Correlation filter tracking

Anže Mihevc

I. Introduction

Object tracking is a fundamental problem in computer vision and has numerous applications, including surveillance, video analysis, and self-driving cars. One approach to object tracking is to use a correlation filter, which compares a template of the object to be tracked with patches in the image. In this report, we will discuss the implementation of a tracker using a simplified MOSSE (Minimum Output Sum of Squared Errors) correlation filter. The MOSSE filter is a popular method for object tracking due to its computational efficiency and robustness. We will examine how the MOSSE filter works, and how it is implemented.

II. Experiments

A. The tracker

For this assignment a tracker using a simplified MOSSE correlation filter was implemented. The tracker is initialized with a starting position and size of the bounding box where the object of interest is located. Using a two dimensional Gaussian filter and a feature patch extracted from the image with the help of a cosine window, we construct a correlation filter. After we calculate the response in the next frame using inverse Fourier transform. We localize the new position of the object using the highest peak in the response. Lastly the filter is updated.

1) Tracking toolkit (lite): For running and evaluating the tracker we used the toolkit provided to us with the instructions. The toolkit is a modified version of the VOT toolkit and is made to simplify integration with the tracker. The toolkit sets-up a working directory in which it automatically downloads and extracts the VOT dataset which includes different sequences of images. The tracker is then ran and evaluated on this sequences using the provided scripts. In our case we used the 2014 version of the VOT dataset as it includes more sequences than the newer versions.

B. Tracking performance evaluation

To fine tune the tracker a few different parameters are used:

- **Enlargement factor** - Controls the search area size. If it is set to more than 1 we search a bigger area than the given bounding box. This can improve results but hurts computation speed.
- **alpha** - Controls the rate of learning. Larger alpha means more of the change in the object is taken into account.
- **sigma** - Controls how steep out Gaussian peak is. Larger sigma means smoother tracking.
- **lambda** - also called regularization factor.

Using this parameters the filter can be controlled and fine tuned for specific tasks. We want to know which values of the parameters are best in general. Some relationship between the parameters might exist but due to time constraints and exponential explosion of tests needed to confirm this theory we assume that the parameters are independent of each other when optimizing them. We also didn't change the parameter lambda as it does not impact the performance of the tracker. The lambda was set to 0.000001 in all test.

1) Alpha - learning rate: Parameter alpha controls the learning rate of our tracker. When choosing the value of alpha we have to be mindful of setting it too high or too low. A high value of alpha will cause the filter to loose the object whit even a small change in the background. But at a very low value our filter will not take into account the changes of the object.

We can see how alpha effects the performance of our filter in the table I and Figure 1. The other parameters were fixed to the following values: Enlargement factor=1.0, sigma=2.0. **Best value was alpha=0.2**

Tracker name	Alpha	Average overlap	Total failures	Average speed
mosse1	0,001	0,46	156	471,84
mosse2	0,01	0,46	121	444,54
mosse3	0,1	0,45	83	550,56
mosse4	0,2	0,46	79	580,96
mosse5	0,5	0,43	79	495,14

Table I: Comparison of the effect of different values of alpha on the performance of the tracker. Bigger alpha produces less failures, increases the speed but lowers the overlap.

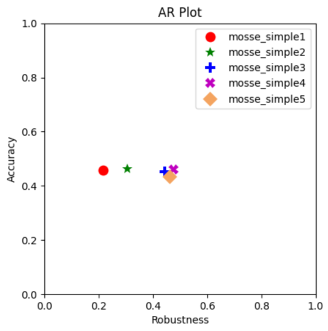

Figure 1: Comparison of the different alpha values. Graph shows robustness vs accuracy. Bigger alpha means more robust while accuracy is comparable.

2) Sigma: Parameter sigma controls the steepness of the Gaussian peak which is then used to create the simplified MOSSE correlation filter. A larger sigma will produce smoother tracking, improve failure rate but slow down our tracker. But a sigma too large will have negative effects on the performance of the tracker. This can be seen in table II. It doesn't have much influence on accuracy and robustness however as can be seen in Figure 2. The other parameters were fixed to the following values: Enlargement factor=1.0, alpha=0.2. **Best value was sigma=2.0**

Tracker name	Sigma	Average overlap	Total failures	Average speed
mosse1	1,0	0,46	81	563,27
mosse2	1,5	0,46	78	567,81
mosse3	2,0	0,46	76	606,54
mosse4	2,25	0,45	85	576,71
mosse5	2,5	0,45	89	556,53

Table II: Comparison of the effect of different values of sigma on the performance of the tracker. Bigger sigma produces less failures. But a sigma too big has negative effects.

