Assignment 3 report - correlation tracking

Luka Boljević

I. Introduction

Object tracking is an important task in computer vision that has many practical applications, such as surveillance, autonomous driving, and human-computer interaction. The goal of object tracking is to locate and follow a target object in a sequence of images or video frames, despite variations in its appearance and motion. In this assignment, we will implement and test a simple version of MOSSE correlation tracker [1]. Different to eg. mean shift tracker, where we update the object representation, correlation trackers update i.e. learn a filter, so that it gives a high correlation response on the tracked object. We will test the correlation tracker on sequences from VOT (Visual Object Tracking) 2014 challenge [2], and evaluate it in terms of total number of failures, and average overlap between predicted and ground truth bounding boxes.

II. Experiments

For testing, the lite version of the toolkit linked in assignment instructions was used. We added some of our code to the toolkit to make testing various hyperparameters of the correlation tracker easier. This allowed us to test a lot of trackers (120 of them to be exact) quite easily. Of course, the original functionality of the toolkit is completely preserved.

As stated, the tracker is tested on VOT 2014 sequences. The tracker has 4 hyperparameters: (1) enlarge factor e, which is used to search the target in a larger region if $e > 1$, (2) σ, used for creating the Gaussian peak, i.e. ideal correlation response $\mathbf{G}$, (3) α, update speed for filter $\mathbf{H}$, and lastly (4) λ, a small value which should control the magnitude of values in filter $\mathbf{H}$ are. Before going into any tests, we should mention that we also played around with λ, however the improvements were insignificant, if any. We decided to stick with $\lambda = 0$ as that is the very first value we used.

To begin with, we will set $e = 1$ (and $\lambda = 0$). In that case, Table I show the two best trackers, one with lowest failure rate, and another with highest average overlap.

	Avg. overlap	Num. failures	Avg. speed (FPS)
Least failures	0.4273	75	996
Highest overlap[1]	0.4562	84	1011

Table I
Two correlation trackers, one with lowest failure rate, and highest average overlap, when $\lambda = 0$ and $e = 1$. Hyperparameters of tracker with lowest failure rate were $\sigma = 1$ and $\alpha = 0.1$, while hyperparameters of the one with highest average overlap were $\sigma = 2$ and $\alpha = 0.15$. Average speed is measured in FPS.

A. *Varying α and σ*

We will now show how varying σ and α affects tracking performance, again in terms of number of failures and average overlap. Our baseline tracker will be the one with lowest failure rate from Table I. Table II shows the effect of changing σ, while Table III shows how varying α affects tracking performance.

[1]The tracker with actual highest overlap, 0.4606, was the tracker with $\sigma = 0.5$ and $\alpha = 0.02$. However, it had 178 total failures, which is worse than NCC tracker, so we decided to exclude it.

σ	Avg. overlap	Num. failures	Avg. speed (FPS)
0.5	0.4111	115	983
1	0.4273	**75**	996
2	0.4451	86	1018
3	**0.4463**	101	1041
4	0.4346	101	1025
5	0.4142	108	1017

Table II
How varying σ affects tracking performance. Other hyperparameters used are $\alpha = 0.1$, $e = 1$ and $\lambda = 0$, as our baseline tracker is the lowest failure one from Table I. Row with $\sigma = 1$ is exactly the baseline tracker.

α	Avg. overlap	Num. failures	Avg. speed (FPS)
0.02	0.4390	136	1002
0.05	**0.4424**	95	991
0.1	0.4273	**75**	996
0.15	0.4298	78	1029
0.2	0.4411	76	983

Table III
How varying α affects tracking performance. Other hyperparameters used are $\sigma = 1$, $e = 1$ and $\lambda = 0$, as our baseline tracker is the lowest failure one from Table I. Row with $\alpha = 0.1$ is exactly the baseline tracker.

From Table II, we see that if σ is too low (eg. 0.5), or too high (eg. 5), tracking performance is degraded, particularly in terms of number of failures. This simply means that the created Gaussian peak $\mathbf{G}$ is too narrow i.e. too wide, and the created filter $\mathbf{H}$ can't distinguish the target well enough. Optimal σ, which strikes a good balance between average overlap and number of failures, is 1 or 2, or perhaps a value in between.

Varying α in Table III, we noticed that setting it to a rather high value, like 0.1 and 0.2 (meaning that each frame we update $\mathbf{H}$ by 10 or 20%), benefited tracking performance quite significantly. This is a bit different than the mean shift tracker from the previous assignment, where optimal update speed α was usually 0, 0.001 or *at most* 0.05, for one VOT2014 sequence. Of course, it is expected that not updating $\mathbf{H}$ would result in worse tracking, but it's interesting to see that correlation tracker needs larger update speeds for good tracking. From trackers in Table III, perhaps the overall best is the very last one, where $\alpha = 0.2$, even though it doesn't have the absolute highest overlap, or lowest failure rate.

