Background Foreground Segmentation Methods in Analysis of Live Sport Video Recordings

Johan Flinke, Fredrik Hammar

Master's thesis
2018:E40

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Master's Theses in Mathematical Sciences 2018:E40
ISSN 1404-6342

LUTFMA-3358-2018

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/

Background Segmentation Methods in Analysis of Live Sport Video Recordings

Fredrik Hammar & Johan Flinke

Supervisors:
Karl Åström, Professor at Mathematics, Lund University.
Niels-Christian Overgaard, Senior Lecturer at Mathematics, Lund University.
Examiner:
Magnus Oskarsson, Senior Lecturer at Mathematics, Lund University.

Mte13fha@student.lu.se & Mte13jfl@student.lu.se
Printed in Lund, Sweden. June 2018.

Abstract

A sports video analysis application was developed by Spiideo for mobile devices. It presents recordings from practices and competitive games for game-play analysis. Available tools, such as on-screen drawings, require a robust background foreground segmentation. The currently used segmentation method have difficulties to master the shifting conditions in weather, shadows and shirt colors.

The purpose of this project was to improve the background foreground segmentation in the application by evaluating alternative methods and examine possible improvements of the currently used method.

A data-set with recordings from Spiideo clients was created and used to evaluate and compare different segmentation methods with the current method. The evaluation included pixel classification scores, complexity measurements and a visual evaluation.

A median background model frame difference approach showed better performance with lower computational time compared to what is currently used. A Mixture of Gaussians method gave the best pixel classification result, but increased the calculation time. Suggested alterations of the currently used method also showed minor improvements in performance.

KEYWORDS: Image Analysis, Background Foreground Segmentation, Segmentation Evaluation, Live Sports Video Analysis

Acknowledgements

We would like to express our gratitude to our supervisors, Karl Åström (Professor at Mathematics) and Niels-Christian Overgaard (Senior lecturer at Mathematics) at the Faculty of Engineering, Lund University. We are very grateful for all the engagement, help and support throughout the project.

Sincere thanks also go to Spiideo, for giving us the opportunity to combine our two biggest interests, football and image analysis, in our master thesis. We would like to thank all the co-workers for welcoming us to the office and answering our questions along the way.

We would also like to acknowledge Håkan Ardö, thank you for taking time to give valuable advice when most needed.

Thank you!

Fredrik & Johan, Malmö 2018.

Contents

Definitions

In this section concepts and abbreviations in need of clarification will be introduced and defined.

- When **segmentation** is mentioned, a background foreground segmentation of the entire frame is always intended.

- The **currently used method** refers to the *ColorCube*, the background foreground segmentation method that Spiideo currently uses their application.

- A **binary mask** is a binary matrix that holds the result of a segmentation. The mask corresponds to a specific frame and are of the same shape and size, every entry in the binary mask corresponds to specific a pixel. If the entry is of value of 1, the corresponding pixel in the image is classified as foreground and if the entry is of value 0, the pixel is classified as background.

- A **scene** can be considered as a geographical place used for recording. Each recording in the data-set are recorded at a specific scene. Some recordings are from the same scene, but recorded at different times and therefore have different characteristics.

- Each video recording is a series of **frames**. A frame is an image in the recording that together with other frames constitutes the video.

- A **key-frame** is a frame that was used for evaluation. Background foreground segmentation methods were applied to recordings in the dataset and were thereafter evaluated at specific key-frames having a corresponding ground-truth binary mask.
- **HLS** stands for HTTP Live Streaming. It is a media streaming protocol that is used for delivering visual and audio media over the internet.
- **FFMPEG** is a complete cross-platform solution to record, convert and stream audio and video. It was mainly used to download media from HLS-streams and to add overlay masks to video recordings [27].
- **OpenCV** is an Open Source Computer Vision Library, see documentation in [34]. It was used to handle videos and images in Python and possess a library of tools used for image analysis and image processing. OpenCV 3.4.0 was used in this project.
- All images, frames and recordings are by default defined with color. Unless anything else is stated is the color in each pixel represented by the three color channels red, R, green, G, and blue, B.

1

Introduction

1.1 Background

Background foreground segmentation is a popular topic in image analysis today. Automatic applications for detection, classification and analysis in images and videos are widely used in many different industries. These applications often require a robust and appropriate background foreground segmentation for optimal performance.

The Malmö-based company Spiideo provides a game-play video analysis service for sport teams. High resolution cameras are mounted and installed next to training grounds and arenas to record practices and competitive games. A video feed from recorded events are presented, live or playback, in the Spiideo mobile application. Coaches, analysts and players can use the application to view and analyze game-play recordings. An example of what the application looks like can be seen in Figure 1.1.

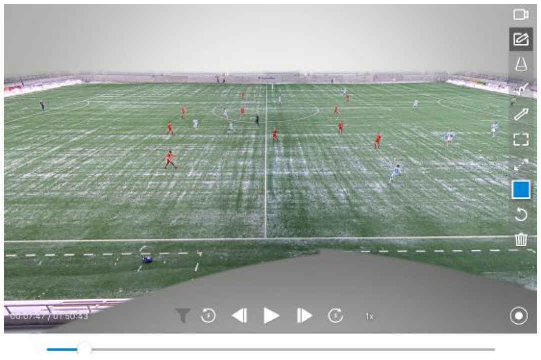


Figure 1.1: A screen-shot from the Spiideo mobile application. The user has the ability to play, pause and use the time-line to navigate between game-play events. The user can also pan, tilt and zoom, or use the drawing tools shown on the right side of the image.

A Spiideo camera set-up can differ from client to client, but (almost) always include two high-resolution cameras at the center-line facing different directions to cover both halves of the field. This can be complemented with one camera on each short-side above the goal line. The two camera feeds from the center-line are stitched together and visualized as one recording, as shown in Figure 1.1. It is possible to switch between the camera feed from the center-line back and forth to the feeds from either goal-line. The user can play, pause and navigate between events in the time-line. It is also possible to pan, tilt and zoom, or use a selection of on-screen drawing tools.

An important feature for game-play analysis is the ability to make drawings, such as arrows and areas, see Figure 1.2. To optimize the visual presentation and thereby the viewer perception, importance lies in the fact that a drawing is projected only onto the background of the frame. Parts considered to be foreground, such as a player, should not be overdrawn and "disappear" behind the drawing. Instead should these pixels be projected on top of the drawing.

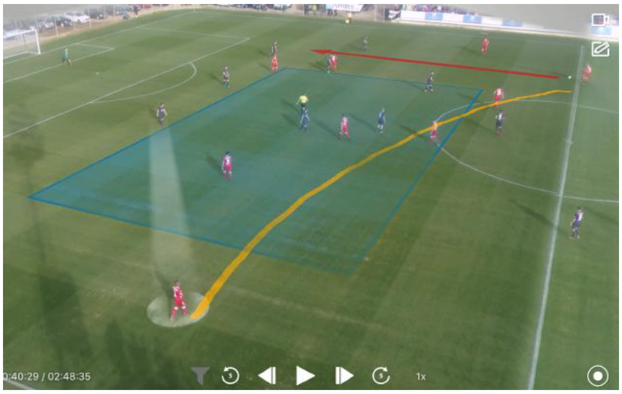


Figure 1.2: The Spiideo application enables on-screen drawings for game-play analysis. The drawings should only be projected on top of the background, and players should not be over-drawn.

To fulfill these demands is a good and applicable method for background foreground segmentation needed. The segmentation method should for each frame in the recording, be able to determine which pixels that belong to the background and which that belong to the foreground. In the case of sport recordings, the foreground consists of moving objects such as players, referees and the ball, while the rest of the frame can be considered to be background.

The currently used method for background foreground segmentation is based upon finding the most dominant color(s) in the frame, and thereafter classify all pixels with a similar color as background. The method is called *ColorCube* and is further explained in chapter 3.2.

The method is computationally efficient and performs well in most basic cases. Harder conditions, such as green player shirts on green grass or patches of white snow on a green grass field, makes it difficult for the method to perform. The green shirts are classified as background and the patches of snow as foreground. Non-static weather conditions such as shifting sunlight, or moving clouds, can also lead to incorrect segmentation results due to the momentarily changes in brightness. Players can therefore disappear behind on-screen drawings, while other parts of field are projected on top of the drawings, as shown in Figure 1.3. By improving the segmentation performance, Spiideo can enhance the visual perception for the users when watching or analyzing game-play recordings.

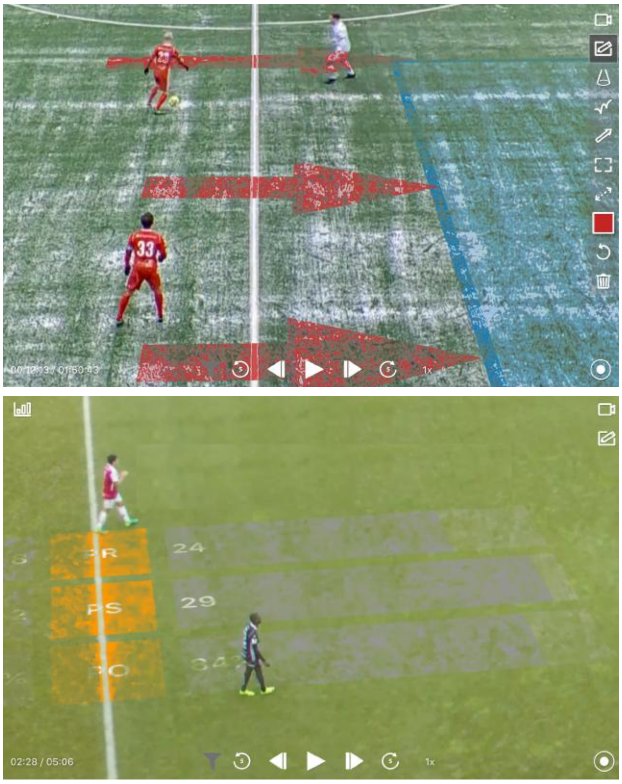


Figure 1.3: Some conditions causes the current background foreground segmentation to fail. This leads to that drawings are incorrectly presented on top of the foreground or behind the background.

The video analysis service is a cloud based system. The camera recordings are directly uploaded to the cloud and then visualized for the users in a mobile application. The current segmentation is carried out directly in the user's mobile device. An option that requires a computationally efficient method as it is run on low-spec hard-ware. A more advanced segmentation method may not be able to be processed in real time on all devices. An alternative way would be to carry out the segmentation in the cloud, where one have access to more computational power.

Implementing the segmentation method in the cloud comes with the problem of transferring the resulting segmentation information from the cloud to the mobile device. A process that is dependent upon internet connection and an efficient transfer method. Many other processes are competing for bandwidth to send and receive information, such as streaming the video feed. Segmentation method complexity is therefore a good thing to have in mind when considering ways to improve the segmentation results.

An important thing is that the segmentation method cannot take any user input to simplify the segmentation. The service is widely used for many different clients and sports which makes it impossible to handle and process user input regarding background and/or foreground information. This implies that a background segmentation method must have the same initial parameters and settings for all scenes.

1.2 Related work

Background foreground segmentation is a popular topic. There are much ongoing work and findings in the area, and many different methods have been developed. In this section a review of background foreground segmentation methods will be presented.

Basic Methods

A simple method to detect differences (such as movement) between two images is to compute the euclidean distance between all pixels in the two frames, as proposed in [18]. The same approach can be used for background foreground segmentation by first creating a background model, and then detect the foreground by calculating the absolute difference between the image and the background model. There are several ways to produce a background model by using past frames in a video recording. One can either create a static background model, or create an adaptive background model that changes over time due to changes in the video.

As mentioned in [19], is one basic approach to initialize and update a background model by taking the the average of past frames. A similar procedure was proposed in [16] where the background model was initialized and maintained by calculating the median of past frames. A weighted median can also be used, which enables the background model to put emphasis on more recent frames.

Similar techniques were presented in [2] where color, edges and texture were combined to create a background model, and in [10] where the background was modeled based on a sub-pixel edge map.

When a background model is initialized, the foreground can be segmented out by detecting differences between the current frame and the background model as showed in Figure 1.4.

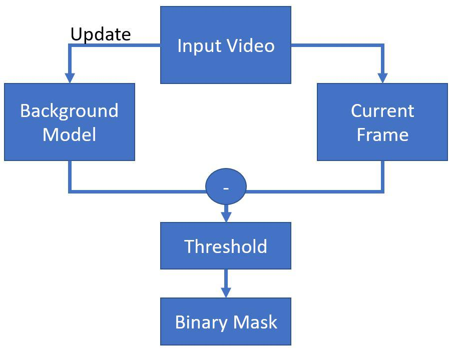


Figure 1.4: A flow chart that describes how background foreground segmentation can be performed by using a background model frame difference approach.

The difference between the model and the current frame can be calculated in several ways. Most straightforward is to calculate the euclidian distance, D, for each pixel as

$$D = \sqrt{(F_\mathrm{R} - B_\mathrm{R})^2 + (F_\mathrm{G} - B_\mathrm{G})^2 + (F_\mathrm{B} - B_\mathrm{B})^2}, \tag{1.1}$$

where F_R, F_G and F_B are the intensity in each color channel (R, G, B) in the pixles of the frame, and B_R, B_G and B_B are the intensity in the corresponding color channels in the background model pixels.

Thereafter, a pixel can be classified as background or foreground depending on the size of D. Preferably by comparing the absolute difference to a suitable threshold value. More advanced ways to discover other dissimilarities, as proposed in [11], are to also consider differences in edges, texture and gradients. It probably gives a more accurate result, but would require more computational power.

Remember that the cameras used for recording Spiideo client videos are fixed, and the background is not expected to contain any movement. Consequently, there should be a high possibility to create a good and useful background model.

Statistical Methods

A statistical approach for segmentation in a static scene without a priori information was presented by Stauffer and Grimson in [20]. The method is called *Mixture of Gaussians* (MOG). The recent history of values for each pixel in the recording was modeled by multiple Gaussian distributions. The distributions were evaluated based on variance and change over time to automatically select which distributions best represents the background. The probability that the current pixel value belonged to a background distribution was calculated, and decided whether the pixel should be classified as foreground or background. The distributions were continuously updated depending on a learning rate.

There have been further research on developments to this method. Zivkovic proposed an improvement in [21], where an adaptive algorithm more efficiently and recursively updated model parameters and chose the appropriate number distributions for each pixel. Another improvement was developed in [12] by Kaewtrakulpong and Bowden to make the model learn and adapt faster with more accuracy, to be able to handle changing environments better. A third addition was implemented by Hayman and Eklundh in [8], where the method was applied to carry out segmentation for a mobile observer in a non-static scene.

These statistical approaches were of interest due to the static scenes from outdoor cameras with varying weather conditions.

Segmentation by Motion Detection

The cameras are fixed and no movement in the background is expected. By assumption, movement can only occur in the foreground. Therefore, it could be expected that by detecting motion in a frame, one could also find the foreground. One approach that further investigated this was proposed in [14], where MOG was combined with *optical flow*.

Optical Flow is a field of image analysis that detects movement between frames. An optical flow algorithm was implemented by Bouguet in [3]. By taking a 3x3 patch around a tracking point, an assumption was made that all pixels in the patch had the same intensity and were subject to the same movement. A least square fit-method solveed the equations which constitutes the movement. Interesting points to track can differ depending on purpose and must be manually provided. A clear drawback when dealing with live recordings for different clients and sports, which makes it hard to generalize tracking points.

Another algorithm was implemented by Farnebäck in [6]. Instead of tracking chosen points, the algorithm tracks the motion of all pixels in the image, which is called *dense optical flow*. By approximating each neighborhood of both frames by quadratic polynomials, a vector field is created by observing how an exact polynomial transforms under translation. The displacement fields are estimated from the polynomial expansion coefficients. The resulting vector field describes the movement for all pixels between two frames.

Artificial Intelligence

There have been much recent work in the field of combining artificial intelligence (AI) with segmentation in computer vision. How neural networks can be used for semantic segmentation and classification were proposed in [4], [17] and [9]. Networks were trained to detect, segment and classify objects in both images and video. Classification were not relevant in this project, as there was no interest to know what kind of object that is detected in the foreground. It would also be at expense of unnecessary computational load.

1.3 Scope and Agenda

The scope of this project was to:

Improve the background foreground segmentation in the Spiideo application.

The segmentation can be improved by either changing to an alternative, better performing method, or by improving the current one. The agenda was to find other relevant and suitable methods, and discover possible improvements of the currently used method. It was also an aim to do so without significantly increasing the computational complexity compared to what is currently used.

A data-set with representative recordings from current Spiideo clients, were created. The methods were applied to the data-set and compared by using scores for pixel classification, complexity measurements and by an visual evaluation.

The rest of this report is organized as follows;

- Chapter 2 (Data) describes the data-set that was used.
- Chapter 3 (Method) describes how the data-set was created. It also explains the methods that have been evaluated on the data set, and how they were evaluated.
- The results of the evaluation are presented in chapter 4 (Results), and discussed in chapter 5 (Discussion).

- Conclusions based on the results and the following discussion are presented in chapter 6 (Conclusion).

2

Data

The data used for this project consist of 13 video recordings from 10 different scenes. The recordings were recorded by Spiideo for their clients during football, ice-hockey and lacrosse. Each recording have different conditions regarding weather, present shadows and player shirt colors. The frame rate for each video is 25 frames per second. How the data-set was created are presented in chapter 3.1 on page 24.

The recordings have different resolutions due to their position relative the field that they are recording. Recordings from cameras mounted on the goal line sides have a 2K resolution (1920x1080). Recordings from cameras mounted on the center line consist of two camera feeds facing different directions and have a 4K resolution (3840x2160). A visualization of the set-up is shown in Figure 2.1. Further details about all recordings are presented in Table 2.1 on page 2.1.

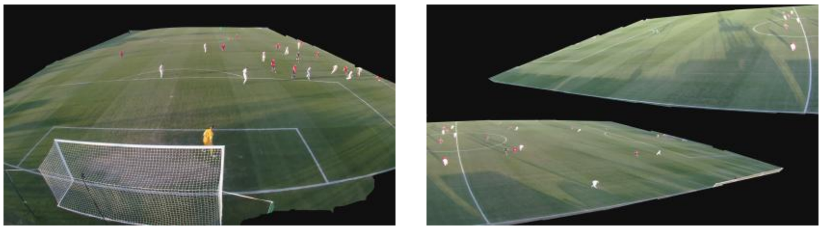


Figure 2.1: Two possible camera positions for the recordings in the data-set. The left figure shows a frame from a goal-line camera. The right figure shows a frame from a center-line camera.

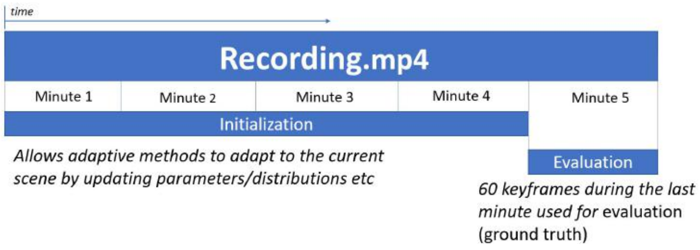

time
Recording.mp4
Minute 1
Minute 2
Minute 3
Minute 4
Minute 5
Initialization
Allows adaptive methods to adapt to the current scene by updating parameters/distributions etc
Evaluation
60 keyframes during the last minute used for evaluation (ground truth)

Figure 2.2: A flow chart explaining how the recordings in the data-set were used. The first four minutes allowed the methods to adapt, the last minute was used for evaluation.

Each recording in the data-set have a duration of 5 minutes. The four first minutes were used to enable adaptive methods to update its parameters to optimally fit the recording specific conditions. The fifth and last minute of each recording was subject to method evaluation, see Figure 2.2. There are 60 frames during the last minute that have a corresponding ground truth represented as a binary mask. Zeros in the binary mask represent background and ones represent foreground. An example of a ground truth binary mask can be seen in Figure 2.3.

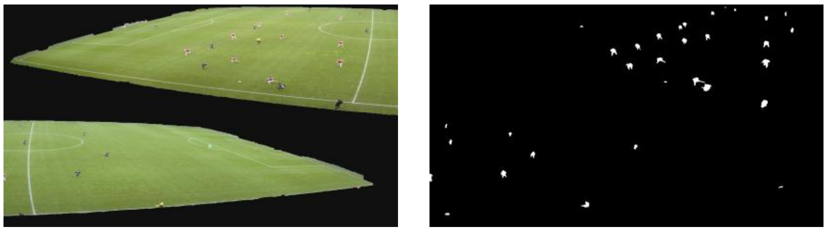


Figure 2.3: The left figure shows a frame from a recording. The right figure shows the corresponding ground truth represented as a binary mask.

Table 2.1: Table describing the characteristics for each recording in the data-set. "Shifting" means that there are variations in the color of the field. "Illuminated" means that the light comes from spotlights around the field, and there is an absence of natural light.

Nr.	Recording	Set up	Description	Resolution
1	Gefle IP	Football, Game-play	Snow on green field. White and red player clothes.	3840x2160
2	Jamkraft Arena (1)	Football, Game-play	Green field. Black, orange and green clothes.	3840x2160
3	Jamkraft Arena (2)	Football, Game-play	Sun on green field. Large shadows. Blue and orange clothes.	3840x2160
4	Kalmar	Football, Game-play	Shifting green field. White/red and dark clothes.	3840x2160
5	Klockener	Lacrosse, Game-play	Green field. Sun on parts of the field. Large Shadows. White and red clothes.	3840x2160
6	La Manga	Football, Game-play	Sun on shifting green field. Large shadows. White and red/blue clothes.	3840x2160
7	La Manga (North Side)	Football, Game-play	Bright sun on shifting green field. Large shadows and backlight. White and red/blue clothes.	1920x1080
8	La Manga (South Side)	Football, Game-play	Sun on shifting green field. Large shadows. Yellow, white and red/blue clothes.	1920x1080
9	Malmö IP (1)	Football, Practice	Shifting green field. Dark. Black clothes.	3840x2160
10	Malmö IP (2)	Football, Game-play	Sun on parts of green field. Large shadows. White and black clothes.	3840x2160
11	Malmö IP (3)	Football, Game-play	Illuminated green field. Multiple small shadows. White and blue clothes.	3840x2160
12	Östgötaporten	Football, Practice	Snow on illuminated green field. Tracks in the snow. Pink, blue and green clothes.	3840x2160
13	Ice Hockey	Ice Hockey, Game-play	Illuminated white ice. Multiple small shadows. White and blue clothes.	1920x2160

3 Method

This chapter describes first how the data-set and the ground truth was created. Thereafter are the evaluated segmentation methods further explained. How the methods were evaluated are also accounted for. Unless anything else is stated, has all computer work been implemented in the programming language Python (Version 3.5.4) [36].

