# Correlation filter

Aljaž Konec

## I. Introduction

Correlation filters like the MOSSE filter match a template to an image by computing the correlation between the template a local search region. Since correlation is computationally expensive, we transform the the images into the frequency domain using the Fourier transform. This transforms correlation into element-wise multiplication and thus speeding up the computation. In this report we showcase a simplified version of the MOSSE filter and test it on the VOT2014 dataset.

## II. Implementation

The implementation of the correlation filter was based on the original paper by Bolme et al.[1]. To improve performance, before FFT we preprocess the image by transforming pixel values by a log function, normalizing the mean to 0, setting norm to 1 and then multiplying by cosine kernel. To test the filter, we integrated the filter into the VOT toolkit and tested it on the VOT2014 dataset. The table I shows results using a kernel size $\sigma = 2$, learning rate $\alpha = 0.125$, regularization $\lambda = 0.7$ and local search region the same size as target. Figure

| Sequence Name | Sequence Length | Overlap | Failures | Speed |
|---|---|---|---|---|
| ball | 602 | 0.451 | 3 | 1773 |
| basketball | 725 | 0.656 | 5 | 761 |
| bicycle | 271 | 0.435 | | 3023 |
| bolt | 350 | 0.622 | 1 | 1085 |
| car | 252 | 0.539 | 1 | 1132 |
| david | 770 | 0.603 | | 1011 |
| diving | 219 | 0.385 | | 992 |
| drunk | 1210 | 0.382 | | 468 |
| fernando | 292 | 0.344 | 3 | 310 |
| fish1 | 436 | 0.380 | 6 | 504 |
| fish2 | 310 | 0.353 | 5 | 680 |
| gymnastics | 207 | 0.607 | 3 | 1074 |
| hand1 | 244 | 0.389 | 3 | 1118 |
| hand2 | 267 | 0.467 | 10 | 1639 |
| jogging | 307 | 0.697 | 1 | 646 |
| motocross | 164 | 0.557 | 2 | 272 |
| polarbear | 371 | 0.461 | | 750 |
| skating | 400 | 0.402 | | 1223 |
| sphere | 201 | 0.557 | 1 | 562 |
| sunshade | 172 | 0.704 | 3 | 822 |
| surfing | 282 | 0.601 | | 2805 |
| torus | 264 | 0.584 | 5 | 1775 |
| trellis | 569 | 0.566 | | 1048 |
| tunnel | 731 | 0.319 | | 683 |
| woman | 597 | 0.677 | 1 | 999 |
| **Average** | **409** | **0.510** | **53** | **1086** |

Table I: VOT toolkit integration test results.

1 shows the output of running the filter using the VOT toolkit.

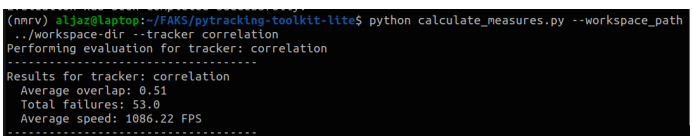


Figure 1: Output of the filter using the VOT toolkit.

## III. Changing kernel size and update rate

As the target moves in the images the filter needs to adapt to the changes. For this reason we update filter $H$ by some update rate $\alpha$. Table II shows the results of changing only the update rate $\alpha$ and keeping all other parameters the same as in the previous section. As can be seen, if the update rate is to small ( 1- 2 % ) the filter fails to adapt to the changes in the target and thus produces worse results. A similar effect canbe seen if the update rate is to high ( 20 - 30 % ) as the filter starts to overfit the target and thus fails to generalize to new images, but the performance is still better than with a low update rate.

| Alpha | Failures | FPS | Overlap |
|---|---|---|---|
| 0.01 | 94 | 1381 | **0.489** |
| 0.02 | 70 | 1435 | 0.47 |
| 0.05 | 64 | 1398 | 0.48 |
| 0.1 | **53** | 1377 | 0.472 |
| 0.125 | 57 | **1481** | 0.469 |
| 0.2 | 57 | 1463 | 0.453 |
| 0.3 | 60 | 1366 | 0.454 |

Table II: Performance of different update rates $\alpha$ while all other parameters are static.