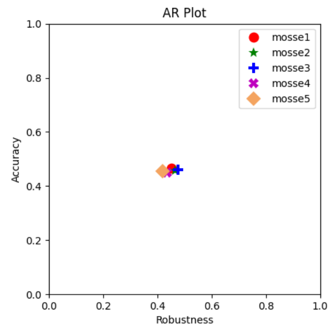

Figure 2: Comparison of the different sigma values. Graph shows robustness vs accuracy. No big differences can be seen.

3) Enlargement factor: Parameter enlargement factor controls the size of the search area around the bounding box provided to the tracker. If the value is 1 the search area is the same size as the bounding box. A larger value can increase the performance by still finding the object if it makes a bigger move or if it moves fast. However a larger search area means more computation thus increased time. We can see this effects in table III and Figure 3. **Best compromise for enlargement factor was 1.25**

Tracker name	Enlargement factor	Average overlap	Total failures	Average speed
mosse1	1	0,46	76	518,24
mosse2	1,25	0,49	72	440,57
mosse3	1,5	0,46	70	337,25
mosse4	1,75	0,47	78	266,75
mosse5	2	0,48	79	199,82

Table III: Comparison of the effect of different values of enlargement factor on the performance of the tracker. Larger values mean slower computation times but better results. Too large values do produce worse results.

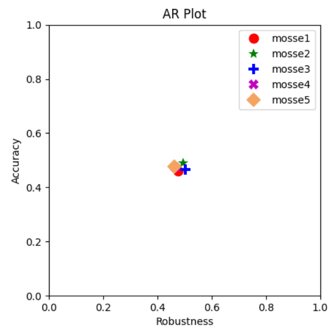

Figure 3: Comparison of the different enlargement factor values. Graph shows robustness vs accuracy. No large deviation is observed.

4) Tracking speed: The tracker is relatively fast. The enlargement factor has the biggest impact on the speed out of all the parameters which makes sense as it adds the most computation to the task. The time difference between initial frame and the rest of the frames is insignificant as we can see on some examples in table IV and thous we cannot draw any conclusions from the results. We can see the tracking speed in frames-per-second for different parameters in their respective tables. The time for different sequences can be sen in table V.

Sequence name	Initial frame time	Average frame time	Diff
ball	0,0018	0,0027	0,00094
bolt	0,046	0,0025	0,0020
drunk	0,0031	0,0038	0,00065
jogging	0,0011	0,0021	0,00095
polarbear	0,0015	0,0014	0,000061
AVG	0,0021	0,0021	0,00067

Table IV: Comparison of initial frame times and average frame times. No significant difference is observed.

C. Best tracker

The best overall parameters from our testing are as follows: Alpha=0.2, Sigma=2.0, Enlargement factor= 1.125. In the following tableV we can see the results of the tracker with this parameters. The tracker had an average overlap of 0.49, total of 72 failures and an average speed of 440 frames per second. We can see that sequences *fish1* and *hand2* have the most failures. This is because the objects move fast and in different directions.

Sequence	**Length**	**Overlap**	**Failures**	**Speed**
ball	602	0.337	4	351.340
basketball	725	0.552	2	372.633
bicycle	271	0.459	3	815.552
bolt	350	0.372	4	369.451
car	252	0.513	1	559.156
david	770	0.706	0	468.799
diving	219	0.423	4	214.854
drunk	1210	0.250	0	262.515
fernando	292	0.328	2	175.301
fish1	436	0.315	11	756.237
fish2	310	0.302	7	297.328
gymnastics	207	0.539	4	164.662
hand1	244	0.514	5	237.512
hand2	267	0.359	13	439.119
jogging	307	0.711	1	460.913
motocross	164	0.542	1	115.142
polarbear	371	0.464	0	681.483
skating	400	0.572	1	405.216
sphere	201	0.681	0	252.518
sunshade	172	0.694	1	760.232
surfing	400	0.522	5	632.262
tiger1	354	0.491	6	480.674
tiger2	414	0.471	6	314.764
woman	1191	0.438	1	336.189

Table V: Performance metrics of tracker using best parameters found in previous sections

III. Conclusion

The correlation filter tracking algorithm discussed in the lectures has been implemented successfully. It exhibits faster than real-time performance and is capable of tracking objects that were previously lost during the sequence. By tweaking the parameters, we were able to observe some impact on the algorithm's performance. The algorithm is not great with fast objects or objects moving moving in random directions. This is also the simplified version of the filter. We could get better performance with the full version.