Exercise 3: Correlation filter tracking

Nikolay Vasilev

I. Introduction

Correlation filter tracking [1] is a widely used approach in computer vision for tracking a target in image sequences. It is based on the correlation between the target template and the search region, and estimates the position of the target by finding the maximum peak in the *correlation response map*.

Among the correlation filter trackers, **MOSSE** (Minimum Output Sum of Squared Error) has gained huge popularity due to its fast processing time and high performance.

In the following paper, we will present our implementation of the MOSSE correlation filter tracker in Python as well as implementation of the actual MOSSE tracker. We will compare their tracking speed and performance by using different parameters, such as: update factor, parameter of gaussian and enlarge factor. The trackers will be evaluated in average on all sequences from the dataset VOT14 [2] as well as on each of the sequence. This will help us observe different sequences, find the failure cases and improve them.

II. Experiments

The MOSSE correlation filter is a specific type of correlation filter that is commonly used in computer vision. It is designed to track a target (object) across multiple frames in a sequence. It is based on a linear regression model that minimizes the output sum of squared errors between the filter response and the desired output.

First we have implemented a normal correlation filter in Python, where we generate the filter for each frame independently from the previous frame's numerator and denominator. This is our baseline MOSSE filter, which we improved as much as possible by adding the following steps:

- **patch preprocessing** – we convert our patch into grayscale, transform its pixel values using log function, which help us with low contrast lighting situations, and we normalize the pixel values to have a mean value of 0.0 and a norm of 1.0. In the end we multiply it with the cosine window to reduce the pixel values near the edge to zero.
- **adaptive scale** – we use the parameter, which is calculated as a multiplication of the adaptive scale constant and the previous patch's scale $\nabla s = adaptive\_scale * s_{prev}$. Then we calculate three possible regions with three different scales - one has the previous patch's scale s_{prev}, the other has increased scale $s_{prev} + \nabla s$ and the third has decreased scale $s_{prev} - \nabla s$. We calculate the correlation response for each of the regions and find the one with the maximum value. This is also the optimal scale, which with the help of the parameter β help us calculate the new size of the region the following way:

$$s_{new} = \beta s_{opt} + (1 - \beta) s_{prev}$$

  We found out that the best values here are $\beta = 0.1$ and $adaptive\_scale = 1e - 8$.

By evaluating the tracker with multiple parameters, we found out that the best values for the other parameters of the tracker are $enlarge\_factor = 1.5$, $\alpha = 0.3$ (update filter factor), $\gamma = 0.1$ (denominator factor), $\sigma = 2$ (parameter for Gaussian).

After we found the parameters, which give us the best result, we implemented the improved MOSSE tracker, which differs from the baseline one in the following steps:

- **filter initialization and online updates** – in this tracker the numerator and the denominator of the filter are stored separately, because we need them in the localization step. In this case we calculate the filter not only with the current frame's numerator and denominator, but also with the previos frame's. This means that the filter at frame t is calculated the following way:

$$\hat{H}_t^{\dagger} = \frac{A_t}{B_t}$$

$$A_t = \tau \hat{G} \odot \hat{F}^{\dagger} + (1 - \tau) A_{t-1}$$

$$B_t = (\tau \hat{F} \odot \hat{F}^{\dagger} + \gamma) + (1 - \tau) B_{t-1}$$

  , where τ is the learning rate and we found that its best value for our tracker is $learning\_rate = 0.75$.
- **failure detection and PSR** – we have also implemented the measurement of peak strength, which is called the Peak to Sidelobe Ratio (PSR). It is computed by splitting the correlation output g into its peak (maximum value) g_{max} and the sidelobe, which is the rest of the pixels excluding an 11x11 window around the peak. The PSR is calculated the following way:

$$PSR = \frac{g_{max - \mu_{sl}}}{\sigma_{sl}}$$

  , where μ_{sl} and σ_{sl} are the mean and standard deviation of the sidelobe. If the PSR value is smaller than some threshold, then this is indication that the object is occluded or tracking failed, so we don't update the target position, the filter, the numerator and the denominator. In our case the best threshold is $psr\_threshold = 7$.

Table I
Performance and processing time of both trackers using the best parameters

Tracker	Overlap	Failures	Init. FPS	FPS
MOSSE	0.45	83	221.68	115.04
Improved MOSSE	0.46	83	231.18	109.43

In Table I, we can see that the improved MOSSE tracker increases the overlap and the initialization speed and decreases the tracking speed. This was expected, since we are implementing more adaptive filter, which is using failure detection. By detecting the failures we increase the accuracy of our tracker, but decrease the initialization speed and in some cases the Failures. In order to prove that these are indeed the best parameters, we will also show the impact of some of the parameters on performance and the speed of the improved MOSSE tracker.

