# Exercise 2: Mean-Shift tracking

Maj Šavli 63150278

## I. Introduction

In various areas such as robotics, surveillance systems, autonomous driving and many other, tracking a specific part of an image can be vary important since the performance of such systems can depend greatly on the performance of such trackers. Nowadays, cnn based trackers mostly outperform other trackers. However, even after so many years, old tracking methods can still perform good. One of such methods is tracking based on the mean shift method for locating the maxima of a density function. In this project, we implemented the mean shift method and used it in our implementation of the tracker. We tested the tracker on multiple video sequences and reported the results in the following section.

## II. Experiments

Mean shift method searches for the maxima of the probability function, which tells us how likely it is that the tracked object is at specific point. We tested the implemented method on artificially made probability function, seen in the figure 1. The probability function has two local maximums, one at (50, 70) with a value of 0,00083 and a global maximum at (70, 50) with a value of 0,00161.

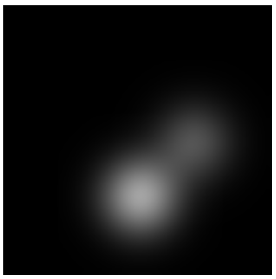

Figure 1: Artificialy made probability function.

The performance (convergence speed and finding the maxima) of mean shift method highly depend on three input parameters kernel size, epsilon and starting position. The performance results can be seen in table 2 below.

| | # of iterations | Function value |
|---|---|---|
| Kernel size 3x3 | 428 | 0,00161 (function max) |
| Kernel size 7x7 | 106 | 0,00161 |
| Kernel size 9x9 | 45 | 0,00161 |
| Epsilon 0.01 | 209 | 0,00161 |
| Epsilon 0.05 | 116 | 0,00159 |
| Epsilon 0.1 | 73 | 0,00147 |
| Start (40, 50) | 209 | 0,00161 |
| Start (65, 35) | 190 | 0,00083 |
| Start (80, 80) | 241 | 0,00161 |

Figure 2: Performance with different parameters.

We can see that increasing the kernel size increases the "sight" of the method, meaning that it has bigger overview of the surrounding pixels, resulting in more efficient moving in the direction of the gradient using lesser iterations till convergence.

Epsilon is the value which we tell the method when the change is too little to continue, or when the mode-seeking has converged. So if we increase the epsilon, we get faster convergence and there is smaller chance for method to get into infinite loop, but we may not get the maximum of the funciton and vice versa.

Probably the most important parameter is the starting point. If we start on a plain, where all the surrounding values are zero, the gradient is zero and we cannot or don't know in which direction to move. Mean shift method does not always find the global maximum as well, as we can see from the results. If we start close to the local maximum at (50, 70), we converge to this local maximum instead of global maximum at (70, 50).

Since change in x and y direction can be smaller than 1 and minimum difference relevant to computer (because of the arrays) is 1, we won't move until the change in any of the directions doesn't exceed a whole number. We can speed up convergence in such way, that we move for 1 pixel in the direction, which will reach the whole number the fastest. For example, if current position if (34.32, 63.57) and x change is 0.4 and y change is 0.02, we increase x by 1, because it will become 35 before y will become 64.

On the figure 3 below, there is another example of probability function. On the image there is a white dot denoting the starting point and black line leading from it which denotes path to convergence (path to maximum).

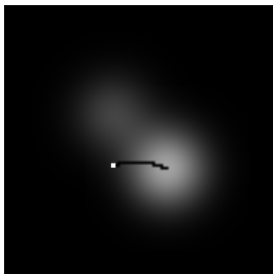

Figure 3: Artificialy made probability function 2, with convergence path.

The tracker uses the discussed method to find the new position of the object tracked, after it moves in the next frame. As well as mean shift's, tracker's performance also depend on multiple parameters. Images or patches of images are represented using color histograms, which take in the parameter number of histogram bins. The kernel, which was used for extracing the color histograms was Epanechinkov kernel. We can regulate the weights of the kernel with parameter kernel sigma. After every frame, we can update the visual model, since it can change a bit during moving. With update alpha we can regulate how much the visual model is updated after every frame.

The output of tracker, new position of the object, was marked as successful, if the bounding boxes around computed position and around ground truth (correct position) were overlapping. If the boxes were not overlapping at all, if was marked as failure. We tested the tracker on five video sequences of different length. The number of failures for each of the sequences can be seen

in the table 4 below. For the testing purposes, all results were acquired with the following parameters: $update\_alpha = 0.1$, $histogram\_bins = 16$, $kernel\_sigma = 0.5$ and $epsilon = 1$.

| Sequence name | Sequence length (frames) | # of failures |
|---|---|---|
| Hand2 | 236 | 3 |
| Polarbear | 371 | 0 |
| Jogging | 307 | 14 |
| Woman | 597 | 14 |
| David | 770 | 9 |
| Sunshade | 172 | 0 |

Figure 4: Performance with different parameters.

The parameters used to acquire results in the table 4 above, were overall good settings for the tracker. Sequences differ in terms of illumination, quality, target size, etc. and changing settings for every sequence separately would result in a better or faster performance. However, changing the parameters can also make tracker perform bad or slow. In the table 5 below, we can see how increasing number of histogram bins affect the speed of tracker (in frames per second) for the sequence *hand*2. We can also see that in this case, the number of failures decreases with increasing bins, which is not always the case.

| # of hist. bins | tracker speed (fps) | # of failures |
|---|---|---|
| 4 | 840 | 6 |
| 8 | 545 | 4 |
| 16 | 145 | 3 |

Figure 5: Tracker speed by histogram bins.

Since *epsilon* parameter only affect the mean shift method, its affect is the same as before, increasing it would probably find better or more accurate new position, but the method may fall into a loop. We found that ideal value for it was 1. The parameter which probably affected the performance of a tracker or failures the most was alpha for regulating visual model update. For the most sequences, the tracker performed the best, when we didn't update the model at all, with alpha being equal to 0. On some sequences, the tracker performed best, when alpha was equal to $0, 1$. In average, for five sequences, increasing the alpha parameter led to more failures, as we can see in the graph 6 below.

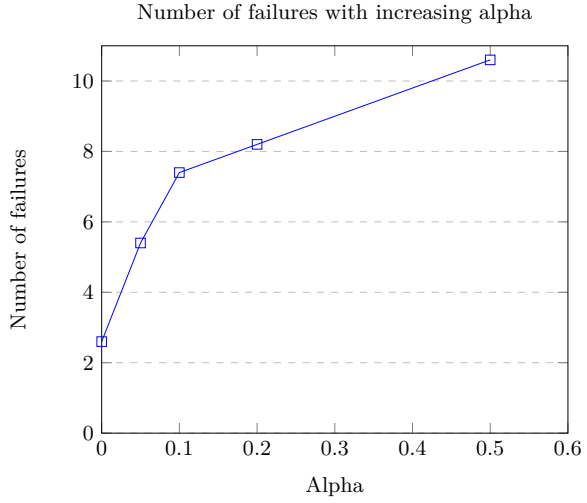


Figure 6: Number of failures with increasing alpha

## III. Conclusion

In this project we implemented a simple tracker based on the mean shift method. We tested the tracker on multiple sequences and tried out multiple tracker settings. We found out, that there exist an overall good setting, which in average performs good. To achieve better performance, we must set settings for every sequence separately. Some parameters, such as epsilon improve trackers' performance, but decreases its speed, while other parameters, such as update alpha, which regulates the update of the visual model in average yields more failures than not updating model at all. Taking the trackers' simple idea into account, it still performs good and with many possible improvements, it could perform even better.