Exercise 3: Correlation filter tracking

Žan Pušenjak, *ACVM 2024/25, FRI, UL*

I. Correlation filter based tracker

We implemented a correlation filter based tracker and tried to make improvements. Further we tried to find the best set of tracker parameters to obtain the best possible performance.

We used the sequences from the VOT challenges[1] more specifically the VOT14 set of challenges that include of twenty five videos hand picked to test the reliability of trackers alongside with a simplified version of the VOT evaluation toolkit[2].

A. Basic implementation

We first implemented the most basic version of Correlation tracker that generates a filter from initially selected target region and then uses correlation score to find the best suitable location in the new frame. Each frame the correlation filter gets updated with the currently found template. Parameters of the basic implementation were:

- σ - since we used a Gaussian peak filter we needed to determine its σ parameter.
- α - the rate of filter update on each frame.

B. Improvements

We tried to extend the implementation and improve the results of our tracker in two ways.

C. Template size

To improve the tracker we introduced a inc_factor parameter that controls the size of the patch that gets filtered. This is necessary because the cosine window will remove the information on the edge of the patches so a larger patch could potentially capture more of the target while still getting rid of the background.

D. Parameter tuning

We used grid search to try and find the best possible set of parameters for our dataset. Because of computational reasons we used a limited set for possible values for each of the parameters. The results can be seen on Figure 2 and 1.

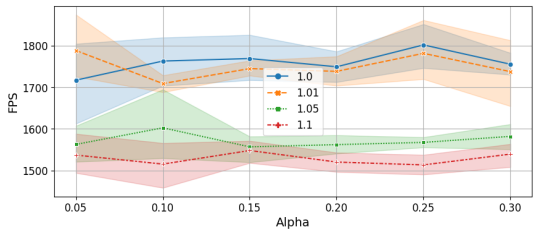
First commenting on the FPS rates. Figure 1 shows that predictably the main reason for a lower FPS is a larger patch size. As the inc_factor increases the FPS drops. Although the drops are not very significant as even with the largest inc_factor the FPS stays way beyond the required value for smooth tracking. Other parameter values

Observing Figure 2 we can conclude the set of parameters with the least fails. Looking at Figure 2a an optimal σ can be found. At all the different parameter settings the minimal number of fails is achieved with $\sigma = 1.5$ and $\sigma = 1.5$. Note that if σ is set to -1 it becomes adaptive and is dynamically set according to formula $\sigma = min(w, h) * 0.005$ where w and h are the width and height of the image. We already used this approach in optical flow computation and it again turned out to be a good parameter value that adapts to the input allowing user not to worry about finding an optimal σ parameter.

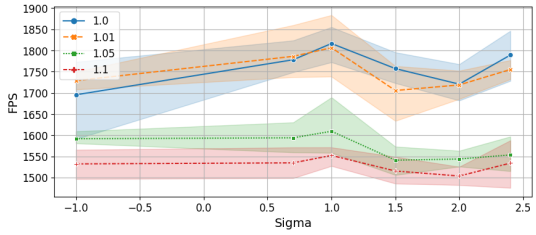
Additionally a bigger patch size increases the performance of the tracker as a larger patch outperforms smaller patches

[1] https://www.votchallenge.net/

[2] https://github.com/alanlukezic/pytracking-toolkit-lite

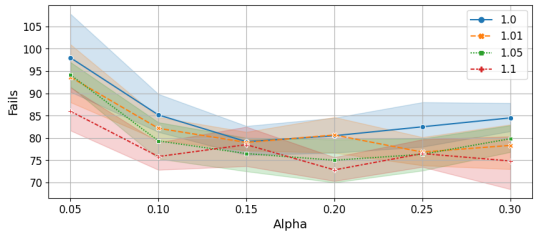


(a) Different σ values

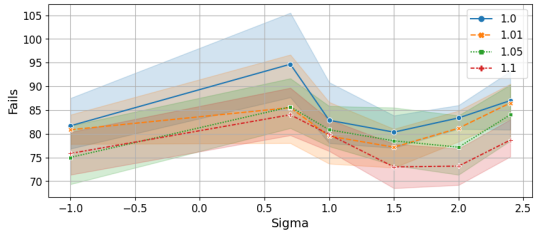


(b) Different N_{iter} values

Figure 1: Average FPS on VOT14 challenge at different parameters



(a) Different σ values



(b) Different N_{iter} values

Figure 2: Total fails on VOT14 challenge at different parameters

at almost every parameter. That can clearly be seen in both Figure 2a and Figure 2b.

Table I: Performance of the tracker according to the VOT toolkit.

Failures	σ	α	inc_factor
64	adaptive	0.2	1.05
64	2	0.3	1.1

After all the tests video sequences two sets of parameter combinations achieved **64 failures**. The parameter values are listed in Table I. We can observe that both combinations are

very similar.

II. Final evaluation

We evaluated the the final set of parameters with the toolkit. For some reason the sets of best parameters in Table I performed worse with the toolkit tester so we took into account the *uncertainty* of the parameter scores and came up with a new set of parameters $\sigma = 1.7$, $\alpha = 0.2$, $inc\_factor = 1.1$ that performed the best and achieved results seen in II.

Table II: Performance of the tracker according to the VOT toolkit.

Avg. overlap	Failures	FPS
0.48	74	1259