First, we will start with the impact of the enlarge factor, which we can see in Table II. Here we see that the enlarge factor indeed effects a lot the performance and gives the best results for the values between 1.25 and 2. Another interesting thing is that even tho the overlap and the failures get less when increasing the enlarge factor after the value 2, the initialization and performance speed decrease. This means that increasing

Table II
Impact of the enlarge factor on the performance and processing time of the improved MOSSE tracker

Parameter	Overlap	Failures	Init. FPS	FPS
$enlarge\_factor = 1$	0.44	115	512.48	236.31
$enlarge\_factor = 1.25$	0.45	86	297.36	154.23
$enlarge\_factor = 1.5$	0.45	84	187.43	108.43
$enlarge\_factor = 2$	0.45	89	120.82	70.20
$enlarge\_factor = 3$	0.43	146	50.75	31.52

the enlarge factor speeds up our tracker, even tho it may also leads to worse performance.

The other parameter, we want to take a look at, is the update factor (α). In Table III, we can see that this parameter also

Table III
Impact of the update factor on the performance and processing time of the improved MOSSE tracker

Parameter	Overlap	Failures	Init. FPS	FPS
$\alpha = 0.02$	0.44	115	219.54	95.03
$\alpha = 0.05$	0.45	101	187.20	113.63
$\alpha = 0.1$	0.47	92	217.91	109.49
$\alpha = 0.2$	0.46	84	241.14	115
$\alpha = 0.3$	0.45	84	187.43	108.43
$\alpha = 0.5$	0.45	83	195.25	120.37
$\alpha = 0.75$	0.44	82	230.40	109.22

has a big impact on the performance, but not so much on the speed. Anyway, after the value 0.1, increasing the parameter α doesn't really change the performance and the speed of the tracker. The main speed that is changed by this parameter is the average initialization speed.

The last important parameter, which has an impact on the tracker, is the gaussian parameter (σ), which is presented in Table IV. As we can see, this parameter doesn't really change the performance so much as the previous two, but this evaluation still help us find the best fit for our tracker. It is

Table IV
Impact of the gaussian parameter on the performance and processing time of the improved MOSSE tracker

Parameter	Overlap	Failures	Init. FPS	FPS
$\sigma = 0.5$	0.46	95	178.95	105.06
$\sigma = 1$	0.46	92	250.13	119.61
$\sigma = 2$	0.45	84	241.14	115
$\sigma = 3$	0.44	94	222.88	104.37
$\sigma = 4$	0.44	109	244.04	105.22

interesting that also the initialization and tracking speed is not changed much, except for the smallest sigma value, where we have the fastest initialization speed. Following these 3 tables, we can clearly see that the best values for these 3 parameters are exactly those, which we have chosen.

Now, when we have found the best parameters, we only need to evaluate the performance and processing time of the improved MOSSE tracker on each sequence. The results from this evaluation are presented in Table V, where we see that for each sequence, with exception of the "bicycle" sequence, the initialization speed is higher than the tracking speed. Another interesting thing that we can see is that for few of the examples the failures are pretty low, but the overlap is not high ("car", "drunk", "tunnel", etc.). It is more important here to point

Table V
Performance and processing time of the improved MOSSE tracker on each sequence

Sequence	Overlap	Failures	Init. FPS	FPS
ball	0.40	8	460.66	152.85
basketball	0.40	9	86.60	67.93
bicycle	0.44	0	86.60	394.51
bolt	0.43	2	168.67	104.13
car	0.39	0	168.67	168.67
david	0.66	0	168.67	140.66
diving	0.37	4	104.10	57.55
drunk	0.33	0	104.11	53.24
fernando	0.37	3	42.60	8.70
fish1	0.33	13	508.29	177.02
fish2	0.28	8	271.23	108.54
gymnastics	0.54	2	117.25	86.26
hand1	0.43	6	435.47	143.61
hand2	0.36	14	551.70	167.98
jogging	0.62	2	72.33	43.71
motocross	0.45	2	34.18	22.84
polarbear	0.45	0	34.18	46.95
skating	0.35	1	44.69	104.74
sphere	0.55	2	163.49	92.41
sunshade	0.68	0	163.49	104.29
surfing	0.69	0	163.49	134.41
torus	0.44	5	569.55	174.78
trellis	0.51	0	569.55	124.68
tunnel	0.30	0	569.55	114.06
woman	0.61	2	120.49	89.85

out that the accuracy of each of these sequence is pretty good even when it has more failures, which is partly because of the failure detection we have implemented in our improved tracker. The performance of the MOSSE tracker could be increased by focusing our attention on the sequences with most failures and lowest overlap such as "ball", "basketball", "fish1", "fish2" and "hand2". By figuring out why the tracker works bad in these cases, we will increase the performance for these sequences and with that make our tracker better performing.

III. Conclusion

We can conclude that these findings highlight the potential of MOSSE correlation filter trackers as a fast and accurate object tracking approach for computer vision applications. Future studies could explore further enhancements to the MOSSE filter or investigate the performance of correlation filter trackers on more diverse and challenging datasets. By improving the tracker for more sequences, we can find techniques and parameters, which are better fit for the tracker and with that increase its performance and processing time.

References

[1] D. Bolme, J. Beveridge, B. Draper, and Y. Lui, "Visual object tracking using adaptive correlation filters," 06 2010, pp. 2544–2550.

[2] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojíř, and G. Fernández, "The visual object tracking vot2014 challenge results," in *Computer Vision - ECCV 2014 Workshops*. Cham: Springer International Publishing, 2015, pp. 191–217.