Mean-shift Object Tracking
Lab Report for IMS

B.J.Buter [9910522], T. Kostelijk [0418889]
{bjbuter,mailtjerk}@gmail.com

1 Introduction

Videos are ubiquitous; in the United Kingdom it has been claimed that there is one observation camera per 14 citizens [1]. YouTube.com, the largest video website in the USA with a marketshare of 40% of all watched online videos serves more than 10 Billion movies a month[2]. Furthermore, the amount of videos uploaded to the site through mobile phones is growing exponentially [3]. From such facts, it is clear that the increasing signifance of video content, ranging from the practical to recreational, is undeniable. This plethora of video content makes automated video analysis desirable. Ideally, we would like to analyse videos to be able to extract their content. This includes making them accessible for search or to alert the viewer of important developments. One of these analyses is tracking an object through multiple frames of video.

In this paper, tracking is understood as follows: given an object description, identify an object at a location in a video frame that is consistent with locations found in previous video frames. An example of an inconsistent location would be that of an object, previously on the left side, suddenly appearing on the right side of the videoframe.

Generally, tracking can be done in two distinct manners; model based and appearance based. The former approach begins by building a substantial descriptive model of the object to be tracked. One can think of a wireframe model of the object together with texture or color information. In every frame this model is searched for, and the location that resembles the model best is the location of the relevant object. A problem with this approach is that a substantial descriptive model has to be constructed for every conceivable object to be tracked. Because of this tedious task, this approach can only be applied to small specialized domains.

This paper, however, will analyse an instance of the latter, appearance based, approach. This approach does not build a full model for tracking, but rather, it looks at the appearance of an object and searches for regions that appear the same. This is thus not a substantial descriptive model; rather it is an appearance based model. The underlying assumption behind this way of tracking is that small local features of the tracked object appear the same over multiple frames. The appearance based tracker discussed in this paper is the mean-shift tracker.

Color is the feature that will be tracked, thus, different color models will be presented and applied. Following this, subsequent tracking results on different videos for both trackers will be discussed. Specifically, we want to explain through experiments how the mean-shift tracker behaves under varying video conditions (such as occlusion and objects of varying scales), and what the

[1] Rt Hon David Davis MP, June 12 2008, speech:"It is incumbent upon me to take a stand" `http://www.conservatives.com/News/Speeches/2008/06/David_Davis_It_is_incumbent_upon_me_to_take_a_stand.aspx`

[2] comScore Video Metrix, August 2009 `http://comscore.com/Press_Events/Press_Releases/2009/9/Google_Sites_Surpasses_10_Billion_Video_Views_in_August`

[3] The Official YouTube Blog, June 25 2009, blogpost:"Mobile Uploads to YouTube Increase Exponentially" `http://youtube-global.blogspot.com/2009/06/mobile-uploads-to-youtube-increase_5122.html`

influence of the used color model is. As said at the beginning of this section, videos are pervasive, and their analysis should therefore ideally be done in realtime. The speed has, therefore, been an integral part in the development of the tracker, and will be discussed.

This paper is organised as follows: after the introduction of the topic in section 1, the theory behind the used trackers and color models will be discussed in section 2. Section 3 will be devoted to the implementation of the presented theories. The experimental setup, a description of the data, subsequent results and their discussion will be presented in section 4. Section 5 will discuss future work and finally, section 6 will present the main conclusions on color based mean-shift tracking.

2 Theory

This section will explain the main theories used in the mean-shift tracker. The color of an object is dependent on what we, for now, will call the intrinsic color of the object, as well as other factors such as lighting. Because the tracker will track an object on its color, it is important to choose a color model that is invariant to all factors except the instrinsic color of the object. Therefore, in the following section, color models in general as well as some specific color models and their invariant properties will be introduced. Following this, histograms, which are probabilistic descriptions, or probability density functions (PDF), of color in image regions, will be discussed. We can compare PDFs using certain distance measures, which enables us to find color similarities between image regions. These concepts are integral to understanding the mean-shift tracker, which will be discussed afterwards. For an interesting comparison and to explain other methods researched in this project, the simpler brute force tracker and histogram backprojection will also be discussed briefly.

2.1 Color models

To clarify the content of this section it is important to take note of some characteristics of color. First, there is a difference between what we have called the intrinsic color of an object and the apparent color of that object. As an example, a white car looks white in normal daylight, whereas, when lit at night by the orange light of a sodium lamp, the white car will look orange. In this example, the intrinsic color of the car is white, while the apparent color is either white or orange depending on the lightsource.

