Exercise 2: Mean-shift tracking

Sara Bizjak

I. Introduction

Tracking various objects can play a huge role in many different fields – from tracking parts in the automotive industry, to tracking people on sporting events to perform personal analysis. One such tracking method is based on the mean shift method, which is a a non-parametric feature-space mathematical analysis technique for locating the maxima of a density function and can be called mode-seeking algorithm.

II. Experiments

A. Mean-shift mode seeking

After the implementation we test how well our algorithm performs and show the results on a *toy function* (can be seen on Fugure 1), generated with the help of function `generate_response_1()`.

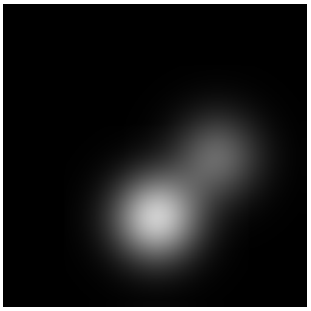

Figure 1. Density function, generated from `generate_response_1()`, with local maximum (50, 70) with function value 0.00083 and gobal maximum (70, 50) with function value 0.0016.

But before rushing into running the algorithm, we can calibrate different parameters, such as starting point, kernel size and epsilon (i. e. the termination criteria). We can quickly see that quite a lot depends on a selection of starting point. If we choose the starting point closer to a local maximum, the convergence can lead to a local maximum instead of the global. This situation is shown on Figure 2 and also highlighted in pink in Table I.

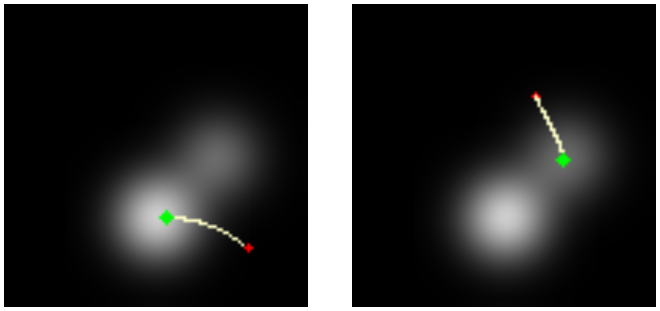

Figure 2. The convergence to the global maximum depends on the choice of the starting point.

As well as staring point, also kernel size and epsilon value can determine the convergence. If we choose epsilon too big, the process may terminate even before it actually started (brown colored text in Table I), or, with the combination with too small kernel size, it may terminate too soon (blue colored text in Table I). All other combinations of parameters converge to global maximum, however, we see that too small or too big kernel size can lead to more iterations than actually needed to obtain the right point. In that cases, we indeed can speed up the convergence.

Parameters			N. iters	Function value
Start point	Kernel size	Epsilon		
(80, 80)	3 x 3	0.01	499	0.0016
(80, 80)	5 x 5	0.01	240	0.0016
(80, 80)	7 x 7	0.01	123	0.0016
(80, 80)	11 x 11	0.01	499	0.0016
(60, 30)	5 x 5	0.5	0	0.000066
(60, 30)	5 x 5	0.1	74	0.00074
(60, 30)	5 x 5	0.01	201	0.00083
(40, 40)	3 x 3	0.01	499	0.0016
(40, 40)	5 x 5	0.01	238	0.0016
(40, 40)	7 x 7	0.1	122	0.0016

Table I
Parameters calibration.

Thus, testing and choosing the right combination of parameters for individual case can be crucial for our algorithm to converge to the right point, and also, to ensure that the convergence is time efficient.

To test the algorithm a little bit more, we generated another *response function*, on which we observe similarly as before – we need to set the parameters carefuly, so that we indeed reach the global maximum through the iterations. On the Figure 3 we see two different starting point. First one is (40, 45) and the second one is (50, 60). From both starting points the path converged where we wanted, however, to achieve this, we needed to enlarge the kernel size from 5 to 7 for the second point, since it is further away from the global maximum.

