Tracking and Identifying People in Real-Time: Classifying team membership based on team uniform's color

Final report

8th January 2016

By:

Justinien Bouron

Bachelor, Year 3, Computer Science,

École Polytechnique Fédérale de Lausanne, Switzerland

and

Nicolas Roussel

Bachelor, Year 3, Communication Systems,

École Polytechnique Fédérale de Lausanne, Switzerland

Supervised By:

Dr. Horesh Ben Shitrit

Prof. Pascal Fua

Computer Vision Laboratory,

École Polytechnique Fédérale de Lausanne, Switzerland,

Abstract

This project is done within the context of our Bachelor's project at the Computer Vision Laboratory of the "École Polytechnique Fédérale de Lausanne" and its spin-off company PlayfulVision which specializes in real-time video tracking in sports environments. The goal of the project is to identify the team membership of players in a team-based sport by using the colors of their uniforms. This is performed on videos captured within the setting provided by the Sports Center of the "Université de Lausanne" and PlayfulVision. A single camera view is used at a time and the performance being aimed for is real-time, or as close as possible.

First a literature review was done to assess existing work which could be applied for this project. The conclusion of this review was that our work would be focused around the use of an object detection algorithm which locates a specified object and its parts on an image. More specifically the Deformable Part Models [10] algorithm, abbreviated DPM. In the context of our project it is used to find the players and their respective torso which contains the necessary information to establish their team-membership. Using only such an algorithm unfortunately does not yield the best results, therefore part of our project relies upon optimizing the use of the algorithm. These optimisations take the form of a background subtraction combined with a blob extraction. They allow us to run the DPM algorithm only on regions which are players. This results in a significant speed and correctness improvement.

The DPM algorithm cannot decide the team membership of the player but it allows us to filter out the irrelevant information on a frame, so that we only have the torsos, in our case the jerseys of the players. This specific information is then extracted to be used as a feature. Finally, every player's feature is given to a classifier to determine the player's team membership.

We select as features the color histograms of the torsos, more specifically, histograms of the hue values of the pixel colors in the HSV color space. In order to classify them, we compare them with pre-computed templates which represent each possible team. The template which is the most similar to a given player's feature determines his or her team membership. Correlation is the similarity metric we use for these comparisons.

The results of this implementation are extremely satisfactory when there is little to no occlusion of players, if the players are standing up straight (not bending over, not crouching, etc.) and finally if the players are neither too close or too far from the camera. However as soon as the postures are altered, the occlusion increased or the positions changed, the performance decreases significantly.

For future work, it would be interesting to evaluate the results when using multiple camera views to overcome the occlusion and position issues. Another interesting aspect would be to see the impact of using a DPM model trained specifically for different postures.

This report ends with acknowledgments and a synopsis of our personal experience.

Table of Contents

1 Project goal

The goal of this project is to achieve player classification based on team membership for a video input of a team sport. We are assuming that we can only use jersey colors to identify the team for which a person is playing. Therefore, the jersey number or the position on the field are not used to classify a given player. Performance-wise, real-time is the goal or at least as close to it as we can get. Also, the input on which our software runs will exclusively come from videos captured at the Sports Center of the “Université de Lausanne”.

2 Introduction

Person re-identification is the challenge of recognizing a previously seen person by one or several cameras. Most often, it is used for video surveillance, but it can also be applied to general-purpose person tracking. Player classification is closely related to person re-identification, as it also aims to recognize a person but only by considering a certain criterion, in our case team membership. Person re-identification is nothing new, it has been tackled previously, but what makes this project challenging is the fact that we are dealing with sports. It’s a very unique environment, because the probability of player occlusion from a given point of view is very high, and players adopt different postures : jumping, diving, crouching. These are all specificities to sports which usually hinder person re-identification.

3 Related work

The first part of our project was to find and read about current uses and implementations of similar projects. This research was mainly focused around two concepts, color-similarity and re-identification. The goal was to introduce ourselves to existing approaches in both of these fields. After having read several papers, we found a few different ways in which these topics were implemented.

Color similarity is a widely used concept when it comes to comparing multiple people and distinguishing them by their appearance, ie their colors. The main idea is to compare the colors extracted from an image to those from another image, and from there decide if there are enough similarities between the colors to establish that both people in each image are in fact the same single person. Of course, the extracted colors can be taken from the skin of the person or his/her clothes, or even both. Now more specifically for this project, the easiest way to separate teams in sports is obviously by their jersey’s colors. Color similarities are therefore a perfect criterion to establish team membership.

Norbert Fuhr [3] professor and leader of the Duisburg Information Engineering Group, gave lectures about color similarity and how we can compare images. His approach is to use color histograms of the images : the colors are quantized to reduce the number of possibilities and the histograms simply give the proportion covered by each color. Then, given two histograms we can define their similarity. A well-known similarity metric is the Euclidean distance, but there are many more.

Another approach given by Norbert Fuhr is to use matrices. Each image is a matrix, and each cell represents the color (quantized) of a pixel (or a group of pixels) of the image. By doing so we can consider the spatial distribution of colors. However this approach is not invariant to rotation, reflection and translation. A trick to overcome these restrictions is to use co-occurrences descriptors, quite more complicated, but invariant to those transformations.

François Fleuret, Horesh Ben Shitrit, and Pascal Fua [4], in their paper "*Re-Identification for Improved People Tracking*" use color distribution as a signature to distinguish player teams and referees. At the beginning of a video sequence, they create a template for each team (the two playing teams and the referees) by using several frames, this created template is a color histogram. In order to deal with closely spaced people, they compute an occlusion map based on the raw probability occupancy map. Then if a specific location is occluded with a high probability for a given camera view, they do not use it to compute color similarity, and instead use other camera views. Thanks to background subtraction, they also manage to only extract the pixels belonging to the players and the referees, and therefore do not take in account the crowd, walls, or floor when creating the color histogram for each player in each view. Finally the similarities between those freshly created color histograms and the templates are computed using the Kullback-Leibler divergence.
The same approach was used for their paper "*Tracking Multiple People under Global Appearance Constraints*" written along with J.Berclaz [5].

The goal of Michael Mason and Zoran Duric [6] is a bit different. They use color similarity to design an algorithm capable of identifying regions of a video containing a moving object, like a background subtraction algorithm. The main idea is to compare each frame with the frame of the same image without foreground objects. To do so, they overlay each frame with a 40x40 grid. The comparison between the background and current frame is made on a cell by cell basis, comparing their histograms to determine which cells contain foreground objects. The frames are 24 bits RGB images, but they only use the 4 upper bits of each color component to compute the histograms as the lower 4 bits can be very noisy depending on the camera. Another reason to use only 12 bits colors is because it greatly reduces the number of colors. They then use 2 functions, the first one is called histogram intersection and the other the chi-squared, both used to compare the histograms. After computing these values, they compare them with predefined thresholds, and decide if the two histograms are similar enough or not.

Rafael Munoz-Salinas, Eugenio Aguirre, Miguel Garcia-Silvente [7] take advantage of using multiple cameras instead of a single one, as seen in many other projects. However, color information is used to actually improve the tracking reliability. As seen before, they use color

histograms to compare objects, but unlike the others, pixels near the center are given more weight to reduce error. Unfortunately, there are not any details on how they compute the similarity of two images.