B. *Varying enlarge factor e*

One downside of all previous trackers is that they fail quite often when our target is moving a lot. This should be improved when we increase the search region, i.e. when we set a higher enlarge factor e. Indeed, we found that the tracker with overall lowest failure rate was a tracker that used hyperparameters $e = 1.5$, $\sigma = 2$, $\alpha = 0.15$, $\lambda = 0$. It had 64 total failures, while still having decent average overlap of 0.4688. We will use that as the baseline tracker to show how varying the enlarge factor e affects performance. Results are shown in Table IV.

First thing we notice in Table IV is that the larger the search region, the slower the tracker is, which is expected -

e	Avg. overlap	Num. failures	Avg. speed (FPS)
1	0.4562	84	**1011**
1.5	**0.4688**	**64**	648
2	0.4649	81	332
2.5	0.4606	94	259
3	0.4453	109	227

Table IV
How varying e affects tracking performance. Other hyperparameters used are $\sigma = 2$, $\alpha = 0.15$, $\lambda = 0$. Tracker in row 2 had the lowest failure rate out of 120 trackers we tested, while still having decent average overlap.

the Gaussian peak $\mathbf{G}$, filter $\mathbf{H}$, cosine window, and extracted image patches are all bigger, hence making the computations slower. Previously, when $e = 1$, speed wasn't really an issue, so we didn't have any particular comments in that case.

While results are not explicitly shown here, the top 10 trackers with lowest failure rate all use $e = 1.5$ or $e = 2$, so is it pretty much mandatory to use a bigger search region. All 10 of those trackers have a lower failure rate and higher average overlap (except two) than the trackers from Table I, where we had $e = 1$.

C. Tracker initialization speed

The last thing we need to cover is tracker speed. We explained that for $e > 1$, the tracker is significantly slower, while for $e = 1$ there are no such problems. Other hyperparameters do not affect speed as drastically as enlarge factor e. However, we only showed the average tracking speed on all frames (across all sequences). Under the hood, a significant amount of computation is done while initializing the tracker, so we want to see if average speed on initialization frames drastically differs from average speed on all frames.

e	Avg. speed (FPS)	Avg. speed (FPS)
1	1011	1019
1.5	648	618
2	332	350
2.5	259	267
3	227	211

Table V
Difference between average speed on all frames, and average speed on frames where tracker is initialized, on all sequences. Other hyperparameters used are the same ones as in Table IV.

We can see in Table V that, as e increases, average initialization frame speed drops by around the same amount as average speed on all frames. Likewise, there isn't really a pattern, where we can say one is always bigger or smaller than the other. Similar thing holds when we look at sequences themselves, as seen in Table VI - more often than not, initialization is faster than the tracker on average (16/25 sequences), but not always.

III. Conclusion

In this report, we implemented a simplified version of MOSSE correlation tracker. We tested it on VOT2014 sequences. We showed how varying its hyperparameters, namely α, σ and enlarge factor e, affected performance. We showed what the overall best tracker we found was, and wrote about the importance of using enlarge factor $e > 1$. However, we also saw that the tracker is slower when $e > 1$. We lastly discussed

sequence	Avg. speed (FPS)	Avg. init frame speed (FPS)
ball	831	**859**
basketball	381	**440**
bicycle	**1225**	1148
bolt	403	**583**
car	**834**	575
david	311	**383**
diving	336	**385**
drunk	265	**285**
fernando	163	**197**
fish1	1116	**1290**
fish2	518	**527**
gymnastics	261	**393**
hand1	418	**560**
hand2	641	**818**
jogging	681	**833**
motocross	**132**	117
polarbear	**995**	531
skating	**564**	454
sphere	382	**427**
sunshade	**1588**	1249
surfing	**2033**	1130
torus	648	**798**
trellis	222	**377**
tunnel	**585**	510
woman	**659**	584

Table VI
Comparing average speeds on all VOT2014 sequences. Hyperparameters used were $e = 1.5$, $\sigma = 2$, $\alpha = 0.15$ and $\lambda = 0$. In each row, we bolded the larger speed, for easier distinction.

that average tracker initialization speed is about the same as average tracker speed on all frames. The tracker performs quite well, but it is not without downsides. For example, even with larger search regions, it tends to fail when the target moves a lot between frames, or when the target blends with the background a lot (examples being fish1 and fish2 sequences from VOT2014).

References

[1] Bolme, David S. and Beveridge, J. Ross and Draper, Bruce A. and Lui, Yui Man, "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, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2137–2155, Nov 2016.