3.1 Data-set

To evaluate and compare segmentation methods on sports video recordings, there was a need to create a purposeful and relevant data-set. Spiideo have a data-base with recordings from many different scenes and clients. All recordings were stored in a HLS format online, and were downloaded using the multi-media framework software FFMPEG. To create a diverse and useful data-set, 13 recordings from 10 different scenes were chosen. The aim was to create a data-set with recordings having with different conditions in weather, presence of shadows and shirt colors.

To avoid bias, recordings known to be both easy and hard for the currently used method to correctly segment, were chosen to the data-set. More details about the chosen recordings are presented in Table 2.1 in Chapter 2 (Data). Each recording has a duration of five minutes. These five minutes were manually chosen to include actual game-play, as this represents when the analysis tools are mostly used.

As can be seen in the Figure 3.1 are the cameras recording more than the playing field. For example, are the supporter stands and the sky also included. The Spiideo application have a *region of interest*-function that automatically distinguishes where the playing field is, and other parts of the recording are thereby ignored. The currently used segmentation method thereby only takes the field in consideration. To replicate this, the unwanted parts were removed from the recordings in the data-set. An overlay mask was manually created and added to each recording using the video converting software FFMPEG [27], see Figure 3.1

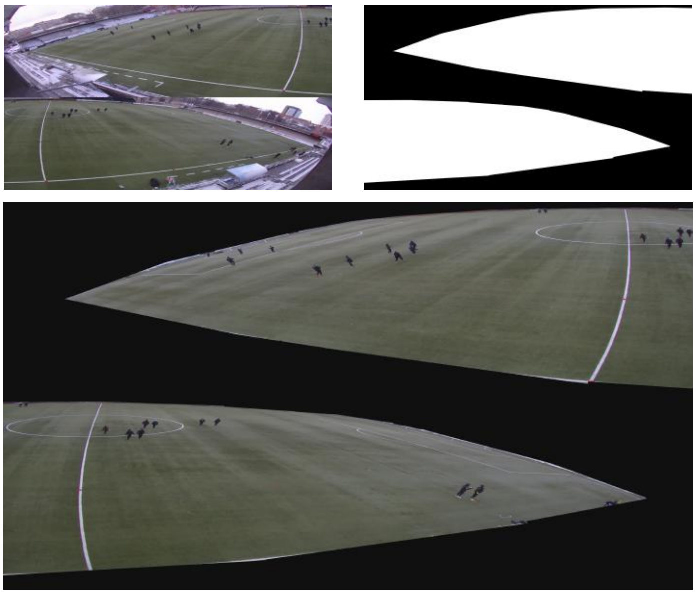


Figure 3.1: The recordings were pre-processed with an overlay mask to remove unwanted parts. The upper left figure shows the original frame. The upper right figure shows a manually created overlay mask, note that the white parts are transparent. The lower figure shows the resulting combination of the other two, where the unwanted parts were removed from the recording.

Ground Truth

A fundamental part of the data-set is the ground truth. It was created to enable an objective evaluation of the different segmentation methods. All recordings in the data-set have a duration of five minutes. The first four minutes enable adaptive segmentation methods to adapt to recording specific conditions, e.g. background color or weather. During the last minute, 60 key frames were used for method evaluation. All key-frames have a ground truth represented as a binary mask, explaining which pixels that are foreground or background in reality.

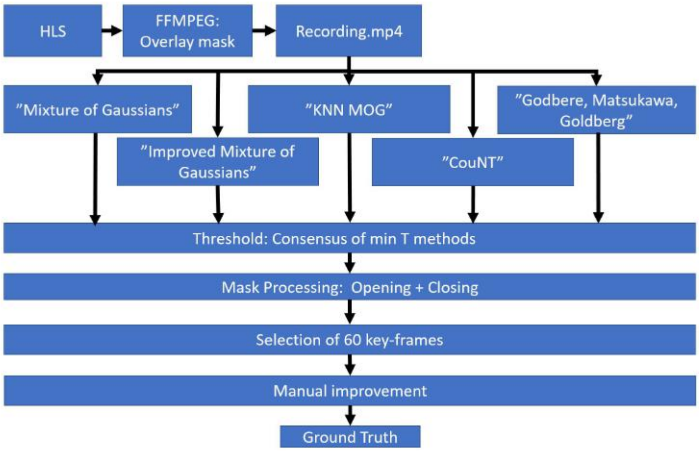

HLS
FFMPEG: Overlay mask
Recording.mp4
"Mixture of Gaussians"
"Improved Mixture of Gaussians"
"KNN MOG"
"CouNT"
"Godbere, Matsukawa, Goldberg"
Threshold: Consensus of min T methods
Mask Processing: Opening + Closing
Selection of 60 key-frames
Manual improvement
Ground Truth

Figure 3.2: The ground truth key-frames were created by using a consensus of minimum T methods of the 5 different segmentation methods. The masks were thereafter processed and manually improved.

To create a robust ground truth for the data-set, a semi-automatic approach that combined five different segmentation methods was developed and used. A brief overview of the procedure can be seen in Figure 3.2. The ground truth were created according to the following steps:

1. Five different segmentation methods (*Mixture of Gaussians, Improved Mixture of Gaussians, GMG, KNN* and *CouNT*) were implemented in

Python with OpenCV. All methods are further explained in section 3.2 on page 29. The five methods were applied, one at the time, to segment each recording into background and foreground. During the last minute, the resulting binary masks were saved. All recordings were segmented in full resolution.

2. The binary masks from all five segmentation methods were summarized for each frame. After this, each frame had a mask where each pixel value was an integer between zero and five. The value indicated how many methods that had classified the pixel as foreground. Pixels which all methods had classified as background had a value of zero, while pixels all methods had classified as foreground will had a value of five.

3. A binary mask was generated by using a threshold value, T, on the summarized mask. This means that the binary mask was set to foreground (pixel value = 1) if at least T number of methods had consensus over segmenting that pixel as foreground. The value of T was chosen manually and differed from recording to recording. The threshold values are shown in Table 3.1.

4. The resulting binary mask were processed. First were *Opening* (*Erosion* followed by *Dilation*) applied to remove noise in the mask. Thereafter were *Closing* (*Dilation* followed by *Erosion*) applied to fill holes in the binary mask. These concepts are further explained below and shown in Figure 3.3. The size of the kernels used for opening and closing differed for each recording, and are presented in Table 3.1.

   - **Erosion** is when a kernel[1] slides over the binary mask and a pixel in the mask is set to 1 only if all pixels under the kernel is 1, otherwise eroded to 0.
   - **Dilation** is the opposite of Erosion. It sets a pixel value to 1 if at least one of the pixel under the kernel has value 1.

[1]A kernel is a small matrix used for processing the image by convolution.

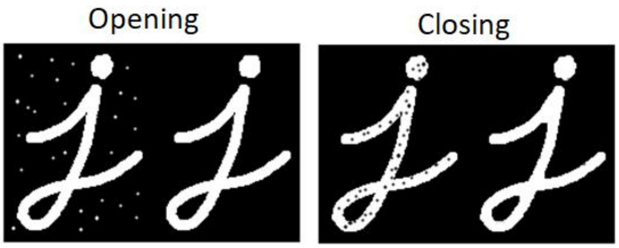


Figure 3.3: The left figure shows the result of opening, a technique used to remove noise. The right figure shows the result of closing, a technique used to fill holes in segments. The images were taken from [32].

5. Binary masks existed for all frames of the last minute of each recording. 60 binary masks were chosen from each recording to represent the ground truth. The masks were manually chosen by visual evaluation to use the masks that best represent the true foreground and background[2].

6. For each recording were the 60 binary masks checked and manually improved to better reflect the truth. Parts incorrectly classified as foreground, such as shadows or noise, were removed from the foreground. Parts that belong to the foreground but were incorrectly classified as background, such as a player, were added to the foreground.

[2]This may have lead to that frames that are easier to segment, were being subject for evaluation. This is further discussed in chapter 5 (Discussion).

Table 3.1: The recording-specific parameters that were used to create the ground truth binary masks.

Recording	Threshold	Opening Kernel Size	Closing Kernel Size
Gevle IP	2	(3,3)	(5,5)
Jamkraft (1)	2	(5,5)	(0,0)
Jamkraft (2)	2	(6,6)	(15, 15)
Kalmar	2	(5,5)	(5,5)
Klockener	2	(6,6)	(6, 6)
La Manga	2	(5,5)	(5, 5)
La Manga North	1	(5,5)	(0, 0)
La Manga South	2	(5,5)	(8, 8)
Malmo IP (1)	0	(9,9)	(5, 5)
Malmo IP (2)	2	(4,4)	(4, 4)
Malmo IP (3)	2	(5,5)	(5, 5)
Östgötaporten	1	(3,3)	(3, 3)
Ice Hockey	2	(5,5)	(15, 15)

The ground truth for each recording was limited to 60 key-frames to simplify the creation of the ground truth binary masks and the method evaluation. It was also considered unnecessary to evaluate every frame, since the video frame rate was 25 frames per second and the frames did not considerably change from frame to frame.

3.2 Segmentation Methods

Relevant methods with different approaches were chosen for further evaluation. In this section, the methods are presented and explained. The abbreviation presented in the method title are used as an identifier in chapter 4 (*Results*).

Median Background Model Frame Difference (Median)

There are multiple ways to initialize a median background model. One can initialize the background model by sampling past frames and then calculate the median for each pixel. Much variation in the foreground in the past frames, often leads to a good background model, not required to be frequently updated. However, without knowledge of past frames, the model can instead be initialized by only taking the first frame in consideration. For example can the most dominant color in the first frame be calculated, and thereafter are all pixels in the background model set to this color. This approaches require that the background model, in the beginning, are updated more frequently updated to quickly resemble the true background.

When evaluating this method, a background model was created by using 50 past frames, sampled every 10:th frame. The median value for each pixel was calculated. By skipping frames between the samples, the foreground was allowed to change and thereby make less impact on the background model.

When the background model was initiated, it was updated recursively with inspiration from the work by Ardö in [1]. Each pixel in the background model P_{B}, was updated every 10:th frame. The intensity in each color channel for each pixel in the background model, P_{B}, was updated as

$$P_{\mathrm{B}} = \begin{cases} P_{\mathrm{B}} + 1, & if \;\; P_{\mathrm{F}} > P_{\mathrm{B}} \\ P_{\mathrm{B}} - 1, & if \;\; P_{\mathrm{F}} < P_{\mathrm{B}} \\ P_{\mathrm{B}}, & if \;\; P_{\mathrm{F}} = P_{\mathrm{B}} \end{cases} , \tag{3.1}$$

where P_{F} is the corresponding color channel for the pixel in the frame used to update the background model. An example of an initialized background model can be seen in Figure 3.4.

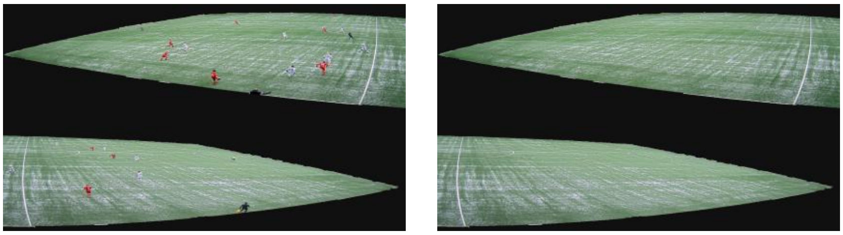


Figure 3.4: The left figure shows a frame from a recording. The right figure shows the initialized background model from the same recording. It was created by calculating the median frame of 50 past frames sampled every 10:th frame.

By recursively updating the background model, the calculation time was reduced as there was no need to save past frames. The segmentation was carried out for every frame in the recording by using a threshold for the euclidean distance between the colors of the pixels in the background model and the pixels in the current frame. The absolute difference D was calculated for each pixel as

$$D = \sqrt{(F_{\mathrm{R}} - B_{\mathrm{R}})^2 + (F_{\mathrm{G}} - B_{\mathrm{G}})^2 + (F_{\mathrm{B}} - B_{\mathrm{B}})^2}\ , \tag{3.2}$$

where F_{R}, F_{G} and F_{B} are the intensity in the three color channels in the frame and B_{R}, B_{G} and B_{B} are the color channel intensities in the background model. Thereafter was each pixel in the binary mask P_{BM} set to foreground or background by using a threshold T for the difference D, as explained by

$$P_{\mathrm{BM}} = \begin{cases} 1, & if\ \ D \geq T \\ 0, & if\ \ D < T \end{cases}\ . \tag{3.3}$$

The threshold T decided how big the difference between the background model and the frame was allowed to be before a pixel was classified as foreground or background. An example of the impact of different threshold values is shown in Figure 3.5.

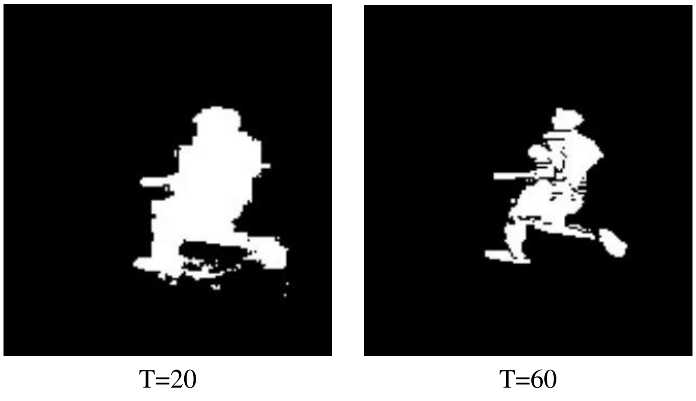


Figure 3.5: A binary mask was calculated by using a threshold for the difference between a frame and the background model. If the difference was bigger than the threshold value T, the pixel was classified as foreground. The figure shows how two different values of T affects the resulting segmentation.

Three different values (20,35 and 60) of T were used to segment the data-set.

Mixture of Gaussians (MOG)

An improvement of *Mixture of Gaussians*, first implemented by Stauffer and Grimson in [20], was presented by Kaewtrakulpong and Bowden in [12]. Every pixel maintains a density function based on past frames. The density function was viewed upon as a mixture of distributions. The distributions were internally ordered, and a number of distributions were considered to represent the background. The probability that a pixel intensity belong to a background distribution were calculated, and used to classify the pixel as either foreground or background. The algorithm has the ability to use different equations at different phases to update parameters, which enabled the system to learn faster. As a result, the system adapts more accurately and efficiently to changing environments.

The method was implemented using the OpenCV background subtraction library with default parameters. More method details are presented in [12] and [33].

Improved Mixture of Gaussians (MOG2)

Another improvement of the *Mixture of Gaussians*, was implemented by Zivkovic in [21]. Recursive equations were used to enhance the efficiency when updating the distribution parameters. An automatic selection of the appropriate number of distributions for each pixel was added as it allowed the system to be more adaptive to changes in the recording.

The method was implemented using the OpenCV background subtraction library with default parameters. More method details are presented in [21] and [30].

Mixture of Gaussians - KNN Improvement (KNN)

A K-Nearest-Neighbour improvement of the *Gaussian Mixture Model* was presented in [22] by Zivkovic and Heijden. Recursive equations were used to update model parameters and to simultaneously select the appropriate number of distributions for each pixel. A K-Nearest-Neighbour method for better kernel density estimation was implemented.

The method was implemented using the OpenCV background subtraction library with default parameters. More method details are presented in [22] and [31]

Godbehere, Matsukawa, Goldberg (GMG)

The method presented in [7] combined statistical background image estimation, per pixel Bayesian segmentation, and an approximate solution to the multi-target tracking problem using a bank of Kalman filters and Gale-Shapley matching. A heuristic confidence model enabled selective filtering of tracks based on dynamic data.

The method was implemented using the OpenCV background subtraction library with default parameters. More method details are presented in [7] and [29]

CouNT

A background segmentation project called *CouNT* was created by Sagi Zeevi in [23] and [24]. The method is computationally efficient and fast on low-spec hardware compared to other similar methods. It uses the "stability" of a pixel to determine whether it belongs to the foreground or the background. If a pixel holds the same intensity over time, it is given credit for being stable, and therefore considered to belong to the background. Changes in the pixel intensity will decrease the stability counter. When a pixel is considered to be "unstable", is it considered to belong to the foreground.

The method was implemented using the OpenCV background subtraction library with default parameters. More details are presented in [23], [24] and [25]

ColorCube (the currently used method)

The method currently used in the Spiideo's application was based on the work in [28]. The dominant color(s) in the frame are detected, and used to classify all pixels with similar color as background.

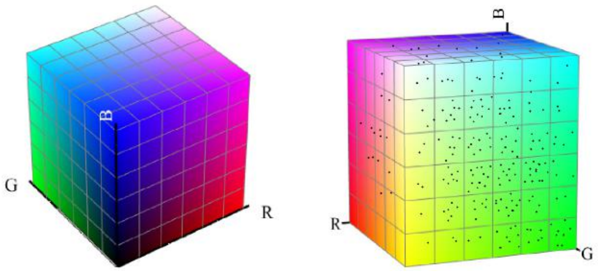


Figure 3.6: The Colorcube, a 3D color space histogram. The dots in the right figure visualizes how all the color of all pixels in a frame were projected into the ColorCube. The bin(s) with the most hits represent the most dominant color(s) in the frame.

The dominant color(s) were determined by looking at the intensity in the RGB color-channels for all pixels in the frame. A 3-dimensional histogram was produced, where each axis corresponds to one of the RGB-channels. The RGB-values for all pixels in the frame were projected into the 3-dimensional color space, as shown Figure 3.6, where the dots represent hits from all the pixels in the frame. The resolution of the histogram axes was set to 30, meaning that each axis was divided into 30 bins, leading to a grid of 30^3 (27 000) bins in the 3-dimensional color space.

Objects far away from the cameras appear smaller, and are therefore represented by fewer pixels than objects closer to the camera. To compensate for this, the hits in the 3D histogram were counted in relation to the pixel's position in the frame. The value of a hit from a pixel in the ColorCube was weighted according to its position. The lower boundary, L_b, and upper boundary, U_b of the region of interest were used to weight each hit, H, as

$$H = 0.1 + (\frac{U_\mathrm{b} - y}{U_\mathrm{b} - L_\mathrm{b}})^2 \ , \tag{3.4}$$

where y is the height of the pixel's position in the frame. How L_b and U_b are defined in a frame, is explained in Figure 3.7.

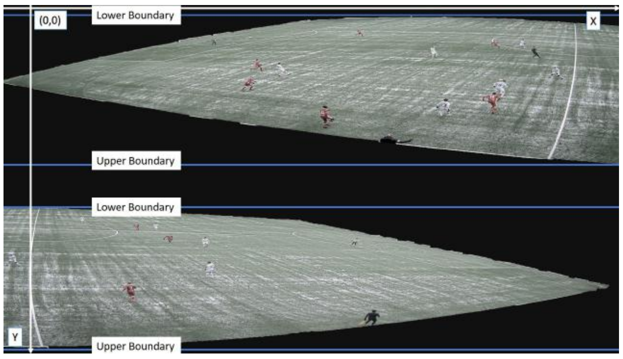


Figure 3.7: Image showing how the boundaries were set to weigh each pixel hit relative its position in the frame. See equation 3.4 to see how the boundaries were used.

The number of hits for each color bin was summarized, and the bins were sorted by most number of hits first. Color(s), represented by the bins, were chosen, by most hits first, to represent the background until 90% of the total number of hits were covered.

The background color(s) were updated every 5:th second. In between, the segmentation of each frame was done by calculating the euclidean distance, D for each pixel's color to the background color(s) in the 3D color space as

$$D = \sqrt{(P_\mathrm{R} - B_\mathrm{R})^2 + (P_\mathrm{G} - B_\mathrm{G})^2 + (P_\mathrm{B} - B_\mathrm{B})^2}\ , \tag{3.5}$$

where P_R, P_G and P_B are the intensity in the three color channels for a pixel and B_R, B_G and B_B are the color channel intensities for the background colors. The axes span from $0-255$ points and if the distance D, was smaller than 44.625 points (17.5% of 255) to any background color, the pixel was classified as background.

In the Spiideo application implementation were the number of possible background colors limited to 16 colors. It was discovered that 90% of the hits were reached for all recordings with six, or less, background colors. In this project, instead of choosing the amount of colors extracted to cover 90% of the hits, the segmentation was carried out six times for each recording having a fixed number of 1-6 background colors respectively. All six results were evaluated, and the best result for each recording are presented. The reason for this was to better understand the dynamics of the ColorCube for different number of background colors. This approach may lead to that the behaviour of ColorCube is not perfectly replicated, but it follows that the best possible result for the ColorCube are presented, even though the number of chosen background colors may differ from the original.

Color Cube Alterations

Five alterations of the ColorCube have been implemented and evaluated. These alterations are further described in this section. For each proposed alteration are only the changes that were made explained. The rest of each alteration was implemented in the same way as the current method, described in the previous section.

Alteration 1. Eight ColorCubes were initiated in eight different parts of the frame, as shown in Figure 3.8.

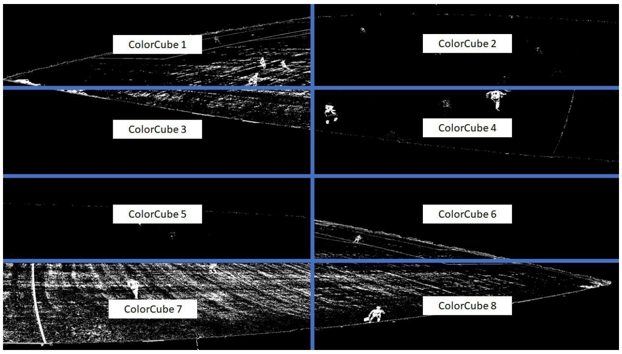


Figure 3.8: To better handle local variations in different parts of the frames, were the recordings divided into eight parts and a ColorCube was applied to each part.

