Exercise 3: Correlation filter tracking

Maj Šavli 63150278

I. Introduction

In various areas such as robotics, surveillance systems, autonomous driving and many other, tracking a specific part of an image can be vary important since the performance of such systems can depend greatly on the performance of such trackers. Nowadays, cnn based trackers mostly outperform other trackers. However, some tracking methods are still popular due to its simplicity and generally good performance. One of such methods is tracking with template correlation filter. In this project, we implemented the simplified version of the MOSSE correlation filter tracker. We tested the tracker on multiple video sequences with VOT toolkit (lite version) and reported the results in the following section.

II. Experiments

The main idea for the tracker is to learn or get such a filter to have high correlation response on the object and low response on the background. For the creation of the filter, besides the patch in Fourier domain, we need a 2-dimensional Gaussian function, which takes in the parameter σ. The target or object moves throughout the sequence, so the the filter must be updated in every frame. The update speed or step can be regulated by α. Teh size of the target is always the same, but the patch or background can be even larger. If we increase the size of background area, the correlation filter could perform better, since it "sees" more around the object. The responses may be different as well, leading to different peak (object) location. The size of patch was regulated with parameter $enlarge\_factor$.

The tracker was tested with VOT toolkit lite on two datasets, one from year 2013 and the other from 2014. We evaluated performance of the tracker based on total number of errors on the dataset, average overlap, which tell us how well the predicted box intersected with ground truth and tracker speed (measured in frames per second). Parameters, which yielded best results we $\sigma = 2, \alpha = 0.1, enlarge\_factor = 1,1$. The results, obtained with there parameters, can be seen in table 1 below.

Dataset	Total errors	Average overlap
VOT2013	35	0,53
VOT2014	75	0,46

Figure 1: Tracker results on the datasets.

Changing the parameters can affect all of the three performance measures. In the table 2 below we can see, how update speed can affect the performance of the tracker. At $\alpha = 0$, meaning the model does not get updated at all, there is the highest average overlap, but there is also a lot more errors, which is in our opinion more important measure. Based on this, we can say, that the best results were obtained with $\alpha = 0,1$, with low errors, almost high average overlap and high tracker speed. High alpha values resulted in more errors and lower average overlap.

With sigma σ we regulate the intensity of the Gaussian function, the filter making function takes in as a parameter. The sigma values, which make sense are in the interval $[0,5,5]$.

Alpha	Dataset	Avg. overlap	Errors	Speed
0	VOT13	0,55	99	947
	VOT14	0,47	184	705
0.05	VOT13	0,48	34	791
	VOT14	0,46	89	645
0.1	VOT13	0,51	38	747
	VOT14	0,47	86	705
0.2	VOT13	0,52	35	828
	VOT14	0,42	89	721
0.5	VOT13	0,5	41	845
	VOT14	0,42	94	656
0.8	VOT13	0,44	43	821
	VOT14	0,41	92	761

Figure 2: Effect of update speed on performance

The effect of sigma on performance of the tracker can be seen in table 3. We can see that the impact on the performance is not as significant as with update speed α. Number of errors being more important measure, the best results were obtained with $\sigma = 2$, which yielded a little lower average overlap than $\sigma = 0,1$ for both datasets, but also lesser errors. The tracker speed did not change significantly with increasing σ.

Sigma	Dataset	Avg. overlap	Errors	Speed
0,5	VOT13	0,51	39	750
	VOT14	0,45	93	631
1	VOT13	0,53	42	726
	VOT14	0,46	85	688
2	VOT13	0,52	34	719
	VOT14	0,45	74	729
3	VOT13	0,52	34	711
	VOT14	0,45	80	680
4	VOT13	0,52	35	677
	VOT14	0,42	89	655
5	VOT13	0,51	41	750
	VOT14	0,43	84	685

Figure 3: Effect of sigma on performance

The last parameter, enlarge factor, tells us, how much of the background we extract from image. That is, how much around the object region we take into equation, when we apply correlation filter to it. We can see a little improvement in terms of average overlap, up to the value of 1,5. After further increasing the size of background, both average overlap and number of total errors started to get worse. Again, taking into account importance of errors, the best results were obtained with $enlarge\_factor = 1,1$, meaning, the extracted patch of image is just a little bigger than object region. Both average overlap and number of errors started getting worse, with increasing the factor. Moreover, with increasing the enlarge factor, tracker speed was greatly reduced.

The tracker speed in the previous tables were acquired as the average speed throughout the whole sequence not differentiating between initalization frame speed and tracking frame speed. In the table below, we can see the difference between

Factor	Dataset	Avg. overlap	Errors	Speed
1	VOT13	0,52	34	719
	VOT14	0,45	74	729
1,1	VOT13	0,53	35	730
	VOT14	0,46	75	632
1,5	VOT13	0,53	43	386
	VOT14	0,48	87	318
2	VOT13	0,49	58	252
	VOT14	0,46	112	283
3	VOT13	0,47	99	132
	VOT14	0,44	153	129

Figure 4: Effect of enlarge factor on performance

average initalization frame speed and tracking frame speed for some of the sequences of the VOT2014 database (in frames per second). As we can see in the table 5 below, the times to process the initialization time is in the most cases significantly bigger than times to process other frames. That is because in the initialization frame we need to create the correlation filter, which takes in as parameter Gaussian function and cosine window, which both also have to be created in the same frame.

Sequence name	Init avg. speed (FPS)	Track avg. speed (FPS)
Ball	1553	1105
Basketball	511	274
Bicycle	796	1100
Bolt	1124	847
Car	2273	1487
David	367	313
Fernando	232	71
Gymnastics	484	256
Motocross	677	350
Motocross	390	446

Figure 5: Initialization frame and tracking frame average speed.

III. Conclusion

In this project we implemented a simplified version of MOSSE tracker and tested it on multiple video sequences with VOT toolkit. We analyzed the effect of various parameters on the tracker performance and speed. We found out that the parameter for regulating filter update speed had the biggest effect on tracker performance in terms of total errors and average overlap. As far as speed is concerned, the biggest effect on it had the enlarge factor parameter. It seems reasonable, since the image patch gets larger and gets computationally more difficult. We also found out that it takes significantly more time to process the initialization frames in contrast to the other frames. However, considering the relatively simple idea of the correlation filter method, the tracker generally performed well.