Another noteworthy detail is that an object rarely has uniform color, therefore, an object patch is defined as a sufficiently small region of an object such that it has uniform color properties.

When a camera is aimed at an object patch, the color it detects is dependent on the following factors:

- the spectral power distribution (SPD) of the light source
- the reflectance factor of a patch of the object
- the color matching functions of the camera sensors
- the geometric arrangement of the lightsource(s), the object and the observer
- the geometric properties of the object itself

The SPD describes for the whole visible light spectrum with what intensity every wavelength is being radiated. The reflectance factor describes how much each part of the spectrum gets reflected and how much gets absorbed by the object patch; this is what has been referred to as the intrinsic color. The color matching functions indicate the magnitude of the response of a camera sensor for

every wavelength of incoming radiation. Last, the geometric arrangements and properties of the object also influence the color [1].

We have noted a difference between the intrinsic and the apparent color of an object. All factors described above influence the appearance of an object patch and therefore the apparent color. However, the reflectance factor is the only factor to stay constant, because this is the only factor intrinsic to the object. Therefore, we want to track an object based on its reflectance factor. Hence, we want to have a color model which is ideally invariant to all factors that influence the appearance of the object except for the reflectance factor. In the following sections, various color models will be discussed in terms of their general properties and their invariance, in particular.

2.1.1 RGB and XYZ

All percievable colors can, according to trichromacy theory [1], be modeled as the response of three sensors, namely: red, green and blue. The RGB color model resembles the human perception of colors in the sense that human color vision can also be said to be the result of a tri-stimulus. RGB is an additive color model, which describes each color as a mixture of the three mentioned primary colors.

Some properties of the RGB color model are that it is a device dependent color model, which means that a given RGB color will be detected or reproduced differently on different devices. The three primary colors are not defined and therefore every device might use three different primary colors to detect or reproduce colors; this causes the device dependent property of this color model. Another property of RGB is that it is invariant to neither illumination intensity, highlights nor geometry [2]. Since RGB is not invariant to any of the factors which influence the apparent color of an object patch, this color model seems ill-equipped for tracking. However, it is a standard model for representing colors and therefore worth mentioning.

The XYZ color model, just like RGB is also a tri-stimulus model, however it is defined with 3 fictive primary colors X, Y and Z. The three colors are a linear combination of the RGB colors. However, the X,Y and Z colors are well defined, therefore, the XYZ colorspace is device independent. All invariant properties are the same for both RGB and XYZ.

2.1.2 Normalised RGB and Normalised XYZ

In the RGB color model, a change in lighting intensity would make the responses of all three color channels increase proportionally to the intensity change. Through normalisation, the intensity is set to a fixed level of 1. This makes normalised RGB (rgb) invariant, to any factor which solely influences the intensity of the perceived object patch. Therefore, rgb is invariant to illumination intensity and geometric arrangements. rgb however is still not invariant to illumination color or highlights [2].

The following equations describe the conversion from RGB to rgb, for each color channel respectively. Let I denote the intensity:

$$I = R + G + B \tag{1}$$

$$r = \frac{R}{I},\ g = \frac{G}{I},\ b = \frac{B}{I} \tag{2}$$

Two things are worth concidering here, first r, g, b are undefined for black when $I = 0$, this can be remedied by recognising that white black and grey are the same except for intensity. Thus black in rgb space should be the same as white or grey which is $r = g = b$. Last the rgb colorspace is 3-dimensional; however, since all 3 channels are normalised, it must be that $r+g+b = 1$. Therefore, rgb is overdetermined as a 3-dimensional space and can be described in 2 dimensions r, g, where b can be deduced through $b = 1 - (r + g)$.

Similar to RGB, the device independent colorspace XYZ can be converted to a normalised XYZ (xyz).

2.1.3 HSV

The hue, saturation, value model (HSV) does not represent color as a tri-stimulus; rather it uses three color properties to describe a color. Hue describes the dominant wavelength, saturation describes how dominant this wavelength is with respect to the whole of the visual spectrum and value indicates the brightness of the color response, similar to intensity. The HSV-space can be thought of as a cone, where the central axis ranges between black at the bottom and white on top, the value indicates the distance along this central axis. The angle around the axis is described by the hue, while saturation indicates the distance from the central axis to the edge of the cone.

When we defenestrate all but the hue, we are left with a 1-dimensional color space[4]. We can derive hue H from RGB as follows:

$$H = arctan\left(\frac{\sqrt{3}\,(G-B)}{(R-G)+(R-B)}\right) \tag{3}$$

Describing color in this manner is invariant to illumination and geometrical considerations as well as highlights [2]. The illumination intensity invariance can be understood in the following way: by removing the value we make our representation illumination independent, since value exactly describes this property.

