Universiteit Leiden
The Netherlands

# Computer Science

Player tracking in water polo

Kevin Noordover

Supervisors:
Dr. A.J. Knobbe
Dr. D.M. Pelt

MASTER THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

30/08/2023

## Abstract

Sports data science is a growing field in both the world of sports and in data science. Recent improvements in the field have made data science more accessible and allow coaches at lower levels or less popular sports to gain access to methods that have only been used at an elite level in more popular sports such as soccer and American football. Water Polo is an example of a niche sport that has not been covered in any public research. The dynamic nature, lack of electronic tracking options and the smaller size of the sport made it a less obvious candidate for data analysis.

This research combines both chromatic feature analysis and visual artificial intelligence to accomplish this task: first multiple steps of color masking are performed to filter out all the water, background elements and splashing. Next, blob detection is used to identify points of interests that a visual AI network will then attempt to identify. Because the ball is made up of bright colors that do not occur anywhere else in the field, only blob detection is sufficient to identify points of interests. By using a similar approach to identify the back line, goal post and the 5 and 6 meter marks, reverse perspective mapping can be used to convert video coordinates to field coordinates.

Points of interest are regularly misidentified or not discovered at all and no or multiple ball candidates are found. Using the frames with an expected number of players from each team as key frames and combining the data of those frames with the incomplete data of all other frames makes it possible to interpolate all players throughout a match.

The use of chromatic features for player detection proved to be a misstep. While the detector was accurate with stationary isolated players, it was less accurate with swimming players or groups of players. Large portions of the match had unidentified players as a consequence, which makes the output unusable for analysis. Field detection had a near perfect accuracy on the other hand, and ball tracking was also more successful. While the final product on its own is too error-prone, it can still be used to build a training data set for future deep learning approaches.

# Contents

# 1 Introduction

## 1.1 Background

Water Polo is a fast and dynamic game played in teams of six field players and 2 goal keepers. The number of offensive actions and goals is significantly higher than other team sports such as football and hockey, and this characteristic leads to a high data density in each match. At the time of writing water polo has not been subject to data analysis methods, which leaves a lot of untapped potential for players and coaches that can be extracted with relatively simple methods.

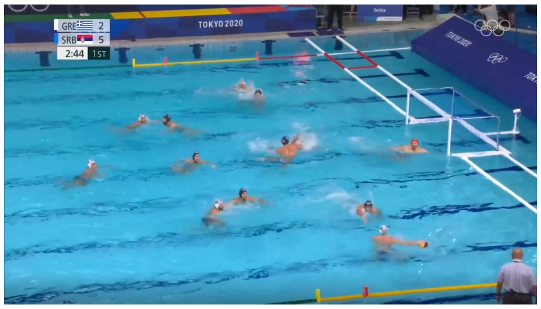


Figure 1: Frame from the analysed match

Data science plays a growing role in sports. Various technical advances in the recent years make it easier to collect and analyse large data sets in sports, and trainers and athletes are also starting to embrace the added benefit of introducing data science to achieve their goals. Sport data science can come in play in various stages of a training regime. It can be used to optimize an athlete's training schedules, identify habits that can lead to injuries and optimize strategy in both individual races and team sports. Methods range from researching new techniques to obtain and improve data sets to analyzing large data sets to find useful patterns.

A common practice in the field of sport data science is obtaining and analysing position data of players in team sports. Large position data sets can be built with relative ease with various methods such as electronic trackers and video data. Advanced tools such as neural networks can then in turn be used to label specific strategies and whether or not they were successful, a job that used to be done by trainers and coaches by manually analysing the video data. Data science not only automates this process, but it also allows for quick analysis of new strategies or other findings on all recorded past matches.

Data science is often introduced in large and popular sports where fast and constant improvement is a requirement on an elite level. Examples of such sports are soccer, hockey, speed skating and American football. These sports are practiced on a professional level, and sponsor and television money plays a large role for all involved teams. A lot of time and money is invested to keep pushing the boundaries further ever so so slightly, and the field of sport data science as a consequence plays an ever growing role in these sports.

## 1.2 Research scope

One sport with very little to no advances in data science is water polo. The dynamic game gets little attention in most parts of the world, with a sole exception in the former Yugoslav countries, and teams often have little money to spend for research purposes. Because water polo is a team sport similar to soccer and hockey, whereby the objective is to advance to the opponents goal without losing possession of the ball, laying a solid base should allow for rapid expansion in how data science can be used in this new sport.

## 1.3 Research goals

The largest hurdle in applying data science to a new field is creating a data set. Because no known research has been conducted on this topic, new data collection methods must be built and tested to create one. The main aim of this thesis is therefor obtaining the data required to perform a tactical analysis in water polo. This data consists of position data of both the players and the ball within the playing field, as well as data such as the current score, remaining time and game events like fouls. Because no kind of tracker can be attached to the players in the pool, this data must be extracted from video footage. An added benefit of using video footage is that position data of the opponent team can also be extracted with this method.

## 1.4 Structure

This thesis contains two main sections that cover the project. The first section describes the methodology used to build a data set containing position data of all players and the ball. It covers the steps taken to recognise the edges of the playing field, markers with known coordinates to generate a reverse perspective, and how players and the ball are detected and tracked. This section also covers how raw data is filtered and smoothed to remove as many errors as possible.

The second section describes how each of the units created in the first section are evaluated. It covers three main units: recognition of the field, movement of players and movement of the ball. Due to the nature of this project and lack of test data, it is not possible to generate numbers that describe the accuracy of the algorithms. Instead I opted for manual examination of the output whereby the original imagery at random moments is compared to the program's output.

# 2 Related work

## 2.1 Player detection and classification

In most sports, a common first step in player and ball detection is background subtraction. Common techniques used for background subtraction are based on either chromatic features ([16]) or motion-based techniques ([10]). Subtraction techniques based on chromatic features use established information such as colors of the pitch and ball to identify regions that are part of the background, while motion-based techniques differentiate between background and foreground elements based on movement of the foreground elements. Motion-based techniques often rely on the use of a static camera, since only non-moving objects are considered background. Simple high-performance examples of a motion-based technique include finding the difference between two consecutive frames, or a frame and the mean of a larger set of frames ([3]).

Yoon et al. [16] showed that chromatic features can not only be utilised to filter out green grass, but also to differentiate between field lines on the football pitch and players, and between players of different teams. Domain knowledge that field lines have on average a higher intensity value than players allows for a quick and easy approach to split all remaining elements in the frame into candidate regions for player detection and candidate regions for line detection. Shirts and shorts of the players are divided using a turnover point in the gray level histogram. The color histograms of both the shirt and the short of a player are then compared to all other players to split up the players into two teams.

Manafifard et al. [9] name differentiating between players and backgrounds with similar colors, partially occluded players and fragmentation of players into multiple regions due to variations in color as common weaknesses in pixel-based detection algorithms. A more advanced modern approach to the problem of player detection that does not suffer from these weaknesses is the usage of deep learning with a convolutional neural network ([6], [8]). Convolutional networks can be used to both detect and classify players and the ball without the need of chromatic or motion-based feature extraction methods. One disadvantage of the usage of convolutional networks is that they can be computationally expensive and require a large labelled data set to be trained upon.

A mixed solution with both background subtraction and deep learning has also been introduced. Kamble et al. [5] describes an approach that makes use of motion-based techniques to identify moving objects. These objects are then classified as one of three categories - ball, player or background - using a VGG-based classifier. Sah and Direkoglu [13] use a similar approach, but with fixed-size images that are extracted using a sliding window approach. These images are filtered using a set of manual rules and are then fed into a simple convolutional neural network. One disadvantage of both methods is that they will fail if two or more objects touch or partially occlude.

This project uses video data of a water polo match with an inconsistent background and high quality loss caused by video compression, which complicates detection and identification of points of interest. Because no public research has been conducted on water polo, the training data for neural networks would also have to be created first, which complicates the use of deep learning to locate and identify the players as well. The cap colors being indistinguishable from the background also makes classification using only chromatic features very difficult, a problem that is magnified by the poor video quality. A mixed solution is therefore favored for this problem. Chromatic differences between the pool itself and the players are generally large enough to detect players and can be used to create a training set for player classification.

## 2.2 Reverse perspective mapping

Detecting the position of players in a video frame is not sufficient to learn about the position of every player within the field. Certain background elements such as lines on the playing field and the edges of the field itself give the necessary information to the viewer about the whereabouts of each player currently on the screen. A player detection algorithm also needs to be able to extract this information from a frame in order to convert the frame coordinates to field coordinates.

A straightforward method of extracting features from the field is an edge detection algorithm to find the edges and other lines. Li et al. ([7]) apply a Hough transform to detect field boundary lines and penalty box lines on a soccer field. A decision tree based classifier then determines the current play position according to the lines' slope and position. Wang et al. ([14]) use a similar approach, but use a Competition Network consisting of 15 dependent classifier nodes for each of the 15 predetermined areas in a soccer field.

Yoon et al. ([16]) use a relatively simple decision tree to classify the recognised lines on the field. Because the intersections of these lines have known coordinates within the field, knowing which lines in the video frame correspond to which line on the field itself allows for mapping these frame coordinates to field coordinates. Three frame coordinates with a known field coordinate are then used to fill in a perspective transform matrix, which can then be used to convert any other frame coordinate to a field coordinate. Although a fourth coordinate is required to account for all forms of deformations caused by the camera lens, the results show that an accurate approximation of the perspective transform can be constructed using only three.

A water polo field doesn't have any clear line markers within the field boundaries, which complicates identification of points with known coordinates in the frame. For this reason a new method for finding these points has to be implemented. If these points can be identified, reverse perspective mapping is a useful method to map frame coordinates to field coordinates.

## 2.3 Detecting swimmers in videos

At first hand detecting swimmers may seem like a relatively easy task, as evolution of human detection algorithms over the years has resulted in excellent results. Detecting swimmers, however, comes with an additional challenge not seen in most regular circumstances: swimmers are mostly hidden under small and noisy waves and their surroundings are also very dynamic in nature, which makes detecting swimmers much harder than humans in other circumstances.

Benerab et al. ([2]) propose a method based on chromatic features using well-known pattern recognition techniques such as optical correlation and histogram based approaches. Similar simpler techniques have been proven helpful in field sports. One large disadvantage of their approach is that it leads to hand-crafted and pool-specific thresholds.