Since the areas were smaller, were no weighting of the hits performed when extracting the dominant color(s) in each part. Instead gave each pixel's color a hit value of 1.0 in the 3-dimensional color space. This approach was developed to handle local variations in the background better. The eight different ColorCubes were allowed to have different amounts of dominant colors when segmenting. For each part, the extracted color(s) were determined by using the color bins with hits corresponding to 90% of the total number of hits in that part. The pixels in each part were classified as foreground or background depending on the euclidean distance in 3-dimensional space, as previously described.

Alteration 2. This alteration was developed to take advantage of the fact that most recordings have a green playing field, and therefore a green background. After the dominant colors had been extracted from the frame, the euclidean distance D, from the pixel color P to each background color B was instead calculated as

$$D = \sqrt{w_0(P_R - B_R)^2 + w_1(P_G - B_G)^2 + w_2(P_B - B_B)^2}\ , \qquad (3.6)$$

where $w_0 = w_2 = \frac{2}{5}$ and $w_1 = \frac{1}{5}$, leading to that differences in the color channels were weighted differently. Differences in the red and blue channels were taken into greater account, making the classification more sensitive for differences in these colors, and allowed more variations of green in the background. This approach was based on the fact that most of the Spiideo customers are football clubs. For a setup at a hockey arena or other sport where the playing field is of another color, the weights must be adjusted.

Alteration 3. An approach that could handle local variations and use information from the entire frame was developed. The dominant colors from the entire frame were extracted as before, but a ColorCube was also applied to a smaller area in the frame. Background colors from both areas were used to classify the pixels in the area. This approach was developed to replicate application usage as the recordings often are viewed in a zoomed in mode. The same background colors as used by the original method were used for the entire frame. The number of colors extracted from the smaller area was chosen based on the what gave the best classification result. This gives a hint on how many colors from the smaller area that are suitable to use. Note that only the pixels in the smaller area were classified. This alteration was not evaluated in full resolution, more information about the evaluation methods are presented in Chapter 3.3 on page 45.

Alteration 4. The original ColorCube was used for segmentation. Post-processing with *opening* and *closing* were applied to the resulting binary mask to remove noise and to close holes in the segments. These post-processing techniques are further described in Chapter 3.1 on Page 26.

Alteration 5. The euclidean distance threshold that decided whether a pixel was classified as background or foreground was changed. As the ColorCube often segmented parts of the background as foreground, a higher value of 63,75 points, 25% of 255, was tested to make the pixel classification more restrictive.

Optical Flow

As the background in the recordings were narrowed down to only include the playing field, movements were only expected to occur in the foreground. A motion detection algorithm called *dense optical flow* was developed by Farnebäck in [6]. A segmentation method based on this was implemented and evaluated. By approximating each neighborhood of both frames by quadratic polynomials, a vector field was created by observing how an exact polynomial transforms under translation. The displacement fields were estimated from the polynomial expansion coefficients.

The movement between two consecutive frames in the recording were calculated. As the algorithm required one channel images, all frames were converted to gray-scale. A 2-dimensional vector field representing the magnitude and direction for the movement of each pixel, was created. A visualization of the process can be seen in Figure 3.9.

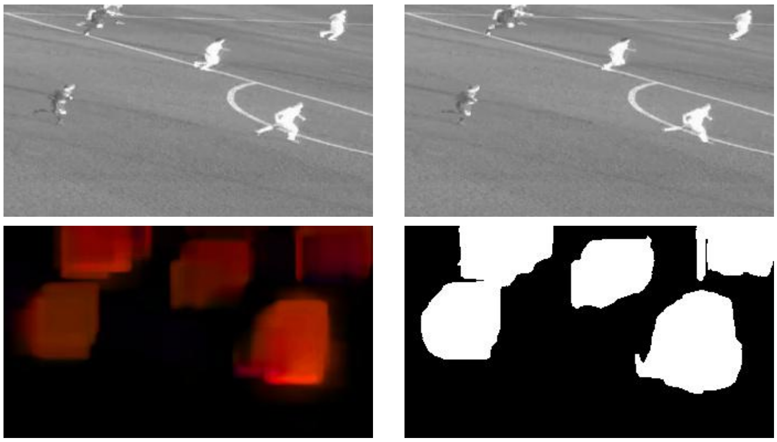


Figure 3.9: The two upper figures shows two following frames, on which the movement have been detected. The lower left figure shows the dense optical flow vector field that describes the movement between the two frames. Color represents vector direction and the intensity represents the magnitude of the vector. The lower right figure shows the resulting segmentation after using a threshold on the vector magnitudes.

A binary mask was created by classifying pixels containing considerable movement to foreground. The vector fields were transformed into an RGB image where color represented direction and intensity represented magnitude, as shown in Figure 3.9. A pixel in the binary mask P_{BM} was set to foreground or background by using a threshold for color intensity in each pixel. If the sum of the color intensity in all three channels was greater than a threshold T, the pixel was set to foreground or background as

$$P_{\mathrm{BM}} = \begin{cases} 1, & if\ P_{\mathrm{R}} + P_{\mathrm{G}} + P_{\mathrm{B}} \geq T \\ 0, & if\ P_{\mathrm{R}} + P_{\mathrm{G}} + P_{\mathrm{B}} < T \end{cases}, \tag{3.7}$$

where P_{R}, P_{G} and P_{B} were the color channel intensities for the pixel. The parameters used in the optical flow algorithm are presented in Table 3.2. They were chosen to be computationally effective while producing a satisfying motion detection result.

Table 3.2: Parameters used in the optical flow algorithm. Further details about impact of the parameters can be read in [26].

Pyramid Layers	Layer Resize	Window Size	Iterations	Polynomial Degree	STD
5	0.5	40	5	5	1.2

Machine Learning Classifier

Segmentation based on color will struggle when the foreground have the same color as the background, e.g. green player shirts on a green grass field. Motion based segmentation will struggle when the foreground stops moving, e.g. a player standing still. A novel machine learning classifier was developed to segment by combining information of both color and movement. The Python machine learning library *scikit-learn* [37] was used to implement the classifier.

Instead of classifying each pixel, was a hierarchical classification approach used. The frame was first divided into 16 equal parts, where each part is hence referred to as a *patch*. Each patch was classified as either background or foreground, by using features based on both color and movement in the patch. If a patch was classified as foreground, the patch was divided into 4 equal parts. Each smaller patch was then classified as background or foreground and the process was repeated. The recursive procedure is further explained in the following steps:

1. Features were calculated on the patch.
2. The patch was classified as foreground or background.
    - If background:
        - All pixels in the patch were classified as background.

- If foreground:
  - If the patch size was greater than 1500 pixels:
    - ∗ The part was divided into 4 equal smaller parts. Repeat steps 1-2 for each part.
  - If the patch size was smaller than 1500 pixels:
    - ∗ All pixels in the patch were classified as foreground.

An example of the hierarchical structure is shown in Figure 3.10.

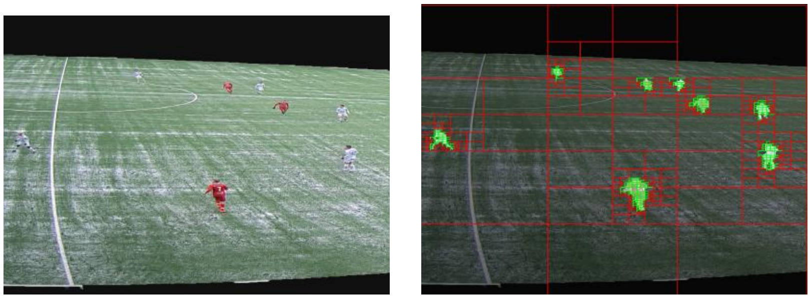


Figure 3.10: The hierarchical structure of the classifier. Red squares show patches classified as background while green squares show patches classified as foreground. Note that this figure shows a zoomed in area of the entire frame.

Two features were calculated for each patch. These features were designed to give relevant information regarding color and movement in the patch.

- Feature 1. Based on motion in the patch:
  - By applying the *Dense Optical Flow* algorithm (as previously described) to the current frame, movement (from the last frame) was measured and described by vector field. The sum of the magnitude of all vectors inside a patch were calculated, S_{M}. The first feature F_1 was then calculated as

$$F_1 = \frac{S_{\mathrm{M}}}{5H}, \tag{3.8}$$

where H was the height of the patch.

- Feature 2. Based on colors in the patch:
  - By applying *Improved Mixture of Gaussians* (MOG2) (as previously described), a binary mask for the patch was created. The sum of all foreground pixels in the patch were calculated, S_{P}. The second feature F_2 was then calculated as

$$F_2 = \frac{S_{\mathrm{P}}}{10H}, \tag{3.9}$$

where H was the height of the patch.

The size of the patches varied and three different classifiers were used to classify the patches depending on their size. The three possible patch sized are defined in Table 3.3.

Table 3.3: The definition of the three possible patch sizes.

Size	**Definition**
Big patches	Patch size > 500 000
Medium patches	100 000 < Patch size < 500 000
Small patches	Patch size < 100 000

To classify the patches as background or foreground, five different classification algorithms were considered (more information about the classifiers are presented in [38]). A sub-set containing 10 key-frames from each recording was used for a 10-fold cross validation to measure the accuracy and classification time for the different classifiers on the sub-set. This means that the subset was split into 10 sets. Each set was used for validation while the other 9 were used for training. The sub-set was randomly divided into 10 sets, and the 10 sets were the same when testing all classifiers. The mean values of accuracy and classification time are presented in Table 3.4, the most accurate methods were chosen, and are presented in bold.

Table 3.4: Mean values of accuracy and classification time for the considered classifiers on a 10-fold cross-validation on a sub-set of the data for the different patch sizes. The most accurate methods for each patch size were chosen and are presented in bold.

Patch Size	Classifier	Accuracy	Time
Big	Logistic Regression	0.75	0.014
	Linear Discriminant Analysis	0.73	0.011
	K Neighbors Classifier	0.91	0.009
	Decision Tree Classifier	0.92	0.006
	Random Forest Classifier	**0.94**	**0.027**
	Gaussian Naive Bayes	0.72	0.005
Medium	Logistic Regression	0.80	0.009
	Linear Discriminant Analysis	0.77	0.006
	K Neighbors Classifier	0.91	0.118
	Decision Tree Classifier	0.90	0.009
	Random Forest Classifier	**0.92**	**0.042**
	Gaussian Naive Bayes	0.73	0.004
Small	Logistic Regression	0.78	0.352
	Linear Discriminant Analysis	0.76	0.120
	K Neighbors Classifier	**0.87**	**1.68**
	Decision Tree Classifier	0.83	0.899
	Random Forest Classifier	0.85	4.046
	Gaussian Naive Bayes	0.64	0.081

The segmentation of each recording was carried out in a similar way as the cross-validation. When segmenting a recording, the classifiers were trained on all key-frames from the other 12 recordings. Default settings and parameters were used for the classifiers [37].

3.3 Pixel Classification Evaluation

All methods were evaluated and compared by calculating the *precision*, *recall* and F_1-score by comparing the segmentation results to the ground truth. The three pixel classification scores are further explained in Section Metrics on page 45.

Full Resolution.
The mean value of the three scores were calculated for all key-frames, for each method and recording, in full resolution. The mean value of the three scores were also calculated for all recordings.

Zoomed area.
A user seldom view the recordings in full resolution, but rather in a zoomed in mode. To perform a more user experience related pixel classification evaluation, the same three scores were also calculated on two zoomed in areas. The ten high-resolution center-line recordings were chosen for the "zoomed area" evaluation. The two areas were the same for all recordings, and are shown in Figure 3.11. Note that the areas were only evaluated when there was foreground present in the area. The mean value for each of the three scores were calculated for all key-frames, for each method and recording. The mean value of the three scores for all recordings were also calculated.

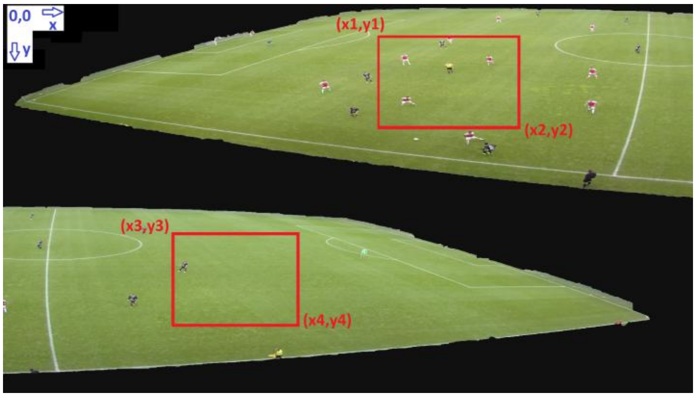


Figure 3.11: A user often watches a recording in a zoomed in mode. As a compliment to the full resolution evaluation, the two areas in the frame were also evaluated for all methods and recordings. The upper right area spans from $(x_1, y_1) = (2083, 244)$ to $(x_2, y_2) = (2917, 712)$. The lower left area spans from $(x_3, y_3) = (948, 1367)$ to $(x_4, y_4) = (1800, 1804)$.

Metrics

When evaluating the segmentation methods, the resulting binary mask was compared to the ground truth for all key-frames. The three metrics used for evaluation were based on the four possible cases for a classified pixel:

- **True Positive** (T_P): A foreground pixel classified as foreground.
- **False Positive** (F_P): A background pixel classified as foreground.
- **True Negative** (T_N): A background pixel classified as background.
- **False Negative** (F_N): A foreground pixel classified as background.

A visualization of these cases can be seen in Figure 3.12 where a binary mask is compared with the ground truth.

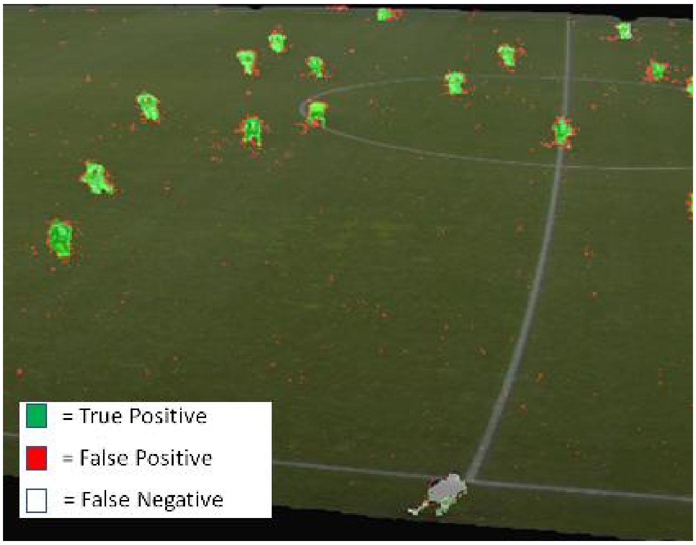


Figure 3.12: Segmented pixels were classified as True Positive, False Positive and False Negative when compared with the ground truth. Note that True negative pixels are shown as transparent.

The recordings contain considerably more background than foreground, leading to the number of pixels classified as True Negative would be at a great majority, regardless of segmentation performance. Measures including True Negative pixels was therefore excluded. The three scores used to evaluate the background segmentation methods were Precision, Recall and F_1-score [15].

Precision, P, is the fraction of correctly segmented foreground pixels of all pixels classified as foreground, and was calculated as

$$P = \frac{T_\mathrm{P}}{T_\mathrm{P} + F_\mathrm{P}} \,. \tag{3.10}$$

Recall, R, is the fraction of correctly segmented foreground pixels of all true foreground pixels, and was calculated as

$$R = \frac{T_\mathrm{P}}{T_\mathrm{P} + F_\mathrm{N}} \,. \tag{3.11}$$

F_1-score is the harmonic mean of precision and recall, and was calculated as

$$F_1 = \frac{2PR}{P + R} \,. \tag{3.12}$$

3.4 Complexity Evaluation

The complexity for each method was estimated by measuring the computational time and the average CPU utilization.

All methods were applied to the same recording, with a length of 8000 frames. Each method was timed ten times, and the mean computational time was calculated. The time ratio relative the currently used method, the ColorCube, was also calculated.

The average system-wide CPU utilization percentage during performed segmentation was measured for each method by using the *psutil*-library in Python [35]. The function *psutil.cpu_percent* was used to extract the values for CPU utilization. The CPU utilization ratio relative the currently used method was also calculated.

The time and CPU-utilization was measured using an Intel Core i7-6700HQ 2.6 GHz processor with 16 GB DDR3 RAM and Windows 10 Home x64.

3.5 Visual Evaluation

When evaluating methods by pixel classification scores, high values in precision and recall does not necessarily imply better visual perception. What a human viewer perceives when watching a recording can be difficult for a computer to measure and quantify.

The pixel classification scores treats all correctly or incorrectly classified pixels the same way, regardless of relevance for user perception. For example, one method may have missed an entire player while another method have some Gaussian noise all over the frame. These two methods may end up with a similar classification score, but are perceived very differently by the viewer.

To evaluate the segmentation methods by user perception, a visual evaluation was performed. Four methods that proved their relevance according to classification and complexity performance, were chosen for the visual evaluation:

- **ColorCube**: The currently used method.
- **ColorCube Alteration 3**: One of the proposed improvements to the currently used method.
- **MOG2**: Relevant due to its good overall segmentation performance and low complexity.
- **Median T=35**: Relevant due to its computational speed and robust segmentation performance.

Four different recordings with different conditions[3] were chosen:

- **Gefle IP**: Snow on a green field.

[3]More details about the recordings can be seen in chapter 2 (*Data*).

- **Kalmar**: Regular green field. No shadows.
- **Malmo IP (2)**: Sun on a green field and large shadows.
- **Ice Hockey**: Ice with artificial lights and small shadows.

Five seconds of each recording were sampled and segmented with the four methods. A zoomed in area with active game-play were chosen. An on-screen drawing was applied to the area according to the resulting binary mask. The drawing was shown "on top" of the pixels classified as background, while pixels classified as foreground were shown as normally, see Figure 3.13.

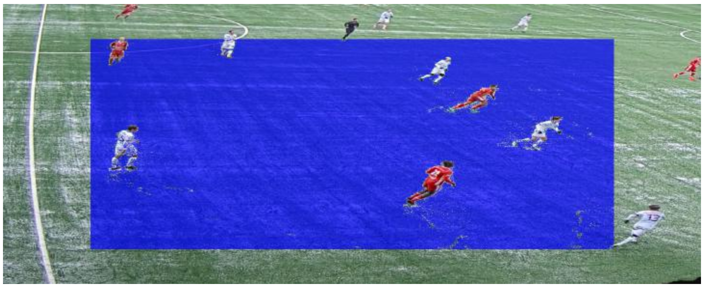


Figure 3.13: An on-screen drawing was applied to a zoomed in area according to the segmentation results of the different methods. It was designed to look like one of the actual analysis tools present in the Spiideo application.

For each recording were the four video sequences, segmented with different methods, presented next to each other as shown in Figure 3.14. The video sequences were looped until the viewer had internally ranked the four videos due to their segmentation performance by looking at the on-screen drawings. The viewer was asked to rank the four methods internally from 1 to 4, where 1 was the best and 4 was the worst. Two methods could not have the same rank. The viewer had no prior knowledge about which methods that were evaluated or where they were placed. The placing of each method was changed for each recording to avoid bias.

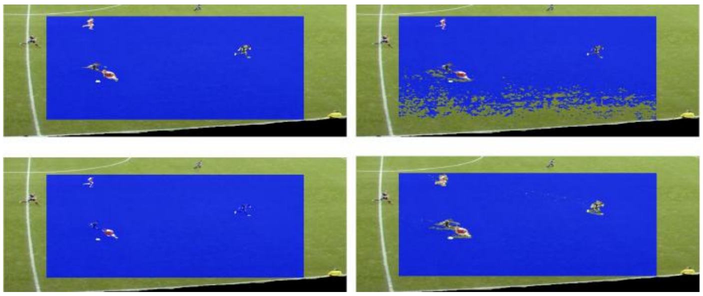


Figure 3.14: The visual evaluation was performed by presenting the same video sequence with on-screen drawings based on the result from four different segmentation methods. The viewer was asked to internally rank the four videos without prior knowledge about the methods or their position.

Six employees at Spiideo were chosen to participate in the visual evaluation. They have relevant knowledge of image analysis and the user needs of the Spiideo application. An example of the evaluation score sheet are shown in Appendix B.

4

Results

4.1 Data-set

A key-frame from each recording in the data-set can be seen in Figure 4.1 on page 52. The corresponding ground-truth binary masks can be seen in Figure 4.2 on page 53.

4.2 Segmentation Methods

Examples of segmentation results from recording Kalmar can be seen in Figure 4.3 on page 54. Examples of segmentation results from recording Gefle can be seen in Figure 4.4 on page 55. Examples from all recordings are presented in Appendix A.

4.3 Pixel Classification Evaluation

The metrics for all segmentation methods for the full resolution pixel classification evaluation can be seen in Table 4.1 on page 56. The metrics for all segmentation methods for the zoomed area pixel classification evaluation can be seen in Table 4.2 on page 58.

4.4 Complexity

The results of the complexity measurements are shown in Table 4.3 on page 60.

The pixel classification scores are plotted in relation to the complexity measurements in Figure 4.5 and Figure 4.6 on page 61 for the full resolution evaluation, and in Figure 4.7 and Figure 4.8 on page 62 for the "zoomed area" evaluation.

4.5 Visual Evaluation

The results of the visual evaluation are shown in Table 4.4 on page 63.