In their paper "*Multicamera tracking of multiple humans based on colored visual hulls*" Pashalis Padeleris, Xenophon Zabulis and Antonis A. Argyros [8] use color histograms as well. The samples for those histograms are taken from tracked blobs. Instead of using the RGB color space they use HSV, but only the hue and saturation are considered. By doing so, the color space is only 2 dimensional. The reason why the intensity component is not being considered is because illumination variations such as shadows or inconsistencies between the brightness values of the cameras change its value, even though it is the same color at any given time. If the color similarity does not provide sufficient disambiguating information (i.e. if color similarity is equivalently high for all candidates) spatial continuity of blob motion is also considered to establish the temporal correspondence. The similarity of the two color histograms is computed by another function, different to what we have seen in the other related works. Unlike the other works where the original template was left unchanged during the whole process, here it is updated at each comparison. This is done by averaging the person's histogram and the template, the person's histogram is given a weight of 1, whereas the template is given a weight of N equal to the number of frames since the beginning. This template update is done only if the similarity between the two histograms is above a certain threshold.

As mentioned previously, we also researched the field of re-identification. Usually, re-identification is used in an environment without overlapping camera views, and the goal is to recognize a person who has previously been seen. The difficulty of this problem remains in the fact that we can not use the person's position as the cameras' views do not overlap and must therefore find other criteria to recognize a given person.

For our project, we are not looking to identify each person individually, teams suffice. Therefore, we have researched existing re-identification work that focuses on color similarities and not on personal traits such as size, posture, accessories, etc.

Tamar Avraham, Ilya Gurvich, Michael Lindenbaum, and Shaul Markovitch [1] try to primarily use color histograms as features to recognize a person. Their approach is to start by getting a boundary box around the person, and separate it into 5 horizontal strips. Then for each strip, they compute a color histogram in the HSV space. From there, they use a SVM classifier to identify the person. The whole idea revolves around the fact that machine learning can be used to recognize a person when extracting his/her features.

In their paper "*Evaluating Feature Importance for Re-Identification*" [2] Hunxiao Liu, Shaogang Gong, Chen Change Loy, and Xinggang Lin have a similar approach. They also start by separating the person's image into several horizontal strips. But then, in addition to extracting color histograms from each strip, they look at the structure and textures of the clothes and also the covariance features. These criteria are then clustered to re-identify the person.

4 Approaches

As explained in chapter 2, we are looking to classify players based on their team membership. In order to achieve this we need to identify the player. However complete re-identification as seen in [1] or [2], is not needed in our context. We are not trying to know the identity of a player, but only his or her team. Therefore, to achieve this “partial” identification of a player we can consider less features than when usually attempting person re-identification. In fact, the only feature which is interesting for this project is the player’s jersey colors.

Throughout chapter 3 most approaches taken to re-identify a person arbitrarily separate the player into grid cells or strips. Colors are then extracted from these cells/strips and are used as features, which are then compared to previously seen features to determine the person’s identity. Our approach is very similar to these, the largest change resides in the color extraction which is not done on cells or strips but on body parts. Through the use of the Deformable Part Models [10] algorithm we can locate the positions of body parts for a person, and thus not use an arbitrary separation. By using such an algorithm, the colors extracted are much less posture dependant than when using arbitrary strips for example. This is an important factor for our environment, because in sports players tend to adopt different postures than a straight standing position which is a key assumption when arbitrarily separating into cells or strips. Of course, for this project the only relevant body part is the torso of the players since it contains the information about the jersey they are wearing.

As mentioned previously, we are not trying to achieve actual person identification - teams suffice. However, we are doing re-identification by considering that every player for a given team has the same “identity”. This could also be seen as using a person template for each team. Again, the only feature of a player which we are comparing with these templates are the colors of the torso. The form which this feature takes is similar to [1], a histogram of the colors in the HSV colors space. This histogram only keeps the hue values of the colors, the reasoning behind this will be explained at section 6.4.

The similarity metric used to compare the histogram of a given player with the histogram of each template is correlation. The highest correlation value obviously determines the player’s team membership.

Of course, these templates need to have been created beforehand. This is the only part of our project which demands training. The way we go about establishing player templates for each team is by using cluster analysis.

We first create a dataset containing features of all possible teams and then analyze it. This analysis allows us to not only cluster the data, but also more importantly to compute the centers of each individual cluster. It is these centers which are used as templates.

5 Player classification pipelines

The pipelines discussed here only concern the computation necessary to classify players on a single frame. These pipelines are then looped upon in order to go through all frames of the video input. Throughout this report, when pipelines are mentioned it is to these we are referring.

5.1 Trivial pipeline

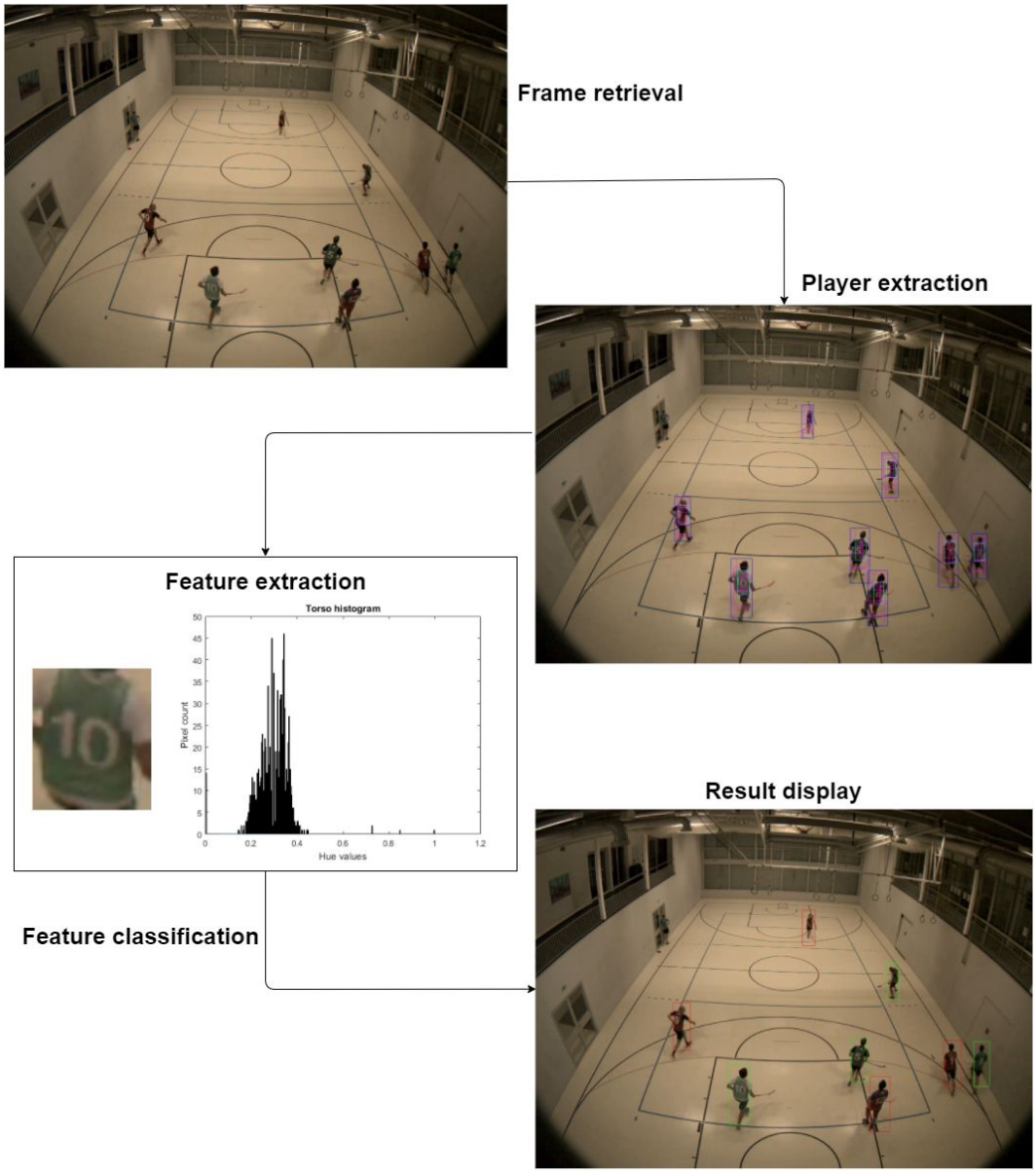

Figure 1 : Diagram of trivial pipeline.