Other models with this goal use deep-learning methods to detect and classify swimmers in races ([15], [4]) make use of deep-learning methods. One named problem with this approach is, however, that there is little available training data for this task, thus making a sophisticated network would not be possible without first manually crafting a large data set from widely available broadcast-style overhead race videos.

Although data sets from past work are available, these data sets are crafted from swimming races only, which makes them less practical for tracking players in water polo footage. There exists no known past work that applied a deep-learning approach to track players in water polo, which means that training data has to be crafted for the purpose of this research to allow the use of deep-learning methods.

Papić et al. ([12]) introduce a method whereby a Fuzzy Support Vector Machine is used to classify certain areas using histograms of chromatic features of each pixel. They differentiate between splashing, bodies, caps and empty part of the pool. The authors propose the use of the Cb and Cr features for further research, since these features score the best together with the Fuzzy classifier. However, the HSI histograms differ the most between the four categories, which could make it the method of choice for other classification methods.

## 2.4 Player tracking

Detection of players alone can already provide meaningful information about how a game is played by the team as a whole. However, tracking individual players might provide more insight about pathing and individual performance. Because it is difficult and sometimes even impossible to correctly identify the person from image data alone, a tracking algorithm is required to link players between frames.

Visual tracking comes in various forms of complexity. Amin et al. ([1]) use a distance parameter to identify players from the previous frame in the current frame. In the event of occlusion between two players this method may fail, however, since the algorithm can not know which player is which after the players separate again. Because occlusion is a common event in most team sports, this method can not be considered an accurate approach to track players for an entire game.

Najafzadeh et al. ([11]) use a more sophisticated approach for tracking multiple players. A state vector of one player is determined to find the movement of every other player relative to the reference player. With this state vector, a motion model of all players is then created to predict the position of every other player. This method may still be vulnerable to occlusion if two players follow similar trajectories, and may therefor not provide added value over a simpler distance-based approach.

The lower speeds in water polo as well as noise in the position data of each player may negatively affect any motion-based player tracking algorithm due to random changes in trajectory in speed in the raw data. Smoothing the raw position data also is not possible if no tracking has taken place yet. Players that belong to the same team also rarely cross paths, which makes occlusion less of an issue. For these reasons a simple distance-based algorithm may work best in the scope of this project.

# 3 Methods

## 3.1 Pool edge detection

In order to extract accurate position data from the video recording of the match, a first requirement is the construction of a transformation matrix that can be utilized to convert coordinates in a video frame to coordinates in the water polo field. Because the video recording used for this research relies on a stationary, rotating camera to cover most of the field, it is not sufficient to manually label known locations in the field, since these locations will move to different positions within the frame over time. A program must be written that can track a set of at least four markers in the frame with known field coordinates, so that the transformation matrix for every frame can be calculated automatically.

An official water polo field for men is 20 meters wide and 30 meters long. The coordinate system that will be the target of the perspective warp will be 3000 by 2000 in size, with the origin being the top left corner of the field from the camera's perspective. The goal line of the white team will be represented by $x = 0$, and the goal line of the blue team will be represented by $x = 3000$. The side line at the top of the video is at $y = 0$. In order to get a proper understanding of how the perspective of the camera must be warped to an orthographic top-down view, at least two of these three lines must be identified in the video frames, since we know that all markers with known coordinates will exist on these lines. The fourth line at $y = 2000$ is not visible in many of the shots and is therefor not a target in this section. it is important to note that a right-handed coordinate system is used, which means that the origin is at the top left and that $x$ and $y$ increase towards the bottom and right.

First identifying lines rather than only the markers comes with a couple of advantages. The most important one is the identification of either the $(0, 0)$ or the $(3000, 0)$ coordinates in the frame, since these two points will exist at the intersection of the two lines. Having these two lines will also help with verifying possible markers that exist along these lines and if the current frame is a top-down representation of the field. The used footage often switches to other camera positions, and these do not give a proper overview of the player locations. If the algorithm is unable to identify at least two field edges, it is likely that the current point of view should be skipped, as it cannot provide meaningful information.

The first step in identifying the edges of the field is creating a mask that filters out all details that are of no interest. In the used video footage, the lines at the field edges are all either white, yellow, red or green and they are surrounded by blue water. Line detection algorithms work best with a black and white image input, so the goal is to create a mask image that creates clear borders from water to one of the aforementioned four colors. The mask should make it relatively obvious where the field edges are, while also removing as much noise as possible from both the background and the pool itself, since it contains black tiled lines.

Constructing the mask happens in two stages: in the first stage, the image is converted to an HSV format to allow filtering based on hue rather than red, green and blue values. This makes it possible to filter out all blue colors regardless of brightness or saturation. Since this also includes white, white and near-white pixels have to be included separately. In the second stage the mask is dilated to fill in small gaps and give all lines a stronger presence, which in turn makes it more probable that they are detected. Since lines that make up the field edges always exist between $y = 150$ and $y = 500$ in the frame, anything found outside can be discarded. Although this will include the part of the back line on one side of the goal, the remaining part of this line is enough to estimate the location of the back line.

The resulting mask image shows clear white lines of 4 to 10 pixels wide where the edges of the field are, as well as other lines behind the field edges such as the goal posts, the edge of the pool and the other lines just outside the field. Splashing is also detected, but due to the random nature of splashing, it does not create straight lines and will therefor not be detected by the edge detection algorithm. A Canny modifier is applied to this mask image to generate an image containing all edges found in this mask. These edges are then widened to 15 pixels in width so that the line detection algorithm also detects less than optimal edges. This is necessary because the mask may contain noise and because the camera itself can curve otherwise straight edges. The result of these operations can be seen in figure 3a.

A Probabilistic Hough Line Transform is then applied on this image. The transformation takes an image as input and then returns a vector of lines. Each line is represented by a 4-element vector $(x_1, y_1, x_2, y_2)$ where $(x_1, y_1)$ and $(x_2, y_2)$ are the start and end point of each line. Its limited to lines that are at least 150 pixels in length, while small 5 pixel gaps are allowed in case the input image

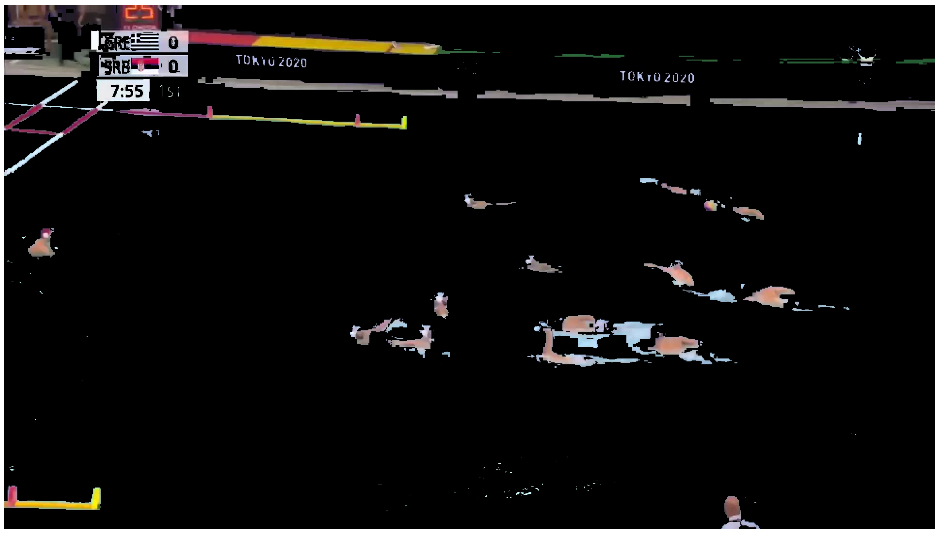


Figure 2: The resulting image after filtering out all water and other blue pixels. Although players cause noise in the output, the lines are clearly visible and identifiable.

contains small errors. The distance resolution is set to 1 pixel and the angle resolution is set to 0.1 degrees. This high resolution allows for many detected lines along the long wide field edges, while optimistically fewer lines will be detected around noise or other unwanted output.

The lines found by the Hough transform do not represent the actual lines as they are in the frame itself. As a result of the Canny operation, only edges are found rather than the center of the lines that make up the border of the field. The additional dilation also comes with the consequence that some detected lines can be up to 10 pixels away from the actual edge. Because of the way a Hough transform works, it detects multiple lines of various lengths where only one large one exists. The true line lies exactly between two line groups: one for each edge of the line. In order to find these lines, all detected lines with similar angles and positions need to be bundled. Important is that both edges of the field lines are put in the same bundle, since they all belong to the same line.

The reason that the edges of the field lines were extracted rather than the field lines themselves is that there is no perfect way of creating a mask containing only the field lines, and thus some objects such as reflections and the players may be part of the mask as well. A Hough transform will attempt to place lines throughout the image, and as long as lines fully cover a white area they are accepted. Blobs such as player masks will also appear as straight lines if the transform would be applied to the color mask itself, and the field edges themselves may also give one or more false lines due to small imperfections. Applying the Canny operation ensures that all detected lines are part of straight objects and greatly reduces the risk on false positives.

Whether or not two lines belong to the same bundle depends on two factors: the angle or slope of the line, and the position of the line. One horizontal line and one vertical that are close to each other may represent two unrelated edges that intersect at some point. Two parallel lines that are far apart are also likely to belong to two unrelated edges. The slope of a line is denoted by $a = \frac{\delta y}{\delta x}$, rounded to the nearest multiple of 0.05. Two lines have a similar slope if their value for $a$ is the same. For determining the position of a line, a second parameter $b = y - a * x$, with $(x, y)$ being the center of the line, is considered. If a line were to be drawn from this center point to $x = 0$, $b$

will be the y coordinate of the end point of this line. Lines are bundled only if they have the same value of $a$ and the same value for $b$ when rounding $b$ to the nearest multiple of 20. If the distance of a line to an existing bundle is greater than 350, a new bundle will be created. This occurs when two entirely separate edges are in line with each other.

Finally, to estimate the course of all edges found by the Canny operation, all lines inside a bundle are merged. The average line is denoted using the formula $y = ax + b$. $a$ is calculated by taking the average slope of all lines in the bundle and $b$ is calculated using the formula $b = y_{avg} - a * x_{avg}$, with $(x_{avg}, y_{avg})$ being the center of the bounding box that holds all lines in the bundle. $x_{start}$ and $x_{end}$ of the average line equal $x_{min}$ and $x_{max}$ of the bounding box.