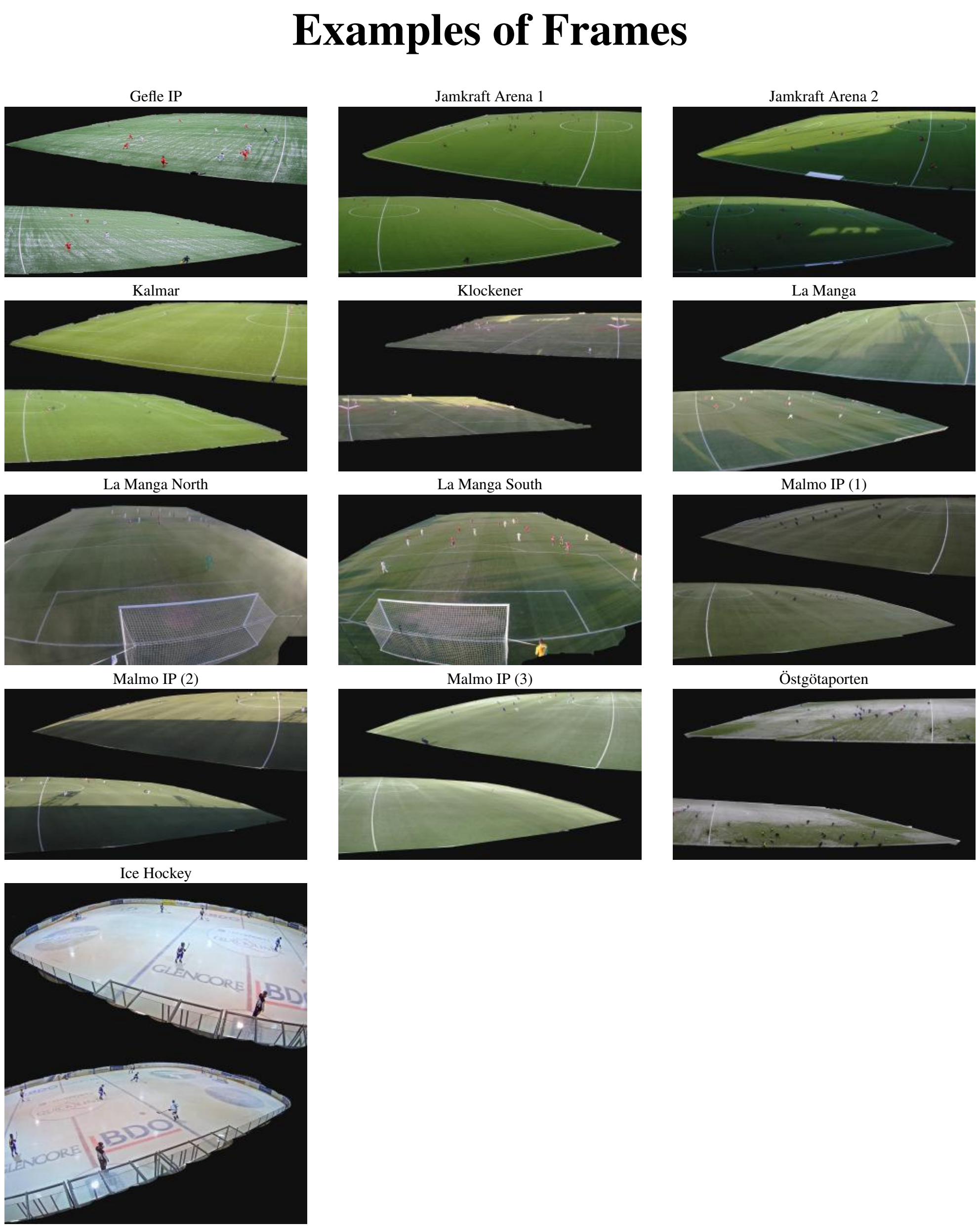


Figure 4.1: A frame from each recording in the data-set.

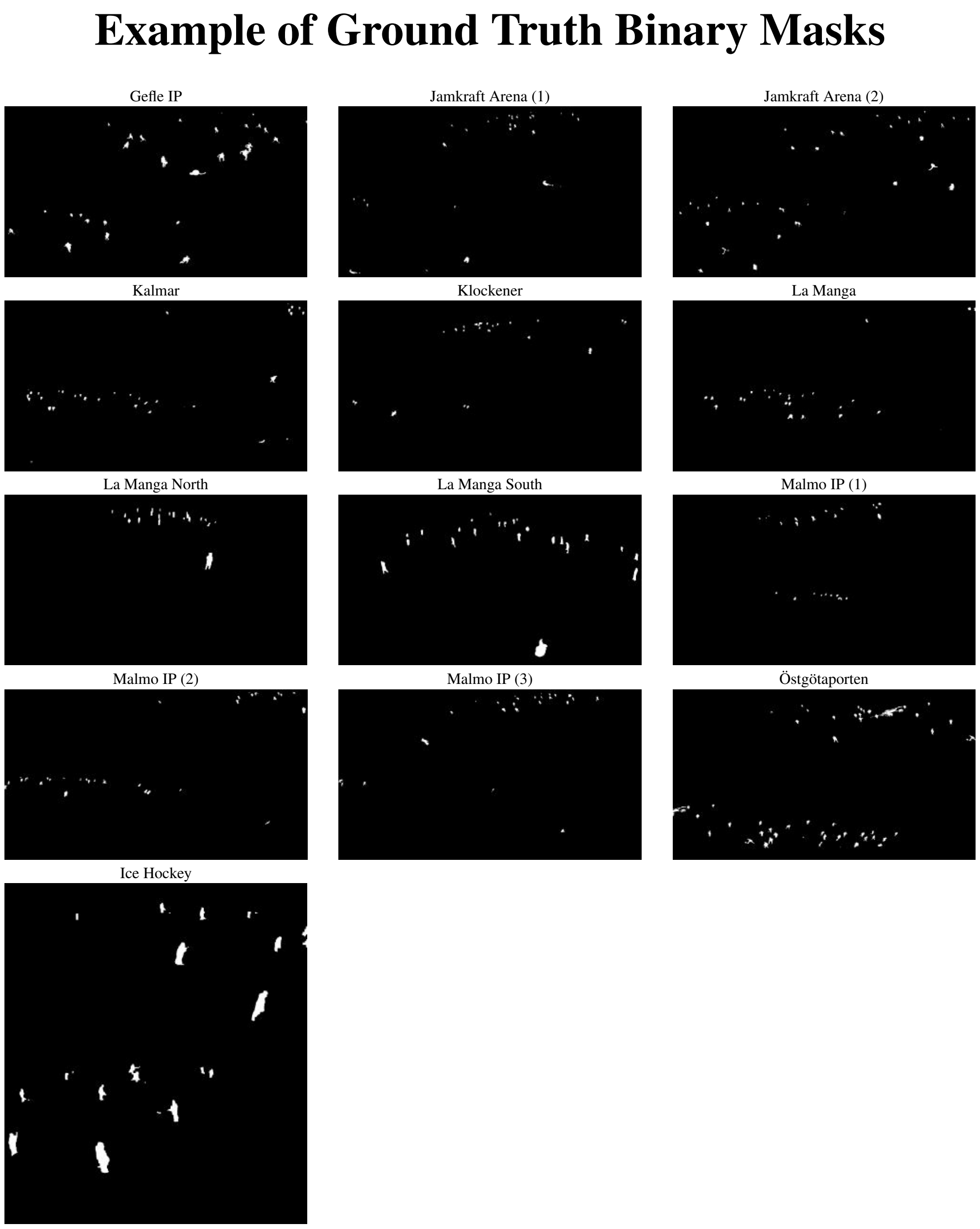


Figure 4.2: A ground truth binary mask from each recording in the data-set.

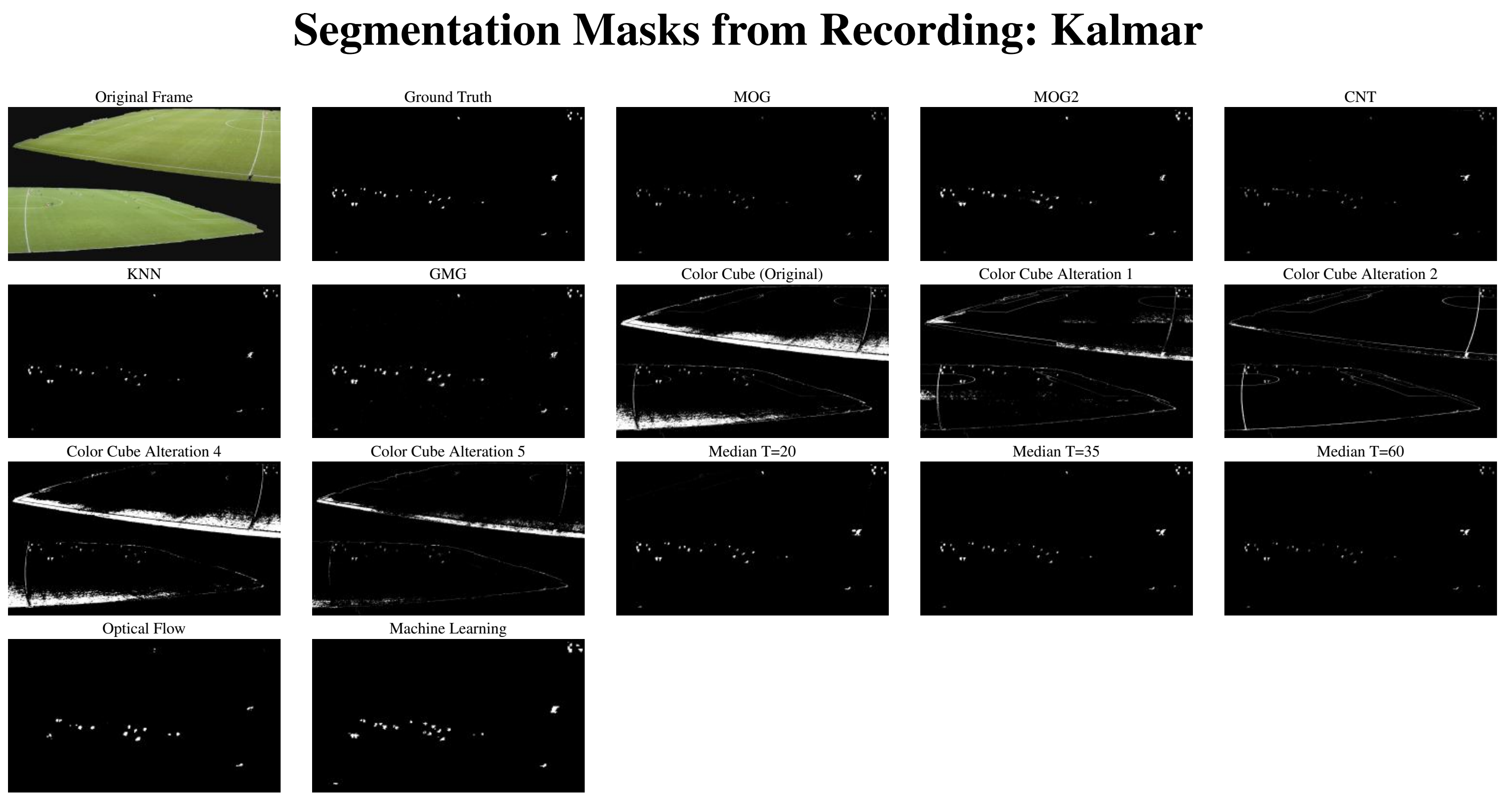


Figure 4.3: A binary mask from each method for recording Kalmar. The most upper left image is the frame which is segmented.

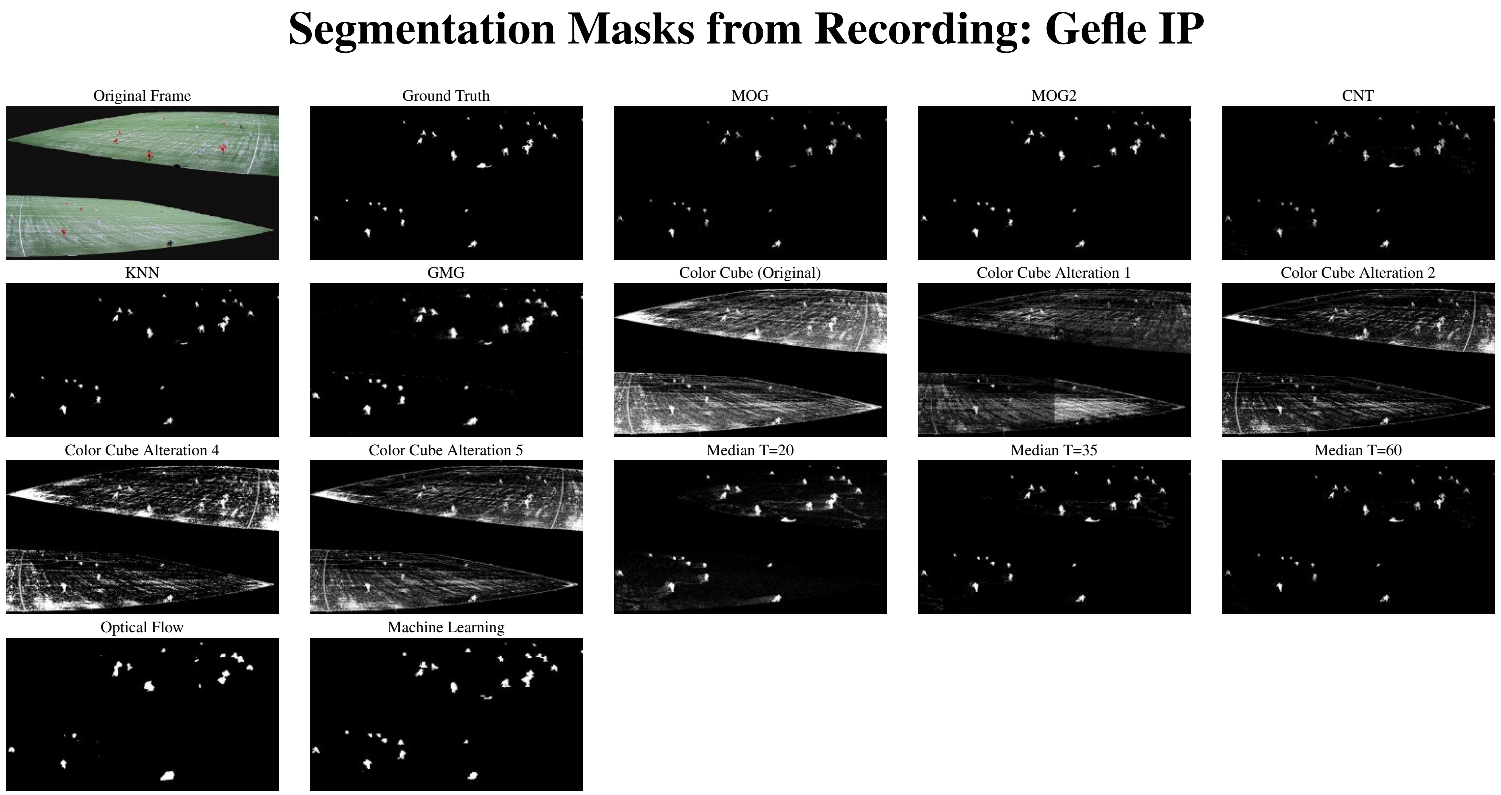


Figure 4.4: A binary mask from each method for recording Gefle IP. The most upper left image is the frame which is segmented.

Full Resolution: Pixel Classification Scores

Table 4.1: Results of the pixel classification scores calculated in full resolution. The most right column shows the mean value and standard deviation for all recordings. The best F_1-score mean values are marked out in bold. Note that: **P** = Precision, **R** = Recall and **F** = F_1-score. The table continues on the next page.

Method		Gefle IP	Jamkraft Arena (1)	Jamkraft Arena (2)	Kalmar	Klockener	La Manga	La Manga North	La Manga South	Malmo IP (1)	Malmo IP (2)	Malmo IP (3)	Östgötaporten	Ice Hockey	Mean (STD)
	P	0.25	0.66	0.45	0.81	0.30	0.34	0.05	0.09	0.51	0.22	0.44	0.26	0.23	0.35
Median T=20	R	0.91	0.97	0.75	0.90	0.75	0.93	0.83	0.94	0.89	0.86	0.93	0.89	0.89	0.88
	F	0.39	0.79	0.56	0.85	0.43	0.50	0.10	0.16	0.65	0.35	0.60	0.41	0.36	0.47 (0.22)
	P	0.67	0.78	0.55	0.90	0.44	0.67	0.07	0.13	0.68	0.61	0.63	0.51	0.59	0.56
Median T=35	R	0.84	0.88	0.60	0.78	0.64	0.82	0.62	0.85	0.77	0.71	0.83	0.79	0.75	0.76
	F	0.74	0.83	0.57	0.84	0.52	0.73	0.12	0.22	0.73	0.66	0.72	0.62	0.66	**0.61** (0.22)
	P	0.85	0.84	0.60	0.94	0.61	0.93	0.07	0.20	0.73	0.52	0.74	0.56	0.65	0.63
Median T=60	R	0.70	0.65	0.36	0.61	0.48	0.66	0.34	0.73	0.67	0.56	0.67	0.68	0.62	0.59
	F	0.77	0.74	0.45	0.74	0.54	0.77	0.12	0.31	0.70	0.54	0.70	0.61	0.63	**0.59** (0.20)
	P	0.91	0.88	0.64	0.96	0.66	0.97	0.53	0.72	0.79	0.73	0.60	0.89	0.95	0.79
MOG	R	0.64	0.54	0.29	0.53	0.41	0.60	0.29	0.66	0.32	0.49	0.64	0.52	0.55	0.50
	F	0.75	0.67	0.39	0.69	0.51	0.74	0.38	0.69	0.46	0.58	0.62	0.62	0.70	**0.60** (0.13)
	P	0.82	0.65	0.57	0.78	0.60	0.77	0.52	0.54	0.80	0.58	0.17	0.83	0.90	0.66
MOG2	R	0.78	0.91	0.75	0.86	0.70	0.90	0.86	0.92	0.44	0.80	0.85	0.66	0.54	0.77
	F	0.80	0.76	0.65	0.82	0.65	0.83	0.65	0.68	0.57	0.67	0.28	0.74	0.67	**0.67** (0.14)
	P	0.81	0.84	0.71	0.92	0.65	0.92	0.36	0.47	0.80	0.64	0.38	0.87	0.92	0.71
KNN	R	0.83	0.90	0.67	0.80	0.68	0.88	0.72	0.90	0.42	0.77	0.87	0.68	0.58	0.75
	F	0.82	0.87	0.69	0.86	0.66	0.90	0.48	0.62	0.55	0.70	0.53	0.76	0.71	**0.70** (0.14)
	P	0.43	0.41	0.34	0.55	0.33	0.29	0.17	0.22	0.22	0.44	0.04	0.58	0.42	0.34
GMG	R	0.86	0.77	0.72	0.88	0.75	0.85	0.91	0.91	0.50	0.74	0.84	0.68	0.89	0.79
	F	0.57	0.53	0.47	0.68	0.46	0.43	0.29	0.36	0.31	0.55	0.08	0.62	0.57	0.46 (0.16)
	P	0.75	0.80	0.36	0.91	0.21	0.60	0.05	0.12	0.71	0.49	0.44	0.64	0.65	0.52
CouNT	R	0.65	0.69	0.30	0.55	0.43	0.55	0.31	0.63	0.70	0.59	0.77	0.71	0.67	0.58
	F	0.69	0.74	0.33	0.68	0.28	0.57	0.09	0.21	0.71	0.53	0.56	0.67	0.66	0.52 (0.22)
	P	0.45	0.31	0.32	0.43	0.28	0.27	0.23	0.31	0.10	0.24	0.20	0.43	0.41	0.31
Optical Flow	R	0.53	0.41	0.42	0.52	0.50	0.41	0.79	0.58	0.17	0.37	0.51	0.36	0.74	0.49
	F	0.48	0.35	0.37	0.47	0.36	0.32	0.35	0.40	0.13	0.29	0.29	0.39	0.53	0.36 (0.10)
	P	0.41	0.37	0.15	0.46	0.36	0.40	0.16	0.22	0.41	0.30	0.05	0.51	0.57	0.34
ML Classifier	R	0.87	0.85	0.72	0.90	0.89	0.90	0.90	0.92	0.63	0.84	0.94	0.8	0.88	0.85
	F	0.56	0.52	0.25	0.61	0.51	0.55	0.27	0.36	0.49	0.44	0.09	0.62	0.69	0.47 (0.18)

P = Precision, **R** = Recall, **F** = F_1-score and **C** = Number of dominant colors.

Method		**Gefle IP**	**Jamkraft Arena (1)**	**Jamkraft Arena (2)**	**Kalmar**	**Klockener**	**La Manga**	**La Manga North**	**La Manga South**	**Malmo IP (1)**	**Malmo IP (2)**	**Malmo IP (3)**	**Östgötaporten**	**Ice Hockey**	**Mean (STD)**
	C	1	3	4	2	3	3	2	3	3	3	3	4	4	-
ColorCube (Current)	P	0.01	0.04	0.08	0.03	0.03	0.04	0.02	0.04	0.04	0.04	0.04	0.08	0.05	0.04
	R	0.89	0.83	0.43	0.77	0.68	0.74	0.59	0.85	0.72	0.69	0.71	0.69	0.63	0.71
	F	0.02	0.08	0.13	0.06	0.06	0.08	0.04	0.08	0.08	0.08	0.08	0.14	0.09	0.08 (0.03)
ColorCube (Alteration 1)	P	0.01	0.05	0.04	0.06	0.05	0.04	0.02	0.04	0.04	0.05	0.04	0.09	0.06	0.05
	R	0.66	0.80	0.51	0.85	0.70	0.87	0.64	0.86	0.82	0.64	0.76	0.73	0.63	0.73
	F	0.02	0.09	0.07	0.11	0.09	0.08	0.04	0.08	0.08	0.09	0.08	0.16	0.11	0.08 (0.03)
ColorCube (Alteration 2)[1]	P	0.01	0.04	0.03	0.10	0.06	0.06	0.01	0.05	0.02	0.08	0.06	0.15	0.07	0.06
	R	0.77	0.22	0.11	0.58	0.41	0.53	0.38	0.75	0.20	0.52	0.63	0.28	0.43	0.45
	F	0.02	0.07	0.05	0.17	0.10	0.11	0.02	0.09	0.04	0.14	0.11	0.20	0.12	0.10 (0.06)
ColorCube (Alteration 3)								Not evaluated in full resolution							
ColorCube (Alteration 4)[1]	P	0.02	0.05	0.08	0.03	0.04	0.07	0.02	0.07	0.07	0.04	0.08	0.11	0.05	0.06
	R	0.87	0.75	0.33	0.72	0.61	0.66	0.48	0.84	0.63	0.62	0.59	0.59	0.42	0.62
	F	0.04	0.09	0.13	0.06	0.08	0.13	0.04	0.13	0.13	0.08	0.14	0.19	0.09	0.10 (0.04)
ColorCube (Alteration 5)[1]	P	0.01	0.09	0.06	0.06	0.05	0.06	0.01	0.05	0.04	0.07	0.05	0.15	0.07	0.06
	R	0.82	0.67	0.21	0.66	0.57	0.62	0.44	0.78	0.50	0.57	0.72	0.42	0.53	0.58
	F	0.02	0.16	0.09	0.11	0.09	0.11	0.02	0.09	0.07	0.12	0.09	0.22	0.12	0.10 (0.05)

[1] Same background color(s) used as in the currently used ColorCube.

