Long term tracking using SiamFC

Anže Mihevc

I. Introduction

For this exercise we ran and evaluated the CNN-based short term tracker SiamFC. We then modified the tracker to add long term functionality of re-detecting the target once it is lost. We added some parameters for which we evaluated the tracker and found the optimal values. We ran both trackers on the ARNES supercomputer cluster, compared the findings and visualized the results.

II. Experiments

A. Short term tracking SiamFC

The first part of the assignment was running the SiamFC short term tracker with pre-trained weights on the long term dataset. The results can be seen in table I. The tracker performed as expected since it does not have the ability to re-detect the target once it disappears behind an occlusion. The precision score is much higher than recall however both are low resulting in a low F-score. The tracker takes a significant amount of time to run even on the ARNES supercomputer cluster.

Tracker	Precision	Recall	F-score
Short term	0.603	0.316	0.415

Table I: Performance of the short term SiamFC tracker on the whole dataset

B. Long term tracking SiamFC

For the second part of the assignment we modified the short term SiamFC tracker and added the ability for it to re-detect the target. This was done by implementing a confidence score based on a moving average and using it in combination with a threshold to determine if the target was lost. After the target was lost we started re-detection which generated possible new positions for the target in the next frame and checked the confidence score on them to see if the target was re-detected. The time to run the tracker only increased when implementing this additional computation and it would take too much time to run the tracker on the whole dataset when optimizing the different parameters even when using the ARNES supercomputer cluster. In majority of cases the tracker was run on only the car9 sequence unless stated otherwise.

We added this parameters to the tracker that effect the performance of the long term tracker

- **not_detected_threshold** threshold for detecting tracker fails
- **redetect_samples** number of samples for redetection
- **sample_method** method of sampling new positions for redetection
- **gaussian_std** standard deviation for Gaussian method

More information about each parameter can be seen in subsections bellow.

1) Confidence score and threshold: To define the confidence score we used a moving average calculated with each frame with the following formula:

$mov\_avg = \alpha * response + (1 - \alpha) * mov\_avg$

The value for alpha was 0.05. This allows us to not relay completely on only the previous response and have a more flexible confidence score that is not specific for one instance. In theory this would work better than a simple threshold of the response when running the tracker on the entire dataset. A threshold was then used on this average to determine if the target has been lost and later if it has been re-detected. We tested different values of the threshold when running the tracker on the car9 sequence as can be seen in table below II. In the table we can see precision, recall and F-score as well as the number of times the tracker lost the target and how many iterations it took for the tracker to find the target again. By this data the threshold of 0.8 is best. This means that when the max value of a response is lower than 80% of the average the target is considered as lost.

The lower thresholds would lock on to the wrong target after redirection and the higher thresholds would cause the tracker to constantly lose the target or not find it at all if the target changed shape during the sequence.

During optimization of the threshold the other parameters were set to optimal values.

Threshold	Precision	Recall	F-score	Num. of losses	Num. iter.
0.3	0.636	0.270	0.79	0	0
0.5	0.599	0.593	0.596	1	13
0.7	0.602	0.593	0.596	2	28,0
0.8	0.602	0.591	0.596	1	34
0.9	0.601	0.586	0.594	2	46,0

Table II: Different threshold effect on tracker

2) Number of sampled regions: The parameter number of sampled regions tells us how many new possible locations we generate each time we are re-detecting the target. This locations are then sampled as if they are the position of the target. Then their confidence score is evaluated against the threshold to determine if the target is re-detected. Two different values were tested(see tableIII) using both the Gaussian and the random method of sampling the new positions (more on them in section II-B3). The tracker preformed better with a higher number of sampled regions however this had an effect on the time it took to run the tracker. The tracker re-detected the target faster which improved the performance as less time was spent off target. We can see the better value is 20.

During optimization of the number of sample the other parameters were set to optimal values.