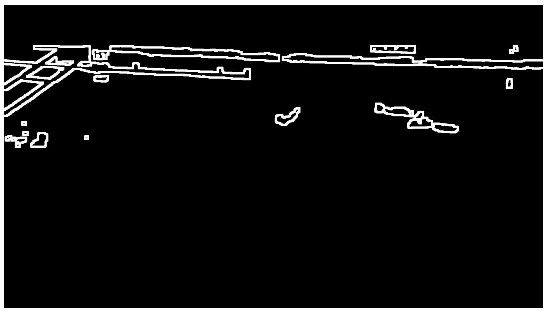
In order to get an understanding of where the edge of the field is rather than the edges of the marker that marks the edge of the field, the line bundles of both edges of this marker need to be merged as well. Two lines that represent their bundles are merged if and only if one of the ending points of one line is at most 60 pixels away from one of the ending points of the other line and the lines are at most 30 pixels apart at either the left or the right side of the frame. This means that lines not only have to be close together, but their slopes also have to be similar enough to not diverge within the borders of a frame. Merging two bundles happens using the same method as the one used to merge lines within a bundle, with the bounding box covering both bundles. The lines that follow from this process ($L$) are drawn in yellow in figure 3b.

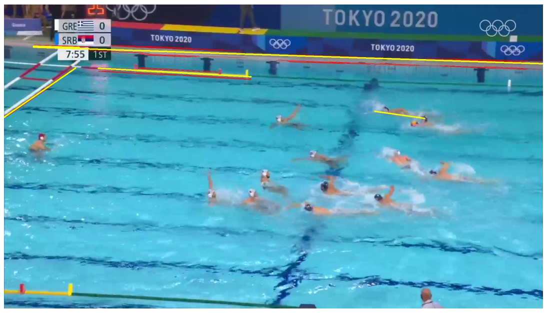
Once all lines have been bundled, the two lines that make up the back and side line of the field have to be identified. Other detected lines are often caused by the players, straight edges next to the pool and the second line behind the back line of the field. Because the field edges are made up of clear white and yellow markers, any line in $L$ that is shorter than 300 pixels is discarded. All lines $\in L$ with both $y_{start}$ and $y_{end} > 400$ are discarded as well. The side of the field that has to be identified is always in the top 300 pixels, and the back line of either side should end where it meets the side line. Therefor it can be assumed that lines that do not fulfill this requirement are not part of the field's edge lines.

The remaining lines are evenly split up in two subgroups, depending on their angle relative to the average angle of all lines in $L$. Lines detected from the video data are generally parallel to the sides of the pool, which means that there is a clear split between lines parallel to the x-axis and those that are parallel to the y-axis, even from the camera's point of view. One of the groups will contain the back line if it was detected, and the other group will contain the side line if it was detected. The two will never be in the same group. If any of the groups does not have any lines, a frame is skipped. From both groups the line closest to the bottom of the frame is selected as field edge. Lines from within the field are always filtered out by either the maximal $y$ requirement or the minimum length requirement, so any remaining false positive must be outside the field. The distance to the bottom of the frame is determined by taking the $y$ value of the line at the $x$ coordinate where it crosses the left edge of the image for lines with a negative slope and the right side of the image for lines with a positive slope.

## 3.2 Field marker recognition

Just marking the back and side borders of the field is not enough to get an understanding of what part of the field is currently visible. it is not possible to tell which of the two lines is the side line


(a) Frame edges



(b) Discovered lines

Figure 3: Figure 3a shows the result of the Canny operation on the color mask of an input frame after the lines have been widened. Figure 3b displays the lines as returned by the Hough transform in red and displays the estimated center of line groups in yellow.

and the current zoom level can not be recognised by the methods used so far. Although the back line is white, while the side line is yellow close to the back line, this distinction can sometimes be subtle and deceitful. To get a full understanding of the current angle of the camera, three points with known coordinates have to be identified. The fourth required point can be estimated using the three other points. Since one of the points is either $(0, 0)$ or $(3000, 0)$, right where back and side lines meet, only two more need to be found.

Water Polo has a number of markers at fixed known locations that can be used to get the perspective information. Examples are the two red markers at $(200, 0)$ and $(500, 0)$ and yellow marker at $(600, 0)$. No such marker exists on the back line however, and it is not possible to obtain perspective information if all markers lie on the same line. The back line does have a small red section, but its length is not standardised and its proximity to the corner of the field would enlarge small errors. Instead the goal post is used to determine the location of $(0, 850)$. The goal post is a white line perpendicular to the back line, so the intersection of both lines denotes the location of this field coordinate in the frame.

First, to learn if the current view is the left or the right side of the field, the yellow marker is looked for. To accomplish this, a Probabilistic Hough Line Transform is performed on a color mask with all blue filtered out. This mask is dilated by 8 pixels to fill in small errors. The discovered line that matches either one of the two field edge lines the best is considered to represent the yellow line that ends exactly at the yellow marker. The line that it matches the best also can be considered to be the side line from here on forward. The point of the line that is the furthest away from the back line is then considered to be the location of the yellow marker, with a 8 pixel correction towards the back line to account for the dilation.

In order to identify the goal post, a Hough Line Transform is performed once again. it is applied on a mask for all white pixels of the frame. Using Canny to limit the algorithm to straight edges rather than objects does not provide any meaningful advantage over using the mask directly, because it may cause the post itself to be missed if water splashing occurs around it. At the same time the position and rotation of the goal post can be estimated, thus making the additional false positives irrelevant. A dilation of 4 pixels is applied to the mask to close eventual gaps or other imperfections. The top 150 pixels and the bottom 500 pixels are discarded from the mask. The top 150 pixels are excluded because it contains mostly background elements outside the pool and the bottom 500 pixels are excluded because it takes away the risk that the bottom goal post is considered. The top goal post is never found in this area, so it comes at no cost to discard this area of the frame.

The goal of finding all white lines in the frame is not only to find the goal post, but also the back line of the field. The point of interest lies exactly on the intersection between the two lines, and using the already found back line could give unexpected results because it is determined using a different method. In order to find this intersection, all lines returned by the Hough Transform are compared to each other. If two lines after being extended infinitely intersect within the bounding box of either one of the two lines, the point where the two intersect is considered. Another requirement for the point is that it lies at most 15 pixels away from the identified back line, since this ensures that one of the two lines that are being compared belongs to this back line. Finally, the point closest to the $y = 500$ line that served as the limit of the mask is assumed to be the intersection between the goal post and the back line. In the video this point is the lowest intersection of any white line and the back line, so it'll always be identified with this assumption as long as it has been recognised as intersection.

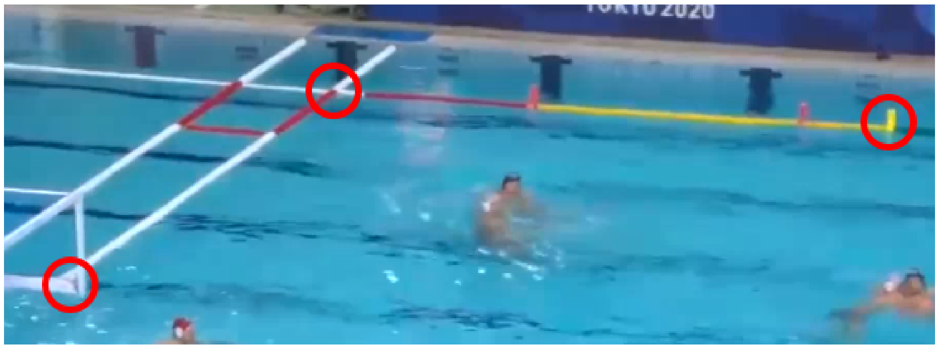


Figure 4: The three target points that can be used to apply a reverse perspective transform

With the three points discovered, a fourth point that completes the rectangle has to be estimated in order to get all information needed about the current camera angle. A naive approach would be to apply the same translation as the one from the corner of the field to the goal post to the yellow marker. This does not provide the correct location in the field, because two parallel lines in the real world will never appear as such from any perspective, unless an observer views them from a perpendicular angle. The two will otherwise intersect at a vanishing point. The exact position of the vanishing point is difficult to find without exact knowledge of the field of view of the camera and its angle. Another method is finding a line that runs parallel to the back line, but since the only line that fulfills this requirement lies only one meter away from the back line, this cannot be a reliable approach due to small errors being magnified.

To estimate the exact location of the point, the assumption is made that all lines parallel to the x-axis of the field are parallel in the video. This is done because the bottom side line and the top side line are only offset by up to 3 degrees in the most extreme cases. The imaginary 6 meter long line drawn from the goal post parallel to the sides would therefore be almost perfectly parallel to the side line in the video. This leaves only the angle $\beta$ of the line parallel to the back line to be estimated.

Given the goal post position within the frame ($A$) and the corner of the field ($B$), angle $\alpha$ equals the angle between the line $AB$ and vertical line $(B, (B_x, A_y))$. Given the position of the 6 meter marker $C$ and the imaginary point $D$, $\beta$ equals the angle between the line $DC$ and vertical line $(C, (C_x, D_y))$. Per the assumption above, lines $BC$ and $AD$ are parallel. In order for lines $AB$ and $DC$ to meet at the vanishing point far above the frame, $0 \leq \beta \leq \alpha$. Manual measurements from selected frames show that the difference between the two angles is always between 15 and 17 degrees. The definition that follows from this is $\beta = \alpha - 16°$. With the trajectory of both $AD$ and $CD$ known, point $D$ can be derived from the intersection of the two lines.

### 3.3 Video coordinates to field coordinates

So far all coordinates were coordinates of objects as they appear inside the 1920x1080 pixel frame. This means that the corner of the field may appear at $(700, 260)$, despite being at $(0, 0)$ in the field itself. This is the result of a perspective transform from the field coordinates to the location of the camera lens. By taking the four points of which both the coordinates in both the frame and the field are known, it it possible to calculate the perspective transformation that is needed to convert between the two coordinate types.

The perspective transformation takes the form of a $3x3$ matrix and can be used to apply an inverse perspective operation on any pixel in the input frame. When the inverse perspective matrix is multiplied with a 3d vector in the form of $(x, y, 1)$, with $x$ and $y$ being the coordinates of a pixel in the frame, the resulting vector in the form of $(s, t, 1)$ represents the real-life location of the object of interest, with $s$ and $t$ representing the x and y location within the field respectively. Once the matrix has been calculated for all frames, it'll be possible to get the real coordinates the identified players based on solely their position within the frame.