Zoomed Area: Pixel Classification Scores

Table 4.2: Results of the pixel classification scores calculated on the zoomed in area. The most right column shows the mean and standard deviation for all recordings. The best F_1-score mean values are marked out in bold. Note that: **P** = Precision, **R** = Recall and **F** = F_1-score. The table continues on the next page.

Method		**Gefle IP**	**Jamkraft Arena (1)**	**Jamkraft Arena (2)**	**Kalmar**	**Klockener**	**La Manga**	**Malmo IP (1)**	**Malmo IP (2)**	**Malmo IP (3)**	**Östgötaporten**	**Mean (STD)**
	P	0.09	0.60	0.34	0.73	0.22	0.27	0.47	0.23	0.33	0.18	0.35
Median T=20	R	0.88	0.91	0.88	0.86	0.69	0.84	0.90	0.81	0.89	0.88	0.85
	F	0.16	0.72	0.49	0.79	0.33	0.41	0.62	0.36	0.48	0.30	0.47 (0.20)
	P	0.39	0.70	0.45	0.84	0.44	0.44	0.68	0.45	0.49	0.53	0.54
Median T=35	R	0.80	0.83	0.76	0.78	0.62	0.75	0.84	0.74	0.82	0.80	0.77
	F	0.52	0.76	0.57	0.81	0.51	0.55	0.75	0.56	0.61	0.64	**0.63** (0.11)
	P	0.63	0.74	0.47	0.86	0.56	0.74	0.72	0.36	0.71	0.55	0.63
Median T=60	R	0.69	0.73	0.54	0.68	0.53	0.65	0.78	0.66	0.71	0.73	0.67
	F	0.66	0.73	0.50	0.76	0.54	0.69	0.75	0.47	0.71	0.63	**0.64** (0.11)
	P	0.82	0.78	0.47	0.87	0.61	0.83	0.58	0.49	0.76	0.82	0.70
MOG	R	0.66	0.66	0.48	0.65	0.48	0.61	0.27	0.63	0.70	0.58	0.57
	F	0.73	0.72	0.47	0.74	0.54	0.70	0.37	0.55	0.73	0.68	**0.62** (0.13)
	P	0.74	0.60	0.47	0.65	0.52	0.67	0.58	0.43	0.15	0.80	0.56
MOG2	R	0.78	0.85	0.89	0.86	0.70	0.83	0.37	0.83	0.93	0.70	0.77
	F	0.76	0.70	0.62	0.74	0.60	0.74	0.45	0.57	0.26	0.75	**0.62** (0.16)
	P	0.72	0.73	0.56	0.83	0.57	0.82	0.62	0.47	0.77	0.82	0.69
KNN	R	0.78	0.86	0.82	0.80	0.66	0.83	0.48	0.78	0.83	0.72	0.76
	F	0.75	0.79	0.67	0.81	0.61	0.82	0.54	0.59	0.80	0.77	**0.72** (0.10)
	P	0.33	0.29	0.26	0.36	0.16	0.16	0.09	0.26	0.05	0.49	0.25
GMG	R	0.88	0.79	0.91	0.90	0.75	0.82	0.43	0.87	0.90	0.78	0.80
	F	0.48	0.42	0.40	0.51	0.26	0.27	0.15	0.40	0.09	0.60	0.36 (0.16)
	P	0.54	0.73	0.25	0.84	0.25	0.43	0.72	0.36	0.61	0.58	0.53
CouNT	R	0.65	0.72	0.48	0.63	0.49	0.60	0.83	0.67	0.73	0.74	0.65
	F	0.59	0.72	0.33	0.72	0.33	0.50	0.77	0.47	0.66	0.65	0.57 (0.16)
	P	0.50	0.37	0.35	0.44	0.27	0.31	0.41	0.30	0.29	0.51	0.38
Optical Flow	R	0.58	0.41	0.67	0.74	0.52	0.72	0.21	0.77	0.82	0.38	0.58
	F	0.54	0.39	0.46	0.55	0.36	0.43	0.28	0.43	0.43	0.44	0.43 (0.08)
	P	0.55	0.43	0.31	0.51	0.39	0.48	0.51	0.31	0.04	0.57	0.41
ML Classifier	R	0.81	0.87	0.9	0.92	0.88	0.9	0.55	0.95	1	0.78	0.86
	F	0.66	0.58	0.46	0.66	0.54	0.63	0.53	0.47	0.08	0.66	0.53 (0.17)

P = Precision, **R** = Recall, **F** = F_1-score, **C** = Number of dominant colors for the entire frame, and **C2** = Number of dominant colors for the zoomed in area.

Method		**Gefle IP**	**Jamkraft Arena (1)**	**Jamkraft Arena (2)**	**Kalmar**	**Klockener**	**La Manga**	**Malmo IP (1)**	**Malmo IP (2)**	**Malmo IP (3)**	**Östgötaporten**	**Total**
ColorCube (Current)	C	1	3	4	2	3	3	3	3	3	4	-
	P	0.03	0.55	0.54	0.21	0.04	0.34	0.70	0.08	0.58	0.14	0.32
	R	0.81	0.70	0.48	0.66	0.58	0.66	0.82	0.68	0.70	0.62	0.67
	F	0.06	0.62	0.51	0.32	0.07	0.45	0.76	0.14	0.63	0.23	0.38 (0.25)
ColorCube (Alteration 1)	P	0.02	0.11	0.12	0.39	0.06	0.05	0.60	0.08	0.24	0.16	0.18
	R	0.61	0.67	0.57	0.73	0.63	0.72	0.81	0.66	0.65	0.66	0.67
	F	0.04	0.19	0.20	0.51	0.11	0.09	0.69	0.14	0.35	0.26	0.26 (0.20)
ColorCube (Alteration 2)[1]	P	0.03	0.32	0.40	0.54	0.09	0.14	0.61	0.20	0.87	0.13	0.33
	R	0.76	0.22	0.19	0.65	0.46	0.54	0.20	0.56	0.56	0.28	0.44
	F	0.06	0.26	0.26	0.59	0.15	0.22	0.30	0.29	0.68	0.18	0.30 (0.19)
ColorCube (Alteration 3)	C2/C	4 /1	2 /3	3 /4	5 /2	5 /3	5 /3	2 /3	4/3	2 /3	2/4	-
	P	0.11	0.50	0.30	0.49	0.15	0.55	0.46	0.59	0.56	0.29	0.40
	R	0.60	0.60	0.46	0.54	0.33	0.49	0.34	0.45	0.63	0.68	0.51
	F	0.19	0.55	0.36	0.51	0.21	0.52	0.39	0.51	0.59	0.41	0.42 (0.14)
ColorCube (Alteration 4)[1]	P	0.05	0.65	0.64	0.46	0.05	0.38	0.75	0.09	0.67	0.18	0.39
	R	0.76	0.61	0.39	0.61	0.54	0.60	0.81	0.57	0.69	0.50	0.61
	F	0.09	0.63	0.48	0.52	0.09	0.47	0.78	0.16	0.68	0.26	0.42 (0.25)
ColorCube (Alteration 5)[1]	P	0.03	0.58	0.57	0.79	0.07	0.37	0.74	0.18	0.83	0.24	0.44
	R	0.73	0.56	0.27	0.58	0.45	0.55	0.71	0.59	0.62	0.35	0.54
	F	0.06	0.57	0.37	0.67	0.12	0.44	0.72	0.28	0.71	0.28	0.42 (0.24)

[1] Same background color(s) used as in the currently used ColorCube.

Complexity Evaluation

Table 4.3: Results of the complexity measurements. The ratios for time and CPU usage are calculated relative the ColorCube (bold), as it is the currently used method.

Method	**Time Used (s)**	**Time Ratio**	**CPU Usage (%)**	**CPU Ratio**
Median	235.87	0.81	3.00	1
MOG	1947.46	6.66	12.00	4
MOG2	817.24	2.79	9.00	3
KNN	2519.55	8.62	8.00	2.67
GMG	3681.68	12.59	27.00	9
CouNT	509.46	1.74	4.00	1.33
ColorCube	**292.40**	**1.00**	**3.00**	**1**
ColorCube (Alteration 1)	384.07	1.31	4.00	1.33
ColorCube (Alteration 2)[1]	-	-	-	-
ColorCube (Alteration 3)	376.34	1.29	4	1.33
ColorCube (Alteration 4)	699.20	2.39	4	1.33
ColorCube (Alteration 5)[1]	-	-	-	-
Optical Flow	34403.22	117.66	8.00	2.67
ML Classifier[2]	-	-	-	-

[1] Alteration 2 and 5 were not measured. The difference lies in the rendering of the frame, which leads to the same complexity as the original ColorCube.
[2] The Machine Learning-classifier was not measured. It used both MOG and Optical Flow for feature extraction, using the complexity for both of these should give a pointer about the complexity.

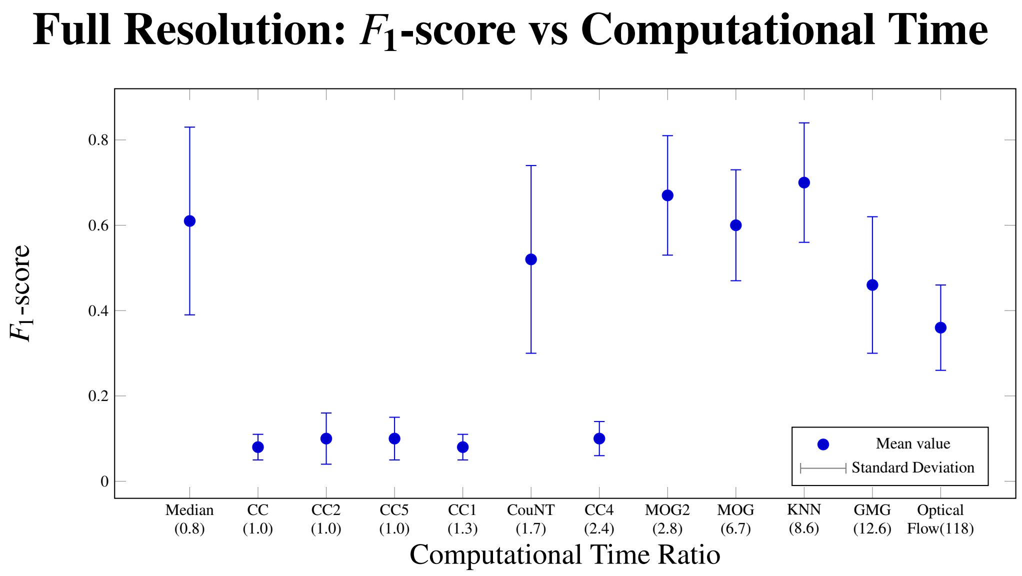


Figure 4.5: The y-axis shows the mean value and standard deviation for the F_1-score calculated in full resolution on the entire data-set. The x-axis shows the evaluated methods and their relative computational time. Note that the x-axis is not scaled. The machine learning classifier is not included due to that its complexity was not measured.

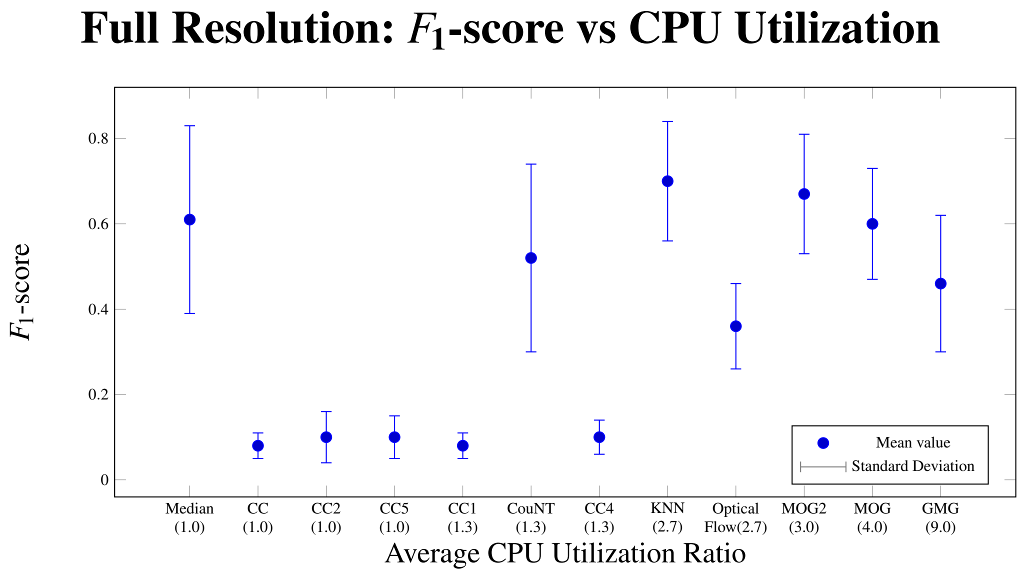


Figure 4.6: The y-axis shows the mean value and standard deviation for the F_1-score calculated in full resolution on the entire data-set. The x-axis shows the evaluated methods and their relative average CPU utilization. Note that the x-axis is not scaled. The machine learning classifier is not included due to that its complexity was not measured.

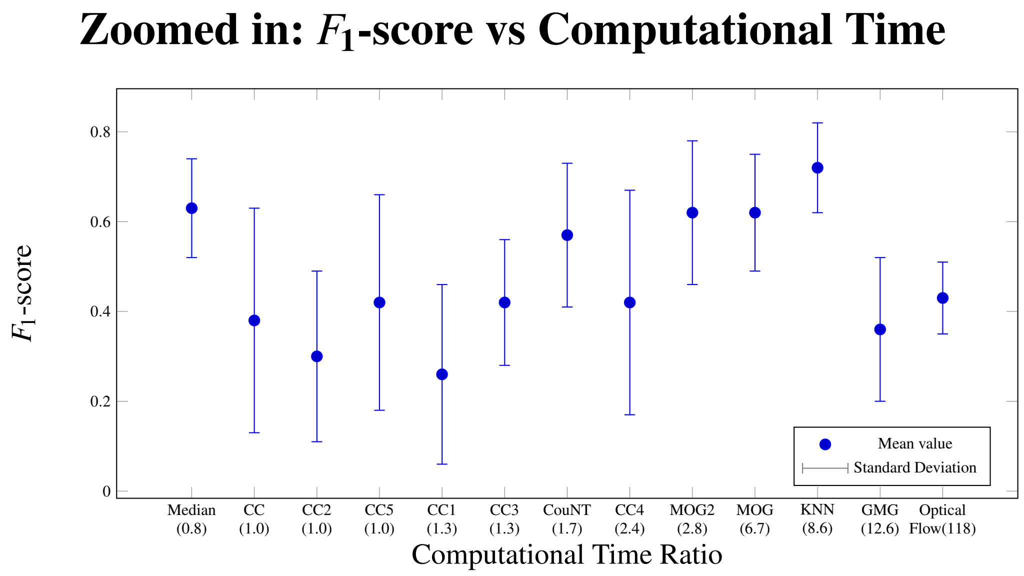


Figure 4.7: The y-axis shows the mean value and standard deviation for the F_1-score calculated in the zoomed in areas on the entire data-set. The x-axis shows the evaluated methods and their relative computational time. Note that the x-axis is not scaled. The machine learning classifier is not included due to that its complexity was not measured.

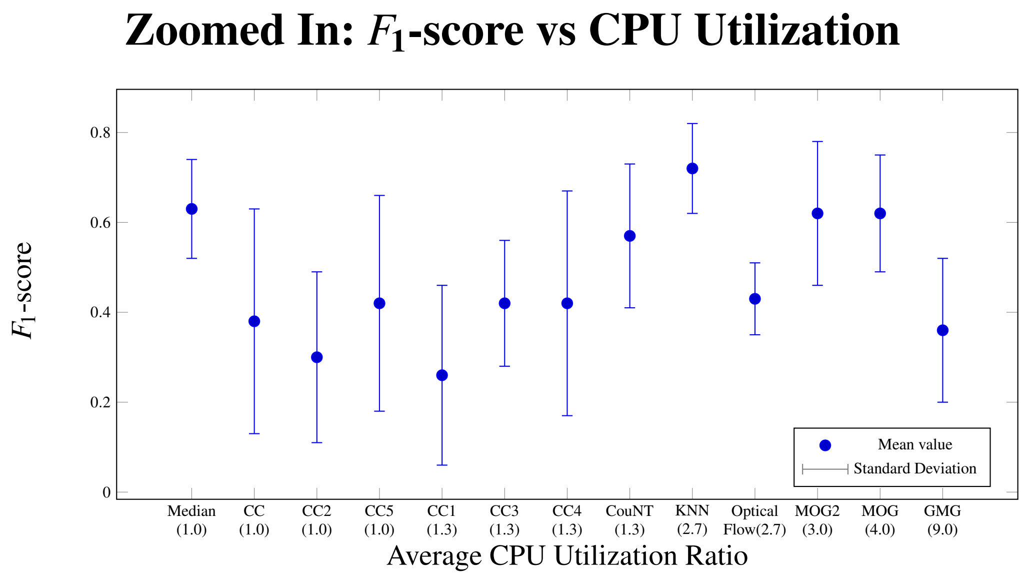


Figure 4.8: The y-axis shows the mean value and standard deviation for the F_1-score calculated in the zoomed in areas on the entire data-set. The x-axis shows the evaluated methods and their relative average CPU utilization. Note that the x-axis is not scaled. The machine learning classifier is not included due to that its complexity was not measured.

Visual Evaluation

Table 4.4: Results of the visual evaluation. The segmentation results of the four methods were ranked from 1 (best) to 4 (worst) for four different recordings. The best mean values are marked out.

Method	Gefle	Kalmar	Hockey	Malmö IP (2)	Mean
CC	4.00	3.00	4.00	3.67	3.67
CC3	2.50	3.67	3.00	2.70	2.95
Median	1.00	1.67	1.50	2.00	**1.54**
MOG2	2.50	1.67	1.50	1.67	**1.83**

5

Discussion

The aim of this project was to improve the current background foreground segmentation used in Spiideo's application, by evaluating the two following options:

- Find an alternative method with better classification performance with similar complexity.
- Find improvements to the currently used segmentation method.

5.1 Results

The two methods that showed the best overall classification results were MOG2 and KNN. However, looking at the computational time, MOG2 tripled it, and *KNN* gave almost a tenfold increase, compared to the ColorCube. The runner up to these two methods was the median background model approach, it gave superior results compared to the ColorCube and reduced computational time. It was also chosen as the best method in the visual evaluation.

Our findings show that there were much room for improvement of the currently used method. It struggled to correctly segment the recordings in the data-set due to the diverse and changing conditions in the outdoor environments. Alternative methods with similar complexity proved to perform better on the data-set that was created to represent Spiideo recordings. Methods using the history for each pixel (e.g. Median and MOG2) performed better than methods that instead used information from other parts of the same frame.

This comes however with the need to sample past frames.

The suggested alterations of the ColorCube (especially alteration 3,4 and 5) showed minor improvements in the zoomed area pixel classification scores, without extensive increase in complexity. The project gives Spiideo more knowledge of when the ColorCube fail, some possible ideas for further improvements, potential alternative methods and what to discard in future work. A better segmentation leads to increased user experience, as the analysis drawings becomes easier to understand.

The good result of the median method implies that the created background model was a good representation of the true background. This can be taken advantage of in other aspects of the application. For example, if one added the feature to move parts of the foreground, e.g. a player, to another position on the screen in a freeze frame. Information from the background model could be used to fill the resulting gap when moving a player. A drawback is its need to initialize a model before a good segmentation can be carried out. The approach used in this project sampled 50 frames every 10:th frame, which corresponded to 20 seconds in the recording. Even if this time probably can be reduced, the segmentation should preferably be good already during the first frame as a user is jumping back and forth in the recording time-line to watch different events. A more direct initialization starting with only the first frame reduces the calculation time and memory usage, but will also decrease the performance until a representative background model have been created.

All statistical methods showed a good overall performance but none showed reduced complexity. The best scores, in both classification and complexity, was given by MOG2. This method, and the other ones, could probably be optimized to perform even better regarding both classification scores and complexity, by testing different parameters. The adeptness of the methods enables them to cope with most recordings, but this is also a drawback as they require time to adapt and learn each pixel's distributions over time.

This project shows the shortcomings of the ColorCube. It was very sensitive to recordings where different parts of each frame had different characteristics, as it treated all parts of the image in the same way. A shadow or a patch of snow could heavily influence the segmentation result, as they often were classified as foreground. This often lead to "over-classification" where to many pixels were classified as foreground, which can be seen in the high values in *recall* and low values in *precision*. Consequently, the F_1-score was more or less inapplicable to use in a comparison. An attempt to compensate for this was the development of the zoomed area evaluation, but it showed a similar result, where the ColorCube struggled compared to the other methods.

One can imagine that football recordings often have a completely green background, but the reality is different. Different parts of the field have different nuances of green and there are many possible disturbances due to the outdoor conditions. The fact that the cameras are mounted in different directions also make the brightness vary in different parts of the frame, and thereby more difficult to correctly segment at the same time.

It can be seen that by applying *opening* and *closing*, or by increasing the background classification threshold, the scores improved for almost all recordings in the data-set. Other alterations improved the result for some recordings but worsened the result in other recordings leading to a lower total score than the currently implemented method. It was discovered that when applying ColorCubes on smaller patches, instead of adapt to that specific patch, it would rather miss the dominant colors of the entire frame. For example, if a shadow covered a big part of the playing field, but only a part of a smaller patch, it would rather be classified as foreground in the smaller patch. Alteration 3, where one ColorCube was applied to the entire frame and one was applied the smaller patch, was an attempt to master this problem. It showed minor improvements and achieved the same scores as applying a kernel or increasing the background classification threshold.