The most trivial pipeline consists of 4 steps as illustrated in figure 1 :

1. **Frame retrieval**
   Retrieve the next frame from the video input.

2. **Player extraction**
   Run a DPM algorithm on the whole frame.
   This would return all the players on the frame and their respective body parts.

3. **Feature extraction**
   Extract the colors of every torso to create individual histograms which are used as a feature for each player.

4. **Feature classification**
   For every feature retrieved from the previous step, we give it to a classifier. The result of this would be the team membership for every player.

Such a pipeline, despite its simplicity, does not perform well enough to match our set goals mentioned in chapter 2. The main problem is the player extraction, which is performed too slowly to meet real-time results. The correctness is however considerably good, this will be discussed later in section 7.2.

5.2 Our pipeline

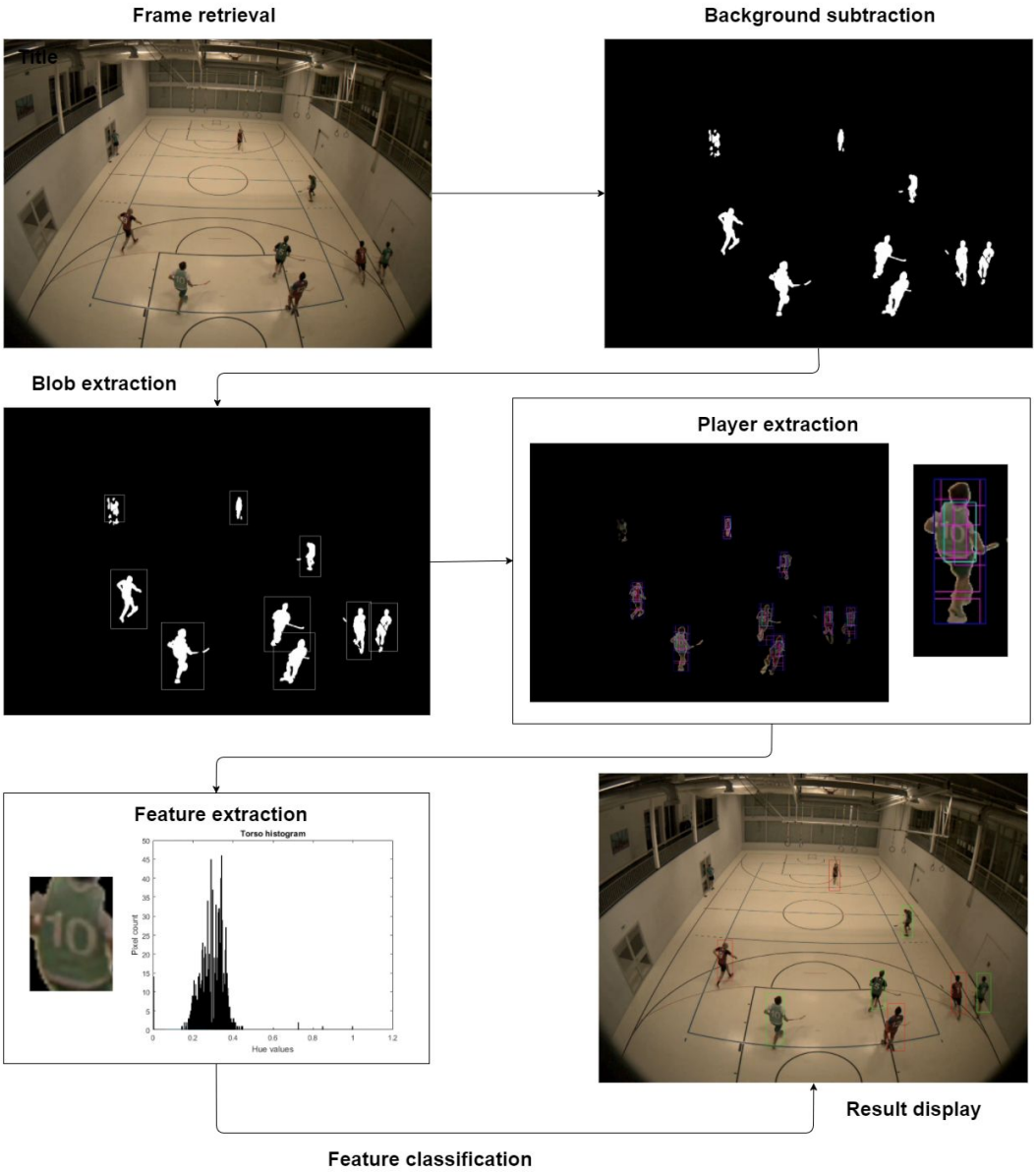

Figure 2 : Diagram of our pipeline.

The actual pipeline that we are using is the following, it is also illustrated on figure 2 :

1. **Frame retrieval**
   Retrieve the next frame from the video input.

2. **Background subtraction**
   Run a background subtraction algorithm on the retrieved frame.
   This allows us to filter out irrelevant information, and for most sports only keep the players - depending on the sport there might also be a ball or something else. The algorithm computes a binary image which highlights foreground objects.

3. **Blob extraction**
   Use a blob extraction algorithm on the binary image computed previously.
   The goal of this step is to associate foreground pixels to players. However, blobs might contain several players, when two players are touching each other for example, or none at all if the blob is a foreground object which is not a player, like a ball.

4. **Player extraction**
   Run the DPM algorithm on every blob.
   As said previously blobs extracted in the previous step might not be players at all, or contain several of them. By running a DPM algorithm on every single extracted blob we can separate blobs or delete them, depending on how many players they contain. The result of this step would be all the players for each blob with their respective parts described by the DPM model.

5. **Feature extraction**
   Extract the colors of every torso.
   From all the parts extracted in the player extraction step, the torsos are the only ones we are interested in since they contain the information of the jerseys the players are wearing. This step extracts the colors of each individual torso and wraps that information so that the classifier can handle it.

6. **Feature classification**
   Classify a player by feeding its feature to a classifier.
   We can create a classifier which decides players' team memberships given their jerseys' colors (the feature extracted in the previous step). The result of this would be the team membership for every player.

The main differences between our pipeline and the trivial one are the addition of a background subtraction and blob extraction algorithm before performing player extraction. These differences allow us to reach an important speedup that will be discussed later in section 7.4. The reasoning behind our pipeline which allows us the get the aforementioned speedup is that feeding the DPM algorithm with a whole frame is extremely inefficient. Most of the information on the frame does not concern players at all, and so the DPM algorithm runs useless computations on irrelevant parts of the frame. Our solution to this is adding steps 2 and 3 to our pipeline. These steps allow us to only run the DPM algorithm on parts of the frame which could potentially be players.

In the following chapter we discuss each step individually. We go over their implementation and performance as well as the reasoning behind their use.

6 Pipeline steps

Figure 3 : A frame extracted from a floorball video.

