Exercise 3: Correlation filter tracking

Sara Bizjak

I. Introduction

Tracking various objects can play a huge role in many different fields – from tracking parts in the automotive industry, to tracking people on sporting events to perform personal analysis. One such tracking method is called correlation filter tracking, which is in fact simplified version of MOSSE tracker, and is about to be implemented and integrated to the VOT evaluation framework, more specifically, VOT2014.

II. Experiments

After the implementation we test how our algorithm performs by integrating the implemented tracker to the VOT2014 dataset using `pytracking-toolkit-lite`.

Before reporting the results and showing tracker performance analysis, i. e. in terms of number of failures and average overlap, we need to carefully choose the right combination of parameters, which will perform, in general, the best possible. Obtained total/averaged results, together with choosen parameters for all sequences, are shown in the Table I. Additionally, some result for individual sequences are shown in Table II.

Parameters:		Results for tracker:	
α	0.1	Total failures	64
σ	2	Average failures	2.56
`increase factor`	1.2	Average overlap	0.42
λ	0.001	Average speed [FPS]	509.63

Table I
Obtained total/averaged results after integration of tracker on VOT2014 sequences, together with choosen parameters.

VOT2014 seq.	Length	Failures	Overlap	Speed [FPS]
basketball	725	2	0.37	361.80
bolt	350	6	0.43	750.51
drunk	1210	0	0.24	214.16
gymnastics	207	2	0.48	424.61
hand1	244	5	0.42	518.31
jogging	307	1	0.64	761.65

Table II
Some individual results after integration of tracker on some of the VOT2014 sequences.

Note that when choosing appropriate parameters, we are looking for a trade-off between number of failures and average overlap – we want to minimize the number of failures and maximize the overlap percentage. Although in some cases number of failers can drop also below 50, the average overlap is quite poor (only around 20%). We thus choose not to use those parameter for further analysis.

Now we show how the tracking performance changes with calibrating parameters α, σ and `increase factor`. It is worth mentioning that calibrating the parameter λ does not have an impact on performance in terms of failures and overlap and is in fact pointless, since it is used only to avoid dividing by 0.

Firstly, we calibrate the parameter α, while others remain as selected and shown in the Table I. Calibration analysis is presented in the Table III and on Figure 1 with the best option for α highlighted.

Parameter: α	Total failures	Avg overlap	Avg speed [FPS]
0.0	179	0.42	512.2
0.001	172	0.42	522.99
0.01	119	0.39	562.52
0.02	89	0.41	527.50
0.05	77	0.4	565.21
0.08	72	0.41	559.66
0.1	64	0.42	509.63
0.15	68	0.42	542.18
0.2	74	0.42	550.21
0.5	69	0.41	511.35
0.9	78	0.42	503.93

Table III
Obtained results when calibrating parameter α.

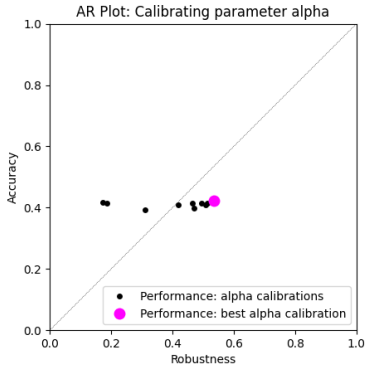

Figure 1. Visualization of α calibration analysis.

From the visualization, we can quickly notice that calibrating parameter α mostly affects the robustness, since the accuracy does not change much. Thus, with properly selected α we can indeed increase the robustness of the tracker performance.

Secondly, we calibrate the parameter σ, while others remain as selected and shown in the Table I. Calibration analysis is presented in the Table IV and on Figure 2 with the best option for σ highlighted.

Parameter: σ	Total failures	Avg overlap	Avg speed [FPS]
0.0	128	0.33	520.24
0.01	112	0.35	509.17
0.25	117	0.35	535.09
0.5	89	0.38	488.44
0.75	76	0.40	469.59
1.0	71	0.39	546.07
1.5	66	0.39	542.23
1.75	64	0.41	547.49
2	64	0.42	509.63
2.25	69	0.43	549.66
2.5	67	0.43	498.00
3	66	0.43	412.75
4	81	0.41	464.53

Table IV
Obtained results when calibrating parameter σ.

