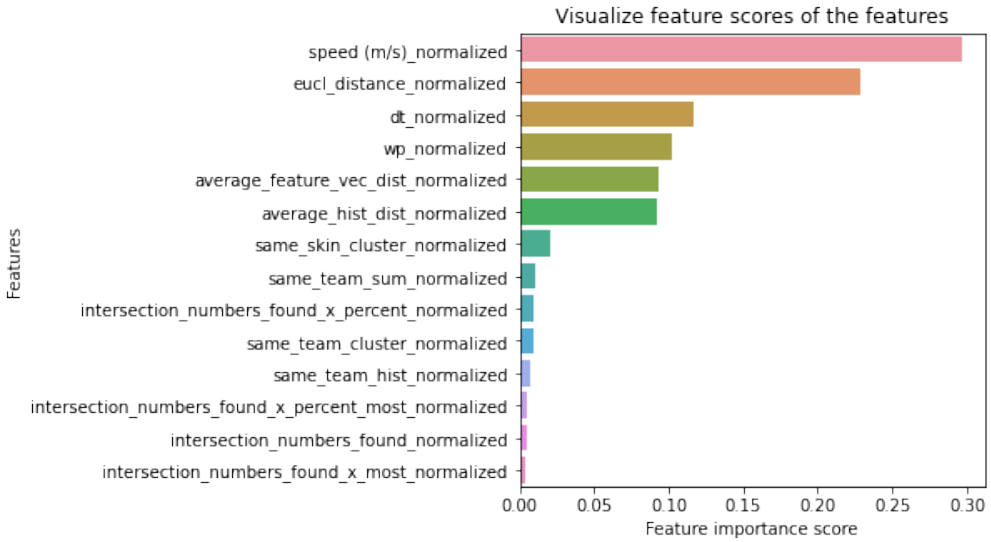

Figure 5.2: Feature importance by Random Forest model

Interestingly, the Random Forest model gives comparable results as when each feature was used alone by giving the features *speed_normalized*, *eucl_distance_normalized* and *dt_normalized* the most importance. While the Linear Regression model gives the features *intersection_numbers_found_x_percent_normalized*, *speed_normalized* and *intersection_numbers_found_normalized* the most importance.

Correlating features bring the same information and they can reduce the model performance. Therefore, it is not desired to have correlating features to train models on. By analyzing the correlation between features, which can be found in figure **??**, it is seen that some features correlate with each other.

