Mean-Shift Tracking

Lenart Rupnik, 63220472

I. Introduction

The main purpose of this exercise was to implement a Mean-shift algorithm to track objects/regions on different sequences. This algorithm is based on a simple idea of mean-shift, where we look for highest density inside our matrix by calculating density inside our defined window and then iteratively moving according to the density gradient until we converge. In our case, we used a mean-shift algorithm to find the best match for our previously defined template of the desired object. Our window was a certain region in rectangle shape defined at the start of the video. In the end, we manage to create a fairly successful algorithm for object tracking.

II. Experiments

A. Mean-shift mode seeking

As was stated in the instructions, we firstly implemented the mean-shift mode seeking method on an artificially created density matrix that contained two maximums and was smoothed with Gaussian Smoothing. Positions of maximums were (50, 70) and (70, 50). To make our algorithm work, we implemented the derivative of a mean-shift kernel, which was shown on slides. Results are shown in Table I.

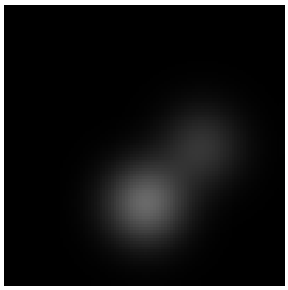

Figure 1: Initial density matrix.

At first, we compared different kernel sizes. From first three examples in Table I we can observe that kernel size primarily influences the number of iterations. When we increased the size of first kernel by 4 in each dimension, we managed to converge twice as fast compared to the first example. To check this theory, we used another kernel of size (31, 31). In this example, we actually didn't converge. After examinating the problem, we figured out the termination criteria was too small for a kernel of this size. Still, we saw that the actual convergence happened in only 6 steps, but then the kernel floated around the same two pixels. Using a bigger kernel for faster convergence does make sense, since we include more pixels in calculations and it can take longer steps along the gradient. For future measurements, we decided to use kernel of size (7,7).

Next we tested different starting location. As expected in all cases we converged to one of two maximums. The number of iterations was around the same for all three examples, as well as the final location accuracy. Even though the second maximum is weaker than the first one, we managed to converge to it when we started close to it. In general, starting position mostly influences number of iterations. Also, if we wanted to always find the biggest maximum, the starting location would be crucial for avoiding converging to local minimum.

Table I: Mean-shift mode seeking comparison.

Kernel	Start loc.	Thres.	Iter.	End loc.
(5,5)	(35, 40)	0.05	133	(50, 69)
(9, 9)	(35, 40)	0.05	51	(51, 70)
(31, 31)	(35, 40)	0.05	∞(6)	(53, 70)
(7, 7)	(35, 40)	0.05	82	(51, 70)
(7, 7)	(20, 70)	0.05	74	(50, 70)
(7, 7)	(80, 20)	0.05	86	(71, 50)
(7, 7)	(40, 40)	0.01	123	(51, 70)
(7, 7)	(40, 40)	0.1	74	(51, 69)
(7, 7)	(40, 40)	0.5	22	(49, 57)

In the last three examples, we compared different termination criteria. That way, we can control how and when we converge. Smaller the values that we use, bigger the chance that we find ourselves in an infinity loop of not converging at all. On the other hand, if we set it too high, we might converge before even reaching the final position. If we take a look at the number of iteration in last three examples, we see that the effect of different termination criteria is as expected, higher criteria - faster convergence and smaller accuracy.

Based on our observations through these nine examples, we can say that we could speed up convergence by using bigger kernel and setting higher termination criteria.

To additionally test mean-shift mode seeking, we included another density matrix where we created two equal global maximums on two edges and one local maximum in the middle as shown in Fig. 2. We used two different kernels, one of size (7,7) and the other of size (39,39). Converged locations are marked with a red dot for smaller kernel and a yellow one for bigger kernel. Both methods successfully converged. What is clearly shown in this example is how kernel size also affects the ability to find global maximum and not be stuck in some local maximum.

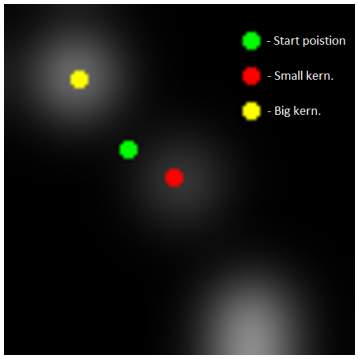

Figure 2: Custom density matrix with convergences.

B. Mean-shift tracker

Mean-shift tracker is based on image histograms and previously described method called mean-shift. The main idea is that we look at sequential images from the video and try to find a

Table II: VOT(14) sequence tracking comparison.

