Exercise 3: Correlation filter Tracking

Klemen Škrlj

I. Introduction

In this exercise we are working on implementation of correlation filter tracker. We evaluate different hyper parameter values and look at how they affect average overlap in number of failures on the VOT14 dataset. To allow tracker to take bigger search region into account, we implement enlargement factor to our CF filter and compare how it performs compared to the original one. We also look at the speed of both implementations. Additionally we implement and evaluate MOSSE tracker[1], which adds some tricks to improve overall performance.

II. Experiments

In the first part we implement basic correlation filter tracker and test it on a VOT14 dataset using VOT toolkit-lite. The results for each individual sequence are shown in Table II. Here we use $\sigma = 2$, $\alpha = 0.125$ and $\lambda = 1e-3$ for our hyper parameters values. We can see that sequence "hand2" seems to be the most challenging for the tracker which is due to quite fast motion and fast changing hand shape. Changes in the searched objects shape are also present in sequence "fish2" which is problematic as well. On average over whole dataset we get overlap of 45% and 79 failures in total.

Seq. name	**#Frames**	**Overlap**	**#Failures**
ball	602	0.368	6
basketball	725	0.431	4
bicycle	271	0.426	2
bolt	350	0.462	7
car	252	0.403	0
david	770	0.674	0
diving	219	0.291	1
drunk	1210	0.252	0
fernando	292	0.292	2
fish1	436	0.337	11
fish2	310	0.292	7
gymnastics	207	0.582	2
hand1	244	0.425	5
hand2	267	0.333	11
jogging	307	0.568	2
motocross	164	0.462	1
polarbear	371	0.441	0
skating	400	0.4	0
sphere	201	0.53	4
sunshade	172	0.701	4
surfing	282	0.637	1
torus	264	0.42	4
trellis	569	0.539	2
tunnel	731	0.306	0
woman	597	0.545	3

Table I
Results of basic CF tracker on each sequence

For the best possible performance we have to tune our hyper parameter values. In Table II we can see how does average overlap and number of total failures change depending on α and σ values. For α we got the best results at values around 0.1 to 0.15, so we choose 0.125 for our final value. Lower values didn't work as well since tracked objects changed their shape over time and correlation with the the stored filter wasn't as good.

After that we compare different σ value for Gaussian and find out that $\sigma = 2$ works the best in our case.

σ	α	**Average overlap**	**#Total failures**
0.01	2	0.45	125
0.05	2	0.47	94
0.1	2	0.43	79
0.125	**2**	**0.45**	**79**
0.15	2	0.45	82
0.125	0.5	0.43	103
0.125	1	0.44	80
0.125	1.5	0.44	79
0.125	2.5	0.44	84

Table II
Results with different hyper parameter values

We then make an improvement to the basic tracker by enlarging the area taken in to account by the filter. By doing this, we make our search region bigger and allow our tracker to be more resilient to quick changes. Since enlarging factor is another hyper parameter, we test different values for it as shown in Table III (here $sigma = 2$ and $alpha = 0.125$ are used). We observe that by enlarging the area we mainly improve in the number of failures, which is expected. We get the best ratio between average overlap and number of failures on the whole dataset with enlargement factor of 1.75.

Enlarge factor	**Average overlap**	**#Total failures**
1.25	0.45	69
1.5	0.45	68
1.75	**0.47**	**66**
2	0.47	79

Table III
Results with different enlargement factor

Table IV shows comparison between average FPS, FPS for initial frames and FPS for tracked frames between our basic filter and filter when using enlarge factor. As we can see, when we use enlargement factor we process approximately less than half frames per second compared to basic tracker. This is due to larger image patches and subsequently larger filter and Gauss response which take more computational power to process. When comparing time needed for processing initialization frames to other frames we don't see big difference. In initialization we are precomputing cosine window and Gauss response, while in track function we extract two patches and each time compute new correlation response. It seems like at the end complexity of this operations evens out between functions and are both pretty similar on average.