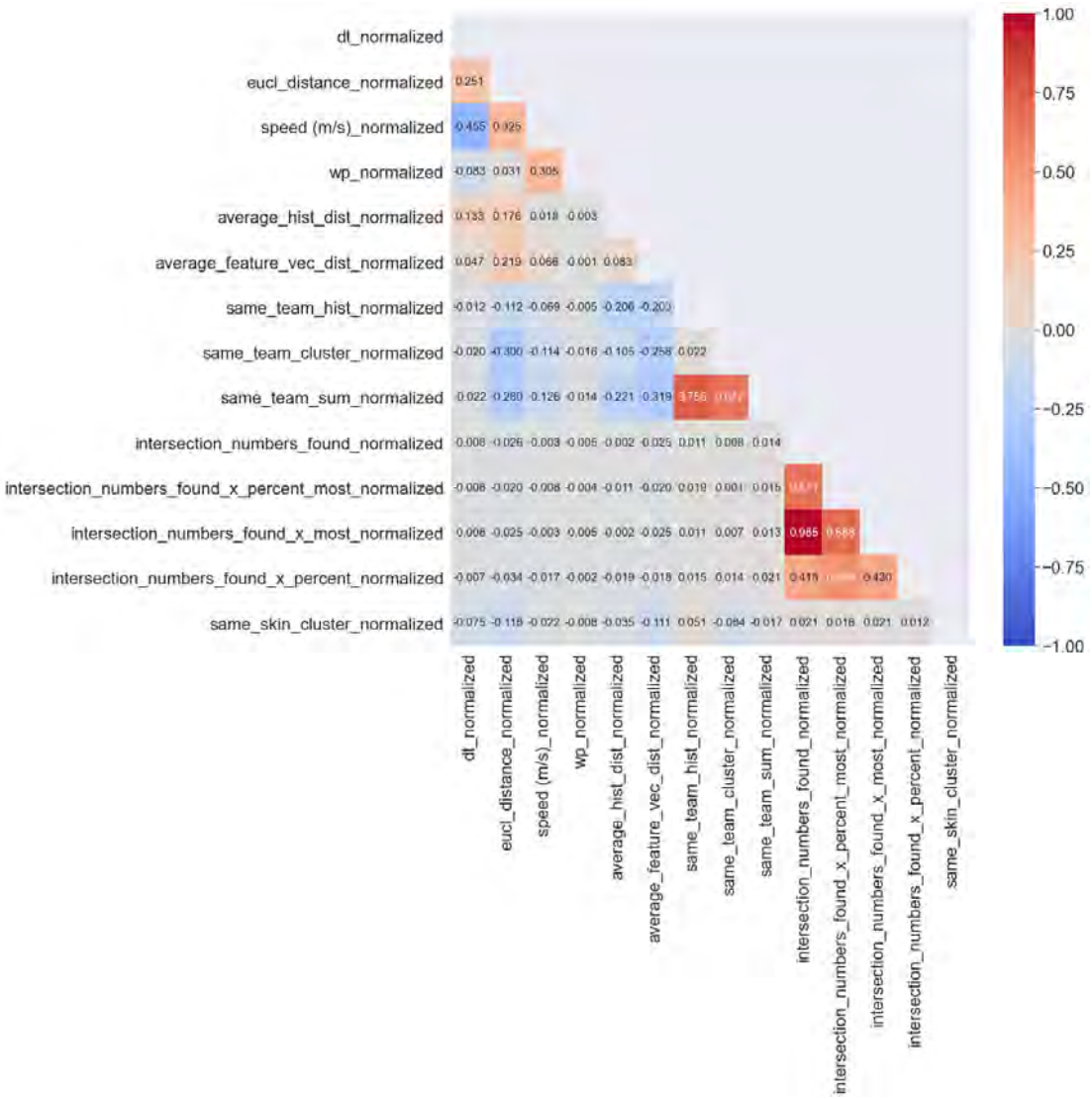

Figure 5.3: Correlation matrix

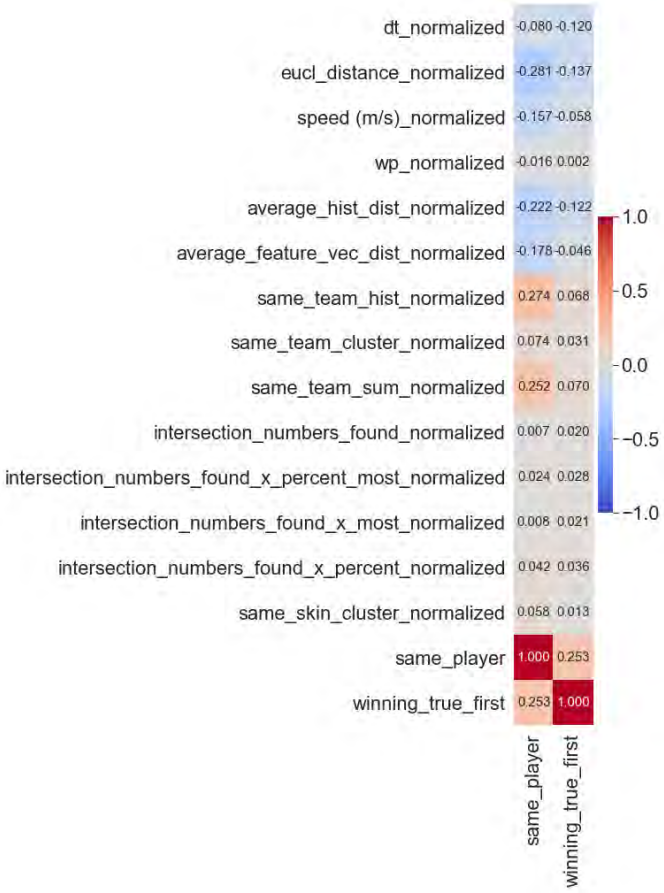

Figure 5.4: Correlations with true value

The features *wp, dt, eucl_distance* and *speed* are correlated with each other. This makes sense since *wp* is a combination of work and power, which depend on time and distance. And *speed* is a combination of time and distance. It is decided to not use *wp* as a feature since, besides being correlated with other features, there are a lot of nan/missing values. For the features *dt, eucl_distance* and *speed* is decided to use them all despite being correlated because they all add value. A solution to diminish correlation could be to combine the features into less features by using for example principal component analysis. However, this causes the loss of variance which causes loss of information. After testing some combinations, it is seen that it either worsen the results or does not improve the results. I conclude that the model is robust for the correlation between those features and therefore decided that all three features are used as is.

The features *same_team_sum, same_team_hist* and *same_team_cluster* are also correlated with each other. This also makes sense since *same_team_sum* is the direct sum of the other two. It is decided to only use the feature *same_team_sum* since this feature captures the other two sufficiently.

Other features that correlate with each other are the features *intersection_numbers_found*, *intersection_numbers_found_x_percent_most*, *intersection_numbers_found_x_most* and *intersection_numbers_found_x_percent*. This also is explainable since they are all created by using numbers that are found by the number detector and classifier. As explained in section 4.5.5, different numbers of the list are used. Since they are highly correlated, only one should be used. The correlation between *intersection_numbers_found_x_percent* and the true value is the highest as can be seen in figure 5.4. Additionally, both the Linear Regression model and the Random Forest model chose this feature to be the most important of the four. Also, rationally thinking this feature should be the most representative for the real number of the four. Therefore, this feature is chosen to be used.

From the correlation analysis the left-over features are: **dt**, **eucl_distance**, **speed**, **average_hist_dist**, **average_feature_vec_dist**, **same_team_sum**, **intersection_numbers_found_x_percent** and **same_skin_cluster**. This is set 1. These are the features that do not correlate too much with each other. The results of using those features for 10 test rounds can be found in table 7.17 of Appendix B. The standard deviations (rounded to 5 decimal points) between the results of those 10 test rounds can be found in table 7.18 of Appendix B. It can be seen that the overall results have been slightly improved. However, the standard deviations have been slightly increased.

Again, both a Linear Regression model and a Random Forest model are fitted to gain insights into the feature importance but now using only the features that are left thus far. The weights for each feature sorted by its absolute value of the Linear Regression model can be found in figure 5.5. The feature importance according to the Random Forest model can be found in figure 5.6.