Depending on the current rotation of the camera, the field coordinates of points $A, B, C$ and $D$ are either $((0, 850), (0, 0), (600, 0), (600, 850))$ or $((3000, 850), (3000, 0), (2400, 0), (2400, 850))$. The half of the field that is currently visible can be determined by comparing the x-coordinate of the yellow marker to that of the field corner. If the yellow marker is at at a higher $x$, the camera is currently pointed at the left half, otherwise it is pointed at the right half. To calculate the transformation matrix to calculate the inverse perspective of the frame, a Gaussian elimination algorithm is used that takes the two sets of coordinates as input. This matrix is then calculated for every frame where the four positions could be identified.

Due to the nature of how all field lines and markers are being detected, the four coordinates that are used for inverse perspective mapping will not always represent the exact points in the field. Instead the found location may be up to 10 pixels off the actual point, which may cause small errors in the position data. In a similar fashion the field might not be properly identified at all, while four points of interest are still discovered. In that scenario the resulting matrix will be wrong in its entirely. Tracked players would appear to constantly jump around in a small area without actually moving and will sometimes disappear altogether if one of the markers was misidentified.

The first action taken to smooth out the array of matrices is to remove all local extreme values for every cell. The value of each cell is compared to the value in the same cell of 30 neighbouring frames. If the value is larger or smaller than any of its counterparts, it is replaced with an average of its neighbours. Although this method will not correctly cut out all outliers if two or more exist in close proximity, it will fully take out the single outliers and lessen the impact these grouped outliers will have on further improvement operations. Analysis of the raw matrix data that follows from the field marker detection shows that clusters of large outliers generally do not exist, thus making this an effective method of removing them from the data.

For the next operation, a larger neighbourhood of 40 frames - 20 in each direction - is used. This neighbourhood is used to calculate a moving average. Before this moving average is calculated, the algorithm tries to filter out possible remaining outliers. It does so by striving for a distribution of cell values where at least 20% of the values is above average and at least 20% of the values is below average. As long as this is not the case, the algorithm discards frames that contribute the most to this uneven distribution and recalculates the average. Not only does this filter out possible remaining outliers, but using this method will also keep sudden changes in camera position in the moving average, without smoothing this transition as much as a pure moving average would.

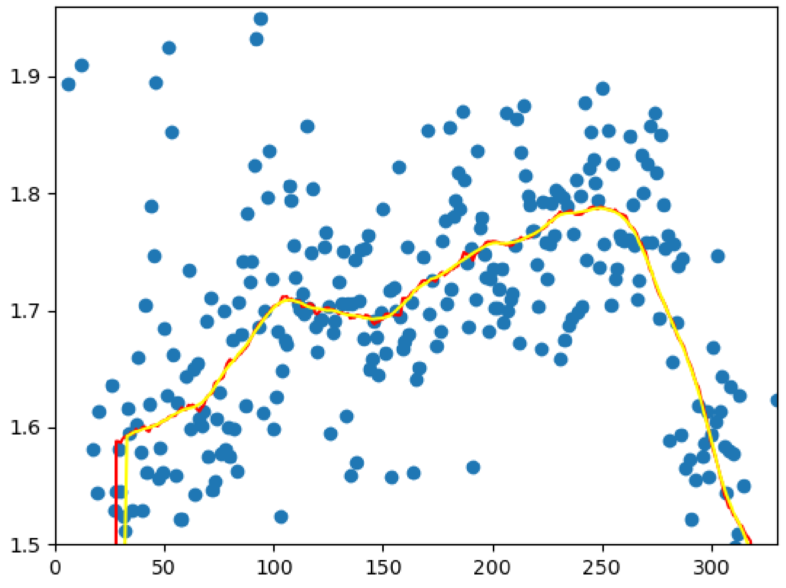


Figure 5: The value of the first cell in the transformation matrix for the first 330 frames of the video. The blue dots show the raw data that follows directly from the marker detection and inverse perspective steps. The graph's y-axis is tuned to show all values in the first 300 frames at the highest resolution possible. The red line shows the effect of a moving average with outliers discarded. The yellow line shows the moving average of the red line.

Because the generated moving average still contains small vibrations, the moving average of the result is applied to generate the final output. This second moving average is calculated from 10 neighbouring frames, 5 in each direction. The results all three smoothing steps is plotted in figure 5. The blue dots represent the value of the first cell in the matrix after all outliers have been filtered out. The large dispersion of the cell's value over time shows the need of applying a moving average with again a need to filter out outliers. The result of this smoothing operation is represented by the red line and the result of the second moving average is represented by the yellow line. The yellow line shows that the second step does not heavily influence the final value, but does filter out all bumps that were still present in the red line.

## 3.4 Player detection

The most difficult task in applying computer vision to a water polo game is recognising the actions in the field itself. Unlike other similar sports such as football, hockey or handball, water polo does not have a clear static field that serves as a background for the players. The water is in a constant motion and splashing from swimming players further complicates recognition of what exactly is going on. The dynamic nature of water also gives problems for digital video recordings, since those depend on lossy compression techniques such as MPEG. If a lot of splashing is going on, the overall quality of the video may drop massively, further complicating the process of finding players.

One property that makes a player stand out from the blue background is the color of their skin, so an algorithm that tries to find players needs to be able to differentiate between the blue color of the pool and the reddish color of the players. Other colors that may appear in the field are white from either splashing or a player's cap, red from the goal keeper's cap and yellow from the side line and the ball. Blue caps and suits would be filtered out with this color filter, but as long as enough of the body or some splashing is visible to the camera, players should be detectable.

A useful tool to find players based on a color-filtered image is OpenCV's SimpleBlobDetector algorithm. This algorithm attempts to find regions in an image that have different properties than its surroundings. These properties can include color and brightness. Discovered blobs can then be filtered further based on properties such as area, circularity and inertia. To make this detector focus solely on objects that fulfill the predetermined color demands, it is ran on a mask image that filters out the irrelevant blue colors. This gives full control over what is being detected and what to filter out.

The mask that is being used for the blob detector is based on two HSV color filters: the first filter removes all pixels with a hue between pure green and purple and a saturation of at least 30%. This filters out all blue pixels of the pool, without also filtering out the grayish and light blue pixels. A second filter filters out all pixels with a light blue hue, at least 30% saturation and at least 90% brightness. This filters out some of the splashing and reflections in the water to remove noise from the mask. The mask is then eroded by 5 pixels to further reduce the amount of noise and dilated by 5 pixels to counter the erode and to remove noise from within blobs. The bottom 90 pixels of the image are then also removed, since referees might give false positives.

As can be seen in figure 6, all players are clearly visible, but in some circumstances two players close together may form one single blob. Some background elements and lines stand out, especially at the very top, and may give false positives. Most of these objects however can be filtered out with the area parameter of the blob detector. The player with the smallest blob has a blob of size 1284 and the largest blob, the one containing two players in the middle, is 7196 in size. All blobs smaller than 1000 or larger than 10000 can therefore be discarded without missing players. The blobs that represent the lines can then be filtered out with a lenient inertia lower limit of 0.02. This ensures that blobs aren't as elongated as the lines are, filtering them out.

One problem with using a mask with a blob detector is that there's no perfect set of conditions that always work. Some players may be hidden behind splashes, causing the blob detector to miss them,

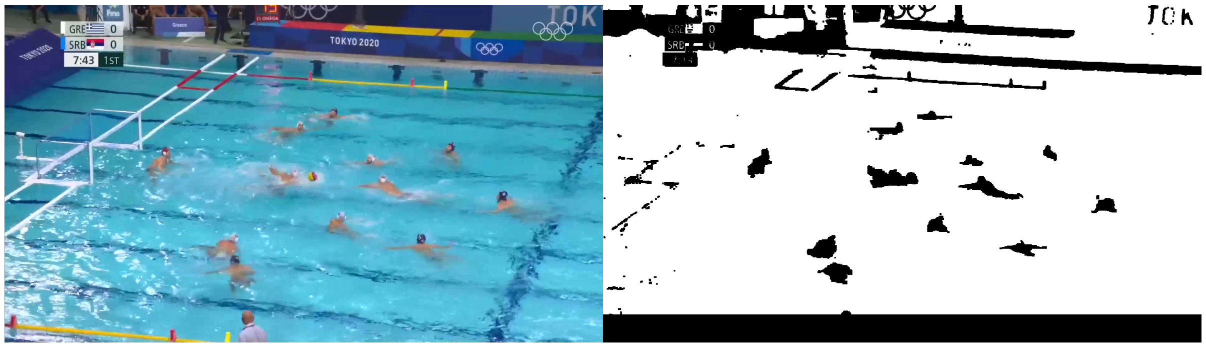


Figure 6: The original frame and the player mask used for blob detection. All players appear as black blobs on a white background, which makes them easier to detect than in the original image.

and in some cases noise such as a reflection or the goal can still pass off as a possible player. One blob representing two players can likewise be an issue. Because false positives can be filtered out at the next stage and because double players can also eventually be recognised as such, lowering the number of false negatives is deemed the most important. Players not recognised by the blob detector will forever be missing in the proceeding steps.

Each detected blob comes with two important properties: the center point and a size metric. This size is the diameter of a hypothetical circle that covers the entire blob. For the identification of players it is important to build bounding boxes that cover the entire player without including too much empty space. Each bounding box is given a size of 100x100 pixels. In the video the width of each stationary player is just under 100 pixels, so using this number as a minimum ensures that the entire player is covered by the bounding box even if the blob is of a very small size. An exception is made when the blob has a diameter $d$ larger than 150 pixels, in that case the width of the bounding box is increased to $1.6d$. The decision to only increase the width is taken because larger diameters are often the result of players swimming horizontally in the frame, so increasing the height would only add more water to the bounding box.

In a final operation, the average red, green and blue intensities for each bounding box is calculated and compared to values that each bounding box containing a player is expected to have. If a bounding box contains too much red, a color that the pool itself does not contain, the width of the bounding box is resized to $1.6d$ for blobs with a diameter larger than 120. Just like the scenario above, it is likely that the detected blob is of a swimming player if this color check did not pass. To avoid having a bounding box that does not contain the cap of a player, this bounding box is slightly increased in size. This resize action is not performed on all blobs larger than 120 pixels because larger diameters can also be caused by players extending their arms, and it is preferable to keep the bounding boxes as small as possible for further identification.