Throughout this chapter, we will be visually illustrating the results of each step of our pipeline when given with the frame shown in figure 3.

6.1 Background subtraction

This step acts as our main filter for our pipeline. As said previously, our goal with our pipeline is to allow us to run DPM only on relevant information - players. By using a background subtraction algorithm we obviously only keep the foreground which is composed of players and, depending on the sport, a ball and/or other equipment. In the case of football for example, it allows us to highlight pixels which concern players or the ball.

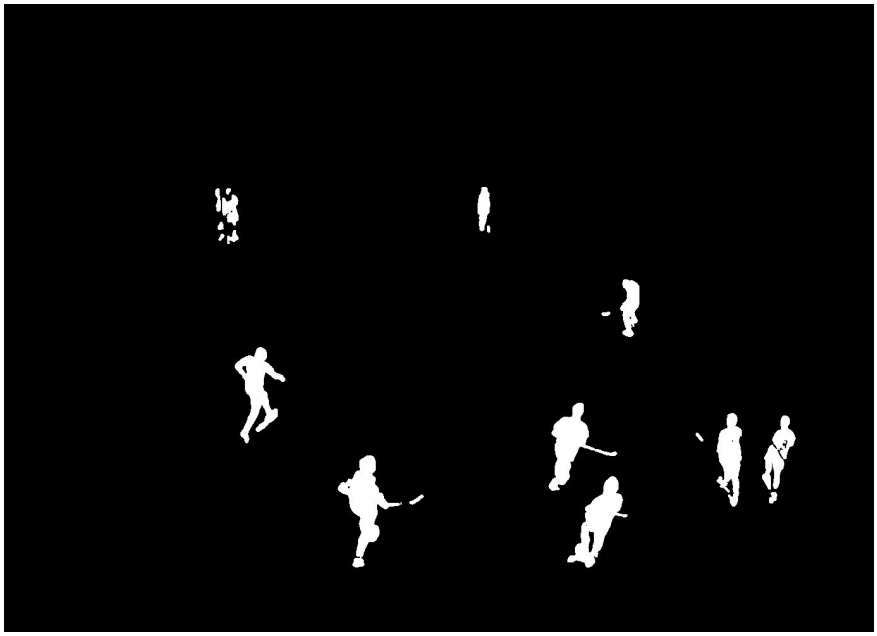

Figure 4 : The result of a background subtraction on a floorball video.

On figure 4 we can see the result of using a background subtraction algorithm on a single frame of a floorball video input. This binary image that we obtain highlights foreground pixels, which in this case are exclusively players and their floorball sticks.

The actual implementation of this step is the algorithm mentioned in [9], a gaussian mixture model for background subtraction. We are not using any training, therefore the video must start with a frame which describes the background (an empty gymnasium). In the software implementation, in order to overcome this inconvenience, we allow the user to pick a background image.

We tune the algorithm mentioned above by adding a static mask to it. Given the fixed setting and camera views that we have, we are able to create a binary image which highlights pixels that do not concern the playing field. This image is then applied as a mask to the output of the algorithm [9]. This is performed because around the field there could be moving objects which would be considered as foreground objects by [9], but have nothing to do with the players. Again, our goal with this step is to filter out information that does not directly concern the players on the field.

Performance-wise, this step has to be as lightweight as possible, allowing us to have the biggest margin for the following steps. Hence, real-time is a bare minimum. As for its correctness, the expectations are much lower. The boundaries of a foreground object do not have to be pixel perfect, but must not be smaller than the actual object as it might result in a loss of important information for the player classification. Obviously, poor correctness of this step will also have a domino effect on the rest of the pipeline.

6.2 Blob extraction

Blob extraction is used to identify regions in an image which have a certain property. In our case, the blob extraction algorithm is used on the result of the background subtraction and the property detected is foreground regions (the white pixels of the binary image). There are two main goals with this step. The first one is to filter out foreground objects which are not players, because these other objects have no purpose in achieving player classification. The second goal is to extract regions which are players, in order to pass them on to the player extraction step. These regions are meant to be as small as possible. Ideally, we would like to have blobs which contain exactly one player, unfortunately there are risks of player occlusions which lead to a single blob containing 2 players.

Using a blob extraction algorithm is useful as it considerably increases the speed and correctness of the next step. As mentioned previously, our pipeline differs from the trivial one because we are looking to feed the DPM algorithm of the player extraction step with as little irrelevant information possible. By doing blob extraction we can pinpoint areas which potentially could contain players, and therefore only give that information to the following step.

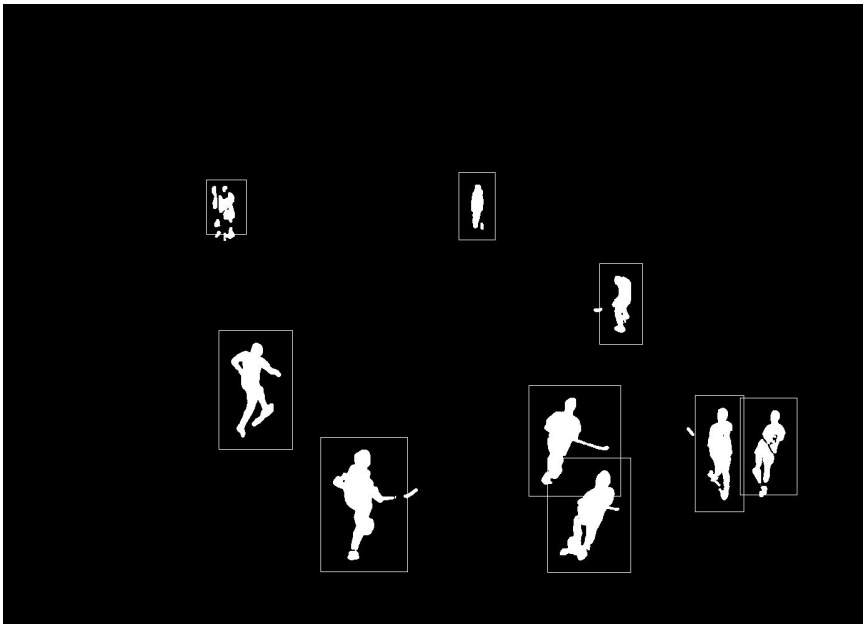

Figure 5 : The result of the blob extraction on a floorball video.

As for the implementation of this step, we are using a two-pass connected component labeling algorithm. The algorithm was changed in order to incorporate a pixel count and a position for every blob. This pixel count is used to filter out blobs which cannot be players due to their size being too small. However, for the same object, depending on how far it is from the camera, its blob size varies. To account for this we use the blob's position and an adaptive threshold based on it.

The correctness of this step is almost entirely based on the result of the background subtraction. For example, if the binary image is missing foreground objects or parts of a foreground object, this step can unfortunately not recover them. This step can however consider two separate blobs as being a single one. We implemented this for cases where a foreground object would be cut into several parts by the background subtraction. For instance, if a player on the binary image would not have his shoulder highlighted it would result in two separate blobs : one for his arm and one for the rest of his body. See the most bottom-right player on figure 5 for a graphical representation of this feature. By doing this we can recover flaws created by the previous step. The only downside of performing this additional feature is that two large distinct foreground objects close enough to each other are considered as a single blob. This event does not directly affect this step but might cause unwanted performance drops in the next step in particular, we will discuss this furtherlater on.