2.2 Histograms

In the previous section, various color models and their invariant properties were discussed. Once a color model has been chosen we are confronted with two problems; first, describing an object, and second, identifying an object in a videoframe given the object description and it's location in previous videoframes. These problems can be overcome with the use of histograms. This section describes histograms as a probability density function (PDF) over a color space. In further sections the histograms or PDFs that represent the object to be tracked will be called the target model.

We can describe an image patch containing an object by listing all the colors contained in the image patch as well as their relative frequencies. This yields a PDF over the color space, that describes the local color. This local color descriptor is scale invariant, since an object image patch of different dimension should still yield the same relative frequencies. The description is also shift and rotation invariant, since a rotated image of the object will again yield the same descriptor, and the exact coordinates of the image patch containing the object are irrelevant to the resulting descriptor. This object description, however, is not object orientation invariant since different sides of an object can have different colors.

A PDF as described above creates a function over the whole color space, (for RGB, if every color channel is described by a byte, this leads to 2^{24} different colors). This will lead to a PDF which is too sparse, since most colors will not be in the image patch. Therefore, we collect similar colors in bins, and calculate the relative frequencies for each bin. In this report the sizes of the bins are equal, thus every bin spans a equally sized portion of the color space. By accumulating colors in bins, we get a more coarse color description. The coarseness of the PDF is relevant since we want to use these PDFs to describe and compare local image descriptors. We want local image regions that look the same, to yield local color descriptors that are similar. Choosing the correct bin size will take care of this. However, this raises the question as to how to pick the correct bin

[4]Note the difference between a color space and a color model. The latter induces the former. For instance, the RGB 3-dimensional color space arises from the tri-stimulus color model, with 1 dimension for each color channel.

size; bins that are too large lead to a PDF which is too coarse, and bins too small to a PDF that is too sparse. Therefore fine-tuning the number of bins is a delicate task of balancing between these two extremes.

2.2.1 Epanechnikov Weighting

Histograms so far weight every color in an image region equally. However, it is reasonable to assume that the color near the center of the region is more important than the color near the borders. Therefore a weighting kernel is used to assign a larger importance to colors near the center than to colors near the border. This means that the relative frequency becomes a weighted relative frequency, where the weights have been based on the location in the image region.

The mean-shift tracker uses a kernel with Epanechnikov profile [3], where "the profile of kernel K is defined as a function $k : [0, \infty) \to \mathbb{R}$ such that $K(x) = k(\|x\|^2)$."[4]. The following equation describes the Epanechnikov profile and subsequent kernel: Let x be a normalised location, such that the borders of our image region lie at distance 1. Then the Epanechnikov profile is

$$k_E(x) = \begin{cases} 1 - x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{if } x > 1, \end{cases} \tag{4}$$

with a subsequent radially symmetric kernel in 2 dimensions, described as:

$$K_E(x) = \begin{cases} \frac{2}{\pi}(1 - \|x\|^2) & \text{if } \|x\| \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Let $x = (x_1, x_2)$ be a 2 dimensional vector indicating a normalised pixel location, such that after some manipulation we derive

$$K_E(x) = \begin{cases} \frac{2}{\pi}(1 - x_1^2 + x_2^2) & \text{if } \sqrt{x_1^2 + x_2^2} \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The Epanechnikov kernel is used because the derivative of the profile equals -1, which makes the derivation and implementation of the mean-shift tracker less troublesome than with another kernel, though kernels such as Gaussians are feasable.

2.2.2 Bhattacharyya Distance

Multiple local color feature descriptors need to be compared to find the one that most closely resembles the description of the object we are tracking. Since the local feature descriptors are PDFs we need a similarity or distance measure between them. One such distance is the Bhattacharyya distance, which for discrete distributions is defined as:

$$d(p, q) = \sqrt{1 - \sum_{u=1}^{m} \sqrt{p(u) \cdot q(u)}}. \tag{7}$$

where m denotes the number of bins, p and q are two discrete PDFs and $p(u)$ is the probability density for bin u in PDF p.

Intuitively it can be seen that when both p and q are similar, $\sqrt{p(u) \cdot q(u)}$ will be large for bins where both have accumulated a large probability mass. Therefore, $d(p, q)$ will be small. When two PDFs are dissimilar, none of the bins with a large probability mass will match up, therefore $\sum_{u=1}^{m} \sqrt{p(u) \cdot q(u)}$ will be small, and $d(p, q)$ will be close to 1.