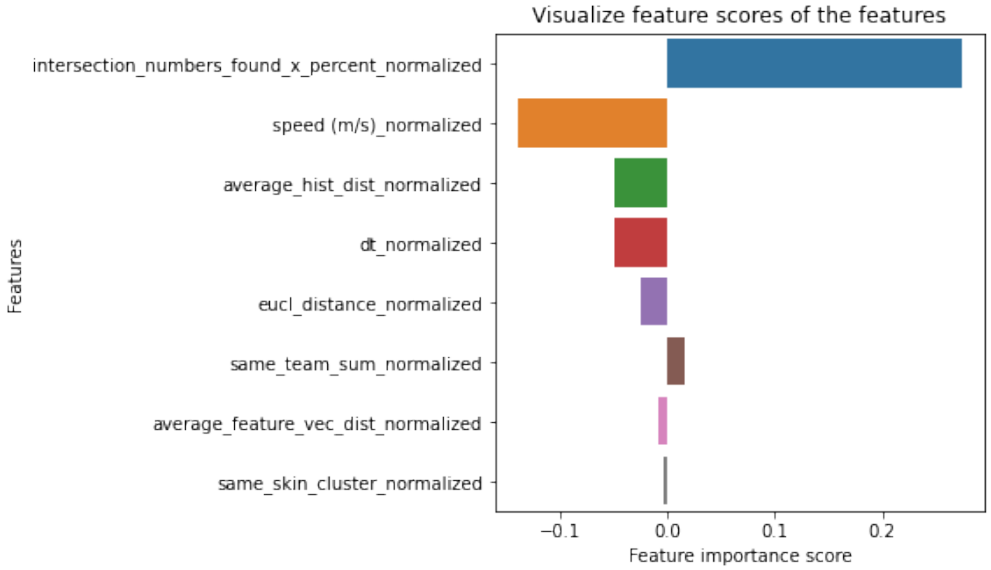

Figure 5.5: Feature importance by Linear Regression model

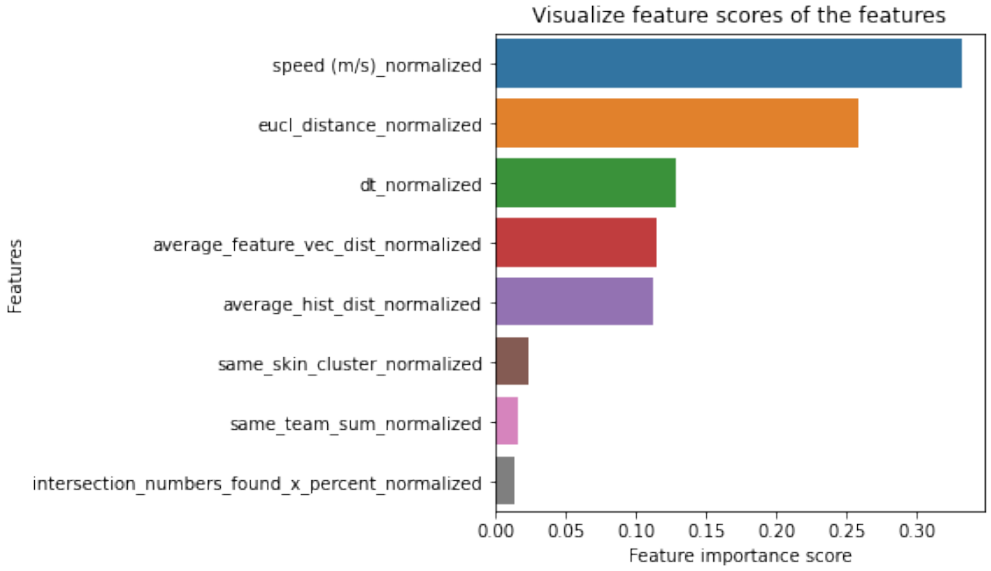

Figure 5.6: Feature importance by Random Forest model

Interesting is that the Linear Regression model gives the feature *intersection_numbers_found_x_percent*, the highest score while the Random Forest model gives that feature the lowest score. On the other features they mostly agree on the importance ranking. They for example both assign the features *same_team_sum* and *same_skin_cluster* to be of the less important features.

Removing also the features *same_team_sum* and *same_skin_cluster* results in set 2 and gives the results of table 7.19 of Appendix B after 10 test rounds. The standard deviations (rounded to 5 decimal points) between the results of those 10 test rounds can be found in table 7.20 of Appendix B. Here it can be seen that the overall results have again been slightly improved in comparison with the previous results. But, the standard deviations are slightly increased.

Also removing the feature *intersection_numbers_found_x_percent* results in set 3 and gives the results of table 7.21 of Appendix B. The standard deviations (rounded to 5 decimal points) between those results can be found in table 7.22. Again, the overall results have been slightly increased in comparison with the previous results and the standard deviations are slightly increased.

Since the features *average_hist_dist* and *average_feature_vec_dist* also belong to the least important features, removing each of those features is also tested. These are set 4 and set 5 respectively. Tables 7.23 and 7.25 of Appendix B give results of using those sets of features. The standard deviations (rounded to 5 decimal points) between those results can be found in tables 7.24 and 7.26 of Appendix B. It can be seen that for some methods the results improve and for some methods the results worsen comparing the results of set 4 to the results of set 3.

The overall results of set 5 are slightly worse than the results of set 3. The standard deviations between the results of both set 4 and set 5 are slightly increased.

To compare the results of different methods using different sets of features, the mean f1-scores are shown in Figure 5.7. Other scoring metrics like the fbeta0.5-scores, the fbeta2-scores and the AUC scores were also used but gave approximately the same results and showed the same trend. It is therefore chosen to only show the f1-scores. Each scoring method, that is used to combine the different scores for each connection, has a different marker and each choosing method, that is used to choose the winning connections, has a different color. The values on the x-axis stand for the feature-sets that are used. The different sets with their features can be found in table 7.27 in Appendix C. The vertical lines on each point represent the standard deviation between the values of the results of the 10 test rounds. In the plot, the standard deviations are divided by 10 to ensure visibility.

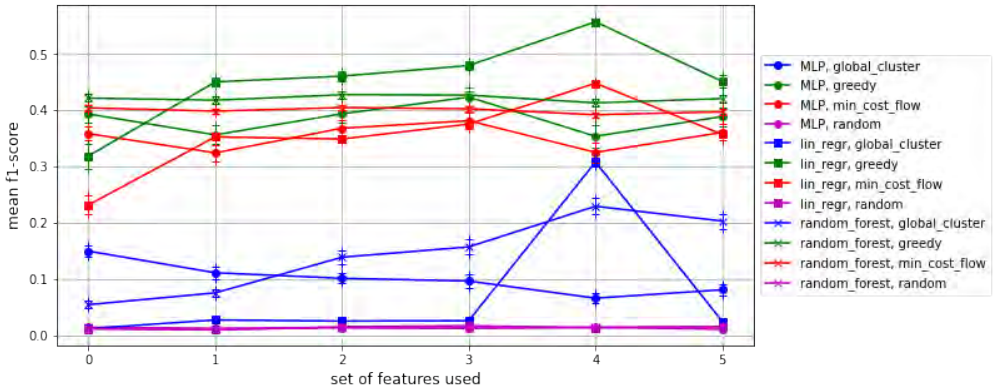

Figure 5.7: Mean f1-scores per set of features

6 Conclusions, Discussion and Future Work

6.1 Features that add value

As seen in the results section, overall, the mean scores of using the different feature sets are increasing up and till feature set 3. The mean scores of using feature set 4 and feature set 5 increase for some combination of methods but decrease for others. This makes sense, since up and till feature set 3, it was chosen to remove some features based on correlation and feature importance. Feature set 4 and 5 were to see if other changes also influence the results.