6.3 Player extraction

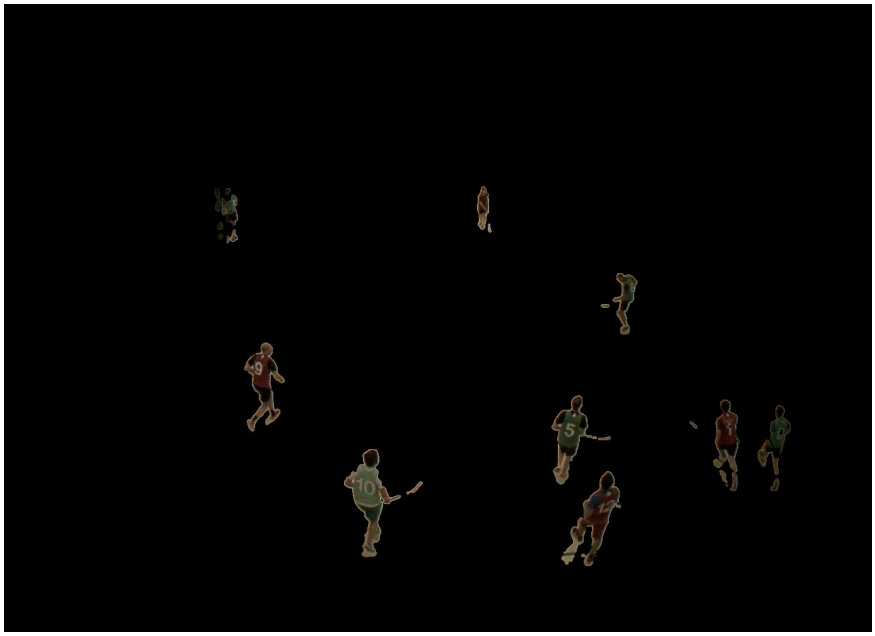

Figure 6 : The result of applying the background subtraction as a mask to the original frame.

This is the core step of our pipeline - every step leading up to this one was done with the intention of maximizing the performance of it. In this step we run the DPM algorithm on every extracted blob. As said in section 5.1, running DPM on the whole frame leads to a lot of unnecessary computations as the the regions which contain players are only a small portion of the whole frame. The blobs extracted in the previous step are foreground objects which could potentially be players, in most sports however these blobs are always players. Therefore, by running the DPM algorithm on these blobs, we are almost only doing necessary computations.

This is the main reason we achieve a large speedup compared to the trivial pipeline, the actual results will be discussed in section 7.4.

Before running the DPM algorithm on a blob, we apply the background subtraction mask to the original frame in order to keep the color information which is essential for the algorithm. By performing this, the players stand out more and consequently the DPM algorithm performs better. Figure 6 illustrates the result of this mask application. Blobs are of course cropped out of the full frame before being given to the DPM algorithm.

DPM uses a model which defines what is being searched for in the input image. In our case we feed it with a model of a person which contains 5 parts : both feet, both thighs, the head, left half torso and a right half torso. These models can be created by training on a large dataset. In our case, we are using a model which was trained on the VOC 2007 person dataset.

The result of the DPM algorithm are boxes for the positions of the players in a blob, and individuals boxes for each body part of the players, this is illustrated in figure 7.

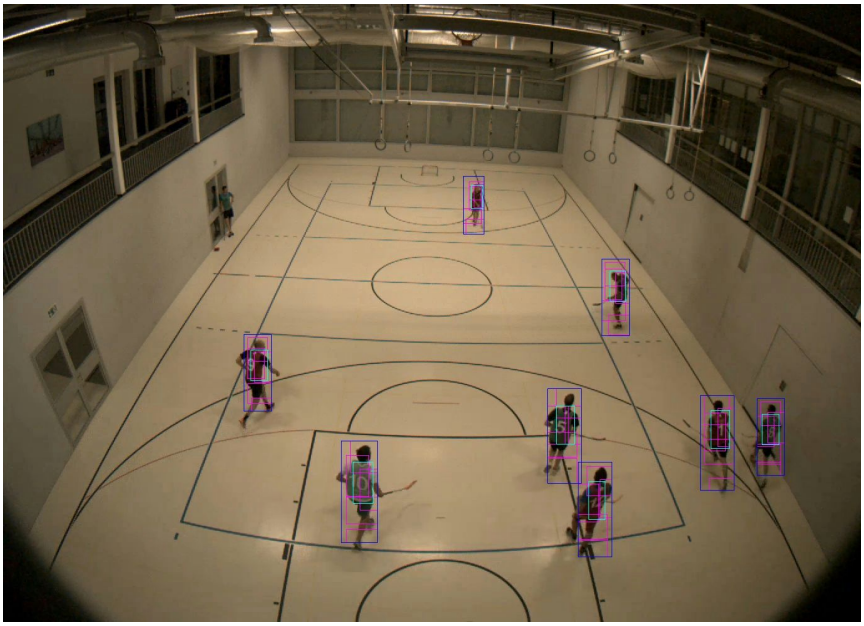

Figure 7: The result of the DPM algorithm on a floorball video.

As mentioned previously the model we are using has two parts which concern the torso : a left one, and a right one. The light-blue boxes seen on figure 7 are the regions we established as the actual torsos, they are calculated by averaging the position of both parts detected by the DPM algorithm.

The speed of this step is determined by the size of the image we feed it with. Again, this is why we only feed it with small blobs which are most likely players. The correctness of the player extraction and its parts relies on the DPM algorithm. We will go more into detail about the results of the player extraction in section 7.2.

In figure 7, we can see that the top left player which is near the door was not detected by the DPM algorithm. Figure 5 illustrates that there is however a blob containing him, thus the algorithm was given that region of the frame. When looking at figure 6, it becomes more apparent as to why the player was not found by the player extraction. His reflection on the door and the background subtraction result make that blob very confusing, even to the human eye. This is a perfect example of the domino effect mentioned in 6.1, because we are unable to recover from poor performance of previous steps (the background subtraction in this case). It should be noted that the best calibration of the background extraction does not yield such results, more on this in section 7.1. However, we still wanted to illustrate the importance of performing good background subtraction and therefore purposely degraded the calibration to create figures 4 through 7.

6.4 Feature extraction

At this stage of the pipeline we know the position of every player's torso, we still need to use the information the torso contains to classify the player. This step extracts this information to pass it on to the classifier which then decides the team membership of the player. As explained in section 2 we are looking to use color similarities to establish the team membership, so in this step colors are the only thing we are extracting from the torso.

We start out by cropping out the torso of the player, and convert that image to a HSV color space. We then extract the hue value of each pixel of the torso. From these hues we then establish a histogram, which is the only feature we are using to identify the team of the player. Only using the hue assures us that the actual color of the jersey is used and that these values are not at all dependant on the light in the gymnasium. Given the fact that we have the background subtraction, we can also use it as a mask so that the pixels extracted for the histogram are only from foreground objects. This is useful because the torso box often has very large boundaries, and therefore background pixels could be taken in account when creating the histogram. In addition to this, we normalize the bin counts of the histogram to compensate for the varying size of the torso depending on its extraction and player position.

6.5 Feature classification

Now that we have finally extracted the features of a player, we can determine the player's team membership. For this step we assume that the amount of classes (teams, referees, etc.) a person on the field can take is known.

The way we go about classifying is by indirectly using cluster analysis. We first start out with a populated data set, unfortunately this means that there is a certain amount of training necessary. In section 7.3 of this report we will discuss the amount of training necessary and the impact it has on performance. We then run a cluster analysis algorithm on that data, so that we

can extract the centers of all the clusters. Each center represents a class to which a player can belong. To classify a player we take its features and compare it to all centers and the center which is the most similar to the feature of the player determines the class of the player. The similarity metric used for these comparisons is correlation. The cluster analysis algorithm we use is a k-means implementation [11].

