Exercise 2: Mean-shift
Advanced Methods in Computer Vision

Dimitar Stefanov

I. Introduction

For this assignment, we needed to implement a very well known algorithm for tracking - the mean-shift. Firstly, a mode seeking function had to be developed and tested in various algorithm settings. The idea behind this method was then used for the implementation of a mean-shift tracker. The performance of the tracker was evaluated on a series of video sequences from VOT(14) (Visual Object Tracking 2014) [1]. The results of this analysis served as the starting point for the search of optimal parameters for the algorithm.

II. Experiments

A. Mode-seeking function

To begin with, we tested our mode seeking function on a provided image representing a probability density (size of image: 100x100 pixels; maxima at (70, 50) and (50,70) with the second being the global maximum). During the tests, we were expected to configure the algorithm in different ways (various starting positions, kernel sizes, etc.) in order see how it performs. So, for the starting positions, we chose 4 points: (50, 50), (65, 50), (80, 20) and (20, 80). The first point was chosen, because we thought it would be meaningful to start at the center of the image. In addition, this point was equidistant from both of our maxima, so we wanted to observe if the algorithm might fail and detect the lower of the two maxima. The second position (65, 50) was much closer to the lower maximum, so it also presented a good test of our implementation. Last two points were chosen to observe the behaviour of the algorithms when it starts near a corner, but also because each point was significantly closer to one of the two maxima. The size and shape of the kernel were investigated as well, by trying 4 different sizes: 5, 9, 15 and 21, and 2 different shapes: square and elliptical. Due to better readabilty, for the elliptical kernel we only reported number of iterations in the cases when the performance was notably worse than the one of the square kernel. For stopping criterion, we utilized the length of the shift vector. The results obtained were summarized in Table II:

As we can notice from Table II, when the center of the image was our start, we always managed to converge to the global maximum. The same can be said, and it's no surprise, about the (20, 80) point. What is perhaps more interesting is the behaviour of the algorithm when initialized from the pixels with coordinates (65, 50) and (80, 20). It turns out kernel sizes 5, 9, 15 (both square and elliptical shape) and 21 for an elliptical kernel converge towards the local maximum at (70, 50). However, when utilizing a square kernel of size 21, hence taking into consideration a wider region of the image, we were able to escape the local maximum and find a path to the global one. Aside from that, there was no unexpected behaviour of the algorithm. The bigger the kernel size, the lower the stopping criterion needed to be in order not to stop with the iterations before reaching the maximum. Also, as can seen from the table, in all cases when the algorithm was initialized closer to a maximum, it needed fewer iterations to locate it.

Nevertheless, the convergence speed presented above can be further increased. One way to achieve this would be by changing the current position for a multiple of the shift vector. In other words, we still move in the direction of the shift vector, but we increase the step we make. Here we assume that the gradient ascent is performed in the correct direction, so we can speed it up by taking larger steps. To be more exact, in the cases of twice or three times larger shift vectors, we were able to reduce the number of iterations by exactly the same number of times. Utilizing bigger multiples of the shift vectors at first still decreases the number of iterations by some factor, but after a certain value it is not useful to consider such high multiples.

B. Testing tracker on video sequences from VOT(14)

In the second part of our assignment, we were tasked with developing a tracker and testing its capabilities on video sequences from VOT(14). The sequences we chose to demonstrate the functioning of our tracker were: "ball", "basketball", "david", "polarbear", "sphere" and "bolt". In terms of parameter setting, after some initial exploration, the update parameter alpha was set to a value of 0 meaning no object template update. In regard to the stopping criterion, at first, we were using the shift vector length as a stoppage measure. But, then we noticed it results in unnecessarily longer iteration process, so we decided to be stopping the algorithm after 10 iterations. For the extraction of color histograms, we opted for the standard choice of 16x16x16 bins. So, the results obtained by utilizing these settings were the following:

Table I: Initial performance of the mean-shift tracker on sequences from VOT(14) was rather satisfactory.

video	alpha = 0, 10 iters, 16x16x16 bins	
	FPS (frames per second)	failures
ball	1588.2	0
basketball	90.0	2
david	1137.5	2
polarbear	1280.7	0
sphere	915.8	0
bolt	707.3	2

To sum up the results on the video sequence tests, we can say the algorithm has no trouble maintaining the position of the target object in almost monochromatic frames as that was the case in the "polarbear" sequence. The model also copes rather well with changing color histogram of the target, although in the sequences "bolt", "basketball" and "david" it had slight issues. Most notably, in the second part of the "bolt" video, the histogram template varies dramatically from one frame to another, hence our failures in following the object. We give ideas to resolve this issue in section II-D.