2.3 Other tracking methods

This section describes other methods of tracking, or object localising that have been explored for this project. Not all methods are used in the experimentation section, though all have been researched. A short description is therefore justified. Furthermore the intuition gained with brute force tracking will come to use in our discussion of the mean-shift tracker.

2.3.1 Histogram back projection

With a PDF as target model, we can take a picture and assign every pixel in that image the probability attached to the corresponding bin in the PDF. In this manner, pixels, which are assigned a high probability, are similar to many pixels in the target, since it falls in a bin with large probability. Likewise pixels with a low probability are largely dissimilar. The object can then be found in a region of the image that has high average probability. A problem with this approach is that it only looks at the dominant colors in the target and not to the similarity of the PDFs between regions. Furthermore, it requires an exhaustive assignment of probabilities to all pixels.

2.3.2 Brute force

This section will give a description of the brute force method, as a simple method of tracking, useful as contrast to the mean-shift tracking method. As with histogram back projection, first a color histogram (the target model), is computed to model the object to be tracked. In a new frame, color histograms are computed at every location; these are the candidate models. The target model is compared with all candidate models using a distance measure. The new location is then determined by the location of the candidate histogram with the smallest distance.

One optimization of the brute force tracker we explored, exploits the knowledge that video consists of multiple frames per second and that objects and camera move within a reasonable range of speeds. Therefore, it follows that the object location should be close to the previous location, and thus only locations that can be reached within the range of reasonable speeds need to be considered. This makes the brute force approach somewhat more tractable.

To understand brute force tracking intuitively, consider an image for which at every location the color has been replaced by the Bhattacharyya distance between the image region centered at that location and the target model. The resulting image can be seen as a height map, where the highest peaks represent the locations least similar to the target model, while the lowest are most similar. If one drops a marble onto this height map at the last known location of our target, the marble will roll to and stop at the locally lowest point. All points for which the marble will roll to a specific local minimum are the basin of attraction of that minimum.

2.4 Mean-Shift

The thought behind the mean-shift algorithm is the following: Instead of computing all Bhattacharyya distances in a region close to the last known object position, one could take steps towards the local minimum of the Bhattacharyya distance. Therefore we need to create a shift vector which indicates in what direction and how big a step needs to be taken move towards the lowest Bhattacharyya distance. If we are close to the local minimum, small steps are prudent, whereas further away larger steps are desired.

The shift vector is computed in the following manner, first weights for evey bin are calculated, in the following manner[5]:

$$w(u) = \sqrt{\frac{q(u)}{p(u)}} \tag{8}$$

[5]For the derivation we refer to [4]

where $w(u)$ is the weight, $q(u)$ is the target model, $p(u)$ is the candidate model all for bin u.

As an intuition when $p(u)$ and $q(u)$ are of equal magnitude, $w(u)$ will be close to 1, which can be considered as a neutral weight, the probability mass in both bins is similar thus no shift is needed. If $p(u) > q(u) \rightarrow w(u) < 1$ and if $p(u) < q(u) \rightarrow w(u) > 1$. Thus the weights are large for bins which have low probability mass in the candidate model, but height mass in the target model.

Every location in the candidate region "pulls" at the center of the candiate region with the weight of its corresponding bin times the relative location to the center. Thus a location that lies far from the current center which has a large bin weight "pulls" hardest at the center, while either locations close to the center or with low weights pull less hard. The mean of this pull at the center, when normalised, is the shift vector. Which is calculated as follows, let v_{shift} be the shift vector and x be a normalised pixel location, with the center of the candidate region as the origin, $Pixels$ be all pixel locations in the candidate region, $c(x)$ be the color of a pixel at location x, $b(c(x))$ the bin corresponding to color $c(x)$ and $w(b(c(x)))$ the weight of the bin corresponding to the color of the pixel at location x. The following equation gives us the calculation of the mean-shift vector[5]:

$$v_{shift} = \frac{\sum_{x \in Pixels} x \; w(b(c(x)))}{\sum_{x \in Pixels} w(b(c(x)))} \tag{9}$$

As a further intuition, when an object moved to the left, the weight values will be high on the left side. Furthermore the weights will be low on the right side because new information (different colors) entered the search window.

The mean-shift algorithm brings all previously discussed parts together; first the target model is obtained. Next, in every new frame the mean-shift algorithm starts at the last known position of the object, on this first candidate location a candidate model is created by computing the color histogram. Using both candidate and target models, $w(u)$ is computed for every bin, after which through the weighted shift or "pull" from every location the mean-shift vector v_{shift} can be obtained. Until the shift vector is smaller than a pixel distance this is repeated. The resulting location is a local minimum and will be used as the location where the tracked object was found. This process is repeated for every image in the video[6].