Running the cluster analysis allows us to get the centers of each cluster and also for every data sample, the cluster to which it belongs. Therefore, if we ran the analysis every time we added a feature to the dataset, we would know exactly to which cluster the feature belonged - there would be no reason to use a similarity metric and the centers of the clusters. Unfortunately, running the analysis for every player feature is too slow when aiming for real-time performance. It can however still be used as a way to update the cluster centers every so often. This is why we use similarity metrics for classifying features, it allows for much faster computation. As for the correctness of this approach, it is determined by the clustering algorithm and the similarity metric used to determine the cluster to which a feature belongs. Quantitative results are discussed in section 7.3.

Figure 8 : The result of our pipeline on a floorball video.

7 Results

Throughout this chapter we discuss the performance of each step of our pipeline. Two different datasets are considered for most measurements. The first one is a floorball video and the second one is a combination of videos of a single player walking around the playing field, which will be called “simple dataset”. The floorball dataset can be seen as a real-world use case of our software.

All the results were measured by manually inspecting frames.

7.1 Background subtraction and blob extraction

We can consider the background subtraction and blob extraction steps together when looking at their performance as they both work towards locating player-related pixels. The implementation of the background extraction algorithm we are running does not use any learning, it computes the foreground pixels of a given frame by comparing it with the first frame of the same video. Therefore, the video must start out with an empty gymnasium in order to detect every foreground object possible. This can be inconvenient, but given our controlled setting we are able to either add empty room frames at the beginning of the video input or directly capture videos starting with an empty gymnasium. Another reason as to why we are not using learning is because we are able to get much more consistent and satisfactory results without it.

There is no point in measuring the results of these two steps, as we calibrate them in order to not miss a single player or blob. These steps can therefore be considered “perfect” performance-wise, the only aspect which could see improving is their speed. They are by no means slow, but faster computation would benefit the overall pipeline speed.

7.2 Player extraction

The results of the player extraction step are the most important as they determine the players’ torsos which play a key role in classifying the player.

We consider that a player is correctly extracted when the DPM box for the whole player contains him or her entirely. False positives are when the previously mentioned box does not contain a player at all or only partially contains a player.

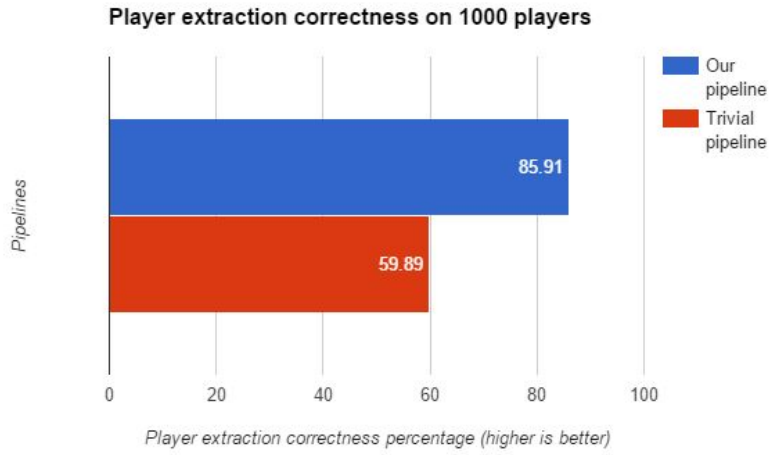

Figure 9 : Comparing results on the floorball dataset of our pipeline and the trivial pipeline for player extraction correctness.

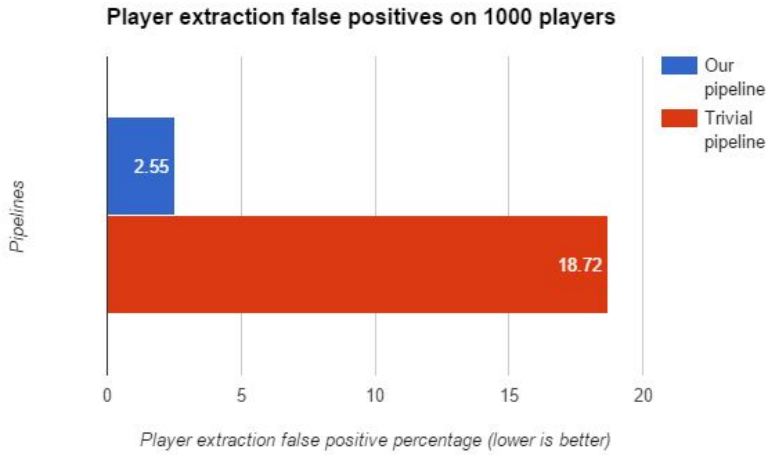

Figure 10 : Comparing results on the floorball dataset of our pipeline with the trivial pipeline for player extraction false positives.

Figure 9 shows that we manage to reach 85.91% correct player extraction with our pipeline, whereas we only have 59.89% with the trivial pipeline. The correctness gain which we can observe here is due to the fact that in our pipeline the DPM algorithm runs on background extracted blobs. Without the background, players are more often detected as they visually stand out more compared to when the background is not extracted (this helps the DPM algorithm significantly).

Figure 10 illustrates that we have 2.55% false positives compared to the trivial pipeline's 18.72%. The additional false positives from the trivial pipeline are mostly extractions in locations

which have no players at all. Again, the background subtraction and blob extraction allow us to filter out areas of the frame which are not players, and thus by not running the DPM algorithm on these parts we obviously avoid detecting false positives in them.

It should be noted that for the same correctly extracted player in both pipelines, DPM creates the same boxes. Therefore, we can conclude that by adding the blob extraction and background subtraction, we do not change the body parts for a given player but only add actual players and remove false positives.

The results of our pipeline illustrated in figures 7 and 8, are obviously not perfect - we are missing 15.09% of the players and have 2.55% false positives. Through visual inspection of our results we come to the conclusion that there are 3 factors which contribute to these 15.09% and 2.55% : player occlusion, player posture, player position.

Player occlusion is the main factor for false positives. When players overlap, the DPM algorithm might not separate them correctly and often outputs boxes which are a combination of both players. This obviously also affects correct player extraction, because both players in the previous example would not be extracted correctly.

Player posture causes a lot of missing player extractions. For example, when a player is bending over, the DPM algorithm sometimes does not detect it as a person. This behaviour is determined by the model we provide to the algorithm. Because the model we use is almost exclusively trained on people standing up straight, the algorithm does not detect a person who is adopting a different posture.

Player position also causes a lot of missing player extractions. There are two distinct cases in which the position of a player hinders his or her extraction : the player is too close to the camera or too far from the camera. In the first case, the camera has a view of the player which is, or almost is, directly above the player. Such a view does not allow us to see all body parts of a given player, in fact it usually only allows us to see the player’s head and shoulders. The DPM algorithm needs all body parts to extract players, and therefore in this case it is not able to correctly extract the player. The second case is when the player is too far, in such a scenario the amount of pixels describing the player is too low for the DPM algorithm to detect him or her - it only considers the player as noise.

In order to assure the previous assumptions are true we use another dataset than the floorball one - the simple dataset. It is made of videos where there is a single player and he or she does nothing else but walk around. This dataset allows us to not have player occlusions and also to eliminate players who are not standing up straight. Therefore, from the 3 factors mentioned above, only player position remains a cause for missing player extractions. We should also see a considerable decrease in false positives as there are no more player occlusions.

