# Correlation filter tracking

Ivan Nikolov

## I. Introduction

For this third assignment, we implemented a simplified version of the MOSSE correlation filter. This algorithm learns a filter that gives high response for the template and low response for the background. The filter computation is done in Fourier domain and this gives additional performance boost.

## II. Experiments

We implemented the tracker by following the instructions given at the lectures and labs. Additional improvements were done with converting the input patch into logarithmic space and normalizing it such that it has a mean of 0, and standard deviation of 1. The implementation of the simplified MOSSE tracker was evaluated using the `pytracking-toolkit-lite` on the VOT2014 dataset.

For our implementation, we figured out that best results are achieved with a regularization factor $\lambda$ of 0.1, so kept it fixed to that value. We tried tuning the parameters $\alpha$ (update rate) and $\sigma$ (parameter of the Gaussian G).

| $\alpha$ | $\sigma$ | $\lambda$ | $SF$ | #Failures | AO | FPS |
|---|---|---|---|---|---|---|
| 0 | 0.5 | 0.1 | 1.1 | 240 | 0.47 | 1122.47 |
| 0.1 | 0.5 | 0.1 | 1.1 | 87 | 0.42 | 1118.94 |
| 0.2 | 0.5 | 0.1 | 1.1 | 75 | 0.42 | 1085.19 |
| 0.5 | 0.5 | 0.1 | 1.1 | 79 | 0.45 | 1104.31 |
| 0.9 | 0.5 | 0.1 | 1.1 | 87 | 0.45 | 1032.12 |
| 0.2 | 1 | 0.1 | 1.1 | 65 | 0.45 | 1041 |
| 0.2 | 2 | 0.1 | 1.1 | 52 | 0.48 | 1175 |
| 0.2 | 3 | 0.1 | 1.1 | 58 | 0.46 | 1046.68 |

Table I: Performance metrics on VOT2014 with different parameters

Due to time constraints, we did the search of optimal parameters in a 'greedy way' (start with inital configuration, change the values for one parameter and take the one that gives the best performance. After that, freeze that parameter and continue tuning the others). On Table I we can see the results under different parameter configurations. The best results we got were with failure number of 52, average overlap of 0.48 and 1175 FPS. Based on our experiments, can see that $\alpha$ values influence the performance more than $\sigma$. With an $\alpha$ value of 0, the algorithm uses only the last frame as a template for the object and due to the high variations in the objects' appearances there are more misses. Although $\sigma$ does not influence the performance of the algorithm in the same scale as the $\alpha$ parameter, choosing too small or too big values can degrade the performance. For our experiments, the best values for $\alpha$ and $\sigma$ were 0.2 and 2 accordingly.

Having chosen values for $\alpha$ and $\sigma$ we continue our experiments and try to find the best value for the scaling factor (SF).

| $\alpha$ | $\sigma$ | $\lambda$ | $SF$ | #Failures | AO | FPS |
|---|---|---|---|---|---|---|
| 0.2 | 2 | 0.1 | 1.0 | 62 | 0.47 | 1250.89 |
| 0.2 | 2 | 0.1 | 1.1 | 58 | 0.46 | 1046.68 |
| 0.2 | 2 | 0.1 | 1.5 | 68 | 0.46 | 562.77 |
| 0.2 | 2 | 0.1 | 1.8 | 69 | 0.48 | 568.77 |

Table II: Performance metrics on VOT2014 with scaling factor

On Table II we can see that best performance was achieved with SF of 1.1. Setting the scaling factor to larger values degrades the FPS performance. As larger image patches mean more operations for calculating the filters.

As always, the parameters depend on the use case. Using our best parameters for the entire dataset, we get 5 fails for the `bolt` sequence. With increasing the SF to 1.5, and $\sigma$ to 5 we get only 1 fail. This pattern of getting better results in fast moving templates with higher $\sigma$ and SF was observed in multiple videos.

## III. Conclusion

For this assignment, we implemented a discriminative correlation filtering method (a simplified version of the MOSSE algorithm). The evaluation and tuning of the parameters was done on the VOC2014 dataset, and best performing parameters had 52 failures in total. Further improvements can be done with fully implementing the MOSSE paper and adding scale estimation during tracking.