The mean-shift algorithm does not exhaustively compute all Bhattacharyya distances at locations in a region where the target might be found, but rather steps towards a local minimum of the Bhattacharya, thus making this algorithm more tractable. A drawback of this algorithm is that the tracked object cannot move too far from the last known position, this would mean that the current position falls outside the basin of attraction of the target and will thus not be found. Another drawback is that only color information is used, which makes the algorithm unstable when faced with changing lightsources or occlusion, especially when occluded by an object of similar color.

3 Implementation

The implementation of the mean-shift algorithm generally follows the implementation as discussed in [4]. However, there are some noteworthy differences as well as some practical issues worth discussing in the following section.

3.1 Mean-Shift Implementation

The implemented mean-shift algorithm performs the following steps: Given :

- a target model q and

[6] some checks and precautions are prudent, such as checking that the Bhattacharyya distance has indeed become less after a shift, or caution for large steps which "overshoot" the local minimum.

- the last target location y_0.

For a new frame:

1. Assign the candidate location, $y = y_0$.
2. Calculate the candidate model p at location y. (keep the 1st candidate model as p_0)
3. Calculate the weights w for every bin, using p and q and equation 8.
4. Obtain the shift vector v_{shift} using equation 9, w and normalised pixel locations x in the candidate region.
5. Rescale v_{shift} to standard picture sizes, "un-normalise".
6. If $v_{shift} < \frac{1}{2}$: (thus rounded the shift is less than a pixel location)
   - if true, goto. 7
   - else, assign $y = y + v_{shift}$, goto 2
7. Check if the Bhattacharyya distance between location y and y_0 has indeed decreased with $d(p, q) < d(p_0, q)$:
   - if true, $y_0 = y$
   - else, $y_0 = y_0$.
8. Load the next frame and goto 1

A difference with the mean-shift tracker as presented in [4] is that there are no abundant checks if the distance is indeed decreased at every step. Also, the "overshoot" prevention of taking half a shift step was not found to be required. Another difference, is the modified Epanechnikov kernel, other than the mean-shift tracker in [4] this implementation does not use a circular kernel but a square kernel such that all four corners lie at distances $x = (\pm\sqrt{\frac{1}{2}}, \pm\sqrt{\frac{1}{2}})$. The total weight of the kernel is then normalised to 1.

3.2 Variable Window Sizes

A version of the mean-shift algorithm with variable window sizes, was implemented, this algorithm was more complex. For instance, with a fixed window size the Epanechnikov kernel can be pre-computed and used whenever a histogram needs to be computed. However with differing window sizes the Epanechnikov kernel changes size and needs to be computed on a 'case by case' basis. The variable window size was implemented as follows, once the mean-shift has sub-pixel magnitude and thus the shifting terminates, the resulting location is fixed. Centered at this location regions, 1 pixel larger or smaller in both vertical and horizontal directions, are taken as candidate regions. The region that has the lowest Bhattacharyya distance between the candidate model and target model will be the new window size.

3.3 Speed considerations

During the development it was noticed that most of the time of the mean-shift tracker was spent loading and writing the tracked video frames. Therefore, all frames are pre-loaded in memory before the tracking starts. This has the benefit that the tracking is extremely fast. On the datasets presented in the next section, tracking (excluding the loading and saving of the frames) on a

singlecore@3Ghz consumer grade PC was more than 10x faster than real time. The downside of this approach is that the computer needs to keep all frames stored in memory; this might thus become a bottleneck[7].

4 Evaluation

Several experiments were performed to test the performance and robustness of the mean-shift tracker. In this section we will describe the datasets and the experimental setup, after which the results will be presented and discussed.

4.1 Experimental Setup

It is worth noting that all experiments use the exact same target region to compute the target model and start with the same first candidate model. That is, the tracked object is not a variable factor within the experiments. Also the measures are all qualitative assessment by the authors. Though it was never any discussion about the assessment if a track was lost. Either the trackers performed well, or performed badly.

4.1.1 Dataset Soccer

The first dataset consists of frames from a movie of a soccer match, see Figure 1-2. It is a trivial task for humans to track these soccer players, therefore, it is interesting to see how the mean-shift algorithm performs. Players have colored clothing and they move on an almost uniform green background. The tracker, with the right configuration, should work on this domain. Furthermore it provides an appropriate setup to compare different colorspaces.

Multiple players can be tracked with and without occlusion by other players. A single player is tracked, the player has variable speed because he is at different points accelerating or decelerating. A problem would occur if a player is moving so fast that he moves out of the basin of attraction in the next frame. For this dataset, this can become a problem since the players are relatively small.