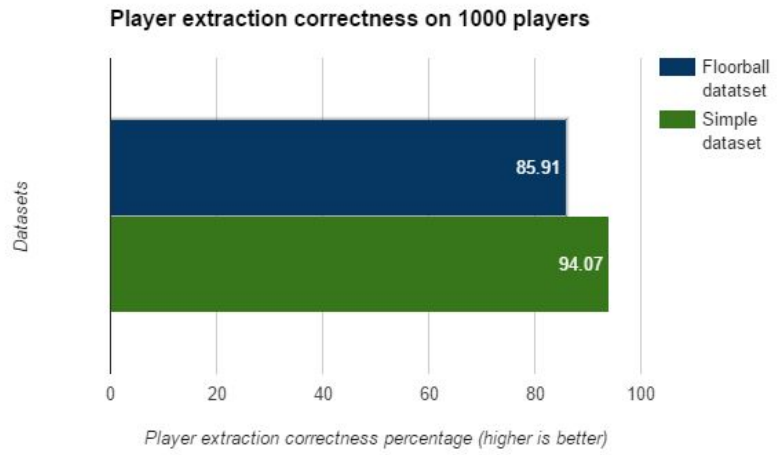

Figure 11 : Comparing results of player extraction correctness of our pipeline on the floorball dataset and simple dataset.

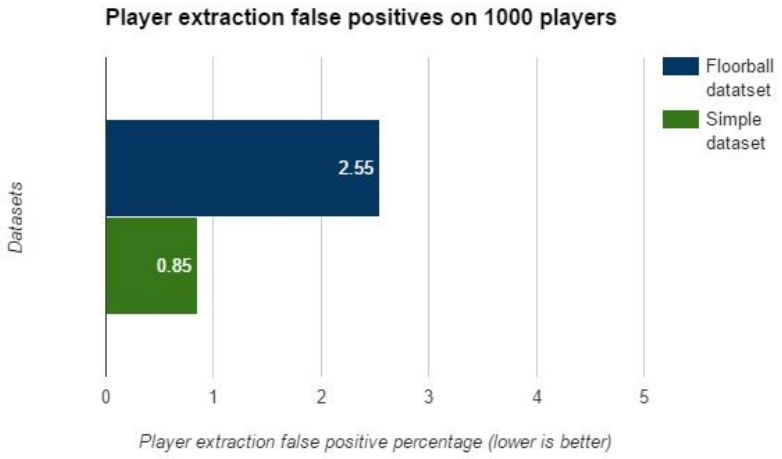

Figure 12 : Comparing results of player extraction false positives of our pipeline on the floorball dataset and simple dataset.

Figure 11 illustrates that we have a 92.37% correct player extraction on the simple dataset, compared to the previously seen 85.91% on the floorball dataset. After checking that the missing 7.63% of extracted players are exclusively players who are either too close to the camera or too far from it, we can conclude that our assumptions are correct.

On figure 12 it is shown that the false positive percentage dropped to 0.85% from the previous 2.55%. It therefore confirms the assumption that player occlusions highly impact the amount of false positives. The remaining 0.85% false positives are players who are about to enter or leave areas which are too close or too far from the camera to extract them correctly.

We can conclude from these results that the only missing player extractions from the floorball dataset are due to the three aforementioned factors - player posture, position and occlusion.

7.3 Feature extraction and classification

In order to evaluate our feature extraction and classification results, we must first look at correct torso extractions as they directly affect these results.

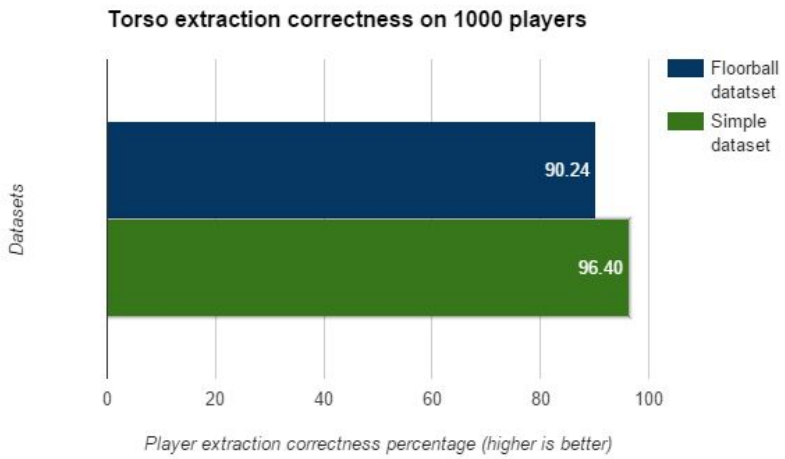

Figure 12 : Comparing results of correct torso extraction of our pipeline on the floorball dataset and simple dataset.

Figure 12 illustrates that 90.24% of torsos are accurate when the player is correctly extracted on the floorball dataset. The simple dataset reaches 96.40% correct torso extraction. The remaining 3.60% and the 6.16% difference can be explained by the same reasoning as in section 7.2 - player posture, position and occlusion.

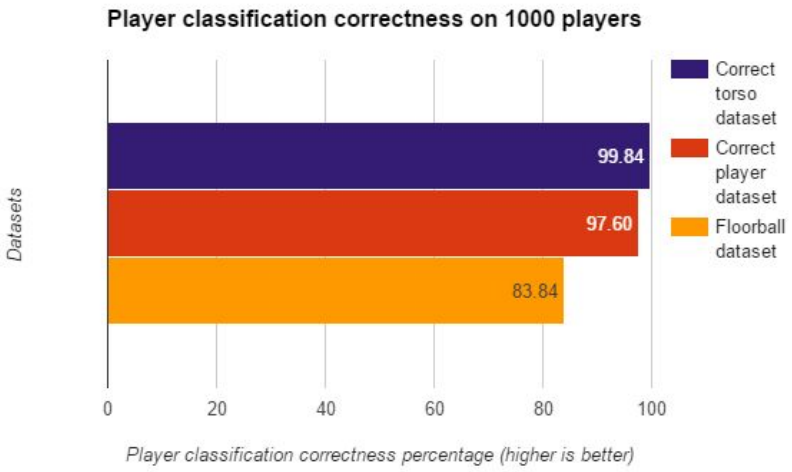

Figure 13 : Comparing results of correct player classification on the floorball dataset.

Figure 13 compares the correct player classification percentage (with only 4 players per team as training) when considering the following data of the floorball dataset:

1. Players who have their torso correctly extracted by the DPM algorithm.
2. Players who are correctly extracted by the DPM algorithm (even if their torsos are not correct).
3. All players seen throughout the video. Players who are not detected by the DPM algorithm are counted as wrong player classifications.

It should be noted that in addition to the results of figure 13, the false positives which were discussed in section 7.2 are still drawn on the pipeline output.

The first case mentioned above results in a 99.84% correct player classification. The few occurring faults are mainly the result of torso boxes being too large, which in the case of player occlusion can contain pixel information about other players.

The second case has a 97.60% correct classification percentage. Knowing the result of the first case and the fact that 90.24% players have their torso correctly extracted as shown in figure 10, we could expect at least a 90.09% correct classification. The additional 7.51% is the outcome of cases where the torso box is not entirely correct but still contains pixels of the player's jersey.

The third and last case achieves 83.84% correct player classification. This is only 2.7% less than the number of correct player extractions seen in section 7.2.

As mentioned previously these results are achieved with only 4 players per team as data samples for our cluster analysis. We do not notice any significant changes when adding more training samples. However when reducing the number of samples, the results become very inconsistent depending on the video input and samples, this is why we do not illustrate them

here.

