Exercise 5: Long-Term Tracking

Klemen Škrlj

I. Introduction

In this exercise we are working on implementation of long-term tracker based on a SiamFC short-term tracker[1]. We test different hyper parameter values and reason about the results. We also implement two different sampling techniques that our tracker can use when performing re-detection. At the end we visualize interesting results and comment on them. We evaluate tracker performance on 9 long-term sequences in total and report precision, recall and F1.

II. Experiments

In the first part of the exercise we look at deep CNN-based tracker called SiamFC[1]. Template for a tracked object is extracted by a pre-trained CNN model. In the update function of the tracker we get image patches of different sizes and push them through the same deep CNN model which transforms them into individual feature vectors. We then compare how similar each of them is to the template with correlation operation and select most similar one for our new object position. This is a short-term tracker since it cannot re-detect target after an occlusion appears or target leaves the scene for some duration. We run this version of the tracker on the whole test dataset and present results in Table I. Here precision, recall and F1 are used as performance metrics.

Seq. name	**Precision**	**Recall**	**F1**
car9	0.638	0.271	0.380
cat1	0.840	0.415	0.555
deer	0.718	0.200	0.312
dog	0.204	0.073	0.108
person14	0.670	0.038	0.073
person20	0.733	0.733	0.733
sitcom	0.535	0.501	0.518
skiing	0.554	0.119	0.197
sup	0.542	0.476	0.507
Overall	0.608	0.308	0.409

Table I
Results of SiamFC as short-term tracker

In the second part of the task we transform our short-term tracker into a long-term one. Now our tracker is capable of detecting when the target is lost and then it goes into re-detection mode. For failure detection we use knowledge from correlation response which acts as confidence level. If the maximum value in correlation map falls bellow some threshold it means that the target needs to be re-detected. To set this threshold value we test different values on "car9" sequence and pick the best. The results of various values are seen in Table II (here we use 50 random samples for re-detection). We see that with higher threshold values our tracker can't really re-detect since it is extremely hard to randomly get a point where the correlation would be that high. Opposite goes with too low threshold values where we are too quickly satisfied and we don't actually re-detect the object that we want but maybe something similar. For our sequence $thresh = 4$ seems to work the best.

Threshold	**Precision**	**Recall**	**F1**
10	0.267	0.026	0.048
6	0.617	0.522	0.565
4	0.603	0.597	0.600
2	0.634	0.269	0.378

Table II
Results at different confidence threshold values

In the re-detection phase we randomly pick N samples from the whole image and from each of them extract a patch at a set scale. Then each patch is compared to the template with correlation and the one with maximum response is picked. If this maximum value goes over our threshold, we go back to tracking mode else we continue trying to re-detect in the next frame. In Table III we show how does number of samples affect performance of the tracker (F1 score) and number of frames needed for re-detection (results are from sequence "car9" with threshold = 4).

# of samples	**F1**	**Avg # of re-detect frames**
10	0.598	10.0
30	0.596	8.6
50	0.600	4.4
70	0.592	7.3
100	0.596	4.6

Table III
Results at different number of random samples

Here we can see that with higher number of random samples we lower the amount frames used for re-detection. This is expected since we are more likely to randomly sample a point on the tracked object and produce higher confidence value than our threshold. But more sampled points also means higher computational requirements. This is why we pick $N = 50$ for the best value as higher value hit level of diminishing returns.

We also test how the results change if we sample points with Gaussian distribution around the last known position of the object. If we lost the object due to short occlusion there is high likelihood that it will appear near it's last known location. But if an object reappears on the other side of the image there is a very small possibility that we will be able to detect it again. So we have to have domain knowledge if we want to properly choose between these two options. In Table IV we show results of out tracker on "car9" sequence when using different number of samples picked based on Gauss sampling. We empirically pick 5000 as variance for covariance matrix.

# of samples	**F1**	**Avg # of re-detect frames**
10	0.597	7.33
30	0.597	7.33
50	0.595	4.2
70	0.597	4.0
100	0.597	4.0

Table IV
Results at different number of Gauss samples

Similarly to random sampling we can observe that average number of frames needed for re-detection drops when using more points. Additionally we also get lower amount of frames compared to random sampling method. One of the reasons why is that is because "car9" sequence doesn't include the problem of object reappearing in different part of the image. All re-detections happen around the tracked object so Gauss sampling is beneficial in this case.

In Table V we present results of our best long-term tracker where $threshold = 4$, $N = 50$ and sampling is uniform in $SiamFC_{uniform}$ and Gauss in $SiamFC_{gauss}$. When compared to short term tracker we see comparable precision score and greatly improved recall. This also means that there is a big improvement in F1 score. Comparing trackers with uniform and Gauss points sampling we see that the final results don't differ as much. The only improvement is in number of frames needed for re-detection as stated before.

Tracker Name	**Precision**	**Recall**	**F1**
$SiamFC_{short-term}$	0.638	0.271	0.380
$SaimFC_{uniform}$	0.603	0.597	0.600
$SaimFC_{gauss}$	0.600	0.594	0.597

Table V
Comparisons of best trackers on "car9"

Lastly we visualize interesting frames from our tracker. In Figure 1 we see how our tracker looses the tracked car since it goes under the sign and is not visible anymore. But then some frames later we see that our tracker is able to re-detect it by randomly sampling particles and trying to find a good enough match. This is seen in Figure 2.

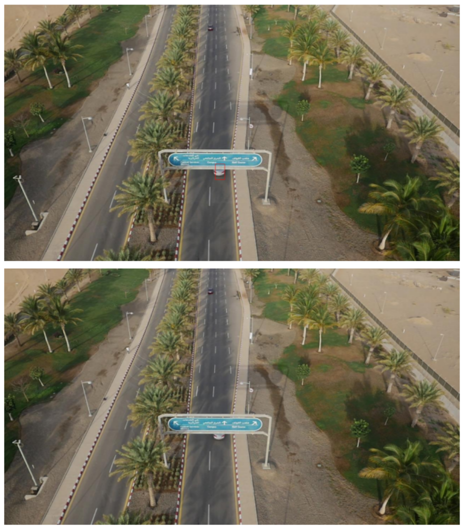

Figure 1. Tracker looses the tracked object

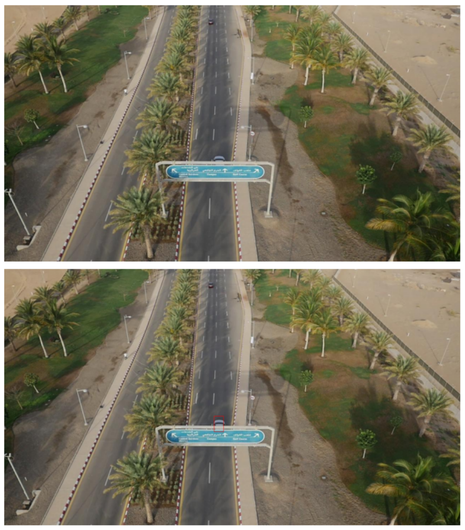

Figure 2. Tracker able to re-detect the car

III. Conclusion

We looked at basic deep CNN-based tracker SiamFC and transformed it into long-term tracker. We tested how different hyper parameter values affect the tracker performance and time needed for re-detection. We tested trackers on 9 sequences and reported results in tables and visualizations.

References

[1] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, “Fully-convolutional siamese networks for object tracking,” *CoRR*, vol. abs/1606.09549, 2016. [Online]. Available: http://arxiv.org/abs/1606.09549