C. Testing mode seeking function on own probability density function

In our own function, we decided to have three maxima, only one of which would be global. The dimensions of the image were as before 100x100 pixels, but the maxima were located

Table II: We show how different parameters can lead to different performance of the mode seeking function, sometimes even to not locating the global maximum.

size	stopping criterion	start position to convergence position and number of iterations (square/ellipse)			
		(50, 50)	(65, 50)	(20, 80)	(80, 20)
5	0.01	(50, 70) 230/314	(70, 50) 153	(50, 70) 191/250	(70, 50) 234/321
9	0.05	(50, 70) 73	(70, 50) 28	(50, 70) 58/75	(70, 50) 68/86
15	0.1	(50, 70) 30	(70, 50) 16	(50, 70) 22	(70, 50) 33
21	0.3	(50, 70) 13/17	(50, 70) / (70, 50) 30/6	(50, 70) 12	(50, 70) / (70, 50) 45/17

at different positions: (50, 25), (50, 50) and (75, 75). We also normalized the response, so that we have a true probability density function. For the tests, we chose three starting pixels: (50, 20), (70, 37) and (87, 87). All three of them were chosen to really test the convergence towards the global maximum, and the first and third point seemed especially promising.

Nevertheless, with all the different kernel sizes, when beginning the gradient ascent from pixel (50, 20) being extremely close to the local maximum at (50, 25), we always managed to converge to the global extremum. On the other hand, when as starting position we had the point (87, 87) and on the line between it and the global maximum, there was the maximum at (75, 75), we were unable to go past this lower maximum with any of the kernels. This result can serve as a proof that certain points inevitably lead to a local maximum no matter what the algorithm parameters are. Lastly, we would like to mention that the performance of elliptical kernels was omitted in Table III, because it corresponded with the observations in section II-A showing that an elliptical kernel converges slightly slower than a square one for our functions and parameter settings.

Table III: Tests of the functioning of the mode seeking function on our probability density function have shown that there exist starting points from which even by using a large kernel we might not be able to find the global maximum.

size	stopping criterion	starting position		
		(50, 20)	(70, 37)	(87, 87)
9	0.05	(50, 50) 61	(50, 50) 49	(75, 75) 44
15	0.1	(50, 50) 23	(50, 50) 25	(75, 75) 17
21	0.3	(50, 50) 13	(50, 50) 10	(75, 75) 8

D. Identification and resolving of failure cases in the video sequences

Once having conducted the initial tests of the tracker on the video sequences, we started to experiment with the algorithm parameters in order to reduce the failure cases in certain cases. As already mentioned above, for the sequences "bolt", "david" and "basketball", the cause of failures was the change of the object template throughout the recording. The constant with which we try to adapt to such situations is alpha. So, we started to slowly increase this constant hoping to see improvements. It turned out that a value of 0.1 for alpha helps filter one of the two fails in the sequences "david" and "basketball". However, this higher value of the constant impacts negatively the video named "bolt", and consequently we register 4 instead of the 2 mistakes that we had before. Bigger values for alpha turned unhelpful for all the recordings.

III. Conclusion

Overall, our implementation of the mean-shift algorithm shows satisfactory results. The algorithm is quite fast as we need only 10 iterations to find a new position of the object. It is also robust, and even in cases when we failed to track the target, by considering some small changes to the parameters, we obtained the desired results.

References

[1] The Visual Object Tracking VOT2014 Challenge Results. https://www.votchallenge.net/vot2014/download/visual-object-tracking.pdf.