It can therefore be concluded that the combination of features of feature set 3 yields the best results. Set 3 consists of the following features: **dt**, **eucl_distance**, **speed**, **average_hist_dist** and **average_feature_vec_dist**.

This feature set consists of both track-based features as well as image-based features. It was expected that they are both needed to correctly connect connections. The features from this feature set are of high quality.

6.2 Best scoring and choosing methods

In the results section, it can be seen that the greedy and the min-cost-flow approach to choose winning connections with yield the best results. Furthermore, the global clustering approach yields better results than the random approach which was expected since every improvement should yield better results than a random approach.
It can also be seen that the three scoring methods yield similar results. It is therefore hard to tell which methods are best.

6.3 Conclusion

It can be concluded that, using a combination of both track-based features as well as video data-based features, different tracks can be connected such that a player's position is being tracked throughout the match. However, there is still a lot of room for improvement since I was not able to connect all tracks to each other to be left with a one-on-one mapping with tracks and players. The positive thing is, each step of the pipeline can be improved separately. So, by optimizing a small part of the whole pipeline, the results will already be improved.

6.4 Discussion

Evaluation is now conducted on only a one-minute video sequence. The longer the video sequence used for evaluation, the more tracks are found and thus more potential connections are found. This will result in more data to train and test models on. This could give better results.

With more data, also the difference between results could be tested for significance. This is now not possible due to lack of data. It would be interesting to see whether the drawn conclusions change when evaluation is conducted using enough data to test for significance.

Since models are only evaluated on a one-minute video sequence of an Ajax match, it is not known if and how the models generalize to other games between other teams. It would be best to train and test the models on matches of different teams. However, due to time constraints, this was not possible for this research.

The tracks used in this research resulted from the tracking software from TNO's Intelligent Imaging team. The quality of those tracks depends partly on the detection software they used. Different detection methods can be compared for this specific problem after which the best can be chosen. This will result in better and less missing detections which then results in better tracks. The better the detections and tracks, the better they can be connected. Now, the players are detected using the ACF detector. Other detectors that can be tested are for example the mask-RCNN detector and the YOLO detector.

6.5 Future Work

This research was focused on finding connections between tracks that are of the same person. However, this only tells that two tracks need to be connected. To complete the pipeline, winning connections need to be connected such that less and longer tracks result.

The results obtained in this research can be further improved by improving each part of the pipeline. All steps can be optimized separately.

This work could be extended to be working in real-time. The global optimization methods used to choose connections will not work anymore but the features can still be used.

7 Appendices and References

7.1 Appendix A

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,991	0,458	0,460	0,459	0,459	0,460	0,728	563	258
lin_regr, min_cost_flow	0,993	0,575	0,205	0,302	0,422	0,235	0,602	200	115
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0

Table 7.1: Results using only feature dt_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,992	0,494	0,480	0,487	0,491	0,482	0,738	544	269
lin_regr, min_cost_flow	0,993	0,670	0,239	0,352	0,492	0,274	0,619	200	134
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,991	0,332	0,135	0,192	0,257	0,154	0,567	229	76
random_forest, min_cost_flow	0,991	0,355	0,127	0,187	0,261	0,145	0,562	200	71

Table 7.2: Results using only feature eucl_distance_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,993	0,560	0,563	0,562	0,561	0,563	0,780	564	316
lin_regr, min_cost_flow	0,994	0,840	0,299	0,442	0,617	0,344	0,650	200	168
MLP, greedy	0,994	0,857	0,342	0,489	0,659	0,389	0,671	224	192
MLP, min_cost_flow	0,994	0,855	0,305	0,449	0,628	0,350	0,652	200	171
random_forest, greedy	0,993	0,616	0,335	0,434	0,528	0,369	0,667	305	188
random_forest, min_cost_flow	0,994	0,775	0,276	0,407	0,569	0,317	0,638	200	155

Table 7.3: Results using only feature speed_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,987	0,064	0,052	0,057	0,061	0,054	0,523	450	29
lin_regr, min_cost_flow	0,990	0,195	0,070	0,102	0,143	0,080	0,534	200	39
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0

Table 7.4: Results using only feature wp_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,992	0,240	0,011	0,020	0,045	0,013	0,505	25	6
lin_regr, min_cost_flow	0,992	0,385	0,018	0,034	0,075	0,022	0,509	26	10
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,990	0,049	0,014	0,022	0,033	0,017	0,506	164	8
random_forest, min_cost_flow	0,990	0,040	0,012	0,019	0,028	0,014	0,505	176	7

Table 7.5: Results using only feature average_hist_dist_normalized_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,986	0,072	0,064	0,068	0,070	0,066	0,529	503	36
lin_regr, min_cost_flow	0,990	0,065	0,023	0,034	0,048	0,027	0,510	200	13
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,988	0,007	0,004	0,005	0,006	0,004	0,500	275	2
random_forest, min_cost_flow	0,989	0,005	0,002	0,003	0,004	0,002	0,499	200	1

Table 7.6: Results using only feature average_feature_vec_dist_normalized_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,986	0,127	0,125	0,126	0,126	0,125	0,559	553	70
lin_regr, min_cost_flow	0,990	0,100	0,036	0,053	0,073	0,041	0,517	200	20
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0

Table 7.7: Results using only feature same_team_hist_normalized_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,986	0,098	0,094	0,096	0,097	0,095	0,544	542	53
lin_regr, min_cost_flow	0,989	0,040	0,014	0,021	0,029	0,016	0,506	200	8
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0

Table 7.8: Results using only feature same_team_cluster_normalized_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,987	0,126	0,118	0,122	0,124	0,119	0,556	525	66
lin_regr, min_cost_flow	0,991	0,335	0,119	0,176	0,246	0,137	0,559	200	67
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0

Table 7.9: Results using only feature same_team_sum_normalized_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,992	0,067	0,002	0,003	0,008	0,002	0,501	15	1
lin_regr, min_cost_flow	0,992	0,067	0,002	0,003	0,008	0,002	0,501	15	1
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0

