Mean-shift tracking

Blaž Erzar

I. Introduction

In this exercise, we have implemented a tracking algorithm using the mean-shift approach for similarity optimisation. First, we use the mean-shift mode-seeking method to find the modes of two functions. Next, the same functionality is used to implement the tracker. Finally, the tracker is evaluated on the VOT14 dataset using various parameter configurations and possible tracker improvements.

II. Experiments

A. Mode seeking

First, we compare the convergence of the mean-shift mode-seeking algorithm for different starting points. We compare two kernels, the uniforms and the Gaussian kernel, with different bandwidth sizes. We repeat iterations until the updated location moves for less than 0.4 pixels. In Figure 1 and Table I we can see convergence results for 6 points. On the left figure, we can see that the Gaussian with the same bandwidth size converges slower than the uniform kernel and the final position is not as close to the maximum as the uniform's one. If the starting position is too far from the mode or the kernel is too small, the method does not converge.

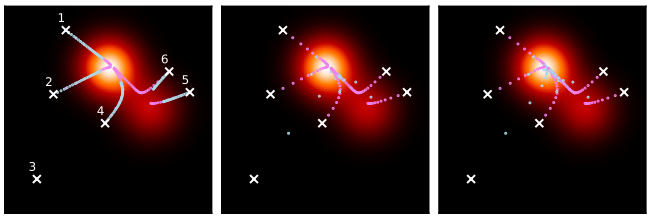

Figure 1: Convergence of mode seeking using the uniform kernel (violet) compared to the Gaussian kernel with the same bandwidth (left), the uniform kernel with a larger bandwidth (middle) and the uniform kernel with adaptive bandwidth (right).

To make convergence faster, we can make steps bigger by increasing the kernel size. This also means we escape the local maximum in some cases, but the final position is again not close to the actual maximum, as can be seen in the middle. To mitigate this, we use an adaptive bandwidth size, starting with a large value and decreasing it in each step. We can see on the right that this approach converges faster, but still comes close to the actual maximum.

	Uniform			**Gaussian**
Point	$h = 23$	$h = 63$	adaptive	$h = 23$
1	13	5	4	26
2	13	5	5	26
3	1	5	10	1
4	17	5	8	33
5	12	7	17	19
6	31	7	9	17

Table I: Number of steps till convergence for points in 1

B. Tracker

We run the implemented tracker on 6 sequences from the VOT14 dataset. The number of failures for 6 sequences can be seen in Table II. There are 9 sequences with 0 failures, and 2 with 2 and 3. The worst performance is on the tunnel sequence with 5 failures and the remaining 11 sequences have a single failure.

Sequence	Failures	**Sequence**	Failures
basketball	0	fish1	1
bicycle	1	hand2	5
bolt	0	tunnel	3
Total failures			26

Table II: Number of failures for 6 selected sequences from the VOT14 dataset and the whole dataset together.

C. Mode seeking on Griewank function

Before looking at the failure cases of the tracker, we look at the mode-seeking convergence for one more function - Griewank in this case. In some cases, the iterations do not converge to the mode due to the step sizes being too small. In this case, the adaptive bandwidth performs worse than in the first example. This occurs because the final adaptive bandwidth size is too small and there are more steps needed for convergence.

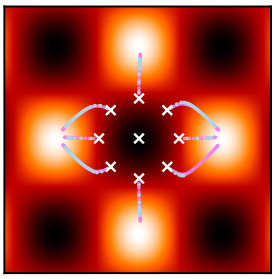

Figure 2: Mode-seeking convergence on the Griewank function with the uniform kernel with bandwidth size $h = 23$ (violet) and an adaptive bandwidth (light blue).

D. Failure cases

The failures in the observed dataset sequences occur because of different reasons. In Figure 3 we show some of the failures of the sequences from Table II.

In the bicycle sequence shown in 3a the target colour has changed from the beginning of the sequence. Because the object is getting closer to the camera, the bounding box is getting bigger. Both of these reasons cause the tracking to slowly drift from the centre of the bounding box and finally fail when the background gets cluttered.

In the fish sequence in 3b, the bounding box and the appearance of the object drastically change because the fish turns sideways. The bounding box becomes almost four times smaller. This is similar to the previous example, and the tracking fails similarly.