All proposed ColorCube improvements showed similar complexity as the currently used method. The aim was to come up with easy alterations which would improve the results without increasing the computational time, and that could be implemented by the company right away. It is probably easier to add improvements to the current method, as it already is integrated in the system, than implementing an unknown and more complicated segmentation method, and make it synchronize with other functions in the current application.

A useful outcome of the project for Spiideo, is the data-set created by the authors. Much effort and time were put into the work to create a diverse data-set with a corresponding ground truth-set. This could be used in further research for evaluation purposes. Another possible future utilization of the data-set lies in appliances of AI in computer vision. The data could be used for training and validation in segmentation, detection or classification.

The optical flow algorithm was very slow and computational heavy. The machine learning approach did not beat the other methods based on only color or motion detection. The classifier also turned out to be very complex. A drawback of the hierarchical approach was when a big patch was incorrectly classified as background, leading to all foreground pixel in the patch was classified as background and thereby wrongly classified. Due to the high complexity and bad performance, more work were put into investigating other methods.

5.2 Limitations

The Spiideo application, and thereby the ColorCube, is implemented in the programming language Swift. In this report the methods have been implemented and evaluated using Python. By looking at the source code of the Spiideo application, the ColorCube were thoroughly disassembled and replicated to greatest extent. However, some differences might occur due to programming language differences or definitions, as for example, how the area of interest on the playing field was chosen. The implemented ColorCube in Python should still give a very good hint on how Spiideo's method act and performs. To speed up the computational time in Python when iterating over all pixels in a high resolution frame, Numba was used [13]. Numba generates optimized machine code from the python code which significantly decreased the computational time. How this relate to code written in Swift was not investigated in this report. However, since all methods have been implemented and compared using the same programming language and hardware, the complexity evaluation should still give a good hint on the complexity ratio between the methods.

There are multiple ways to create a ground truth. The most basic approach, and probably most time consuming, would be to manually annotate each frame into background and foreground. Due to the allocated time, this approach was considered unreasonable and instead, the ground truth was created by the semi-automatic procedure further described in chapter 3.1. This approach came with some bias as some of the evaluated methods were used to create the ground truth. Naturally, this could imply that these methods perform better in the pixel classification scores than methods not used for creating the ground truth. It should also be noted that the manual selection of key-frames could lead to bias, as the most easy frames for the used methods were chosen to be used in the evaluation. To prevent the influence from this, all key-frames manually edited and improved to resemble manually annotated frames as much as possible.

Another limitation was the pixel classification scores. There are several ways a method can fail. A method may randomly classify incorrect pixels all over the frame, but still be considered as a good method from a viewer perspective. Another method may have no random "noise-pixels", but have missed to segment an entire player as foreground. The two methods may share pixel classification scores despite the difference in performance and viewer perception. An attempt to compensate for this was to perform the visual evaluation on the most relevant methods.

The foreground in the ground truth-images does not include any side-lines, goal-lines or parts of the goal. Some segmentation methods, e.g. the ColorCube, will often classify these parts as foreground. Due to the few true foreground pixels, goal/side-line pixels classified as foreground may become a majority of all foreground pixels. The pixel classification scores will be greatly influenced by this, even though the viewer perception are not harmed, as the sidelines, or the goal posts, could be accepted as foreground. A similar phenomena occur when a method correctly segments a majority of each frame, but fails on lesser relevant parts, as shown in Figure 5.1, and thereby receive very bad scores.

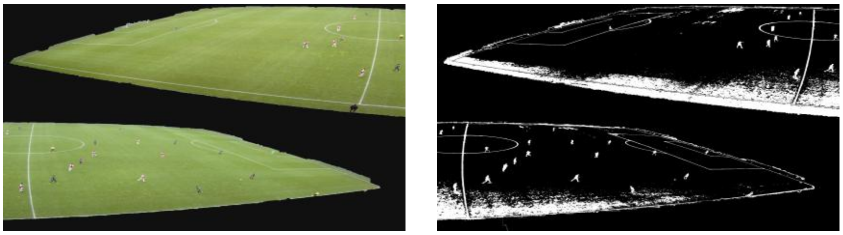


Figure 5.1: A segmentation can sometimes be sufficient where the game-play takes place, but have incorrectly classified pixels in other parts of the frame. It results in a bad score that may be somewhat misleading when having viewer perception in mind.

An attempt to compensate for this was made by performing the zoomed area evaluation. One can directly see the differences by comparing the full resolution scores with the zoomed scores. The ColorCube receives significantly better scores when zoomed, but when the zoomed area also contained snow, shadows or different nuances of green, the scores were still very bad. One can argue that the used scores did not manage to explain the good features of the ColorCube as it was effective at finding the true foreground pixels but often also included other parts.

An obvious drawback of the adaptive methods were the initialization time that prevents an optimal segmentation until the methods have adapted to the recording. As previously mentioned, the game-play is often viewed in short sequences, for example a corner or the build-up to a goal. This way of usage prevents adaptive method to perform optimally. It should be noted that all segmentation methods were allowed to fully adapt before being evaluated. A possible solution to this problem could be to have some kind of data-base with frames sampled from recordings, recorded at the same scene. These frames would represent past recordings, and could therefore be used to build a background model or set method parameters.

One should also have in mind the differences between the visual evaluation and the drawings in the Spiideo application. The drawing tools in the application uses a smoothing function to avoid sharp edges between the foreground and the background. This was not implemented in the visual evaluation. Smoothing the edges leads to a more forgiving environment were incorrectly classified pixels are harder to detect.

5.3 Further research

Further research could include to implement the evaluated methods in Spideo's system to see how they perform in the actual application. This would give a better understanding for the method performance and enable a more user like visual evaluation. The more complex methods could be implemented back-end instead of in the mobile device due to hard-ware restrictions. This leads to further research regarding how to transfer the resulting segmentation to the device for each frame. Should one send over a binary mask as it is? As the majority of each frame are background, represented by zeros in the binary mask, one could further investigate if the masks could compressed without crucial loss of information. Should the binary masks be represented in some other way that are more efficient? Perhaps could the transferred information be reduced by only sending over the coordinates for a given part of each segmented player to the device. A "standard player shape" could be used to avoid drawings on top of the player. An efficient information transfer allows the usage of a more complex segmentation method back-end, not depending on the mobile device hard-ware.

As previously mentioned, there were some limitations to the classification scores used for the evaluation. By better understanding how the user experience is affected by the characteristics of the segmentation one could develop evaluation scores more specific to the project. One idea was to use the distance for a false positive pixel to the closest true positive pixel as a penalty function, penalizing false positive pixels far way from true positive ones. Another idea was to weight the pixels differently depending on where in the frame they were, and thereby penalize incorrectly classified pixels in areas more likely to contain game-play. One could also argue that *recall* may be of more importance than the *precision* or vice verse, depending on what one are looking for. The two measures could thereafter be weighted together in different ways regarding what is of greater importance.

There are also numerous of other method variations, and completely different methods, that would be interesting to test in order to fully investigate all aspects to improve the current segmentation. This project gives a hint on which kind of method one could focus more on. To optimize method performance one could further investigate the result of varying method parameters. There were many different parameters, e.g. method learning-rates, number of pixel distributions or pixel classification thresholds, that could improve method performance regarding both scores and complexity. However, one should meanwhile have in mind that the data-set is not a true representation reality, and optimizing methods on the data-set could lead to over-fitting.

Another interesting field to investigate is the possibility to resize the frames before segmentation, an approach that leads to fewer pixels to classify and thereby reduced calculation time. The trade-off is the image quality and what artifacts that will appear as the frame is first down-sampled before segmentation, and the resulting binary mask is then up-sampled to fit the current frame in the recording.

Another parameter that could be changed, that the motion detection based segmentation could benefit from, is calculating movements between every second or third frame, instead of every frame. This allows more changes to take place and parts of the foreground would not be missed due to small movements. The specificity of this algorithm could also be increased to detect smaller movements, at the cost of being more complex. Another possibility that can improve the motion detection is to run the algorithm for all three color channels, instead of only on the gray-scale conversions of each frame. This would however triple the already large calculation time.

One could also evaluate pre-processing techniques of the image that could "fix" the frames of the recording instead of using a more complex segmentation method. To compensate for differences in different parts of the field, e.g. sunshine causing large shadows, or varying brightness due to camera positions, image processing techniques could be used to even out these differences before segmentation takes place. This is however a different field of image analysis, but solving this could also lead to enhanced user perception and experience, as it will become easier to watch and distinguish what happens in the recording when all parts have the same characteristics.

There are many ways to create a background model, and the median background model initialization could be further developed to better fit Spiideo needs. The usage of the application often includes jumping in the recording time-line between specific sequences, and the segmentation needs to be carried out already during the first few frames. More work could be put in how to initialize a good background model that enables satisfying segmentation directly. One advantage of the ColorCube is the one frame initialization. One solution could be to use the ColorCube during the initializing time for another better method that needs time to sample past frames to learn and adapt. One could also further investigate how to build a data-base with sampled frames from past recordings that could be used. Another solution is to create a background model for every scene that is updated every time a recording is started. One could also start start with the lastly used background model.

In this project, when looking at differences between two frames, only the euclidean distance between the colors were used as a difference measure. One could also consider and test the possibility to detect differences in texture, edges or gradients in the frame. However, this will increase method complexity.

Further research on improvements of the ColorCube could include testing different resolutions in 3D color space and thereby changing the number of color bins. Higher resolution enables the possibility to detect more specific background colors. By adding background colors, it would probably be required to change the pixel classification distance thresholds. A higher resolution would also lead to an increased complexity. One could also classify background pixels with other characteristics than using the three color channels. For example, one could add dimensions to the ColorCube that describes texture, edges or gradients that dominate the image, and thereby represent the background.

Another interesting approach would be to make the ColorCube more adaptive. By using the segmentation result of past frames could parts with abnormal values of foreground pixels be applied with a better and more specific method, or with multiple ColorCubes. These parts probably contain snow, shadows or game-play, and could therefore be better segmented. Another possibility is when dividing the field into smaller parts with individual ColorCubes, these could be given access to information from the other ColorCubes in order to use knowledge from the entire frame.

Another adaptive approach is to include a memory of past frames and/or past background color(s). The detected background color(s), used for pixel classification, are sampled every fifth second without any remembrance of the past, and are only based on the current frame. Slight changes in the frame can lead to that the background colors are significantly changed, and the segmentation result from frame to frame considerable differs, as shown in Figure 5.2. One could argue that this harms the perception more than a method with lower performance but consistent over time.

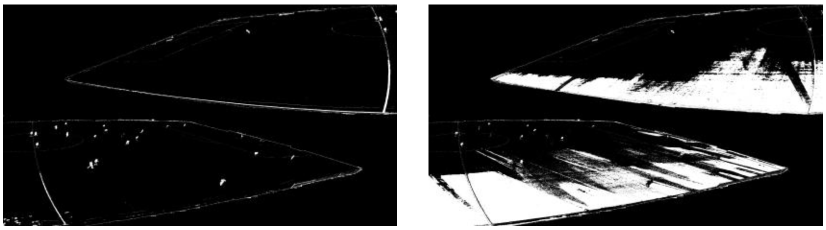


Figure 5.2: Due to changes in the recording, the background colors determined by the ColorCube can change significantly. The two binary masks in the figure are two consecutive frames with very different binary masks due to a "bad" ColorCube background color update.

One could also improve the results by combining methods that are good in different ways, either by alternating the methods due to video characteristics or by combining them. This work could be based on further research on when methods are successful, and what parts of the foreground they manage to correctly classify. If two completely different methods both have a accuracy of 50%, one could investigate whether they succeeded with the same 50% or if they together could complement each other and score 100%. This was the original thought behind the development of the machine learning classifier. There are endless possibilities to improve the classifier by adding more, better and less complex features which better describes the differences between foreground and background. Artificial Intelligence in computer vision is a popular topic, and gives more possibilities than only background foreground segmentation, as one also could consider object detection, tracking and classification.

5.4 Ethics

Better and automatic segmentation algorithms that are used to segment human beings can be used in surveillance cameras that automatically can detect and classify objects. A better segmentation enables better object recognition. One can argue that by improving computer vision techniques, one also takes a step towards a surveillance society with cameras that can recognize citizens and is a threat to integrity and privacy.

Another ethical aspect of video cameras are the audience that may end up in the recordings without their knowledge and consent. The recordings are privately used by the clubs and can not be accessed by the public. But one interesting question is what happens if e.g. a crime is committed and was caught on tape. Who has the rights to access what recordings, and when?

The General Data Protection Regulation (GDPR) will take action in May 2018. GDPR regulates how companies, organizations and similar stores and collect data. Spiideo's system is an analysis tool, where statistics and data about the players can be presented in the application on top of the recordings. How this data is stored and accessed is of ethical importance. It could be personal information about the players, or medical information such as heart rate which would be affected by the GDPR if stored. [5]

5.5 Contributions

The report was written in equal parts by the authors. The creation of the data-set and the statistical segmentation methods were also implemented and performed in equal parts by the authors.

The median background model method, the motion detection method and the machine learning classifier was implemented by Fredrik Hammar. The ColorCube, the proposed alterations and the evaluation-scripts was implemented by Johan Flinke.

6
Conclusion

The aim was to find a way to improve the current background foreground segmentation in Spiideo's sports analysis mobile application. A data-set with 13 recordings and a corresponding ground truth was created. Relevant segmentation methods and possible improvements to the current method were applied to the recordings in the data set. The outcome was evaluated and compared in terms of pixel classification scores, complexity measurements and by a visual evaluation.

Segmentation methods that significantly improved the segmentation results were found. The *Median Background Model Frame Difference* method showed a robust performance on all recordings in the data-set with a reduced computational time compared to the current method. It was also chosen as the best method in the visual evaluation. The *Improved Mixture of Gaussians* (MOG2) method also showed a good performance in both classification scores and complexity. These methods take each pixels history in consideration and therefore have a need to sample past frames.

Suggested alterations to the current method, the *ColorCube*, have been developed and evaluated. The resulting scores showed some minor improvements but did not manage to compete with the two previously mentioned methods (Median and MOG2). The ColorCube struggles with the outdoor weather conditions as it takes the entire frame into consideration when extracting background colors. As the background often had different colors in different parts of each frame, it was hard to cover all variations. The results show that the current method and the suggested alterations, struggled to correctly segment the recordings in the data-set.

A method based on motion detection was developed without great success due to its high complexity. A novel machine learning classifier, taking both color and motion into account, was implemented and evaluated. It did not beat the methods based on only color or motion.

Future work could include to implement the best and most relevant methods in Spiideo's system for a more reality-based evaluation. Better and more purposeful classification scores and evaluations could be used to better compare the methods relevance. More research can be put in how the median background could be initiated during the first few frames for better performance. To further improve the segmentation one could also find and evaluate other methods, or try to combine methods that can complement each other.

Bibliography

Articles

[1] H. Ardö. (2009). Multi-target Tracking Using on-line Viterbi Optimisation and Stochastic Modelling. Centre for Mathematical Sciences, Lund University.

[2] M. M. Azab, H. A. Shedeed and A. S. Hussein, "A new technique for background modeling and subtraction for motion detection in real-time videos," *2010 IEEE International Conference on Image Processing,* Hong Kong, 2010, pp. 3453-3456. doi: 10.1109/ICIP.2010.5653748

[3] J. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm", Intel Corporation, Microprocessor Research Labs, 2000

[4] V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 12, pp. 2481-2495, Dec. 1 2017. doi: 10.1109/TPAMI.2016.2644615

[5] European Parliament and Council Directive on Citizens' on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC. *General Data Protection Regulation*. OJ L 119, 4.5.2016. 1–88. 2016.

[6] G. Farnebäck, "Two-Frame Motion Estimation Based on Polynomial Expansion," Computer Vision Laboratory, Linköping University, 2003.

[7] A. B. Godbehere, A. Matsukawa and K. Goldberg, "Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation," *2012 American Control Conference (ACC),* Montreal, QC, 2012, pp. 4305-4312. doi: 10.1109/ACC.2012.6315174

[8] E. Hayman and J. O. Eklundh, "Statistical background subtraction for a mobile observer," *Proceedings Ninth IEEE International Conference on Computer Vision,* Nice, France, 2003, pp. 67-74 vol.1. doi: 10.1109/ICCV.2003.1238315

[9] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," *2017 IEEE International Conference on Computer Vision (ICCV),* Venice, 2017, pp. 2980-2988. doi: 10.1109/ICCV.2017.322

[10] V. Jain, B. B. Kimia and J. L. Mundy, "Background Modeling Based on Subpixel Edges," *2007 IEEE International Conference on Image Processing,* San Antonio, TX, 2007, pp. VI - 321-VI - 324. doi: 10.1109/ICIP.2007.4379586

[11] Xu Jian, Ding Xiao-qing, Wang Sheng-jin and Wu You-shou, "Background subtraction based on a combination of texture, color and intensity," *2008 9th International Conference on Signal Processing,* Beijing, 2008, pp. 1400-1405. doi: 10.1109/ICOSP.2008.4697394

[12] P. Kaewtrakulpong, R. Bowden, "An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection," *Proceedings of 2nd European Workshop on Advanced Video-Based Surveillance Systems,* London, U.K, 2001, doi: 10.1007/978-1-4615-0913-4_11.

[13] S. K. Lam, A. Pitrou, S. Seibert, "Numba: a LLVM-based Python JIT compiler," *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure*. HPC, 2015, pp. 1-6, doi: 10.1145/2833157.2833162.

[14] A. Lepisk, "The use of Optic Flow within Background Subtraction," Master Thesis, KTH, Stockholm. 2005.

[15] J. Makhoul, F. Kubala, R. Schwartz, R. Weischedel, "Performance Measures For Information Extraction," *In Proceedings of DARPA Broadcast News Workshop*, 1999, p. 249-252.

[16] N.J.B. McFarlane and C.P. Schofield, "Segmentation and tracking of piglets in images," *Machine Vision and Applications 8*, 1995, pp. 187-193, doi: 10.1007/BF01215814

[17] G. Nanfack, A. Elhassouny, R. O. H. Thami, "Squeeze-SegNet: A new fast Deep Convolutional Neural Network for Semantic Segmentation," *10th International Conference on Machine Vision,* (ICMV), 2017.

[18] N. Singla. Motion Detection Based on Frame Difference Method. *International Journal of Information & Computation Technology,* Volume 4, Number 15, , 2014, pp.1559-1565.

[19] A. Sobral, A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding,* Volume 122, 2014, Pages 4-21, ISSN 1077-3142, https://doi.org/10.1016/

[20] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, Fort Collins, CO, 1999, pp. 252 Vol. 2. doi: 10.1109/CVPR.1999.784637

[21] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," *Proceedings of the 17th International Conference on Pattern Recognition, 2004.* ICPR 2004, pp. 28-31 Vol.2. doi: 10.1109/ICPR.2004.1333992

[22] Z. Zivkovic and F. Heijden. "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters,* Volume 27, Issue 7, 2006, Pages 773-780, ISSN 0167-8655, https://doi.org/10.1016/

Web-pages

[23] CouNT GitHub Documentation and source files. `https://sagi-z.github.io/BackgroundSubtractorCNT/`, retrieved [2018-05-17]

[24] CouNT GitHub Documentation and source files. `https://github.com/sagi-z/BackgroundSubtractorCNT`, retrieved [2018-05-17]

[25] CouNT Segmentation Method OpenCV Class Reference. https://docs.opencv.org/3.3.0/de/dca/classcv_1_1bgsegm_1_1BackgroundSubtractorCNT.html, retrieved [2018-05-17]

[26] Farneback Optical Flow Method Matlab Documentation. https://www.mathworks.com/help/vision/ref/opticalflowfarneback-class.html, retrieved [2018-05-17]

[27] FFMPEG Webpage, Software Documentation. https://www.ffmpeg.org/documentation.html, retrieved [2018-05-17]

[28] Github ColorCube Documentation https://github.com/pixelogik/ColorCube , retrieved [2018-05-17]

[29] Godbehere, Matsukawa, Goldberg (GMG) Segmentation Method OpenCV Class Reference. https://docs.opencv.org/3.3.0/d1/d5c/classcv_1_1bgsegm_1_1BackgroundSubtractorGMG.html, retrieved [2018-05-17]

[30] Improved Mixture of Gaussian Segmentation Method OpenCV Class Reference. https://docs.opencv.org/3.3.0/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html, retrieved [2018-05-17]

[31] KNN Mixture of Gaussian Segmentation Method OpenCV Class Reference. https://docs.opencv.org/3.3.0/db/d88/classcv_1_1BackgroundSubtractorKNN.html, retrieved [2018-05-17]

[32] Morphological Transformations in OpenCV. https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html, retrieved [2018-05-17]

[33] Mixture of Gaussian Segmentation OpenCV Class Reference. https://docs.opencv.org/3.3.0/d6/da7/classcv_1_1bgsegm_1_1BackgroundSubtractorMOG.html, retrieved [2018-05-17]

[34] OpenCV, Official Webpage and Documentation. https://opencv.org/about.html, retrieved [2018-05-17]

[35] PSUTIL Documentation. https://psutil.readthedocs.io/en/latest/, retrieved [2018-05-17]

[36] Python Webpage, Software Documentation. `https://www.python.org/`, retrieved [2018-05-17]

[37] Scikit-Learn Python Documentation. `http://scikit-learn.org/stable/index.html`, retrieved [2018-05-17]

[38] Scikit-Learn Python Documentation for different classifiers. `http://scikit-learn.org/stable/supervised_learning.html`, retrieved [2018-05-17]

Appendix A

Appendix A contains examples of a frame from each recording, and the resulting binary masks produced by all evaluated segmentation methods. Examples from recordings Gefle and Kalmar are shown in chapter 4.