Table 7.10: Results using only feature intersection_numbers_found_normalized_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,991	0,065	0,007	0,013	0,025	0,009	0,503	62	4
lin_regr, min_cost_flow	0,991	0,109	0,012	0,022	0,043	0,015	0,506	64	7
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0

Table 7.11: Results using only feature intersection_numbers_found_x_percent_most_normalized_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,992	0,071	0,002	0,003	0,008	0,002	0,501	14	1
lin_regr, min_cost_flow	0,992	0,143	0,004	0,007	0,016	0,004	0,502	14	2
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0

Table 7.12: Results using only feature intersection_numbers_found_x_most_normalized_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,992	0,188	0,011	0,020	0,044	0,013	0,505	32	6
lin_regr, min_cost_flow	0,992	0,188	0,011	0,020	0,044	0,013	0,505	32	6
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0

Table 7.13: Results using only feature intersection_numbers_found_x_percent_normalized_normalized

	accuracy	precision	recall	f1	fbeta 0,5	fbeta 2	AUC score	positive pred	TP
lin_regr, greedy	0,987	0,051	0,039	0,044	0,048	0,041	0,517	430	22
lin_regr, min_cost_flow	0,990	0,140	0,050	0,074	0,103	0,057	0,524	200	28
MLP, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
MLP, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, greedy	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0
random_forest, min_cost_flow	0,992	0,000	0,000	0,000	0,000	0,000	0,500	0	0

Table 7.14: Results using only feature same_skin_cluster_normalized_normalized

7.2 Appendix B

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.992	0.560	0.091	0.150	0.253	0.108	0.545	82.2	50.1
MLP, greedy	0.994	0.766	0.279	0.393	0.527	0.316	0.639	204.6	154
MLP, min_cost_flow	0.993	0.779	0.242	0.359	0.507	0.278	0.621	174.5	133.6
MLP, random	0.985	0.015	0.014	0.014	0.015	0.014	0.503	497.9	7.6
lin_regr, global_cluster	0.992	0.151	0.006	0.012	0.027	0.008	0.503	13.4	3.5
lin_regr, greedy	0.992	0.351	0.296	0.319	0.337	0.304	0.647	328.5	163.1
lin_regr, min_cost_flow	0.993	0.436	0.158	0.232	0.322	0.181	0.579	140	87.1
lin_regr, random	0.985	0.013	0.011	0.012	0.012	0.012	0.502	497.2	6.3
random_forest, global_cluster	0.992	0.750	0.031	0.055	0.107	0.037	0.515	25.8	17.1
random_forest, greedy	0.994	0.810	0.289	0.422	0.587	0.330	0.644	198.8	159.7
random_forest, min_cost_flow	0.994	0.818	0.271	0.404	0.577	0.312	0.635	183.8	149.5
random_forest, random	0.985	0.013	0.012	0.012	0.013	0.012	0.502	495.4	6.5

Table 7.15: Results using all features

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.00049	0.24541	0.07006	0.10273	0.14499	0.08022	0.0349	60.39463	38.73112
MLP, greedy	0.00082	0.07107	0.11491	0.14794	0.17423	0.12633	0.05734	80.55943	62.46243
MLP, min_cost_flow	0.00065	0.0697	0.08769	0.12439	0.16202	0.09951	0.04376	61.53093	47.55395
MLP, random	0.0004	0.00478	0.00441	0.00457	0.00469	0.00447	0.00223	13.32875	2.41293
lin_regr, global_cluster	0.00029	0.16011	0.00744	0.01417	0.03103	0.00918	0.00367	11.35488	4.06202
lin_regr, greedy	0.00087	0.2492	0.21256	0.22591	0.238	0.21719	0.10547	238.94362	118.32296
lin_regr, min_cost_flow	0.00057	0.30985	0.11186	0.16425	0.22863	0.12821	0.0557	96.60918	61.97033
lin_regr, random	0.00044	0.00419	0.00394	0.00404	0.00412	0.00398	0.00202	14.89817	2.00278
random_forest, global_cluster	0.0003	0.2023	0.03959	0.06648	0.11403	0.0472	0.01968	38.38634	22.02751
random_forest, greedy	0.0005	0.05311	0.05976	0.06676	0.05556	0.06343	0.02981	44.61639	32.22163
random_forest, min_cost_flow	0.00047	0.05371	0.0445	0.05303	0.04899	0.04818	0.02222	30.01777	22.55487
random_forest, random	0.00035	0.00342	0.00325	0.00331	0.00337	0.00327	0.00162	13.35997	1.71594

Table 7.16: Standard Deviations of results using all features

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.992	0.432	0.069	0.111	0.187	0.081	0.534	60	38.3
MLP, greedy	0.994	0.706	0.248	0.356	0.494	0.282	0.623	175.9	136.8
MLP, min_cost_flow	0.993	0.707	0.215	0.324	0.472	0.249	0.607	151.9	118.8
MLP, random	0.985	0.014	0.012	0.013	0.013	0.012	0.503	494.9	6.8
lin_regr, global_cluster	0.992	0.439	0.014	0.028	0.062	0.018	0.507	21.6	8
lin_regr, greedy	0.992	0.497	0.427	0.450	0.473	0.435	0.712	482.2	237.7
lin_regr, min_cost_flow	0.993	0.666	0.240	0.353	0.491	0.275	0.620	200	133.2
lin_regr, random	0.985	0.011	0.010	0.010	0.011	0.010	0.501	496.2	5.4
random_forest, global_cluster	0.992	0.846	0.042	0.076	0.148	0.051	0.521	32.8	23.6
random_forest, greedy	0.994	0.806	0.287	0.418	0.583	0.328	0.643	198.3	158.4
random_forest, min_cost_flow	0.994	0.814	0.266	0.399	0.572	0.307	0.633	181.4	146.9
random_forest, random	0.985	0.011	0.010	0.010	0.011	0.010	0.501	497.8	5.5