## 3.5 Player identification

With all bounding boxes found, their role in the field has to be determined. Bounding boxes can be put in one of four categories: white team, blue team, two opposing players, goal keeper. Because the blob detection algorithm was tuned to keep the number of false negatives as low as possible, there's also a chance that a bounding box is a false positive that has to be discarded. What makes this a challenging task is the lack of obvious characteristics that a bounding box may have, since both white and blue are common colors everywhere in the pool. Some players may also be stationary with most of their body under water, swimming with their body flat on the surface or in a fight with another player, hidden behind splashing. Because this step has to identify errors from the previous steps and because errors by this step may remain undetected, it is important to strive for a high accuracy, further complicating the task.

Due to the complex and dynamic nature of the sport, a convolutional neural network (CNN) is constructed to identify what team a player in a bounding box belongs to. The training data is generated by extracting the bounding boxes from the frame and then resizing them to 100x100. Each extracted image is then manually labelled as one of the five aforementioned categories or discarded if an image does not fit any. This may happen if two players of the same team or a larger group is part of the same blob. Due to the rarity of such events, it'd not be possible to train a neural network to differentiate between them and the other categories. The data set used to train the neural network contains 4545 images, of which 1409 white players, 1297 blue players, 566 player pairs, 333 goal keepers and 940 false positives.

The convolutional neural network is built using Keras. Keras is a deep learning API for Python, running on top of TensorFlow, an open-source platform for machine learning. The model has to be trained with very noisy and complex data, which requires more training data, but also with a relatively small data set. To avoid overfitting because of the relatively small data set, the model can transform any given input image by flipping it horizontally and by a applying a random rotation between -15 and 15 degrees. CNNs typically have three layers: a convolutional layer that identifies patterns in the image, a pooling layer to down-sample the output to summarize patterns and a fully connected layer to label found features. These three layers will also be present in the model used for player identification.

A common issue during construction of the network was that the validation accuracy would go up to 80% in a very early epoch, only to stop in the following epochs. The size of the training data set was enough to allow the model to quickly learn how to classify the simple most common images, though it would still have a high loss due to uncertainty about its decisions. It also struggled when input images contained splashes or if the cap of the player wasn't clearly visible. Even though the accuracy and loss converged early on, later epochs didn't show signs of overfitting, which means that the small random transformation to the input is enough to prevent it. The high accuracy showed that this method did have potential to finally be a useful classification method. Even if it doesn't correctly classify all players in every frame, the correct classifications should be enough to detect and solve errors.

The final model built for this task contained both image transform layers to randomise the input, followed by three convolution layers with two max pooling layers between them. The first convolution layer contains 32 kernels of size 3x3, the second contained 64 5x5 kernels and the final layer contains 128 9x9 kernels. The max pooling layers both use a 2x2 kernel. Finally two fully connected rectifiers (rectified linear unit, ReLU) of 32 and 16 nodes respectively lead to five output nodes activated by a softmax function. A ReLU activation function works similar to a linear activation function, except that negative inputs will result in zero. The advantage of using a sigmoid-like activation function for classification is that the five output nodes may each represent a likelihood that a given input can be classified as any of the five respective categories, which can be useful for further processing.

The model is trained using 25 epochs to ensure that the accuracy finally converges.

Training the network takes a considerable amount of time, but as the data set grows, a trained network will be capable of tracking players in numerous different pools and competitions without the need of retraining the network. For the purpose of creating a larger data set that can be utilised to create better networks the slow network times are acceptable. For this project this network will also be used to extract position data to avoid making the project scope too large.

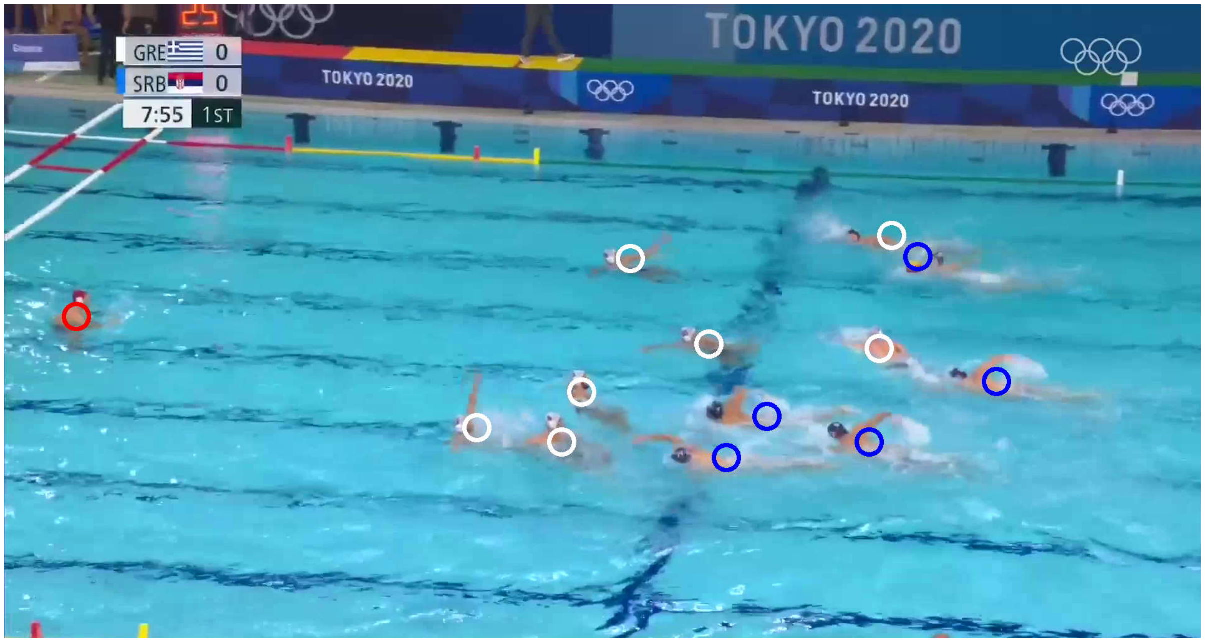


Figure 7: Result of both the player detection and player identification algorithms. All white players are correctly identified in this frame, but one of the blue players is wrongly assigned a white color.

## 3.6 Ball detection

Another important asset in water polo is the ball. The ball often consists of bright colors such as yellow and pink, which makes it easier to track the ball based on colors without the need of neural networks. However, just like players, the ball is also subject to splashes and other events that complicate recognition of the ball. The colors of the ball may also occur elsewhere in the pool: yellow is the color of some field lines and some players may have a skin color close to pink. Both false positives and false negatives may in this case lead to errors that are difficult to identify, leading to possible mistakes in the final path. The main aim of the ball detection algorithm is to recognise the ball in just enough frames to create a reliable path using interpolation, while keeping the number of false positives as low as possible. The ball may not be visible at all for some time, while invalid ball candidates may last for seconds, so simply relying on the longest lasting ball candidate will not be sufficient to identify the false positives.

Similar to the color-based detection algorithm used for field lines and players, a mask is generated that filters out all colors that don't fit a given set of requirements that can only be true for the ball. A blob detection algorithm is then used to find areas in this mask image that may be the ball. An important difference with player detection is that the ball consists of two distinct colors that may overlap due to the video compression. These overlapping sections may not be part of either color, thus resulting in two separate blobs. This may result in the ball not being identified correctly. To lower the number of false positives, it is also possible to add an additional requirement that both colors must be present in a close proximity to each other before a ball candidate is considered. That can be especially useful when one of the colors has a low saturation due to splashes, video compression artefacts or motion blur.

For mask generation a large number of filter rules is utilised. First, all pink pixels with a hue between 246 and 360, a saturation between 30 and 60, and a value of at least 40 are added to the mask. This broad range in hue means that some blue pixels may also be included, but with the saturation constraint these pixels are still more probable to be part of the blue border between the yellow and pink halves of the ball than part of the pool. Noise with these hue levels is often caused by very bright or very dark pixels. The result is finally eroded by 3 pixels to remove noise from the output. Because this output is combined with other similar colors and may already extend beyond the boundary of the ball due to the broad hue range, no dilation is required for proper blob detection.

The clearest feature of a ball is the yellow color. Yellow does not appear anywhere else in the field other than at the edges, so it allows for a broad color range with little risks of including unwanted objects. Two separate yellow masks are generated. The first one contains all orange and yellow colors with a hue between 30 and 80 and saturation of at least 30 and also serves as a reference point for the last three color masks. Only new colors within 20 pixels of a chunk of a 3x3 yellow area are considered, since these are more common throughout the field. These three colors include darker yellow or green, low saturation purple and high value orange. All these colors combined encompass most values found on or directly around the ball, thus lowering the chance that the ball is not recognised. Finally all pixels in the mask that are not at most 70 pixels away from any orange pixel that follows the rule above are pruned. The mask is then dilated and eroded by 5 pixels twice to remove eventual noise.

The blob detection on the mask takes into account that the blob size can range from an area of 150 to an area of 1200 and has an inertia ratio of at least 0.3. These two combined will exclude most noise and the field lines from the algorithm. An additional parameter is circularity, which is calculated using the formula $\frac{4\pi A}{d^2}$, with $A$ being the area of the blob and $d$ its perimeter. Since the ball is always the same round shape, with only the noisy video affecting it, the minimum circularity of each blob is set to 0.5. This still allows basic geometric shapes such as squares (0.785) and right-angled triangles (0.53), but excludes complex shapes with a smaller area. All identified blobs that follow these rules are then taken into consideration during post-processing. Frames with no identified balls often don't have a ball clearly visible at all, which indicates that the algorithm is working correctly. Frames with two or more balls either incorrectly label red reflections near the edge of the field or the back of a goal keeper. Any attempt to remove these false positives also lowered the success rate on frames where the ball is not clearly visible, which is not a preferred alternative.

## 3.7 Post-processing player data

With all players and ball identified, it is already possible to create an output of the field with player movements and ball movements clearly visible. The output generally matches the video data, but it still contains disappearing and re-appearing players and falsely identified players. Because the bounding box of each player is not of a static size, the position of each player may also fluctuate at a high speed around their true position. The ball is also still unknown at this point for similar reasons. For this reason the player and ball data has to undergo further processing to filter out the clear errors and improve the overall quality of the output. This process includes steps such as filling in single-frame blanks, filtering out temporary false positives and estimating a player's trajectory based on known position data.