Sequence name	FPS	Failures	Failures (improved)
bolt	634	5	0
torus	548	2	0
fish1	631	4	2
skating	279	6	5
bicycle	545	4	1

selected object in the next image by using mean-shift on image histograms. To simplify our method of mean-shift convergence, we use image colour histograms as a reference. For extracting the colour histogram, we use Epanechnikov kernel. Since we find the convergence in colour histograms, we need to use histogram backprojection to construct a foreground similarity distribution and identify new positions. For evaluation of our tracker, we used image sequences from VOT(14), where we have labelled ground truth, so that we can compare our method in every step of tracking. In case we slip off the target, we mark it as failure, reset our tracker template and continue with tracking on new template. In the end, we compare number of failures for specific video.

After implementing this method, we tested it on 5 sequences from VOT(14) and the results are shown in Table II. For initial tests we ran our algorithm with parameters: $bins = 16$, $sigma = 0.7$, $epsilon = 0.1$, $termcriteria = 1$ and $enlargefactor = 2$. Effects of these parameters will be discussed later.

All the sequences presented in Table II gave relatively successful results with few failures. Next we focused on some typical examples where our tracker failed.

The first thing that we noticed is that our algorithm failed where the background and selected object changed dramatically, e.g. rotation. This makes sense since we are using a static template for mean-shift algorithm. In order to tackle this problem, we introduced template updating. After each successful iteration, we update our initial template by 5% of the new histogram. This way we made our algorithm more adaptable to object changes.

A second fail that was quite common was where the background had similar colours as our selected object and that resulted in tracker focusing on an object in the background. This can be seen nicely in bolt sequence, where Bolt's yellow shirt is similar to the starting pad. To solve this issue, we added background normalization. The main idea is to look at the bigger patch around the selected object and then normalize the template histogram with the one from the background so that we don't focus on colours that also appear in the background but on colours that are unique for our template.

Another failure that can be seen is when the selected object is occluded by another object. This is well represented in sequences like bicycle and skater. For this failure we didn't find any solution since it's hard to track the object that is actually not on the image. Perhaps we could set some threshold of matching. If we went below that, we would remove the tracking rectangle and inform the user that object is not present anymore and then search for the object in the whole image.

The last common failure that we noticed was when lightning in the image sequence changed significantly. A good example is the skating sequence. We didn't manage to find any improvements in this area. Perhaps if we would update our template faster but this would influence drifting of the object which would result in issues elsewhere. After our failure analysis, we combined all the mentioned improvements and ran the algorithm again. Results can be seen in the last column of Table II.

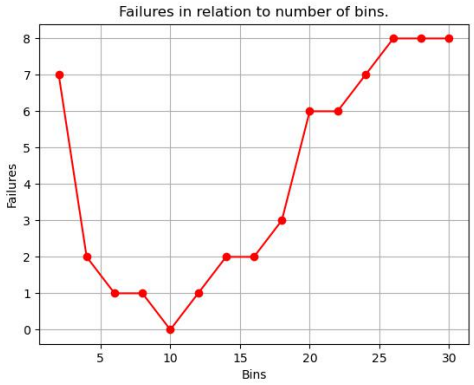

Figure 3: Effect of number of bins on failures.

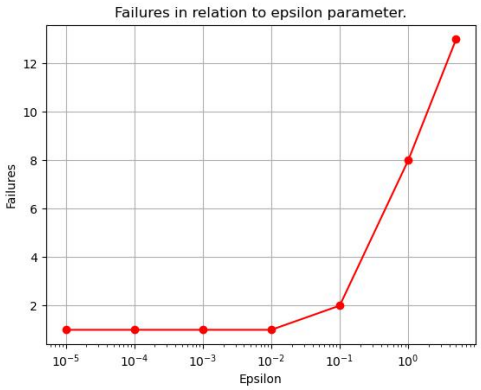

Figure 4: Effect of epsilon on number of failures.

To find the best parameters for our tracker we tested different configurations with changing number of *bins*, *sigma*, *epsilon*, *threshold* and $enlargefactor$. In our study case (bolt) we found out that the only two parameters that made significant difference were number of bins and epsilon. Results are shown in Fig. 3 and Fig. 4.

Observing Fig. 3 we see that using the general rule of 16 bins doesn't produce the best results. In our example, we see that the best results can be obtained by using only 10 colour bins. But this probably doesn't hold for every sequence. If we have a sequence with a lot of different colours, it might be better to use more bins.

Based on Fig. 4 we can say that there are many working settings for epsilon as long as it isn't so high that it influences the actual histograms in calculations. Still we can't set it to zero because then we might divide by zero which can't be done.

III. Conclusion

We successfully implemented mean-shift algorithm on both mode seeking and sequence tracking. With mode seeking, we learned how to implement and improve our mean-shift algorithm that we then applied in tracking. By additionally configuring parameters including template updating and background-foreground normalization, we got great results in terms of number of failures. Although being quite simple, this algorithm proved to be relatively successful.