From the results detailed in this section, we can conclude that using hue histograms is an appropriate feature when using correlation as a similarity metric to achieve player classification. Our approach, which uses training to create templates for each team through cluster analysis, is also suitable as it demands very little training. In most cases, using a single frame as a training dataset is sufficient.

A visual result of our software on a floorball video can be found on YouTube[1].

7.4 Pipeline speed

The trivial pipeline is extremely slow and can not be used to achieve real-time performance on consumer-grade hardware. Our pipeline achieves a 600% to 700% speedup compared to the trivial one depending on the video input. This significant performance gain allows us to reach real-time computation. We cannot give quantitative results as the computation time is very hardware and input dependant. For example, a video input which has more players per frame obviously takes more time to be processed.

We can conclude that our approach is far from being computationally intensive - it can achieve real-time computation in most cases without any correctness decreases.

8 Software implementation

The software built for this project uses OpenCV 2.4.12 as a core library. Most of the algorithms mentioned throughout this report were already implemented in the library. However, some of these had to be modified in order to add small features or even just to be compatible with our pipeline, in particular the DPM algorithm, which only allowed us to get the full player and not the body parts. We had to rework most of the code concerning that algorithm to be able to use it.

An unfortunate point is the lack for more detailed documentation in OpenCV. We attempted to train our own DPM model, however the model file in OpenCV uses a different standard than the one created using [10]. OpenCV does provide a tool to convert from one to the other, but it is outdated and barely has any documentation. The possibility of creating our own tool to perform the conversion was also disregarded because the output model expected by OpenCV is not documented at all.

As for actual implementation details which were not mentioned previously, we are doing basic thread interleaving. Each thread runs the pipeline on a different frame, this can be done since there is no inter-frame dependency. The result of performing this is a significant speedup (the actual numbers are of course hardware dependant).

[1] Link to the result video : https://youtu.be/pOEk0HC6Kvc

9 Conclusion

9.1 Summary

In this project, we achieve player classification based on team membership in real-time. Our approach uses background subtraction and blob extraction algorithms to locate regions on the frame which are players. On these areas we then use a DPM algorithm to extract players and their torso from which we create a hue histogram. This histogram is used as the unique feature to classify a player. The actual classification is done by comparing the feature to pre-computed templates which represent each possible team. These comparisons use correlation as a similarity metric. The highest similarity determines the player's team. The previously mentioned templates are computed by using the centers of clusters retrieved by a cluster analysis algorithm on a populated dataset of player features.

Our approach results in an overall 83.84% correct player classification with 2.55% false positives (classification of objects which are not players). The players who are not correctly classified are the outcome of incorrect player and torso extraction. In fact, when the torso is accurately extracted it leads to a 99.84% correct player classification. This proves that our feature and comparison metric selections are justified and are not the cause of the missing 16.16%. The flawed player and torso extractions are caused by three main factors.

The first one is player position. When the player is too far from the camera, his pixel count is too low for the DPM algorithm to extract him or her. In the inverse case, where the player is too close, his position is almost right below the camera. This situation does not result in correct extraction as we can only see the player's shoulders and head (all body parts must be visible for the DPM algorithm to function properly).

Player posture is the second factor. The DPM algorithm uses a model which defines what the algorithm is trying to detect. The model which we use was only trained on straight-standing persons. Therefore, when a player's posture is different from the previously mentioned one, the DPM algorithm struggles to correctly extract the player and his or her body parts. Unfortunately, as explained in chapter 8 we were unable to train our own models.

Player occlusion is the third cause for inaccurate player and torso extraction. When player occlusion occurs, some body parts are not visible by the camera and as explained previously this is necessary for correct extraction. In most cases, the DPM algorithm combines body parts of several players which obviously results in false positives.

In our approach we use background subtraction and blob extraction algorithms before running the DPM algorithm. By performing these steps, we significantly reduce the number of false positives as shown in figure 11. It also allows for an important speedup without which we could not reach real-time performance.

As mentioned previously, training is necessary to compute the templates for feature

classification. Our results show that very little training is needed to achieve the aforementioned results. In general, after adding 4 player features per team to the data on which we run the cluster analysis, additional data samples do not significantly increase the correctness of the classification. In addition to this, the background subtraction implementation we use does not need any training, it does however require that input video's first frame describes the background (an empty room) for optimal performance. Therefore the overall training needed for our software to obtain the best results possible with our approach is very little. Generally a single frame (with 4 players per team) is sufficient.

9.2 Future work

We've seen that a single camera view cannot handle player occlusions very well with our approach. Experimenting with multiple cameras would be a first step in resolving that issue, it could be implemented by using a similar method as in [5] for example. It would also resolve the issue we have where players are not extracted correctly when they are too far or too close to the camera.

Our initial intentions were to use the Probability Occupancy Maps approach described in [5], but an unexpected problem with the setup at the Sports Center prevented us from using it.

Another important issue of our approach was the correctness of the DPM algorithm when the player was bending over or simply adopting a different posture than standing straight. Our initial guess to solve this would be to train a DPM model with players in their specific sport environment. As explained in chapter 8, we unfortunately could not do this ourselves.

There still is room for software improvements, which could drastically increase the speed of the pipeline. One of them is using GPU computational capabilities.

In future work, it would also be interesting to evaluate our approach when using jerseys that are not plain, but have more details (stripes, text, logos, etc.).

9.3 Acknowledgements and personal experience

Firstly, we would like to thank our supervisor, Dr. Horesh Ben Shitrit for making this a great academic experience and for guiding us throughout the 17 weeks during which this project took place. Secondly, we would like to thank Prof. Pascal Fua for accepting this project and supervising it. Lastly, we would like to thank the whole Sports Center staff for always being willing to help us out and use their infrastructures.

We had very little experience in computer vision before starting this project, having only seen very basic notions about color spaces and edge detection. This made it a powerful

learning experience for that field, we now have a better understanding of the main problems encountered when trying to achieve real-time people tracking and identification.

As for software development, we both have never had projects of this size which were as loosely guided. It taught us a lot about team development and provided us with a concrete example of what to expect as computer science engineers.

Overall, we are very proud of this piece of software, and will cherish the experience we gathered producing it.

10 References

[1] Tamar Avraham, Ilya Gurvich, Michael Lindenbaum, and Shaul Markovitch. (2012). *Learning Implicit Transfer for Person Re-identification.*

[2] Chunxiao Liu, Shaogang Gong, Chen Change Loy, and Xinggang Lin. (2013). *Evaluating Feature Importance for Re-Identification.*

[3] Norbert Fuhr (2012). *Colour similarity search in images.*

[4] François Fleuret, Horesh Ben Shitrit, and Pascal Fua. (2014). *Re-Identification for Improved People Tracking.*

[5] Horesh Ben Shitrit, Jerome Berclaz, François Fleuret, Pascal Fua. (2011). *Tracking Multiple People under Global Appearance Constraints.*

[6] Michael Mason, Zoran Duric. (2001). *Using Histograms to Detect and Track Objects in Color Video.*

[7] Reid Sawtell. (2007). People detection and tracking using stereo vision and color.

[8] Pashalis Padeleris, Xenophon Zabulis, Antonis Argyros. (2013). *Multicamera tracking of multiple humans based on colored visual hulls.*

[9] Zoran Zivkovic.(2004). *Improved adaptive Gaussian mixture model for background subtraction*.

[10] Pedro Felzenszwalb, Ross. Girshick, David McAllester, Deva Ramanan. (2009). *Object Detection with Discriminatively Trained Part Based Models*

[11] Stuart Lloyd. (1982). *Least squares quantization in PCM.*