The first step in improving the data is by filtering out likely false positives. First the immediate neighbourhood of each found player in a frame is scanned ten frames in each direction. True positives are expected to last for a long time, with short interruptions if a misidentification took place. So it can be assumed that at least five of the ten frames in either direction has at least one player in a 90 centimeter radius. Likewise, if a tested player passes this check but still has less than eight matches in the 20 frames surrounding the tested frame, it is still unlikely that this is a real player. With these metrics it is unlikely that real players that disappear for a number of frames are incorrectly pruned from the data and it may also miss false positives that are visible for longer than five frames. It does however filter out the short lived noisy data that would otherwise have affected further optimisation steps.

A similar check is then performed to the remaining players to see if there are any gaps that need to be filled in. If a given player appears at least three times in the next five frames, but doesn't appear in all five, it may have been missed by the earlier detection algorithms despite still being there. By taking the player's position in the final frame before it disappears and the first frame it reappears, its position in any of the intermediate frames can be estimated by performing a linear interpolation between these two frames. If more than one player appear close to the missing player in the frame after the gap, the position of the player closest to the original is considered. The radius for the false negative scan is lower than that of the false positive scan - 50 centimeter instead of 90 - to lower the chance that two points of different players or even a false negative are taken as base for the linear interpolation.

With small gaps in player movements filled in, the number of frames without any errors will increase to the point that these alone are sufficient to determine the player movements. A frame is deemed perfect if it contains exactly six white players, six blue players and one or two goal keepers and these perfect frames can provide a good understanding of the alignment and position of all players in all intermediate frames. Since perfect frames can still contain coincidental errors such as a missing white player, but a false positive identified as white player, each perfect frame is compared to each counterpart in the next 100 frames. If the majority of those frames is not compatible with the reference frame, the reference frame is removed from the list of perfect frames. Two frames are said to be compatible with each other if all players of both frames are compatible, meaning that their positions in the second frame are possible given with either speed limit of 2.5 m/s or a maximum distance of 50 centimeter.

Up until this point the alignment of players is not taken into consideration. Although a proper alignment is important, all optimisations so far have focused on position data only. Each new player was given the alignment of the reference player, and players may switch alignment from frame to frame. The identified perfect frames provide a useful base to determine the alignment of each player, since it can be assumed that each player in those frames is identified correctly. All intermediate frames are compared to the closest perfect frame preceding it and the closest perfect frame following it to find possible errors it may have. The comparison is the same as the one used to compare perfect frames, so if a white player has the wrong alignment for a frame, it'll register as both a missing white and as an additional blue player. If the estimated location of one of the missing players matches the location of one of the additional players, it is assumed that it is a misaligned player and fixed accordingly. The position of each missing player is estimated using a linear interpolation between the two selected perfect frames. Additional players that cannot be linked to any real players are removed altogether.

Since some frames may still contain errors, this entire process is repeated with a new set of perfect frames containing all fixed frames as well as the original set. That will however still not solve all issues. In some cases a player may be identified twice, causing them to create two identical data points in close proximity. Distance checks will not find this error, as both points can be the correct position of the corresponding players, so an additional step needs to be taken to remove these double points. Since double points are almost exclusively caused by one player creating two blobs during the blob detection phase, these points have to be in very close proximity. If this wasn't the case, these points would likely have been removed during the previous optimisation step already. If a frame has more than six white or blue players, it can therefor be assumed that two players of the same alignment that appear within 50 centimeter of each other are actually the same player. So by removing one of the two a frame can be fixed.

After the removal of the duplicates, the player locations for most of the match are fully accurate. Because identification of swimming players is proven to be difficult and unreliable, not enough data points could be obtained for counter actions. The video also does not show the counter actions from a single shot, which further lowers the number of suitable data points. For this reason, only offensive strategies are considered for the evaluation of the player detection algorithms. These strategies provide the most meaningful information about a team's offensive and defensive capabilities and can be subject to further data analysis in future work. Although counter actions too may come with some strategy, such cases are fairly rare, so no further attempts are conducted to identify players switching from offense to defense or the other way.

## 3.8 Player and ball tracking

The final step in player and ball detection is tracking individual players and ball candidates. Paths of the players and the ball can provide meaningful information about how the game is played and how strategies are deployed. It can show if an offensive team is playing a static or more dynamic strategy and how the defending team moves around depending on the position of the ball. In the case of the ball a tracking method can also be used to identify the correct ball if there are still more than one ball candidates, since the ball will always jump to the closest candidate rather than a new invalid one.

The difficulty in following players comes from determining which point in a frame belongs to which player. Points may have overlapping or non-continuous trajectories. In the scope of this project, these possible issues do not arise. Offensive strategies in water polo are often static in nature; players hold their position and movements that do happen are often quick actions that are easy to track as long as the player is identifiable both before and after the action takes place. If offending players do switch positions, they leave at least two meters of room to avoid blocking each other. This allows the use of a relatively simple and naive approach for player tracking that solely depends on correctly identified players.

For each perfect frame (frame $x$) the algorithm tries to track all points that are not already part of a tracked player. Tracking is done by scanning a circle surrounding the previously confirmed point of a given player. The radius of this circle is based on a speed limit of 10 centimeter per frame, with a minimum radius of 50 centimeter to account for small errors. If more than one point of the correct alignment exists within this radius, the algorithm picks the closest point that fulfills the following rule: given a confirmed point $A$ in frame $a$, a point $B$ in frame $b$ is only considered if there exist no point in frame $a$ that is closer to $b$ than $A$. If none of the points can fulfill this rule, the algorithm defaults to the closest point.

The tracking algorithm performs a recursive search both backwards and forwards and continues as long as it can find new points on a player's route. The recursive search ends if no point can be found for 25 successive frames. Trajectories that have less than 25 confirmed points are discarded. If there are gaps smaller than 25 frames, the position of a player in these frames is estimated with a linear interpolation between the last known location and the next known location. Each discovered trajectory is smoothed with a moving average of the 10 closest neighbours. All points that make up a trajectory are then removed from the pool.

The same method to track players is also used to track ball candidates. The maximum speed is changed to 50 centimeter per frame, with a minimum radius of 100 centimeter to account for the higher degree of uncertainty. The maximum gap between points and the minimum length of trajectories is changed to 20 frames, since the ball may be visible for only a short time before it is hidden again. This methods generates a set of potential ball trajectories that provide more information that can be used to determine the trajectory of the actual ball.

# 4 Experimental Evaluation

The work from section 3 can be divided into three units: one for finding and reversing the perspective, one for player movements and one for ball movements. Due to the novelty of data analysis in water polo, the accuracy of neither of these units can be quantified due to the lack of verification data, and manually crafting verification data would take a lot of time and effort. For this reason there's opted for random sample testing for all three units, as well as cherry picking moments in the game where the 2d representation of the output data contains obvious errors to analyse what went wrong.

## 4.1 Field perspective recognition

Field perspective recognition was conducted by finding four points of interest that were then used to calculate a reverse perspective matrix that could be used to calculate the field coordinates from the video coordinates. These four points are the back line, the side line, the goal post and the yellow 6 meter marker. The latter two are placed on their respective field line regardless of where they are detected. The original raw output of this operation appeared very noisy, which shows that this method isn't perfect yet and has a risk of containing errors that can not be neglected.

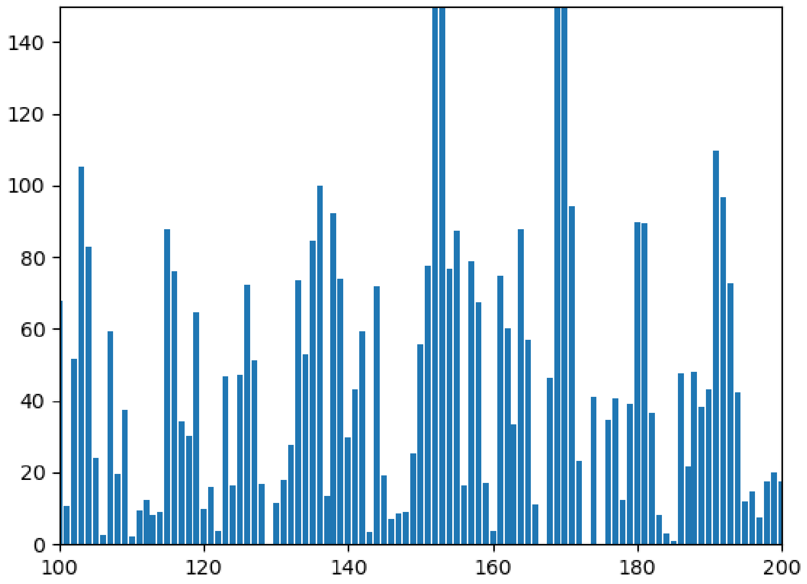
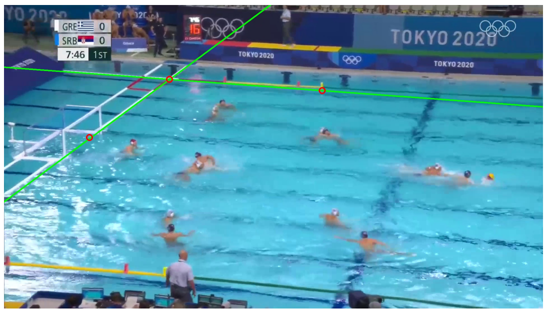


Figure 8: The Euclidean distance between the reverse perspective matrix of frame $x-1$ and that of frame $x$ for frames 100 to 200, a period with little camera movements. It shows how noisy the output is over time. Only large outliers ($distance > 1000$) are cut off.

To quantify the extent of how large the error from these imperfections is, two neighbouring frames need to be chosen that contain a large delta between the two extracted reverse perspective matrices. Figure 8 shows the Euclidean distance between the matrices of two neighbouring frames, which can be used to identify periods with a large amount of noise. Figure 5 shows that the period between frames 100 and 200 sees little camera movement, which makes the true distance between two adjacent matrices neglectable compared to the distance caused by noise.

The Euclidean distance between two reverse perspective frames generally only captures the x-translation and y-translation values in the matrix, cells 3 and 6 respectively. These values are important enough to capture the perspective change between frames, even though the values in other cells are neglected. The graph shows that point of interest recognition failed in frames 152 and 169, and also a number of periods where the data was significantly noisier than in the surrounding frames. Notable outliers are frames 114 and 180, because the two surrounding frames are very similar, which makes these two clear+ errors.