4.1.2 Experiment Soccer1

This experiment is setup to test the mean-shift tracker for variable

- image quality
- color models
- histogram bin size

In the first experiment, we use a simple setup without occlusion. We downsampled the image 2x without smoothing or interpolation to test on data with bad quality. As a comparison we also used images without downsampling. Another comparison is made for the performance of the mean-shift tracker with different color models and different bin sizes. We use 4 different color models, RGB, XY, rg (normalised red and green), H (hue) and HSV (hue, saturation, intensity). We also use 4 different bin sizes[8].

4.1.3 Experiment Soccer2

This experiment is setup to test the mean-shift tracker for variable

- occlusion

[7]The authors did not observe this bottleneck, with any of the short movies described in this document.

[8]The bin size is directly dependent on the number of bins per color channel used to model the target/candidate histogram.

- histogram bin size.

The second experiment tests for occlusion. During tracking, a player is fully occluded by a player of another team. Moreover, to make it even harder for the tracker, a smaller target and candidate region is used. Here also different bin sizes are experimented on. The variable for downsampling has been eliminated, since the tracker already performed badly on most tests in the simpler un-occluded domain.

4.1.4 Dataset Freefall

To test the performane of the tracker in a different domain, a movie of Tjerk skydiving was also used for tracking, see Figure 3-4. In this movie a dual jump is made, which is filmed from an airplane. The starting frames of this movie are aliased, this movie starts with bad image quality. The skydivers (object) diminish in size quickly, furthermore the object rotates in the air, presenting different angles to the camera. The object has a bright color, while the background is a patchwork of white and grey clouds, green and brown fields. It is also worth mentioning that the object falls through a cloud, blending it's color with that of the cloud. This dataset can be considered more difficult than the soccer domain.

4.1.5 Experiment Freefall

This experiment is setup to test the mean-shift tracker with

- variable window size
- differing datasets

In this experiment we test the performance of the mean-shift tracker on a different dataset, to confirm that the tracking is not dataset, or domain, dependent. Furthermore, we test the variable-window size implementation. The tracker performed well on previous experiments with most color models as long as 8 bins per color channel were used. Therefore, this experiment only uses the standard RGB color model with 8 bins per color channel. The experiment consists of judging the performance of the tracker in the new dataset compared to the tracking in the soccer dataset, with and without the variable window size.

4.2 Results

The results of experiment Soccer1 are represented in Table 1. Experiment Soccer2 has its result dispalyed in Table 2. While experiment Freefall, can easily be judged by examining Figure 3-4.

4.3 Discussion

Table 1 conveys some interesting results. First, the tracker performed very well on the frames that where not downsampled. Only for a very large number of histogram bins per color channel (32), the tracker failed. This can be interpreted as overfitting. The large number of histogram bins per color channel causes the model to contain too much detail. If the model contains too much detail, it cannot generalise among the slight differences in the appearance of the soccer player and therefore loses track.

The tracker performs quite badly on the downsampled frames. This is a consequence, because the downsampling uses no interpolation or smoothing, which results in an image with poor quality. Only a few colorspaces survived: rg, H and HS. These colorspaces performed well because they are intensity invariant. Because the soccer player is moving, he recieves different illumination at different locations. This different illumination on different locations hold for many tracking tasks, therefore intensity invariant colorspaces are very useful.

Table 1: Results of Soccer1, mean-shift performance tested on differing color models, downsample rate and histogram bins per color channel (4,8,16,32). + means the tracker kept track for a minimum of 10 seconds.

scene	color model	downsampled	4	8	16	32
soccer	RGB	1x	+	+	+	-
soccer	RGB	1x	+	+	+	-
soccer	rg	1x	+	+	+	-
soccer	H	1x	+	+	+	-
soccer	HS	1x	+	+	+	-
soccer	HSV	1x	+	+	+	-
soccer	RGB	2x	-	-	-	-
soccer	rg	2x	-	+	-	-
soccer	H	2x	-	+	-	-
soccer	HS	2x	-	+	-	-
soccer	HSV	2x	-	-	-	-

Table 2: Results of Soccer2, mean-shift performance tested on differing color models, and histogram bins per color channel (2,4,8,16,32) for a scene containing occlusion. + means the tracker kept track for a minimum of 10 seconds.

scene	color model	2	4	8	16	32
soccer	RGB	-	-	-	-	-
soccer	XY	-	-	+	-	-
soccer	rg	-	+	+	-	+
soccer	H	-	-	-	+	+
soccer	HSV	-	-	-	-	-

In the results for experiment Soccer2 Table 2, where occlusion was introduced, it can again be seen that only the intensity invariant colorspaces keep track of the player. Furthermore, we see that the hue color model only keeps track on very large bin sizes, because it uses only one dimension. The fewer dimensions a colorspace needs, the more bins it needs to model a certain detail, and not over-generalise.