(a) (b) (c)

(d) (e) (f)

Figure 3: Tracking failure cases for (a) bicycle, (b) fish1, (c, d) hand2 and (e, f) tunnel sequences.

The tracking in the hand2 sequence fails for other reasons. The main difference from the previous sequences is that this one contains fast motions that become blurry in the captured video. When the hand passes over the face quickly, the tracker starts to track the face as can be seen in 3c. This happens because it contains similar colours to the hand. In 3d the tracking fails again, but this time it starts tracking some background object due to fast blurry motion.

The final sequence we look at is the tunnel sequence. It starts in bad lighting in the tunnel, meaning the target model does not contain a lot of colour information. The tracking first fails when the harsh lighting produces a lot of white pixels similar to the initial target appearance. The second target is captured in a very dark part of the video, meaning the histograms do not contain a lot of colour information. The tracking again fails, see 3e, when the video gets even darker and the object's vicinity is almost completely black. The tracking continues working but fails one more time before the end when another very similar scooter appears right next to the tracking target. This frame is shown in 3f.

E. Tracker parameters

We compute the performance of the tracker with respect to multiple parameters independently. Changing the maximum number of iterations does not affect the results much. Decreasing it to 5 increases the total number of failures by 2, so we choose 20 for the optimal results. In Table III we first compare the different number of histogram bins and the parameter σ of the Epanechnikov kernel. The optimal values seem to be around 16 bins and σ between 0.5 and 0.75.

	Bins				Kernel σ			
Sequence	4	8	12	16	0.25	0.5	0.75	1.0
basketball	1	0	0	0	0	0	1	2
bicycle	2	1	1	1	3	1	1	1
bolt	2	2	0	0	3	0	0	1
fish1	3	4	2	1	9	1	1	6
hand2	6	5	4	5	8	5	4	4
tunnel	6	3	3	3	5	3	3	4
Total failures	54	28	28	**26**	65	**26**	26	34

Table III: Number of failures for 6 selected sequences with different number of bins and difference kernel σ.

The other two parameters we optimise are the weighting coefficient between the new and previous target model, α, and the threshold distance ε used to stop iterations. The optimal threshold is around 0.5. We might think that higher α values would be beneficial since the target model would contain recent information. This is not the case here. Probably the performance decreases because the detections are not as precise as ground truth, meaning the updated target models are incorrect.

	α					ε			
Sequence	0.0	0.01	0.05	0.1	0.2	0.25	0.5	0.75	1.0
basketball	0	0	0	0	2	0	0	0	0
bicycle	1	1	1	2	3	2	1	1	1
bolt	0	0	2	3	4	2	0	2	2
fish1	1	4	4	5	3	1	1	1	1
hand2	5	4	5	4	5	5	5	5	5
tunnel	3	2	2	2	2	3	3	3	3
Total failures	**26**	34	37	35	42	30	**26**	28	28

Table IV: Number of failures for 6 selected sequences with different weighting coefficients α and convergence thresholds ε.

F. Improvements

In the end, we try a few improvements of the implemented tracker. We implemented the computation of the background features histogram, which is then used to weight the target and proposal histograms. This approach resulted in the lowest number of total failures till now, 24.

Sequence	BG	HSV	YCrCb	Lab
basketball	0	1	0	1
bicycle	1	1	2	2
bolt	0	0	1	0
fish1	5	1	1	2
hand2	6	5	5	6
tunnel	4	5	2	3
Total failures	**24**	29	31	31

Table V: Number of failures for 6 selected sequences with tracker improvements: background features (BG), using different colour spaces (HSV, YCrCb, Lab).

We also tested three additional colour spaces, but none of them increased the performance. We also tried averaging their trackers, but that approach performed even worse.

III. Conclusion

We can conclude that the mean-shift tracker is a quite simple tracker with a decent performance which runs very fast on modern hardware. To some extent, it can work even with occlusion or changes in the appearance model.

Even though the used dataset is now 10 years old, it is still a good indication of the tracker's performance. In most cases, the tracker performs without or with a single failure. We can further improve the performance by tuning the parameters, or even implementing some simple improvements. More possible improvements could still be made to the tracker, perhaps using a model of the dynamics or even combining multiple colour spaces in a way.