Both frames 114 and 180 contain a similar error: the location of the goal post is not correct. The white reflections in the water might have played a role in this misplacement, since these make distinction between the goal and the surroundings more difficult. Another common issue, with less severe effects, is a slightly wrong inclination of the side line. Because this inclination places the side line only a few centimeters further in the field, player tracking should not be negatively influenced


(a) Identified field points

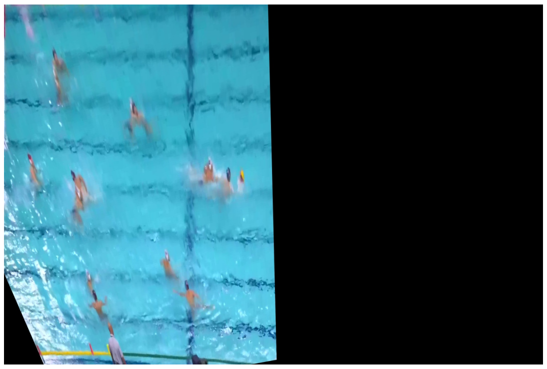

(b) Corresponding reverse perspective

Figure 9: Figure 9a shows the field edges as how the algorithm determined them in green, with the three points of interest circled in red. Figure 9b shows how these errors affect the inverse perspective.

by this small misalignment.

Figure 9 shows how these errors affect the creation of a reverse perspective matrix for frame 266. In the warped perspective, the field lines should appear at the edges of the image, but the side line at the bottom of the image is shifted up 60 centimeter and the side line at the top is shifted up by a few centimeter as well. These errors cause misplacements within the field that fall barely outside the error margin threshold, making a smoothing operation essential.

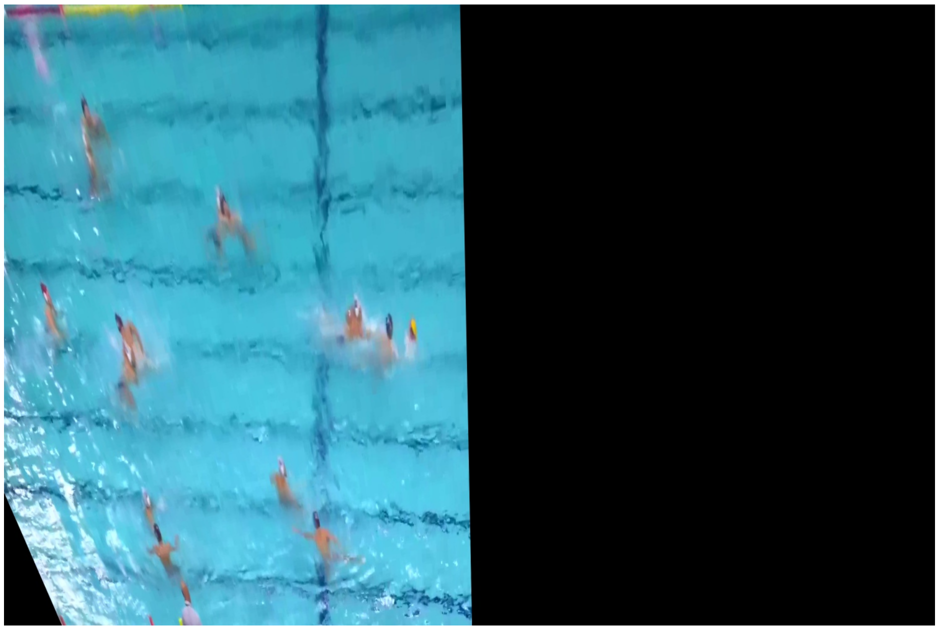


Figure 10: The result of smoothing the inverse perspective matrices on frame 266.

The result of the smoothing operations is shown in figure 10. The side line at the bottom is now moved to the edge of the image, and the other side line is now visible as well. One minor error in this version is that the corner of the field at the top left is no longer correctly placed in the corner of the warped image, despite this error not occurring in any of the neighbouring frames before the smoothing operation. Instead, this error is the result of the rapid change in camera perspective around frame 266. Sudden changes in camera movement get lost after smoothing, so for several frames errors such as this one may be introduced. Because this shift is only 30 centimeters in the most-affected areas - the corner of the field and near the side line at the other end of the frame, it can still be considered a huge improvement over the raw data.

One property of the warped frames is that the same location in the pool should always appear at the same location in the warped frame no matter the current orientation of the camera, even in the case of the dark blue lines at the bottom of the pool. This property makes it possible to test the overall accuracy of the generated matrices by creating an average image. The amount of blur in the pool lines and edges of the field may give an impression of the overall accuracy of the algorithms. In a perfect scenario, the only blur comes from the players and waves.

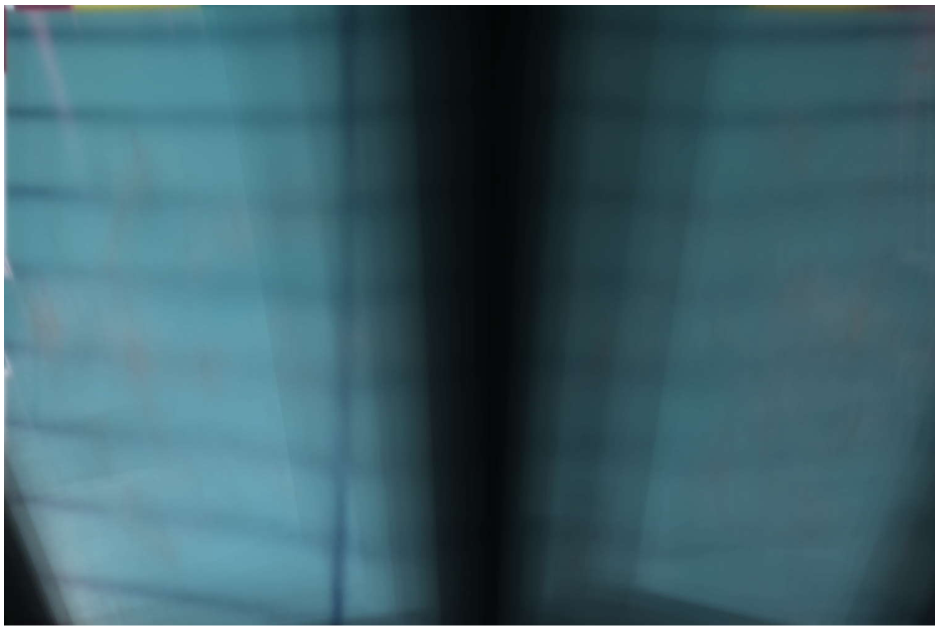


Figure 11: The average color of all frames after a reverse perspective is applied

The result of averaging all warped frames is shown in figure 11. The image shows that the left half of the field is recognised in a fairly consistent manner. The line at the top left is clearly visible, as well as the two goal posts. The right half of the field contains more errors, most notably the back line being placed behind the actual back line. This may cause errors of up to 50 centimeters in the player positions. The smoothing ensures that player tracking is not affected and the error is minor enough to not have a large impact on further analysis, but it shows that even the smallest changes to the environment can disrupt the localisation of certain key points.

## 4.2 Player movements

Tracking players has proven to be a difficult task. The angle of the camera makes it difficult to recognise swimming players because of the splashing and the team of discovered players was not always clear either. For this reason swimming players are often not found at all in certain frames with the blob detection algorithm, and players are regularly put in the wrong team if the convolutional network makes a wrong guess. The proposed solution to these problems is a more advanced tracking algorithm that also predicts the alignment of players or the location of players that went missing. This algorithm fully depends on the input data being correct in at least a handful of frames during each offense, which means that offenses with no correct reference frames may contain errors or be entirely discarded.

A simple way to quantify the accuracy of this tracking algorithm is counting the number of players. At any time there must be six field players of each team. Figure 11 shows that the camera is never aimed at the midfield, so we can assume that there has to be one goal keeper in view at all times as well. This method can not prove the validity of the found players, since the sixth white player in a frame may just as well be a reflection in the water, but it is a great tool to filter out frames with errors. The error of each frame is calculated with the formula $E = \Delta w + \Delta b + \Delta r$, with $\Delta w/b/r$ being the number of white, blue and red players respectively compared to the expected number.

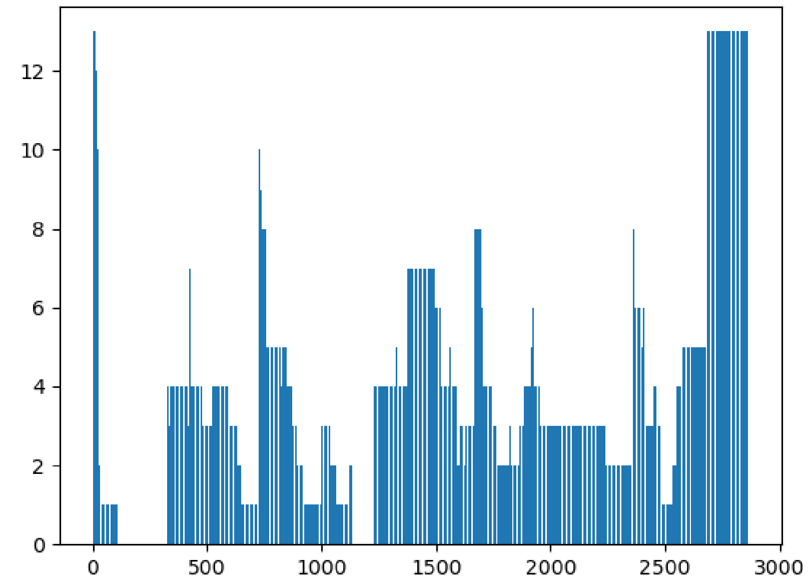


Figure 12: The error in number of players in all analysed frames of the match. Frames where the field could not be detected because the current view does not look down at the field are left out.

The resulting graph (Figure 12) shows that the player tracking algorithm is not as accurate as it should be to be a useful method of analysing matches. Early offenses regularly reach an error of zero, while later offenses rarely reach this point. Although it was probable that not every player could be found in every offense due to factors such as splashing and contact with other players, having only two offenses that reach an error of zero shows shortcomings of the current algorithms.

Figure 13 shows how only five players are detected. The six other players are very close to each other and as a consequence form one large blob that is discarded by the blob detection algorithm because of its size. This gives the tracking algorithm that follows very little information about player positions, which in turn leads to wrong estimations. Other offenses show similar situations, but swimming players are also more common later in the match, further complicating the process.