Unlike with calibrating α, here we can see some improvement in accuracy as well. So with calibrating σ we can improve robustness as well as accuracy.

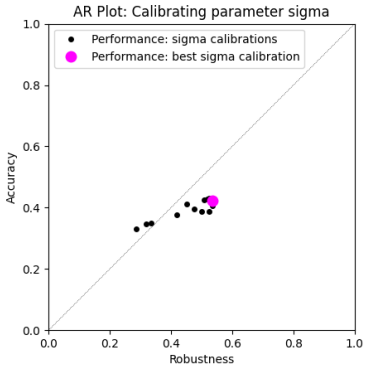

Figure 2. Visualization of σ calibration analysis.

Lastly, we calibrate the increase factor, i. e. constructing a filter using larger template by capturing more background, while others remain as selected and shown in the Table I. Calibration analysis is presented in the Table V and on Figure 3 with the best option for `increase factor` highlighted.

Parameter: `increase factor`	Total failures	Avg overlap	Avg speed [FPS]
1	79	0.45	601.86
1.2	64	0.42	509.63
1.5	62	0.34	372.27
1.75	55	0.27	297.72
2	47	0.23	258.34
2.5	52	0.16	210.89

Table V
OBTAINED RESULTS WHEN CALIBRATING PARAMETER `INCREASE FACTOR`.

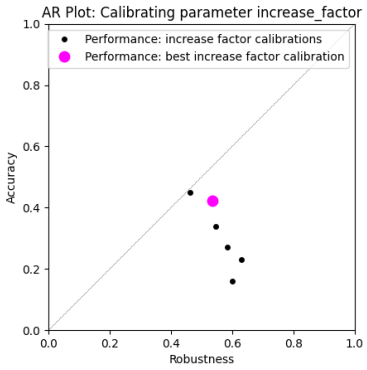

Figure 3. Visualization of `increase factor` calibration analysis.

We can quickly notice that with larger increase factor, failure count gets, in general, smaller, however, average overlap becomes too poor. To compromise, we thus choose increasing factor bigger than 1 (which means that we do construct a larger template) that is, on the other hand, not too enlarged. The choice of 1.2 is in that case the appropriate one. Additionally, we see that with calibrating increase factor we are mostly improving the accuracy and not so much robustness. We can also see that tracker gets slower as increase factor is getting larger.

Tracing speed was reported in all previous analysis (more specifically, speed for some equences was included in Table II), however, we now measure speed separately for initialization step and tracker updates, which can be seen in Table VI.

VOT2014 sequence	Avg initialization speed [FPS]	Avg tracking speed [FPS]
basketball	340	362
bolt	573	756
drunk	261	214
gymnastics	390	425
hand1	642	516
jogging	1095	760

Table VI
MEASURED AVERAGE SPEEDS [FPS] FOR INITIALIZATION AND TRACKING SEPARATELY ON SOME OF THE VOT2014 SEQUENCES.

We see that in most cases, initialization is faster than tracing (since [FPS] number is bigger), while in other cases when this is not true, both speeds are very similar. The only "outlier" is sequence `bolt`, where initialization is quite slower than the tracing itself. This could be the consequence of tracker causing more failures, and initialization after a failure may be slower due to the faster movement of tracked object.

III. CONCLUSION

As a part of the exercise, we implemented a simplified version of MOSSE tracker, based on the correlation filters. The tracker was tested on several sequences, on which a set of paramaters was set, such that they are overall a good choice. However, to achieve even better performance, we must set parameters separately for each sequence. When calibrating parameters, we saw that parameter α mostly affected the number of total failures, while with calibrating σ we improved both total number of failures and overlap. With `increase factor` we needed to be more careful, since improving only total number of failures could lead to a bad result in terms of overlap, thus reaching a great trade-off played a huge role here. Even more, `increase factor` had the biggest affect on speed measurements, which was expected, since the tracking computational complexity depends on the size of an image patch. We then, additionally, separately measured the speeds for initialization and tracking process for a few sequences. In general, the average initialization steps are faster than tracking, however, we also have some exceptions. In general, the tracker yields satisfactory results, however, it could perform even better with some corrections in filter updating step (i. e. real MOSSE tracker implementation).