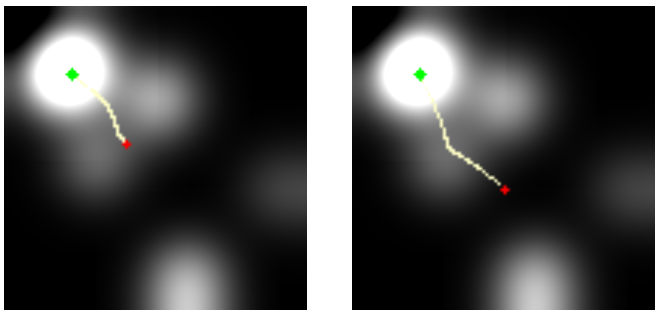

Figure 3. Two different starting points *demand* two different kernel sizes to reach the global maximum.

B. Mean-shift tracker

Mean-shift tracker uses the mean-shift algorithm to find the new position of the object that is being tracked. The performance of a tracker can be measured with counting the mistakes it does – the less, the better. The mistake is defined as bounding box, calculated from the tracker, not overlapping the correct positioned bounding box. The results from testing on 6 video sequences from VOT2014 are shown in the Table II.

Sequence	Number of frames	Number of failures
Basketball	725	12
Gymnastics	207	0
Skating	400	3
Bolt	350	7
Drunk	1210	3
Jogging	307	2

Table II
PERFORMANCE OF MEAN-SHIFT TRANCKING IN TERMS OF FAILURES, TESTED ON 6 SHORT VIDEO CLIPS.

After testing the tracker on many different video sequences, we can see that the tracker has problems if the tracked object changes the size, i. e. coming closer/further to the camera (like seen on sequence `bolt`), or changes the shape, i. e. jumping, running etc. (like seen on sequences `hand1` and `hand2`). This all makes sense, since we are not estimating the whole object, but ony its center point. To improve the tracker in such cases, we would probably try to estimate the size and shape of the object, so that we would know by how much the object changed during the sequence.

Before tracking, we can calibrate some parameters to ensure the better performance. For each sequence, we can try several different options and select the one that results into least failures.

Epsilon, together with number of max iterations, must leave enough, but at the same time not too much space for corrections of the tracker, since the tracker can then quickly become less smooth and starts to shake around the object. In that case, the bounding box may *lose* the target object, which can result into more failure. With iteration number being set to 20, we actually regulate the situation well enough, so that we can set epsilon a little bit lower. Thus, the trade off between corrections made and correctness is reached. In all our examples, the small values for α worked best. This can also be seen on Figre 4, where we are increasing parameter α with n_bin is set to 8.

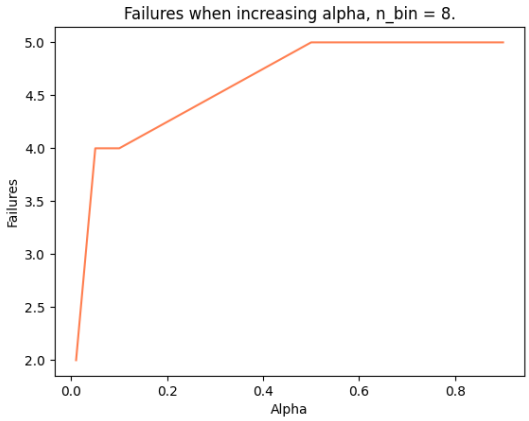

Figure 4. Calibration of parameter α on sequence `jogging`.

We need to be a little bit more careful with number of bins. It looks like increasing the number of bins decreases the number of failures, but it is not always the case, so we must calibrate the parameter manually and separately for every example.

After the research we decide that the most general choice of parameters that work well enough for all cases are (can also be found listed in the class `MSParams` in file `mean_shift_tracker.py`):

- sigma = 1
- number of histogram bins = 16
- number of max iterations = 20
- eps = 0.05
- epsilon = 0.01
- patch factor = 1.05
- alpha = 0.01
- enlarge factor = 2

Note that those parameters were applied also for obtaining the result in Table II. Those parameters were not choosen only becauce of the performance in terms of failure counts, but also on visual effect (bounging box was nicely following the object and not shaking too much) and in terms of time. Even if we the tracker may perform equally in failure counts, it does not neccesserely mean that the tracker *actually* yields the same results.

III. CONCLUSION

As a part of the exercise, we implemented mean-shift mode-seeking and mean-shift tracking algorithms. The tracker was tested on several sequences, on which a set of paramaters was set, such that they are overall a good choice. However, to achieve even better performance, we must set parameters separately for each sequence. With parameters set properly, the algorithm performed better than expected, with few problems when tracked object was changing the shape or size significantly. We can conclude that the tracker could perform even better with some small corrections and improvements.