Table 7.17: Results using non-correlating features left thus far

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.00037	0.38511	0.08013	0.11757	0.17773	0.09141	0.03983	79.45229	44.65684
MLP, greedy	0.00084	0.25485	0.13048	0.1699	0.20415	0.14405	0.0651	94.07261	72.15077
MLP, min_cost_flow	0.0007	0.25533	0.10246	0.14465	0.18966	0.11612	0.05113	71.39164	56.1007
MLP, random	0.00032	0.00407	0.00353	0.00378	0.00395	0.00362	0.00176	12.62669	2.04396
lin_regr, global_cluster	0.00028	0.21616	0.00568	0.01078	0.02332	0.00701	0.00281	8.51404	3.23179
lin_regr, greedy	0.00071	0.04396	0.10944	0.07816	0.0501	0.09886	0.0543	132.41333	65.33002
lin_regr, min_cost_flow	0.00046	0.10255	0.03206	0.04891	0.07139	0.03719	0.01617	0	20.50908
lin_regr, random	0.00036	0.00362	0.0031	0.00333	0.00349	0.00319	0.00159	12.78715	1.7127
random_forest, global_cluster	0.00031	0.13934	0.04335	0.07266	0.12405	0.05166	0.02157	39.26208	24.1854
random_forest, greedy	0.00048	0.05411	0.06283	0.06837	0.05464	0.06615	0.03133	48.54791	34.37118
random_forest, min_cost_flow	0.00043	0.04998	0.04249	0.05017	0.045	0.0459	0.0212	30.18167	21.81972
random_forest, random	0.00038	0.00216	0.00224	0.00219	0.00217	0.00222	0.00113	12.44365	1.08012

Table 7.18: Standard Deviations of results using non-correlating features left thus far

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.992	0.573	0.058	0.102	0.189	0.070	0.529	45.1	32.3
MLP, greedy	0.994	0.829	0.273	0.394	0.541	0.311	0.636	185.5	152.2
MLP, min_cost_flow	0.994	0.830	0.247	0.368	0.523	0.285	0.623	167.8	137.7
MLP, random	0.985	0.015	0.013	0.014	0.014	0.013	0.503	495.4	7.3
lin_regr, global_cluster	0.992	0.422	0.013	0.026	0.057	0.017	0.507	21.3	7.4
lin_regr, greedy	0.992	0.517	0.432	0.461	0.488	0.442	0.714	470.5	240.6
lin_regr, min_cost_flow	0.993	0.659	0.238	0.349	0.486	0.272	0.618	200	131.8
lin_regr, random	0.985	0.017	0.015	0.016	0.016	0.015	0.504	496.1	8.2
random_forest, global_cluster	0.993	0.773	0.085	0.139	0.238	0.100	0.542	65.9	47.5
random_forest, greedy	0.994	0.806	0.295	0.428	0.591	0.337	0.647	203.7	163.1
random_forest, min_cost_flow	0.994	0.810	0.272	0.405	0.576	0.313	0.636	185.9	150.1
random_forest, random	0.985	0.015	0.013	0.014	0.015	0.014	0.503	495.6	7.4

Table 7.19: Results using non-correlating and more important features left thus far

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.00035	0.34718	0.05251	0.09063	0.16077	0.06312	0.0262	38.97136	29.66873
MLP, greedy	0.00074	0.08113	0.12306	0.16533	0.20675	0.13712	0.06146	81.05588	70.01397
MLP, min_cost_flow	0.00061	0.0866	0.10109	0.14558	0.19549	0.11524	0.05049	66.29362	57.7024
MLP, random	0.0003	0.00847	0.00735	0.00787	0.00822	0.00755	0.0037	12.58924	4.19126
lin_regr, global_cluster	0.00028	0.22511	0.00459	0.00875	0.01926	0.00567	0.00226	8.3006	2.63312
lin_regr, greedy	0.00076	0.05399	0.11425	0.08243	0.05633	0.10334	0.0567	137.09547	68.15538
lin_regr, min_cost_flow	0.00046	0.10219	0.032	0.04881	0.0712	0.03712	0.01614	0	20.43853
lin_regr, random	0.00039	0.00481	0.00405	0.00439	0.00463	0.00418	0.00207	13.42013	2.29976
random_forest, global_cluster	0.00043	0.12732	0.08711	0.12579	0.17046	0.09935	0.04338	75.27498	48.85409
random_forest, greedy	0.0005	0.04411	0.06217	0.06689	0.05377	0.06517	0.03101	46.76905	33.89346
random_forest, min_cost_flow	0.00037	0.03222	0.03794	0.04435	0.03845	0.04088	0.01893	26.54326	19.68897
random_forest, random	0.00037	0.00855	0.00787	0.00819	0.0084	0.00799	0.00395	11.90891	4.32563

Table 7.20: Standard Deviations of results using non-correlating and more important features left thus far

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.992	0.645	0.058	0.097	0.167	0.069	0.529	44.9	32.5
MLP, greedy	0.994	0.855	0.299	0.423	0.570	0.339	0.649	197.4	165.5
MLP, min_cost_flow	0.994	0.863	0.256	0.382	0.543	0.295	0.628	166.6	141.3
MLP, random	0.985	0.014	0.012	0.013	0.013	0.013	0.503	494.9	6.8
lin_regr, global_cluster	0.992	0.365	0.014	0.026	0.059	0.017	0.507	21.5	7.6
lin_regr, greedy	0.992	0.527	0.459	0.480	0.502	0.466	0.728	491.6	255.6
lin_regr, min_cost_flow	0.993	0.709	0.256	0.376	0.523	0.293	0.627	200	141.7
lin_regr, random	0.985	0.018	0.016	0.017	0.018	0.017	0.505	496.1	9.1
random_forest, global_cluster	0.993	0.767	0.098	0.157	0.263	0.115	0.549	77	54.7
random_forest, greedy	0.994	0.799	0.295	0.427	0.588	0.337	0.647	205.7	163.2
random_forest, min_cost_flow	0.994	0.801	0.270	0.403	0.572	0.311	0.635	187.2	149.5
random_forest, random	0.985	0.014	0.013	0.013	0.014	0.013	0.503	496.1	6.9