It can also be seen that rg performs better than XY. This is likely because the green background of the soccer field is modeled in more detail in the rg colorspace and therefore rg has more discriminate power.

An example where the tracker fails can be seen in Figure 1. On the other hand an example of a succesful tracking result can be seen in Figure 2.

During the Freefall experiment Figure 3-4, the track was never lost even though the skydivers are small in the end. From this we can conclude that the tracker works well for different domains. Also, the variable window size works, in this domain with a quickly in size diminishing object, the tracker had no problem resizing the window to include the object. Though critcism can be expressed about the fact that in the the target model is the head of the skydiver, whereas in the end the track is on the whole of the skydivers. Yet this does not take away the fact that it kept track on the whole of the object.

4.3.1 Brute Force Discussion

During the project the brute force tracker was implemented, but later modified to the mean-shift tracker, therefore we will not give a comparison of all the results obtained for the mean-shift tracker. A discussion of some anecdotal evidence, from the soccer domain, about results obtained by the brute force tracker can however be given. Computing the Bhattacharyya distance for every possible candidate region in an image is extremely slow, not to mention memory intensive[9]. However, if a sufficiently large region around the last known position is selected the object is tracked well, for un-occluded players. In the soccer domain, half the width of the player, added to each side of the last found object region, gave good results. Occlusion remained a problem, in one instance during a track, a player was tracked from left to right, while a teammate passed right to left in front of the tracked individual. The track of the 1st player was lost while the second was now (wrongly) considered as the new tracked object. Since they were teammates they were both wearing the same sporting outfits, therefore, this confusing tracking behaviour is not unforseen if only color is used.

5 Future Work

It is remarkable that downsampling has such poor effects for tracking in all color models, while non-downsampled frames get tracked well. This could partly be explained by the lack of proper smoothing in the downsampling used. Therefore, Gaussian smoothing before downsampling should improve performance on downsampled images. Possibly a critical point can be determined where further downsampling would have disasterous effects. One might also think of a Gaussian pyramide, experiments might determine what the smallest picture in the Gaussian pyramid is, that still gives a good track. With this information tracking can be done on the smallest possible image, improving the speed of the tracker. Another speed improvement can be investigated by skipping frames and determining how many can be skipped while keeping track.

Another topic for further investigation, follows from the experiments with occlusion, only tracking on color seems to be too sparse of a description. For example, when 2 players bypassed each other, the tracking sometimes passes from one to the other player. If speed of the object is also

[9] the MATLAB approach of using sliding windows creates memory problems, in that the matrix containing all sliding windows for an image becomes huge.

Figure 1: Tracking performance in RGB colorspace modeled with 8 histogram bins per color channel. The tracker loses track at the moment of occlusion.

used in the description this behavior could be avoided. This would include implementing a Kalman filter to keep track of the speed.

Furthermore, further research into why and when certain color models and histogram bin sizes work better than others, should make it possible to automatically detect what color model will perform best in a movie. This would enable automatic color model selection, which again should improve the tracker.

Some minor future work would make the tracker more robust, for instance in avoiding window sizes which are extremely small, or extremely big, or problems when the tracked object comes close to the boundary of the image frames, is also needed.

Last an important aspect which requires improvement is a good evaluation for tracking. This report relies on qualitive assessment of the trackers by the authors. Better metrics could be used; for instance, one can think to measure the Bhacharattya distance along a track, which is a measure