Method	Num. samples	Precision	Recall	F-score	Num. iters
Random	10	0.600	0.586	0.593	44
Random	20	0.601	0.588	0.594	40
Gaussian	10	0.600	0.589	0.595	35
Gaussian	20	0.602	0.591	0.597	34

Table III: Performance of different methods with different value of number of sampled regions parameter

3) Sampling method: Two different methods of sampling used during re-detection were implemented in the long term tracker. The random sampling method simply generates N number of samples randomly anywhere on the image using the ***np.random.uniform*** function and then process the samples. But since we know the location of the target before it is lost we can generate samples around that location as it is most likely that the target will re-emerge and be re-detected in that neighborhood. This is done when using the *"gaussian"* method of sampling. This method uses the ***np.random.normal*** function which is given the last known position of the target, the standard deviation and the number of samples. Empirically we determined that the best value for standard deviation is 20 but we also increment the standard deviation each consecutive frame that the target is not re-detected yet since the chance that the target moved significantly is higher as more time passes. Using the *"gaussian"* method of sampling greatly decreases the number of iterations it takes to re-detect the target as can be seen in table IV. Changing the method does not improve the precision, recall and F-score significantly however.

During optimization of the number of sample the other parameters were set to optimal values.

Method	Precision	Recall	F-score	Num. iters.
Random	0.601	0.588	0.594	40
Gaussian	0.602	0.591	0.597	34

Table IV: Difference between using random *sampling* and *"gaussian"* sampling

C. Disclaimer

It is important to note that the re-detecting is based on some randomness in both methods thus the results should be ran multiple times and averaged to obtain proper values however due to long run times and time constraints this was not feasible for this report. Additionally since the tracker was mostly ran on the ARNES supercomputer cluster using the slurm srun command it is not guaranteed that the same node is assigned to the process each time. As such time of calculation can differ significantly between runs.

D. Visualization of results

In image 1 we see the tracker tracking the target before it disappears behind the blue road sign. In image 2 we see the tracker lose the target but the new estimated position is close to the previous position of the target. This is because here the tracker is using the *"gaussian"* method of sampling the positions. In image 3 we can see the tracker using the *random* method of sampling and the new estimated position is far away from the target in the top right corner.

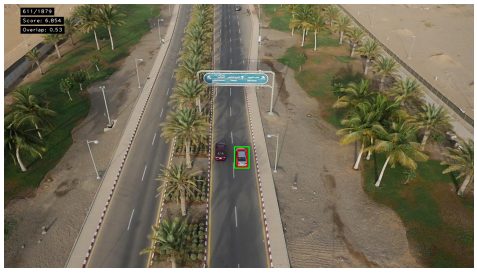

Figure 1: Image of the subject before the tracker losses it.

Figure 2: Top image is when the tracker losses the target and is using the *gaussian* method of re-sampling. Bottom image is when the tracker re-detects the target.

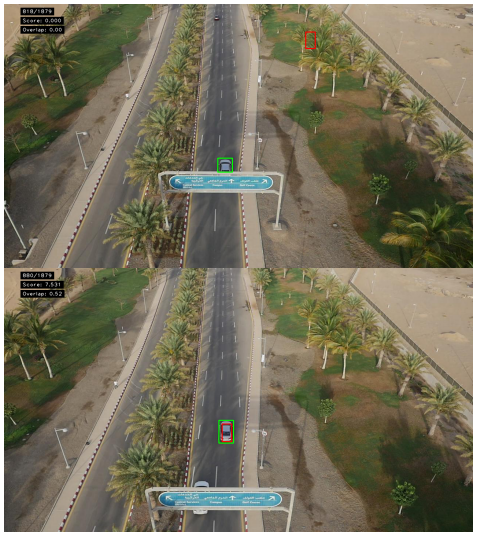

Figure 3: Top image is when the tracker losses the target and is using the *random* method of re-sampling. Bottom image is when the tracker re-detects the target.

III. Conclusion

We successfully adapted the short term SiamFC tracker into a long term tracker. The performance increase is substantial since the tracker can now re-detect the target after it is occluded. There is still room for more improvement as the tracker is quite slow. Better methods for re-detection could also be implemented.