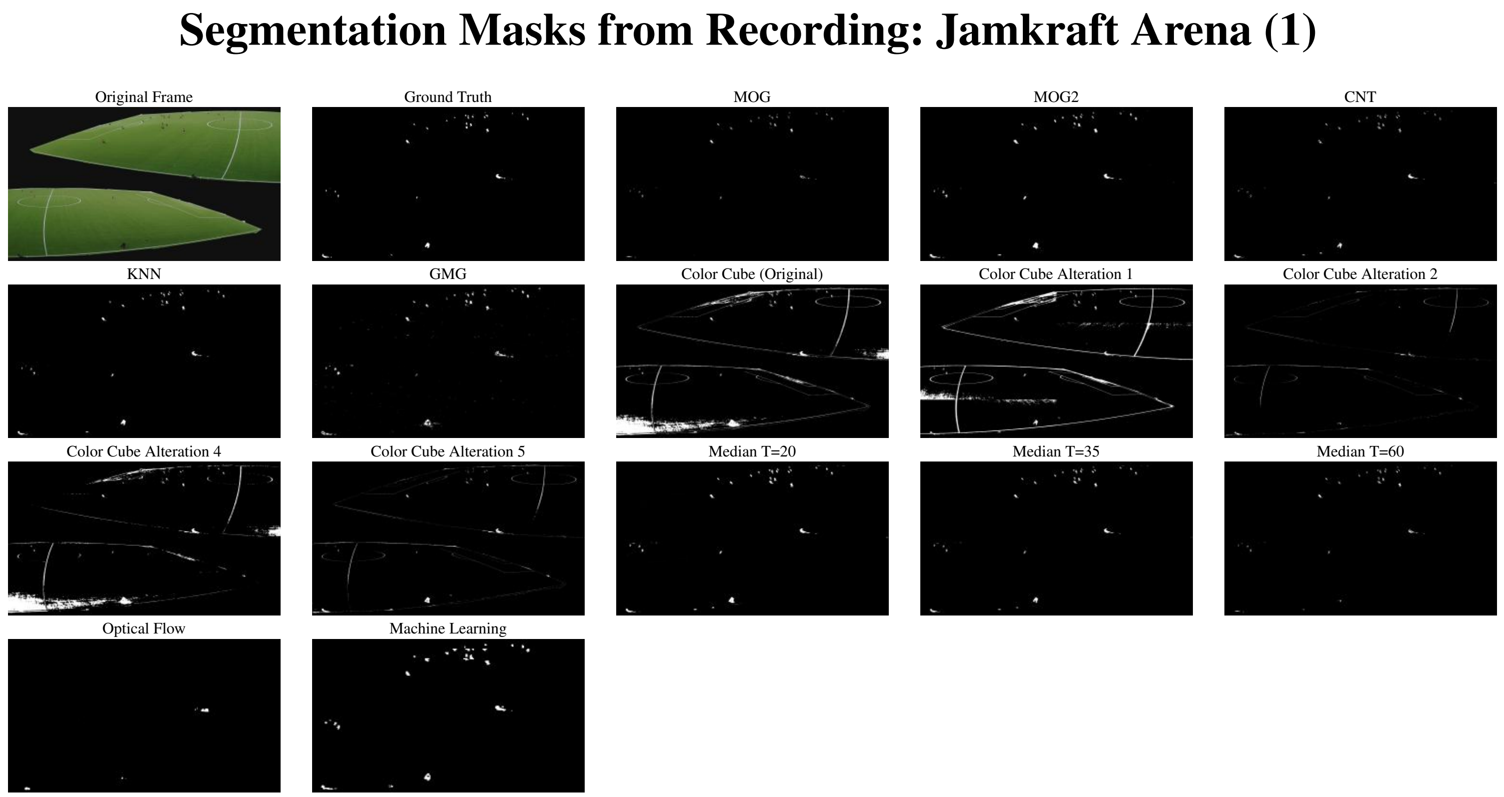


Figure 6.1: A binary mask from each method from recording Jamkraft Arena (1). The most upper left image is the frame which is segmented.

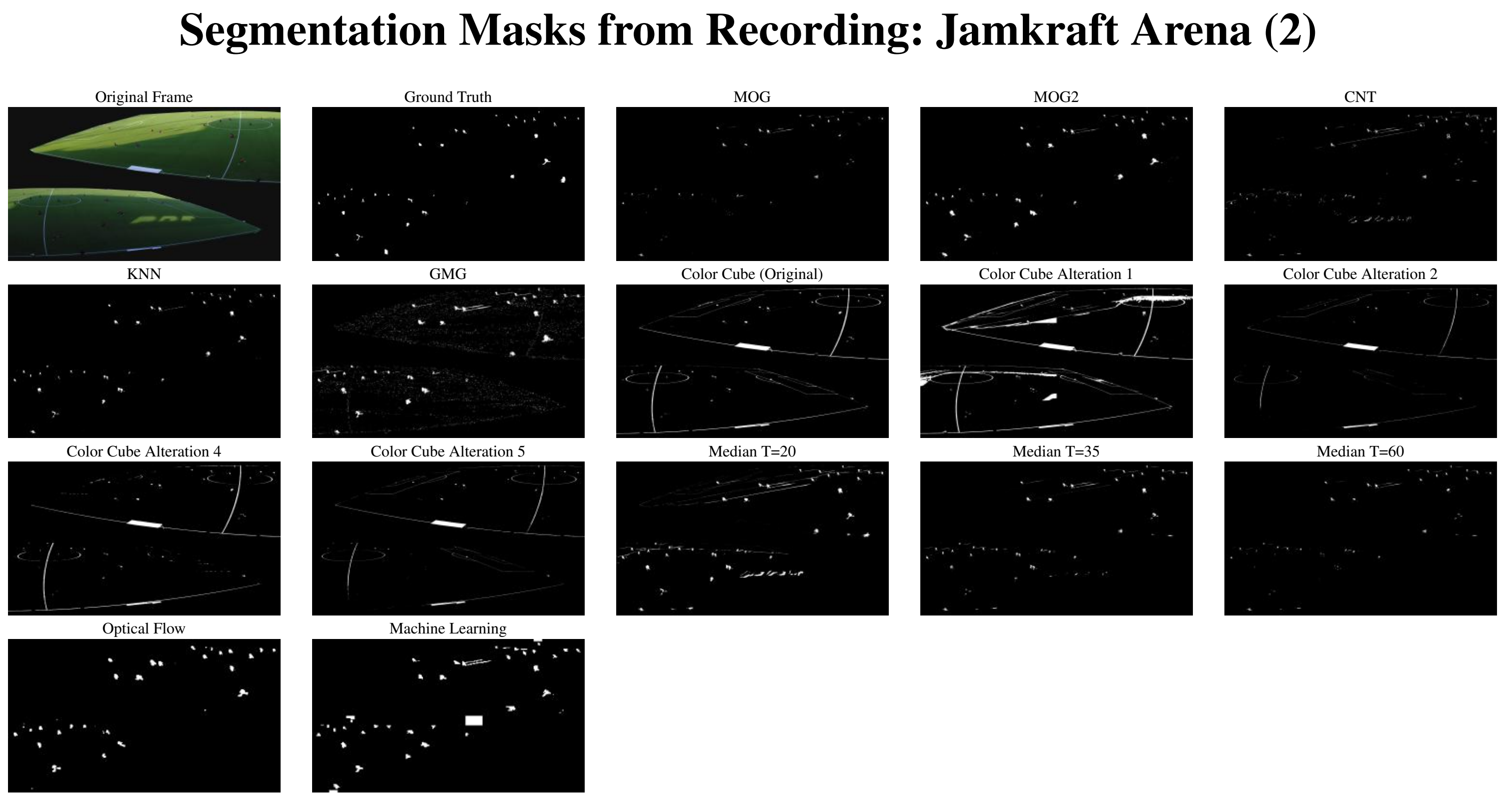


Figure 6.2: A binary mask from each method from recording Jamkraft Arena (2). The most upper left image is the frame which is segmented.

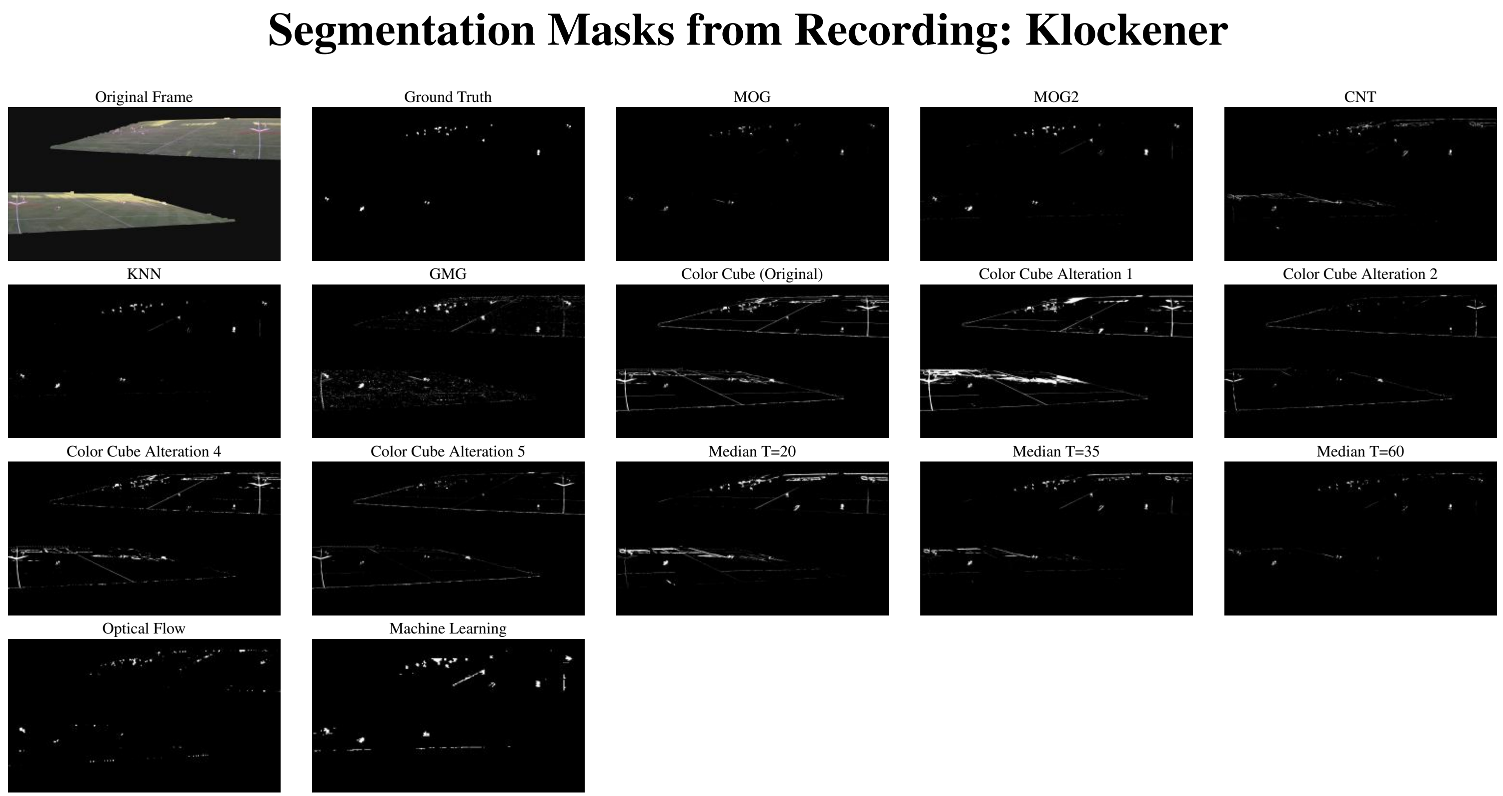


Figure 6.3: A binary mask from each method from recording Klockener. The most upper left image is the frame which is segmented.

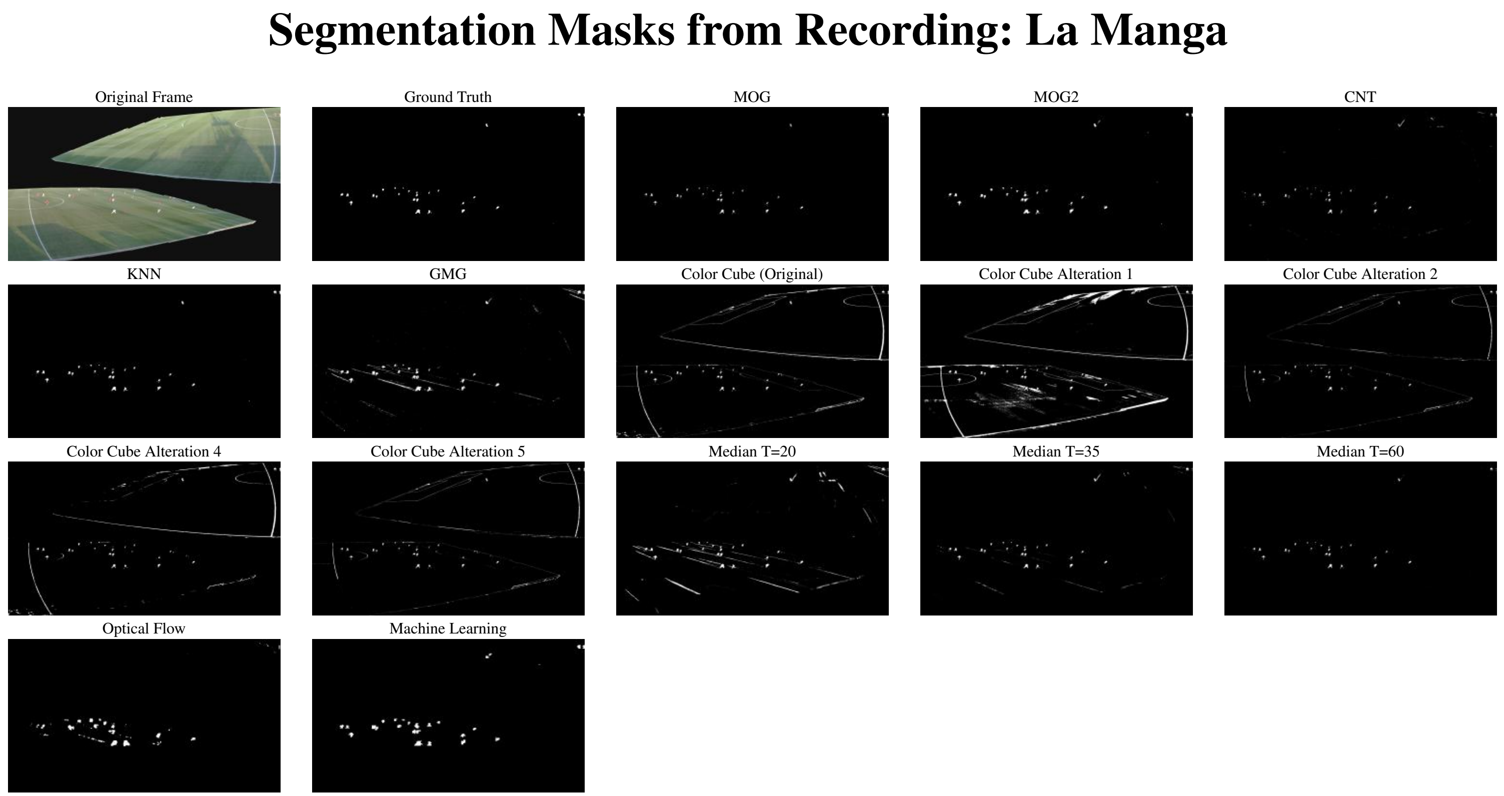


Figure 6.4: A binary mask from each method from recording La Manga. The most upper left image is the frame which is segmented.

Segmentation Masks from Recording: La Manga (North Side)

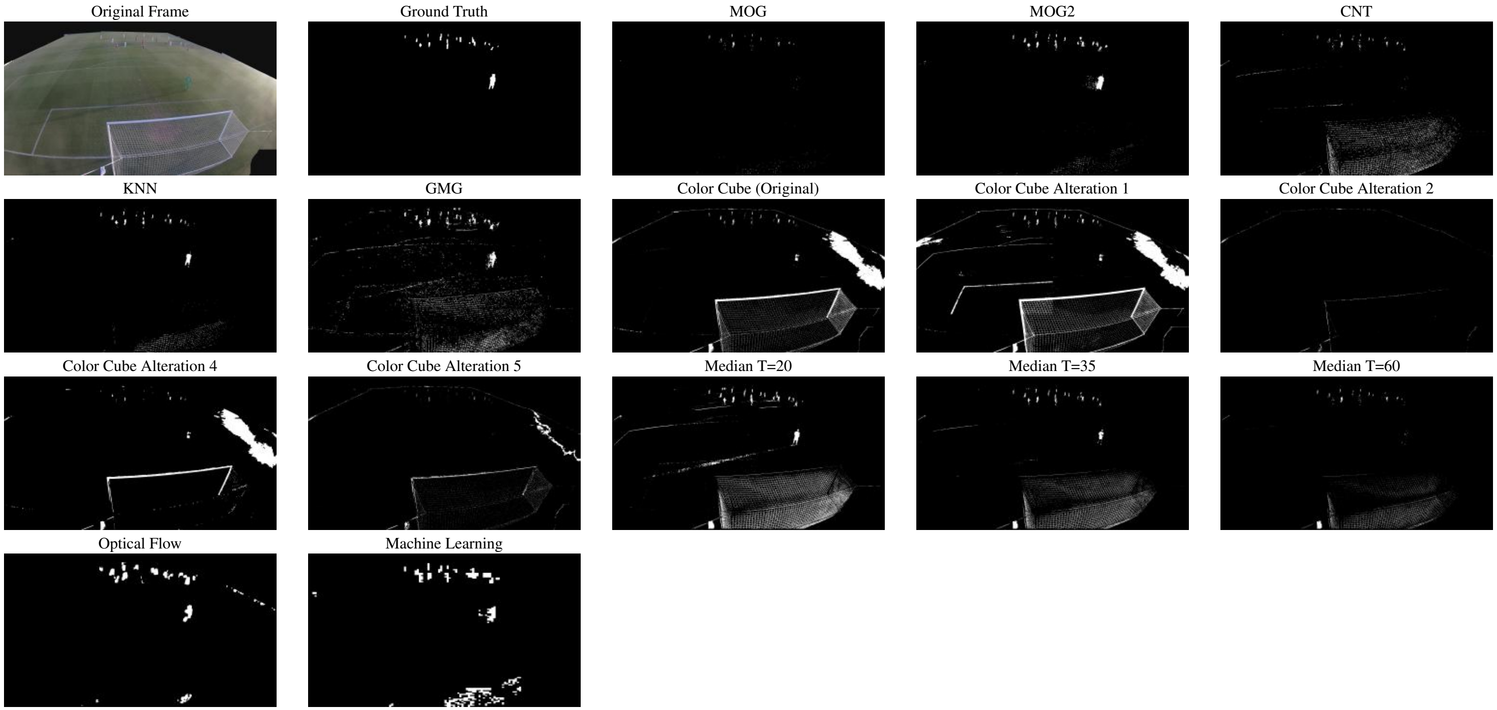


Figure 6.5: A binary mask from each method from recording La Manga (North Side). The most upper left image is the frame which is segmented.

Segmentation Masks from Recording: La Manga (South Side)

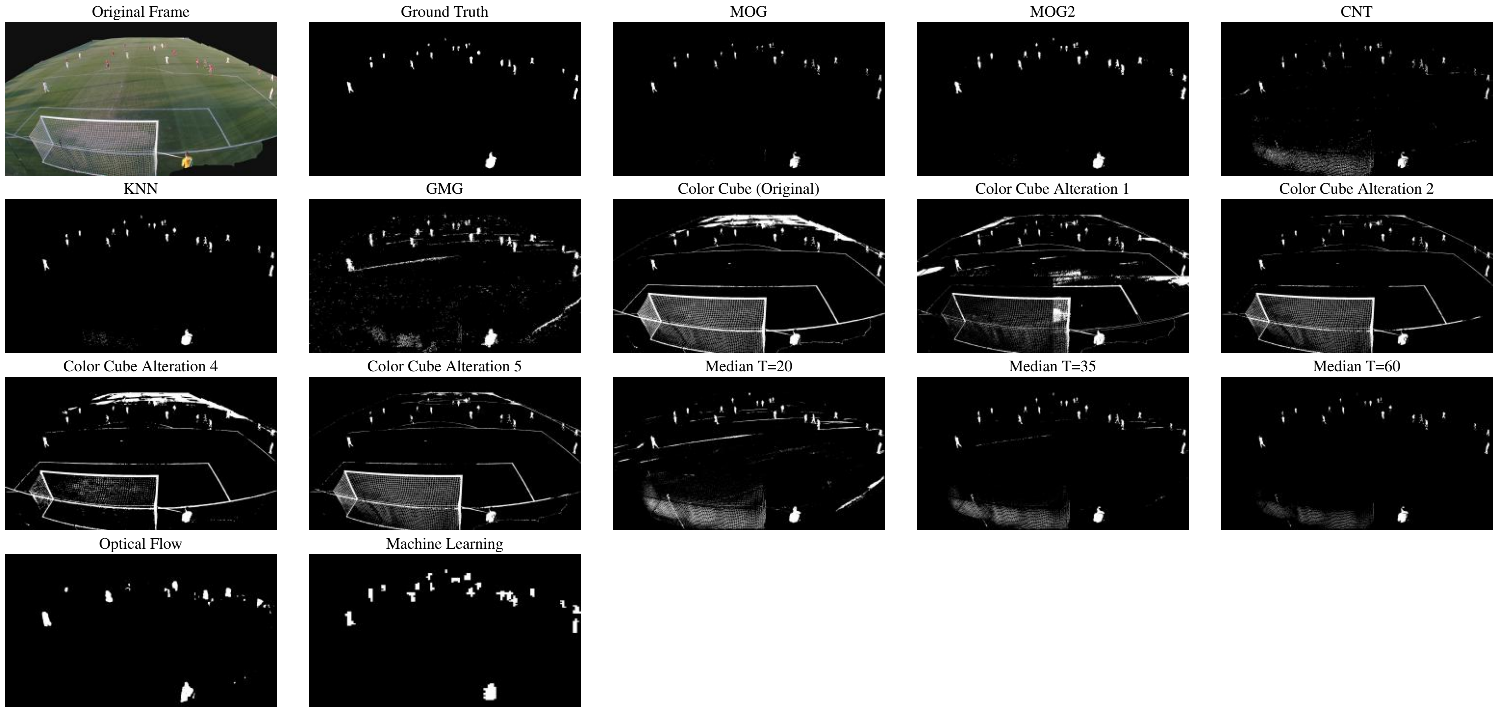


Figure 6.6: A binary mask from each method from recording La Manga (South Side). The most upper left image is the frame which is segmented.