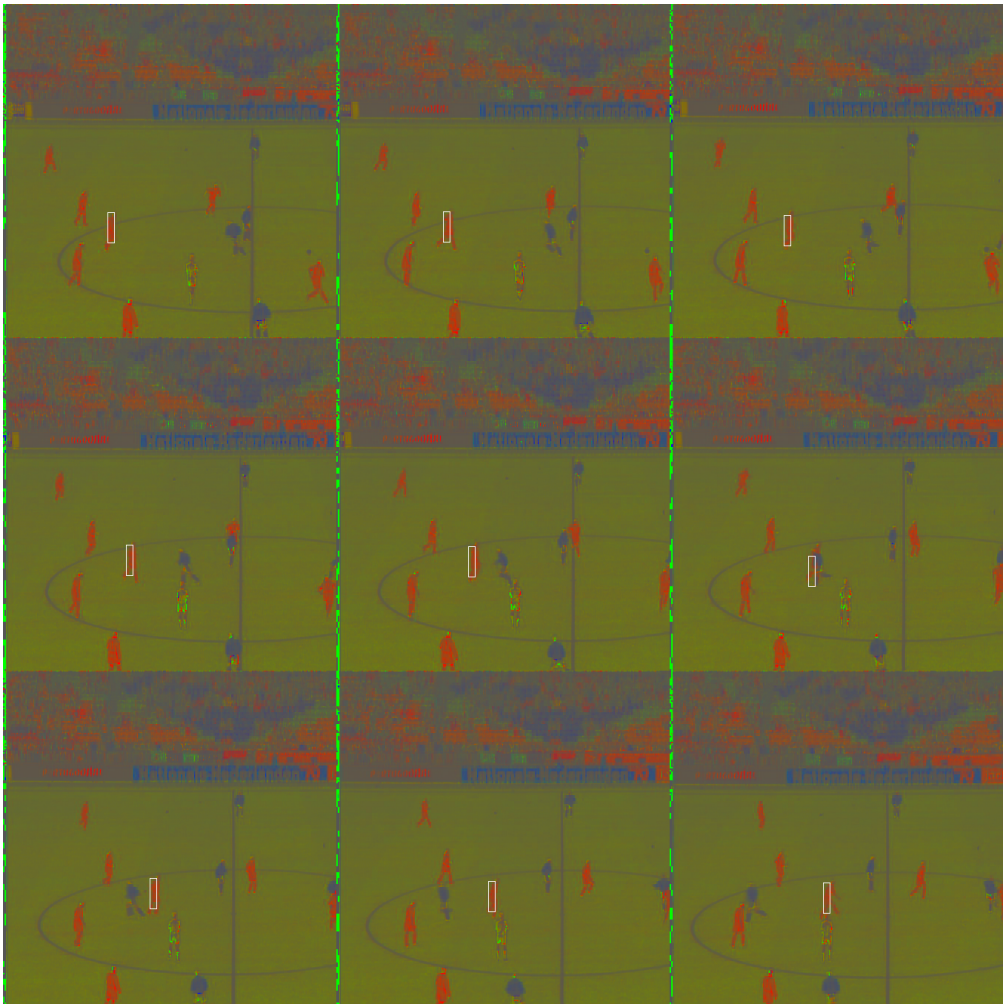

Figure 2: Tracking performance in rg colorspace modeled with 8 histogram bins per color channel. The tracker keeps track even if the player is occluded.

for the quality of the match of the found object. Also a ground truth hand-labeled track would enable one to measure the distance from the tracker to this ground truth. This would also make it possible to quantitatively compare different trackers, such as the brute force tracker with the mean-shift tracker.

6 Conclusion

The evaluation of the mean-shift tracker shows that it performs well once the right parameters for color model and histogram bin size are discovered. However the sole reliance of the tracker on color as a feature, shows in its performance on occlusion, can be poor because it depends on the color model used. The speed of tracking is fast which means that there is enough processing time left for more computationally complex video analysis tasks. However, for real-time applications the loading and saving of images will have to be considered.

Figure 3: Tracking performance of an alternative scene (Freefall) where the object is quickly decreasing in size. The tracking is performed in a RGB color space with 8 histogram bins per color channel. The tracker performs very well.

An interesting similarity between both the mean-shift tracker and the brute force tracker is that the tracked object cannot move too far from the last known position. The brute force tracker cannot exhaustively search the whole image for performance reasons and thus if the object falls outside of its search window it will lose track. For the mean-shift tracker, even though it does not have a search window it has a basin of attraction. If the player moves too far the last known position falls outside the basin of attraction and the track is also lost.

Some suggestions towards improving the tracker and its evaluation have been presented, the Kalman filter seems a likely candidate to improve the tracker in domains with occlusion. While better evaluation methods would also prove that its performance would be better.

These improvements would enable the mean-shift tracker to obtain even better results that the ones presented in this document.

Figure 4: Tracking performance on Freefall, with changing window size, note that the size gets smaller compared to Figure 3. The tracking is performed in a RGB color space with 8 histogram bins per color channel. The tracker performs extremely well.

References

[1] T. Gevers, *Color-Based Retrieval*. Springer-Verlag, 2001, ch. 2, pp. 11–48.

[2] T. Gevers, J. van de Weijer, and H. Stokman, *Color Feature Detection*. CRC Press, 2007, ch. 9, pp. 203–224.

[3] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.

[4] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–575, 2003.