	Basic	**With enlarge**
Average FPS	723.04	310.59
Init frames FPS	765.88	300.11
Tracked frames FPS	721.97	310.34

Table IV
Comparison in speed

When looking at average FPS per sequence in Table V, we can see that they are not the same for every sequence. This is due to different size of tracked objects (ground truth) since some of them are much bigger (e.q. "motocross" sequence) than other (e.q. "fish1"). This means that the patch and filters are bigger as well which leads to more computationally heavy calculations. Comparison between trackers and their speed per sequence show that they are proportional to one another.

Seq. name	**#Frames**	**Basic FPS**	**Enlarge FPS**
ball	602	652.31	442.12
basketball	725	312.38	134.83
bicycle	271	2053.52	570.5
bolt	350	662.42	270.24
car	252	1194.4	1070.8
david	770	198.91	193.32
diving	219	385.32	150.52
drunk	1210	316.39	147.09
fernando	292	179.51	72.159
fish1	436	1014.96	430.43
fish2	310	603.78	213.76
gymnastics	207	579.31	140.57
hand1	244	604.85	293.91
hand2	267	767.22	257.03
jogging	307	972.88	412.78
motocross	164	131.02	52.013
polarbear	371	501.84	250.19
skating	400	592.84	247.45
sphere	201	763.92	141.69
sunshade	172	979.97	432.56
surfing	282	1351.19	709.28
torus	264	823.74	428.5
trellis	569	956.27	297.07
tunnel	731	680.04	148.16
woman	597	796.99	257.56

Table V
Comparison in speed per sequence

Lastly we implement MOSSE tracker[1] which is similar to the one that we used beforehand with some notable differences. Instead of storing and updating the whole filter at once, we update numerator and denominator separately. In addition we also perform 8 random perturbations on the extracted patch in the initialization which makes our initial filter representation more robust to changes. We do this by rotating original patch by a random angle between (-10,10) degrees. Table VI shows results for this algorithem that we get when using the following hyper parameter values: $\sigma = 2$, $\alpha = 0.15$ and enlarge factor $=$ 1.75.

Average overlap	0.47
#Total failures	58
Average FPS	313.09
Init frames FPS	96.09
Tracked frames FPS	320.81

Table VI
Results for MOSSE tracker

With this implementation we mainly see improvements in the number of total failures which went from 66 in CF tracker with enlargement to 58 in MOSSE tracker while the average overlap stayed the same. For major improvements in average overlap we would need to implement a tracker, which can change scale depending on the searched object size. When we compare the speed of the tracker, we see that there isn't much difference with respect to improved CF tracker. We pay a price at the initialization, when we perform the random perturbations, but because of this we also don't have as many failures and don't have to reinitialize often. The complexity of updating the filter as a whole versus storing numerator and denominator separately isn't so different so we don't see big difference in speed on tracked frames.

When comparing number of failures per sequence seen in Table VII, we see that we have less misses on "fernando", "fish1", "hand1", "motocross", "skating" and "trellis" sequences which all include significant changes in shape and benefit from initial random perturbations. But there are still challenging sequences like "hand2" where MOSSE doesn't have any improvements and "bolt" where we see additional miss compared to more basic CF tracker.

Seq. name	**$CF_{enlarge}$ #Failures**	**MOSSE #Failures**
ball	2	2
basketball	2	2
bicycle	1	1
bolt	3	4
car	0	0
david	0	0
diving	1	1
drunk	0	0
fernando	3	1
fish1	13	11
fish2	8	8
gymnastics	3	3
hand1	6	4
hand2	12	12
jogging	1	1
motocross	3	2
polarbear	0	0
skating	2	1
sphere	0	0
sunshade	0	0
surfing	0	0
torus	4	4
trellis	1	0
tunnel	0	0
woman	1	1

Table VII
Comparison in number of failures between MOSSE and $CF_{enlarge}$

III. Conclusion

We looked at how different improvements to the basic correlation filter affect overall performance of the tracker. We see that taking bigger area into account results in less failures but we pay a price in tracker's speed. Also some additional tricks presented in MOSSE algorithm improve performance and make object filter representation more robust to changes.

References

[1] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2544–2550.