Table 7.21: Results 3

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.00049	0.3883	0.07995	0.12272	0.18639	0.09274	0.03984	65.72071	45.07093
MLP, greedy	0.0009	0.05531	0.13182	0.17155	0.20968	0.14532	0.06581	87.53945	73.35038
MLP, min_cost_flow	0.00073	0.05721	0.09983	0.14349	0.1946	0.11368	0.04985	64.13735	54.86559
MLP, random	0.00038	0.00268	0.00243	0.00253	0.00261	0.00247	0.00123	14.69278	1.31656
lin_regr, global_cluster	0.00027	0.09431	0.00274	0.0052	0.01132	0.00338	0.00137	4.67262	1.64655
lin_regr, greedy	0.00065	0.04507	0.11726	0.07881	0.04351	0.1045	0.05817	141.15571	70.21902
lin_regr, min_cost_flow	0.00033	0.07315	0.02168	0.03331	0.04949	0.02518	0.01094	0	14.62912
lin_regr, random	0.00033	0.00326	0.0025	0.00284	0.00308	0.00263	0.00129	12.80148	1.52388
random_forest, global_cluster	0.0004	0.09439	0.09292	0.1312	0.17411	0.10516	0.04625	82.03929	52.11323
random_forest, greedy	0.00048	0.0428	0.05991	0.06396	0.05076	0.06265	0.02988	46.26506	32.74752
random_forest, min_cost_flow	0.00036	0.03802	0.03528	0.04149	0.03709	0.03806	0.0176	25.36314	18.45264
random_forest, random	0.00043	0.00506	0.00488	0.00494	0.00501	0.0049	0.0025	13.287	2.42441

Table 7.22: SD 3

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.992	0.402	0.037	0.066	0.129	0.045	0.518	23.9	20.5
MLP, greedy	0.994	0.685	0.245	0.354	0.490	0.279	0.623	159.9	135.7
MLP, min_cost_flow	0.994	0.693	0.215	0.325	0.472	0.249	0.608	138.1	119
MLP, random	0.985	0.015	0.014	0.015	0.015	0.014	0.503	494.8	7.6
lin_regr, global_cluster	0.971	0.202	0.739	0.308	0.234	0.465	0.856	2297.5	408.9
lin_regr, greedy	0.993	0.550	0.565	0.557	0.553	0.562	0.781	568.5	312.9
lin_regr, min_cost_flow	0.994	0.844	0.305	0.448	0.624	0.350	0.652	200	168.8
lin_regr, random	0.985	0.015	0.013	0.014	0.014	0.013	0.503	495.3	7.3
random_forest, global_cluster	0.993	0.758	0.150	0.230	0.357	0.173	0.575	116.3	83.5
random_forest, greedy	0.994	0.759	0.288	0.413	0.565	0.327	0.643	210	159
random_forest, min_cost_flow	0.994	0.777	0.263	0.392	0.557	0.303	0.631	187	145.5
random_forest, random	0.985	0.016	0.014	0.015	0.015	0.014	0.503	494.4	7.7

Table 7.23: Results 4

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.00041	0.44252	0.04787	0.08585	0.16467	0.05816	0.0239	31.42345	26.7426
MLP, greedy	0.00103	0.36227	0.15468	0.20955	0.27031	0.17263	0.07722	103.48747	86.29414
MLP, min_cost_flow	0.0009	0.36605	0.12581	0.18439	0.25716	0.14411	0.06282	81.90435	69.92695
MLP, random	0.00043	0.00563	0.00599	0.00582	0.00571	0.00592	0.00302	11.85842	2.75681
lin_regr, global_cluster	0.01269	0.08	0.03967	0.08088	0.08187	0.06252	0.01529	934.45424	24.55583
lin_regr, greedy	0.00044	0.02553	0.02908	0.0272	0.02617	0.0283	0.01461	24.6813	22.69826
lin_regr, min_cost_flow	0.00038	0.04287	0.02005	0.02632	0.03307	0.02215	0.01007	0	8.57386
lin_regr, random	0.00038	0.00411	0.00351	0.00378	0.00397	0.00361	0.00179	12.78932	2.00278
random_forest, global_cluster	0.00049	0.11041	0.11182	0.14604	0.17303	0.12347	0.05567	97.99099	62.98545
random_forest, greedy	0.0005	0.04808	0.06374	0.06843	0.05859	0.06663	0.03179	48.77613	35.12517
random_forest, min_cost_flow	0.00047	0.04874	0.03923	0.04932	0.05295	0.04302	0.01963	20.6989	20.39744
random_forest, random	0.0004	0.007	0.00618	0.00656	0.00681	0.00632	0.00313	12.85129	3.4657

Table 7.24: SD 4

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.992	0.459	0.046	0.081	0.155	0.055	0.523	31.2	25.3
MLP, greedy	0.994	0.757	0.267	0.389	0.543	0.305	0.633	174.9	147.6
MLP, min_cost_flow	0.994	0.754	0.239	0.361	0.523	0.276	0.619	156.8	132.1
MLP, random	0.985	0.012	0.010	0.011	0.011	0.011	0.502	495.9	5.8
lin_regr, global_cluster	0.992	0.344	0.013	0.025	0.056	0.016	0.506	21.8	7.2
lin_regr, greedy	0.993	0.551	0.419	0.451	0.492	0.429	0.708	436.5	233.9
lin_regr, min_cost_flow	0.993	0.694	0.242	0.357	0.503	0.278	0.620	191.8	134
lin_regr, random	0.985	0.016	0.015	0.015	0.016	0.015	0.504	498.1	8.1
random_forest, global_cluster	0.993	0.725	0.128	0.203	0.329	0.150	0.564	100.6	71.5
random_forest, greedy	0.994	0.766	0.294	0.421	0.573	0.334	0.646	212.6	162.5
random_forest, min_cost_flow	0.994	0.781	0.268	0.397	0.562	0.308	0.633	189.6	147.9
random_forest, random	0.985	0.017	0.015	0.016	0.016	0.015	0.504	496.9	8.3

