Exercise 3: Correlation Filter Tracking

Ana Poklukar

I. Introduction

In this exercise, we implemented a MOSSE correlation filter tracker [1] and evaluated its performance on the VOT2014 dataset [2]. We analyzed the impact of different parameters on tracking accuracy, robustness, and speed, and conducted experiments to compare initialization and per-frame processing times across sequences. The results provide insights into the behavior and efficiency of the tracker under various settings.

II. Experiments

Firstly, we implemented the correlation filter tracker and integrated it with the Tracking Toolkit Lite [3], then tested it on the VOT2014 dataset. The experiments were conducted on an Apple M1 Pro processor.

Table I
Tracking performance of correlation filter tracker across different sequences.

Sequence	Length	Overlap	Failures	FPS
ball	602	0.402	3	2204
basketball	725	0.626	3	742.2
bicycle	271	0.396	1	3208
bolt	350	0.624	2	1504
car	252	0.439	0	3629
david	770	0.684	0	340.2
diving	219	0.394	2	730.4
drunk	1210	0.338	0	1108
fernando	292	0.389	1	287.0
fish1	436	0.391	6	3174
fish2	310	0.351	7	1305
gymnastics	207	0.612	2	1183
hand1	244	0.542	4	1023
hand2	267	0.486	10	1568
jogging	307	0.735	1	1804
motocross	164	0.541	1	285.9
polarbear	371	0.454	0	1081
skating	400	0.437	0	1867
sphere	201	0.723	0	989.8
sunshade	172	0.734	2	2553
surfing	282	0.646	0	4017
torus	264	0.601	4	2407
trellis	569	0.543	0	854.8
tunnel	731	0.320	0	1643
woman	597	0.721	1	1673
Avg/Total	**10213**	**0.525**	**50**	**1647**

In Table I, we present the results obtained using the following parameters: the learning rate $\alpha = 0.15$, which controls how much the filter adapts to new frames; the Gaussian peak width $\sigma = 2.0$, defining the extent of the response window and the regularization parameter $\lambda = 0.1$, which helps to avoid overfitting by penalizing large filter values. Our tracker achieved an average overlap of 0.53, with a total of 50 failures, and an average speed of 1647 frames per second across the entire dataset.

Then, we experimented with different values of the learning rate parameter α and the Gaussian peak width parameter σ to observe their influence on tracking performance.

In Table II, we present the results of varying α while keeping the other parameters fixed at $\sigma = 2.0$ and $\lambda = 0.1$. The average overlap remains relatively stable across different values of α, suggesting that the tracker is not highly sensitive to the learning rate in terms of alignment accuracy. Similarly, the average speed (in FPS) varies only slightly, indicating that the computational efficiency of the tracker is mostly unaffected by changes in α. However, a more noticeable trend appears in the number of failures, which is minimized when α is set between 0.1 and 0.2. This indicates that moderate values of the learning rate allow the tracker to adapt effectively without overfitting to recent frames or being too slow to update, thereby improving overall robustness.

Table II
Comparison of different learning rate (α) values on average tracking performance.

α	Avg. Overlap	Failures	FPS
0.01	0.49	89	1647
0.05	0.59	59	1583
0.10	0.50	55	1508
0.20	0.52	51	1624
0.30	0.50	57	1583

Table III
Comparison of different Gaussian peak width (σ) values on average tracking performance.

σ	Avg. Overlap	Failures	FPS
0.5	0.51	55	1495
1.0	0.52	57	1684
1.5	0.51	50	1660
2.0	0.53	50	1658
2.5	0.52	50	1582
3.0	0.52	57	1542

Next, we evaluated the effect of different values for the Gaussian peak width parameter σ, with all other parameters fixed ($\alpha = 0.15$, $\lambda = 0.1$). The results, shown in Table III, demonstrate that the average overlap remains relatively consistent across all tested values of σ, suggesting that the tracker's ability to localize the target is not highly sensitive to this parameter. Similarly, the average speed (FPS) varies only slightly, indicating that σ does not significantly impact the computational load. However, a more notable pattern is observed in the number of failures. The fewest failures were recorded for σ values between 1.5 and 2.5, suggesting this range provides a good balance between a sharp response peak (for confident tracking) and robustness to noise or small displacements. Values outside this range may lead to either overly narrow or overly diffuse response maps, making tracking less stable.

We also experimented with constructing the correlation filter using a larger search region by increasing the enlargement parameter while having all other parameters fixed ($\alpha = 0.15$, ($\sigma = 2.0$, $\lambda = 0.1$). This parameter controls the size of the template F extracted around the target, effectively allowing the tracker to include more surrounding context or background information during filter learning.

The results in Table IV show that increasing the enlargement value leads to a noticeable degradation in performance. Specifically, the average overlap steadily decreases as the enlargement grows, likely due to the filter incorporating more background, which introduces noise and reduces localization precision. Additionally, the number of failures increases significantly with

Table IV
Comparison of different search area enlargement values on average tracking performance.

Enlargement	Avg. Overlap	Failures	FPS
1.0	0.53	50	1729
1.5	0.52	55	857.5
2.0	0.50	69	575.8
2.5	0.47	87	421.2
3.0	0.46	106	285.3

larger enlargement values. This is likely because a broader region introduces more distractors and potential for drift, especially when the target appearance is similar to the background. The average tracking speed (FPS) also drops dramatically, as processing larger regions requires more computation per frame.

Table V
Average initialization and per-frame tracking speed (in FPS) across individual sequences.

Sequence	Init. FPS	Tracking FPS
ball	963.3	2285
basketball	1086	746.2
bicycle	2648	3303
bolt	2521	1545
car	3214	3255
david	459.5	321.8
diving	894.3	709.7
drunk	1560	1060
fernando	378.2	278.3
fish1	2615	2969
fish2	1448	1145
gymnastics	1296	1082
hand1	1268	748.7
hand2	1740	1458
jogging	2313	1415
motocross	391.9	271.2
polarbear	1253	1024
skating	1700	1738
sphere	1440	934.7
sunshade	2241	2433
surfing	3933	3890
torus	2573	2249
trellis	1199	828.6
tunnel	1792	1528
woman	1524	1567

Lastly, we analyzed the tracking speed by measuring the time required to process individual frames. All other parameters were fixed to the same values used in the previous experiment ($\alpha = 0.15$, $\sigma = 2.0$, $\lambda = 0.1$), with enlargement value 1.

Table V shows the average initialization and tracking speeds (FPS) across all evaluated sequences. In most cases, the initialization phase is faster than the subsequent frame-by-frame tracking, although there are some exceptions. This behavior is expected because the initialization typically involves only a single filter computation and template extraction, whereas the tracking phase repeatedly applies correlation operations and may include additional steps such as scale estimation or failure handling. Notable exceptions, such as in the *ball* and *bicycle* sequences, may result from smaller object sizes or scene complexity affecting the initialization stage disproportionately.

III. Conclusion

We implemented and evaluated a correlation filter tracker on the VOT2014 dataset. The tracker achieved an average overlap of 0.53 and ran at over 1600 FPS. Parameter tuning showed that moderate values for learning rate and Gaussian width yield better robustness, while larger search areas reduce both accuracy and speed. These results confirm the efficiency of correlation filter trackers for real-time applications.

References

[1] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2544–2550.

[2] M. Kristan *et al.*, "The visual object tracking (VOT) challenge 2014," https://www.votchallenge.net/vot2014/, 2014.

[3] A. Lukežič, "Pytracking toolkit lite," https://github.com/alanlukezic/pytracking-toolkit-lite.