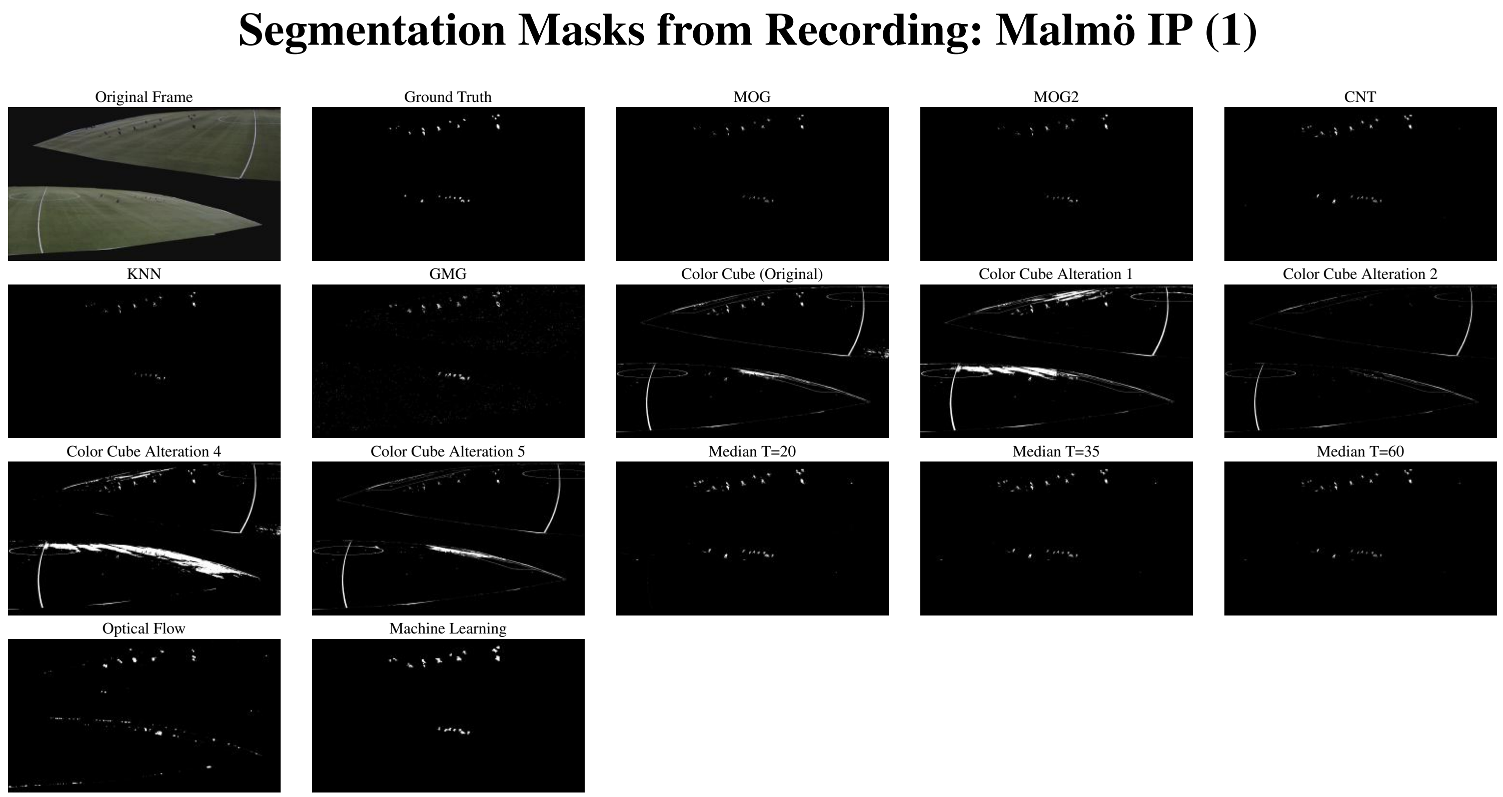


Figure 6.7: A binary mask from each method from recording Malmö IP (1). The most upper left image is the frame which is segmented.

Segmentation Masks from Recording: Malmö IP (2)

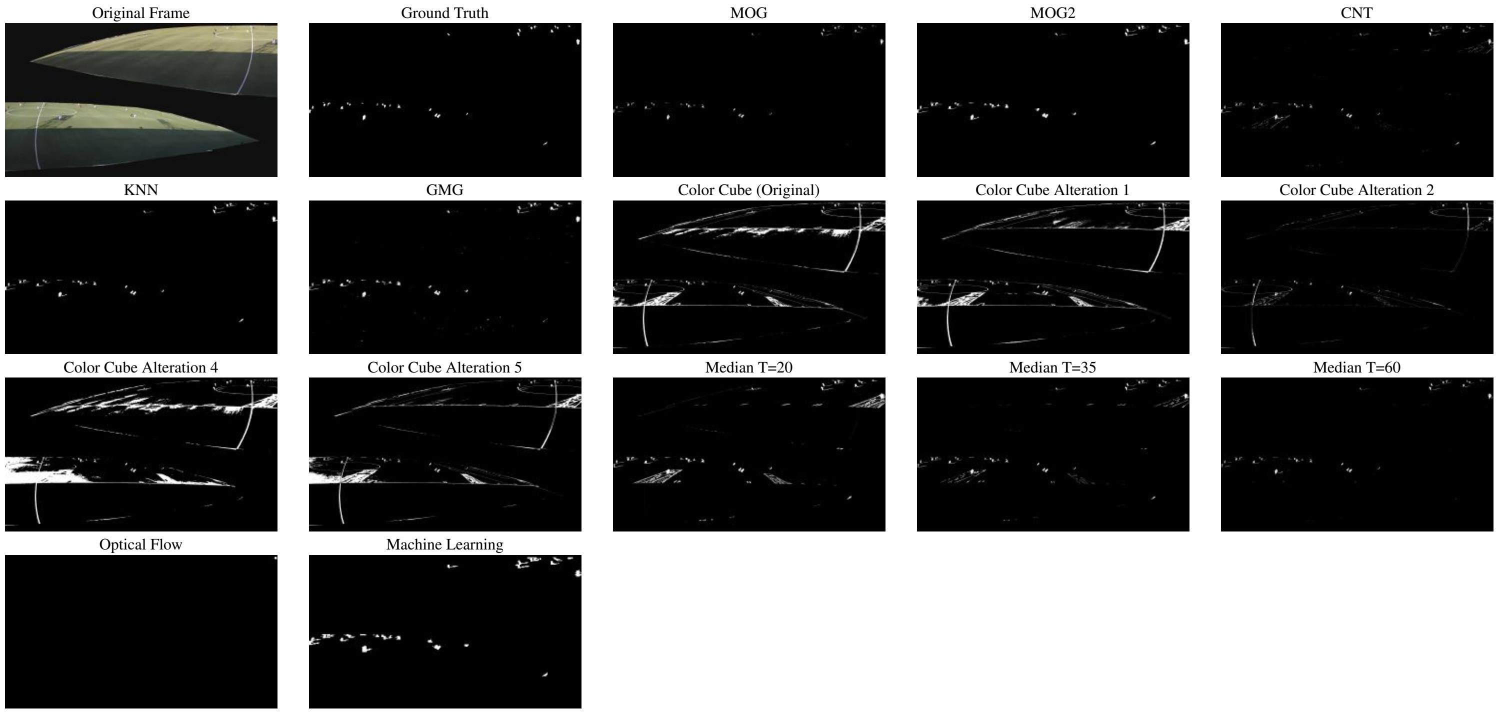


Figure 6.8: A binary mask from each method from recording Malmö IP (2). The most upper left image is the frame which is segmented.

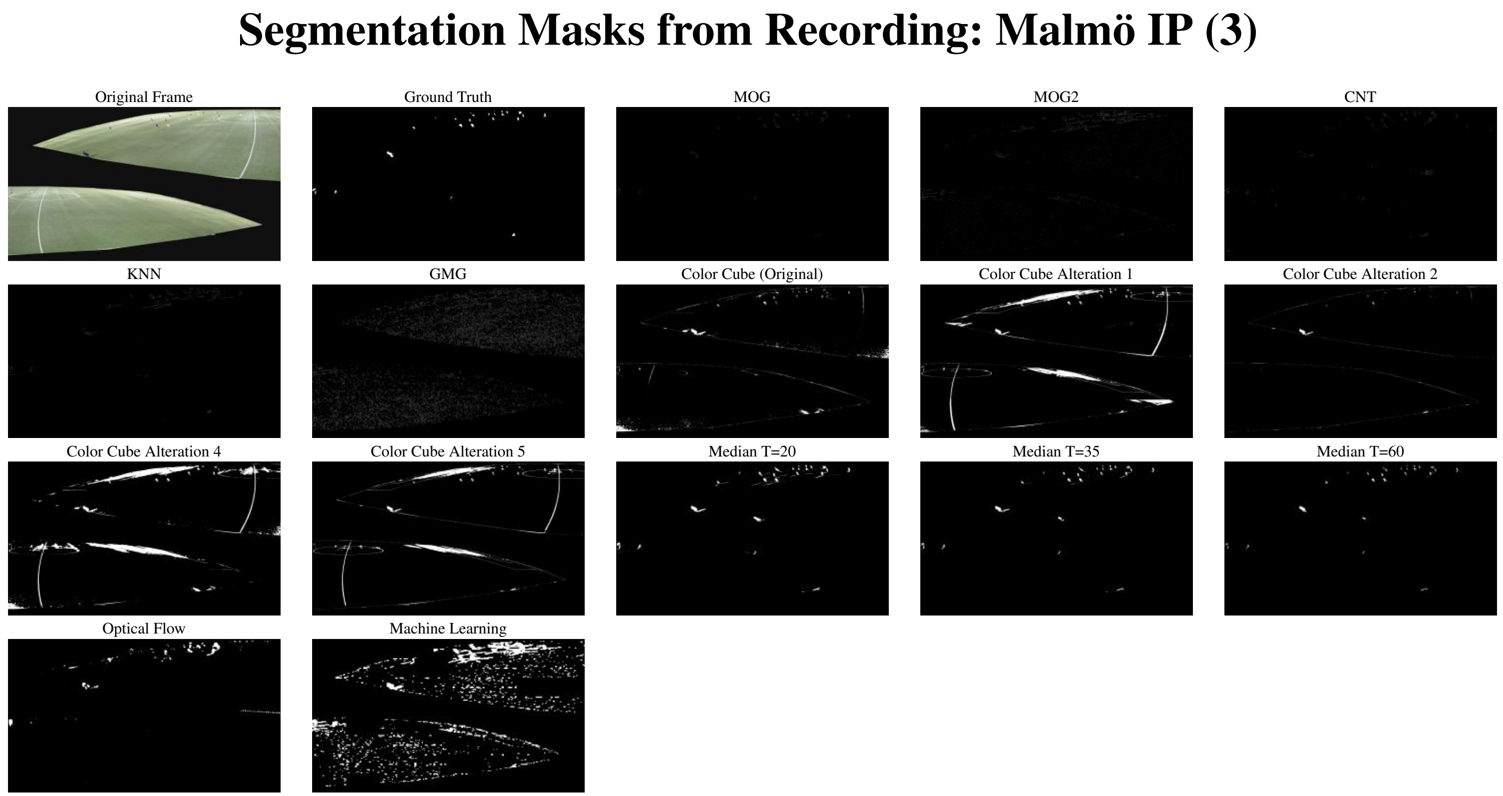


Figure 6.9: A binary mask from each method from recording Malmö IP (3). The most upper left image is the frame which is segmented.

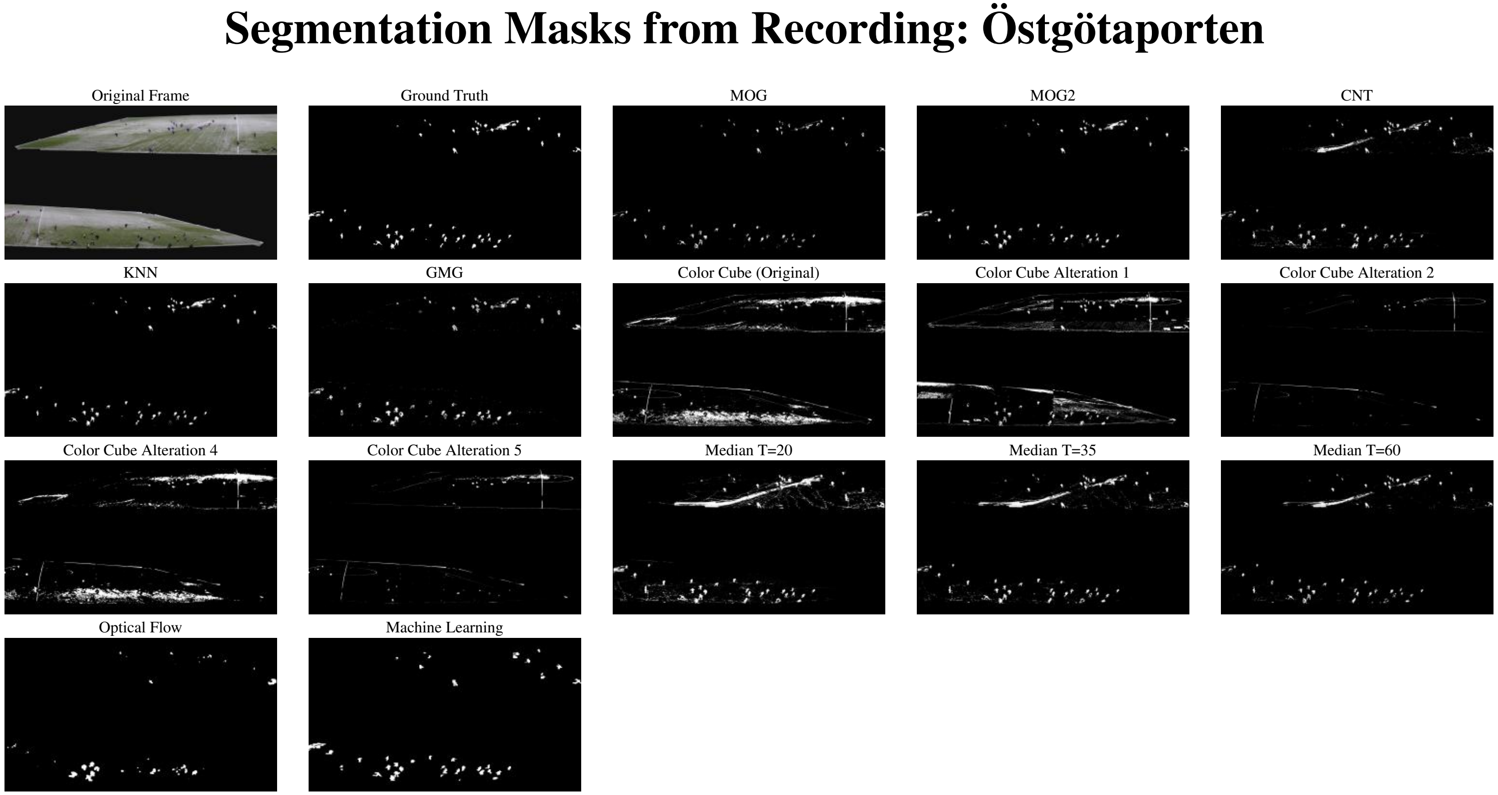


Figure 6.10: A binary mask from each method from recording Östgötaporten. The most upper left image is the frame which is segmented.

Segmentation Masks from Recording: Ice Hockey

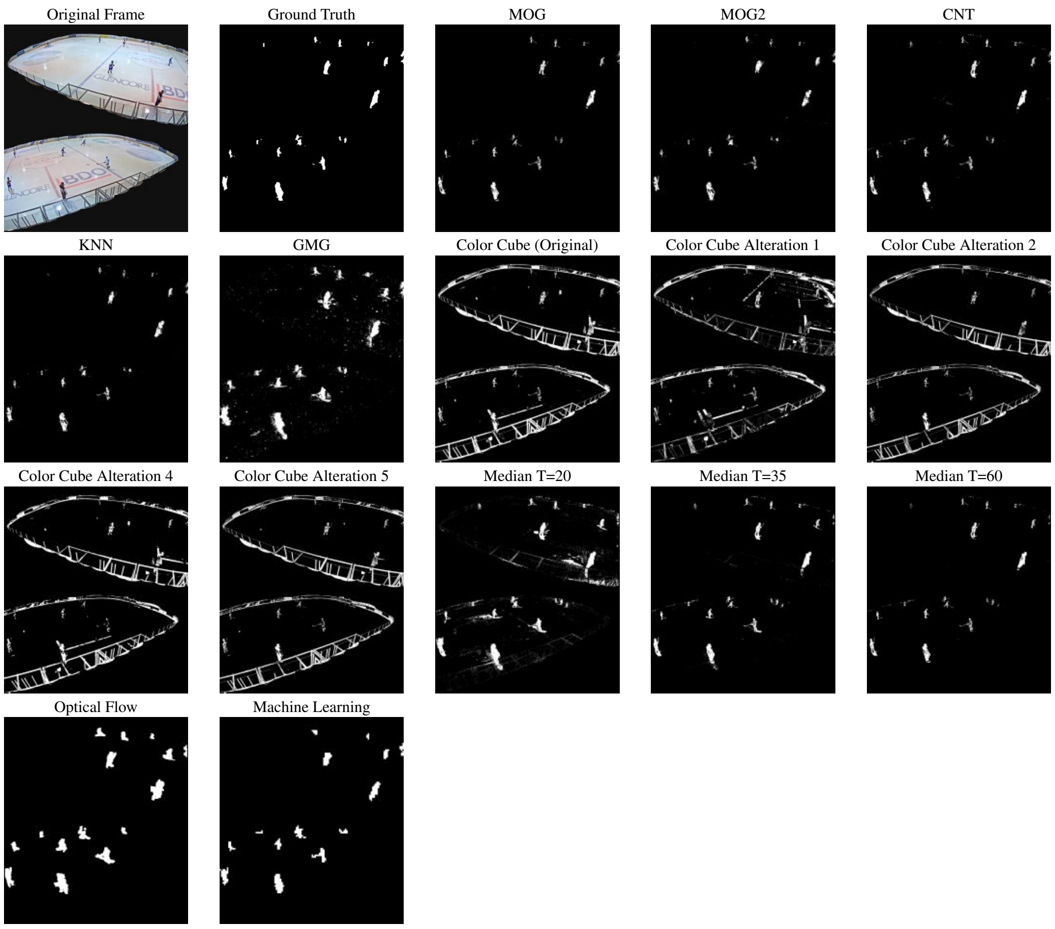
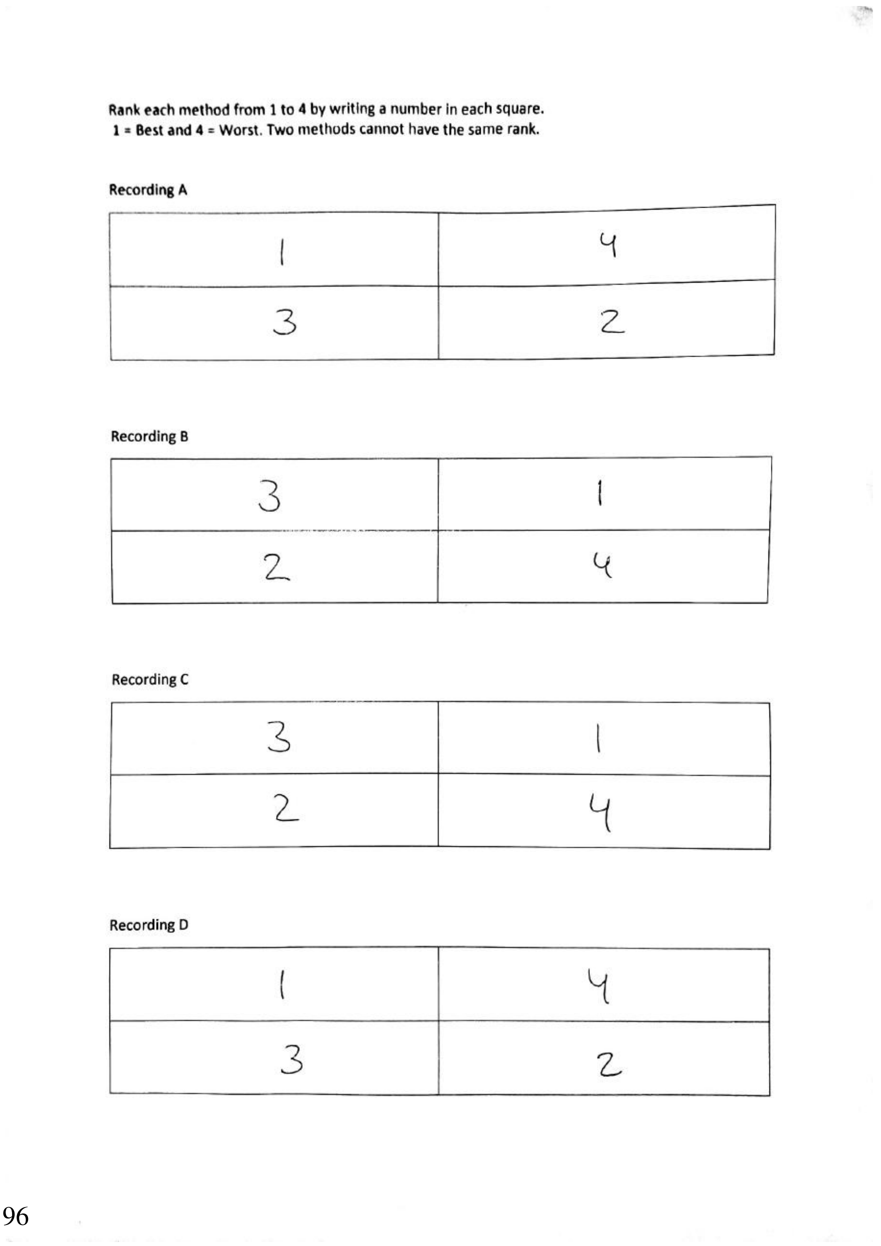


Figure 6.11: A binary mask from each method from recording Ice Hockey. The most upper left image is the frame which is segmented.

Appendix B

An example of the score sheet used in the visual evaluation are shown on the next page.



Rank each method from 1 to 4 by writing a number in each square.
1 = Best and 4 = Worst. Two methods cannot have the same rank.

Recording A

1	4
3	2

Recording B

3	1
2	4

Recording C

3	1
2	4

Recording D

1	4
3	2