Table 7.25: Results 5

	accuracy	precision	recall	f1	fbeta 0.5	fbeta 2	AUC score	positive pred	TP
MLP, global_cluster	0.00047	0.42164	0.05523	0.09544	0.17125	0.0664	0.02759	34.53115	30.67047
MLP, greedy	0.00087	0.27008	0.12546	0.16639	0.20918	0.13912	0.06265	81.44862	69.82391
MLP, min_cost_flow	0.00074	0.27285	0.09972	0.14401	0.19779	0.11368	0.04981	62.52786	55.3262
MLP, random	0.00036	0.00368	0.00307	0.00334	0.00354	0.00317	0.00157	11.82699	1.75119
lin_regr, global_cluster	0.00027	0.08948	0.00166	0.00315	0.00695	0.00204	0.00083	4.39191	1.0328
lin_regr, greedy	0.00037	0.04285	0.16112	0.11763	0.05925	0.14768	0.07989	187.79673	93.60372
lin_regr, min_cost_flow	0.00044	0.08695	0.04729	0.0643	0.07895	0.05302	0.02368	25.93068	27.66064
lin_regr, random	0.00034	0.00242	0.00198	0.00217	0.00231	0.00205	0.00101	12.31485	1.19722
random_forest, global_cluster	0.00051	0.1439	0.09564	0.13252	0.1659	0.10778	0.04763	82.76634	53.80675
random_forest, greedy	0.0005	0.05039	0.06358	0.06839	0.0585	0.06651	0.03173	46.81927	35.60041
random_forest, min_cost_flow	0.00048	0.0618	0.03766	0.04777	0.05339	0.04138	0.01885	21.8388	19.92458
random_forest, random	0.00037	0.00469	0.0039	0.00425	0.0045	0.00403	0.00204	12.63549	2.16282

Table 7.26: SD 5

0	'dt_normalized', 'eucl_distance_normalized', 'speed (m/s)_normalized', 'wp_normalized', 'average_hist_dist_normalized', 'average_feature_vec_dist_normalized', 'same_team_hist_normalized', 'same_team_cluster_normalized', 'same_team_sum_normalized', 'intersection_numbers_found_normalized', 'intersection_numbers_found_x_percent_most_normalized', 'intersection_numbers_found_x_most_normalized', 'intersection_numbers_found_x_percent_normalized', 'same_skin_cluster_normalized'
1	'dt_normalized', 'eucl_distance_normalized', 'speed (m/s)_normalized', 'average_hist_dist_normalized', 'average_feature_vec_dist_normalized', 'same_team_sum_normalized', 'intersection_numbers_found_x_percent_normalized', 'same_skin_cluster_normalized'
2	'dt_normalized', 'eucl_distance_normalized', 'speed (m/s)_normalized', 'average_hist_dist_normalized', 'average_feature_vec_dist_normalized', 'intersection_numbers_found_x_percent_normalized'
3	'dt_normalized', 'eucl_distance_normalized', 'speed (m/s)_normalized', 'average_hist_dist_normalized', 'average_feature_vec_dist_normalized'
4	'dt_normalized', 'eucl_distance_normalized', 'speed (m/s)_normalized', 'average_feature_vec_dist_normalized'
5	'dt_normalized', 'eucl_distance_normalized', 'speed (m/s)_normalized', 'average_hist_dist_normalized'

Table 7.27: Feature sets

REFERENCES

[1] Optasports website. https://www.optasports.com, 2018.

[2] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011.

[3] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Online multi-person tracking-by-detection from a single, uncalibrated camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1820–1833, 2011.

[4] S. Edgecomb and K. Norton. Comparison of global positioning and computer-based tracking systems for measuring player movement distance during australian football. *Journal of Science and Medicine in Sport*, 9(1):25 – 32, 2006.

[5] C. Gough. European football market size 2006-2018. https://www.statista.com/statistics/261223/european-soccer-market-total-revenue/, Oct 2019.

[6] W. Grossmann, G. Guariso, M. Hitz, and H. Werthner. A min cost flow solution for dynamic assignment problems in networks with storage devices. *Management Science*, 41:83–93, 01 1995.

[7] A. Hermans, L. Beyer, and B. Leibe. In defense of the triplet loss for person re-identification, 2017.

[8] G. Khan, Z. Tariq, and U. Ghani. *Multi-Person Tracking Based on Faster R-CNN and Deep Appearance Features*. 05 2019.

[9] S. Krile. Application of the minimum cost flow problem in container shipping. In *Proceedings. Elmar-2004. 46th International Symposium on Electronics in Marine*, pages 466–471, 2004.

[10] L. Leal-TaixÃl', M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese. Learning an image-based motion context for multiple people tracking. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3542–3549, 2014.

[11] Li Zhang, Yuan Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[12] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[13] W. Lu, J. Ting, J. J. Little, and K. P. Murphy. Learning to track and identify players from broadcast sports videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1704–1716, 2013.

[14] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22:761–767, 09 2004.

[15] R. Mesa-Arango and S. Ukkusuri. Minimum cost flow problem formulation for the static

vehicle allocation problem with stochastic lane demand in truckload strategic planning. *Transportmetrica A: Transport Science*, 13:1–22, 07 2017.

[16] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler. Online multi-target tracking using recurrent neural networks, 2016.

[17] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. *NIPS*, 01 2011.

[18] P. Ondruska, J. Dequaire, D. Z. Wang, and I. Posner. End-to-end tracking and semantic segmentation using recurrent neural networks, 2016.

[19] J. B. Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming*, 78(2):109–129, 1997.

[20] Penny4860. Svhn-deep-digit-detector. `https://github.com/penny4860/SVHN-deep-digit-detector`, 2018.

[21] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR âĂŹ11, page 1201âĂŞ1208, USA, 2011. IEEE Computer Society.

[22] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR 2011*, pages 1201–1208, 2011.

[23] T. L. M. R. K. Ahuja and J. B. Orlin. *Network flows: theory, algorithms, and applications*. 1993.

[24] K. Thanikasalam, A. Ramanan, and A. Pinidiyaarachchi. Online multi-person tracking-by-detection method using acf and particle filter. 12 2015.

[25] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr. End-to-end representation learning for correlation filter based tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[26] G. Wang, J. Lai, P. Huang, and X. Xie. Spatial-temporal person re-identification, 2018.

[27] T. Xiao, H. Li, W. Ouyang, and X. Wang. Learning deep feature representations with domain guided dropout for person re-identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[28] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. H. Hoi. Deep learning for person re-identification: A survey and outlook, 2020.