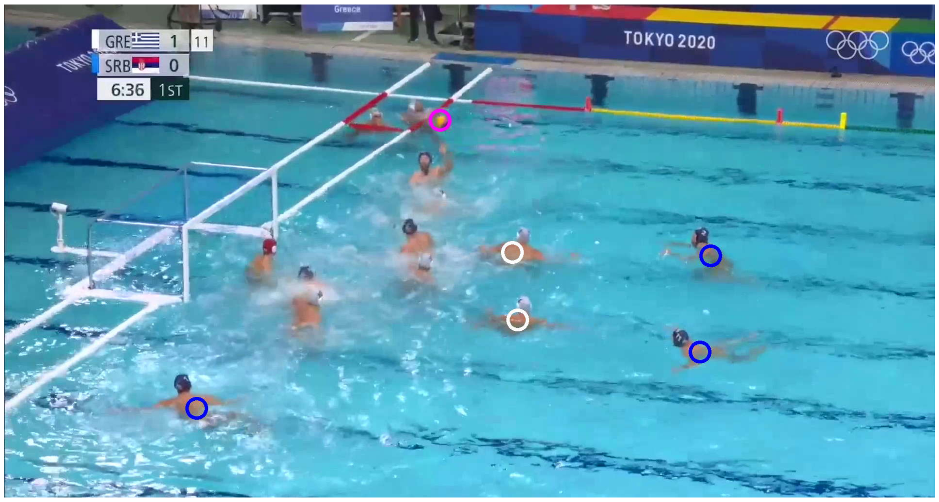


Figure 13: An example of an offense with a high error. Only five of the eleven players on the screen are found due to multiple players overlapping.

### 4.3 Ball movements

Ball tracking comes with the same set of challenges as player tracking. It requires the ball to be recognised in enough frames to draw a path, while false positives must be avoided. One additional layer of complexity is the given that there can be only one ball at all times. This means that in the event of more than one ball candidate for a given frame, a ball has to be chosen. There were also many frames where no ball could be identified at all because the player currently holding it was holding it under the water, which further complicates the process if a false positive pops up at the same time.

To determine the accuracy of the ball tracking algorithm, four random offenses - two on each half- from the match are used during which the ball is tracked manually. If the ball is not clearly visible during a frame, it will also not be included in the hand crafted data set. A straightforward error function would then be to calculate the distance from the tracked ball to the coordinates where the algorithm placed the ball, but this approach does not truly show errors in the algorithm for two reasons. Firstly, it is impossible to accurately track the ball in two dimensions if it travels in all three dimensions on the footage, so small errors will always appear. The second reason is that a misidentified ball should always be treated the same, regardless of the location of the ball relative to the output of the algorithm.

For this reason a simple error measurement is used to quantify the accuracy of the ball tracking algorithm: the number of frames that the ball is correctly tracked compared to the total number of frames. Frames during which the ball is not visible are not counted, even if the algorithm correctly predicted the location. All trajectories are verified manually, as creating a data set of true ball locations requires manually selecting the location of the ball for every frame, while manual labelling allows for accepting or declining a large range of frames.

The results (table 1) show that the ball recognition algorithm significantly outperforms the player recognition algorithm, despite the simpler approach. A possible deciding factor in the high accuracy is the high contrast between the ball and the background. Player bodies, often the goalie, being

|   | Ball visible | Ball found | Ball not found | Wrong coordinates | Accuracy |
|---|---|---|---|---|---|
| 1 | 207 | 207 | 0 | 0 | 1.00 |
| 2 | 224 | 159 | 0 | 65 | 0.71 |
| 3 | 310 | 297 | 0 | 13 | 0.95 |
| 4 | 331 | 263 | 5 | 63 | 0.79 |

Table 1: The number of frames the ball was visible, the number of frames the ball was correctly identified, the number of frames the ball is visible but not found at all and the number of frames the ball is visible, but something else is identified as ball. Each row corresponds to one offense.

identified as ball by the blob detection algorithm combined with the ball itself being not visible for a short duration was the most common cause for errors in the output. Players being labelled as possible ball is a common occurrence, but in most cases these players are only recognised as ball for a short period in which they also do not come close enough to the actual ball location. The tracking algorithm can switch to this player if the ball is not visible for long enough while the player is not too far away.

## 5 Conclusions and further research

This project used a combination of existing and new methods to track players and the ball in a water polo game, a sport that has seen little attention in the past when it comes to automated player and ball tracking. The continuous splashing and wrestling between players make water polo a more challenging sport for this task than many others, and new solutions had to be found to handle the low accuracy of existing proven methods. Existing methods for example often rely on background subtraction, a technique that can not be used in water polo.

The work was divided in three components: field recognition, player tracking and ball tracking. The field recognition component introduced a method that allows calculation of the reverse perspective matrix without the help of clear points of interest within the field. The player tracking algorithm was divided into two steps: a blob detector with a convolutional neural network to find and identify players, and a tracking algorithm that can handle incomplete and invalid data. Detecting the ball was done in a similar fashion, but without the neural network. Both player and ball tracking is done with an algorithm made to specifically deal with unreliable input data.

The method used for field recognition proved to be a reliable one. While the exact accuracy is difficult to determine without manually labelling the points of interest that were used to calculate the reverse perspective matrix, it was possible to quantify the accuracy by hand picking frames with a high distance to its neighbours and the local average. These outliers can give a good impression of the maximal error in the data, and how this affects the following steps of the project. Examination of these frames showed that the displacement that results from this error is minimal and that a rolling average of the matrices provides near perfect results. Very large outliers were also present, but these occurred only once in every 100 frames on average and can thus easily be discarded.

Player tracking was proven to be a more challenging task, one that could not be perfected in the scope of this project. A large challenging aspect of tracking swimming players is the existence of artefacts and other imperfections in the video caused by bitrate limitations. Swimming players in particular were very difficult to recognise because they were surrounded by splashing water. While the algorithm was able to correctly identify and predict all players in some cases, it struggled to correctly extract useful information in the majority of the offensive actions. Other than swimming players, groups of players in close proximity to each other were mostly to blame for this. The blob detection algorithm could not differentiate between the players, which in turn led to no or only one recognised player within this group.

This result was disappointing, but it can still be improved upon using some techniques used for this project. An important factor in using a blob detection algorithm over more modern approaches such as convolutional neural networks for object detection was the lack of training data. The training data used for player identification consisted of a fairly small data set that would not be sufficient for more advanced deep learning approaches. While the algorithm was not entirely accurate, it was still able to correctly track players in almost every offense that it was tested on. This makes it possible to improve the training set with each iteration, which in turn allows the use of more advanced techniques for player detection.

While the algorithm for player detection and tracking had a poor accuracy, its counterpart for the ball had significantly better results. The high contrast makes it easier to find and narrow down ball candidates and further processing steps have more accurate data to further improve the ball tracking output as a consequence. The algorithm however wasn't always perfect. Short time windows during which the ball was not visible while a player was seen as ball candidate by the blob detection algorithm was the most common situation in which the algorithm started to diverge from reality. Because the accuracy of the blob detector is already acceptable, creating a training set for an additional convolutional neural network to distinguish between the ball and a player is a logical next step to further improve this component.

Although the shortcomings of the player tracking algorithm led to an output that is not usable for further game analysis, these three components are still a large step in the process of introducing sport data science to water polo. Existing hand crafted methods rely on a solid unchanging background and consistent player poses, while more modern deep learning approaches require training data that does not exist yet. While the use of chromatic features has been proven problematic, it can still be used to automate the construction of a training set for deep learning approaches. The tracking algorithm also proved to be reliable enough to estimate many player's positions, as well as the location of the ball, despite the unreliable input data.

## References

[1] Jiten Amin. Soccer player tracking system. *International Journal for Research in Applied Science and Engineering Technology*, 6:3455–3461, 03 2018.

[2] Djamel Benarab, Thibault Napoléon, Ayman Alfalou, A. Verney, and P. Hellard. A novel multitracking system for the evaluation of high-level swimmers performances. volume 9094, page 90940A, 05 2014.

[3] David Higham, John Kelley, Chris Hudson, and Simon Goodwill. Finding the optimal background subtraction algorithm for eurohockey 2015 video. volume 147, pages 637–642, 12 2016.

[4] Nicolas Jacquelin, Romain Vuillemot, and Stefan Duffner. Detecting swimmers in unconstrained videos with few training data. 09 2021.

[5] Paresh Kamble, Avinash Keskar, and K. Bhurchandi. A deep learning ball tracking system in soccer videos. *Opto-Electronics Review*, 27:58–69, 03 2019.

[6] Jacek Komorowski, Grzegorz Kurzejamski, and Grzegorz Sarwas. FootAndBall: Integrated player and ball detector. In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2020.

[7] Jianguo Li, Wei Hu, Mingliang Sun, and Yimin Zhang. Soccer highlight detection using two-dependence bayesian network. pages 1625–1628, 07 2006.

[8] Keyu Lu, Jianhui Chen, J.J. Little, and Hangen He. Lightweight convolutional neural networks for player detection and classification. *Computer Vision and Image Understanding*, 172, 03 2018.

[9] Mehrtash Manafifard, Hamid Ebadi, and Hamid Moghaddam. A survey on player tracking in soccer videos. *Computer Vision and Image Understanding*, 02 2017.

[10] Pier Luigi Mazzeo, Marco Leo, Paolo Spagnolo, and Massimiliano Nitti. Soccer ball detection by comparing different feature extraction methodologies. *Advances in Artificial Intelligence*, 2012, 10 2012.

[11] Nima Najafzadeh, Mehran Fotouhi, and Shohreh Kasaei. Multiple soccer players tracking. *Proceedings of the International Symposium on Artificial Intelligence and Signal Processing, AISP 2015*, pages 310–315, 06 2015.

[12] Vladan Papić and Vladimir Pleština. Features analysis and fuzzy-svm classification for tracking players in water polo. *WSEAS Transactions on Computers*, 13:528–537, 07 2014.

[13] Melike Sah and Cem Direkoglu. *Evaluation of Image Representations for Player Detection in Field Sports Using Convolutional Neural Networks*, pages 107–115. 01 2019.

[14] Jinjun Wang, Changsheng Xu, Eng Chng, Kongwah Wan, and Qi Tian. Automatic replay generation for soccer video broadcasting. pages 32–39, 10 2004.

[15] Timothy Woinoski and Ivan V. Bajić. Swimmer stroke rate estimation from overhead race video, 2021.

[16] Ho-Sub Yoon, Young-lae Bae, and Young Yang. A soccer image sequence mosaicking and analysis method using line and advertisement board detection. *Etri Journal - ETRI J*, 24:443–454, 12 2002.