The kernel size $\sigma$ determines the size of the gaussian kernel which determines the probabilities of the target being in a certain location. Table III shows the results of changing the kernel size $\sigma$ and keeping all other parameters the same as in the previous section. When multiplying the feature patch with a gaussian that has a small $\sigma$ produces a filter that only works well on targets that do not move very fast. This leads to overall lower robustness and more failures. Overlap follows a similar trend of increasing with the kernelsize, with a slight drop for $\sigma = 10$. The results show that on the VOT2014 dataset the best overall results are achieved with $\sigma = 2$ and $\sigma = 10$.

| Sigma | Failures | FPS | Overlap |
|---|---|---|---|
| 1 | 82 | 1302 | 0.468 |
| 2 | 53 | 1318 | 0.472 |
| 3 | 60 | 1373 | 0.483 |
| 4 | 59 | 1453 | **0.496** |
| 5 | 60 | **1510** | 0.492 |
| 10 | **49** | 1411 | 0.483 |

Table III: Performance of different sigmas.

## IV. Changing template size F

Extracting a larger template allows the tracker to track the target even if its movement is fast and sudden. This is at the cost of the filter learning more of the background and thus being less robust. To test the effect of larger template size we tested different window magnifications. Table IV shows the results of multiplying the initialization true window size by different factors. As mentioned, too large of an enlargement of the template size introduces more of the backgorund into the filter $H$, making the discrimination between backgorund and target worse. This is why the number of failures increases and the FPS decreases as the template size increases. A slight increase in the template size does reduce the number of failures by 1.

| Window size multiplier | Failures | FPS | Overlap |
|---|---|---|---|
| 1.0 | 53 | **1385** | 0.472 |
| 1.1 | **52** | 1200 | 0.462 |
| 1.2 | 61 | 1152 | 0.471 |
| 1.5 | 59 | 793 | 0.493 |
| 2.0 | 63 | 524 | **0.527** |

Table IV: Performance of template size manipulations.

## V. Testing combinations of parameters

To perform a more thorough analysis we tested the effect of changing multiple parameters at once. The Table V shows results of performning a gridsearch on the two best values of each parameter. We can observe that the FPS for all of the combinations does not deviate intensly and is very high with an average of 1324 FPS. The best performing combination of parameters is $\sigma = 10$, $\alpha = 0.125$, $\lambda = 0.6$ and window size multiplier 1.1. This combination produces only 44 failures which is 30% better than the baseline of 62 failures while only decreasing the overlap by 1%. For all the tested combinations the overlap stays is in the range of 0.45 to 0.5.

| $\sigma$ | $\alpha$ | $\lambda$ | **Window** | **Failures** | **FPS** | **Overlap** |
|---|---|---|---|---|---|---|
| 2 | 0.1 | 0.6 | 1 | 56 | 1399 | 0.474 |
| 2 | 0.1 | 0.6 | 1.1 | 51 | 1245 | 0.466 |
| 2 | 0.1 | 0.7 | 1 | 53 | 1371 | 0.472 |
| 2 | 0.1 | 0.7 | 1.1 | 52 | 1212 | 0.462 |
| 2 | 0.125 | 0.6 | 1 | 58 | 1490 | 0.469 |
| 2 | 0.125 | 0.6 | 1.1 | 53 | 1249 | 0.468 |
| 2 | 0.125 | 0.7 | 1 | 57 | 1440 | 0.469 |
| 2 | 0.125 | 0.7 | 1.1 | 55 | 1242 | 0.468 |
| 2 | 0.2 | 0.6 | 1 | 58 | 1407 | 0.458 |
| 2 | 0.2 | 0.6 | 1.1 | 54 | 1243 | 0.475 |
| 2 | 0.2 | 0.7 | 1 | 57 | 1516 | 0.453 |
| 2 | 0.2 | 0.7 | 1.1 | 56 | 1351 | 0.482 |
| 10 | 0.1 | 0.6 | 1 | 49 | 1477 | 0.485 |
| 10 | 0.1 | 0.6 | 1.1 | 52 | 1301 | 0.499 |
| 10 | 0.1 | 0.7 | 1 | 49 | 1443 | 0.483 |
| 10 | 0.1 | 0.7 | 1.1 | 50 | 1296 | 0.497 |
| 10 | 0.125 | 0.6 | 1 | 51 | 1430 | 0.481 |
| **10** | **0.125** | **0.6** | **1.1** | **44** | **1338** | **0.498** |
| 10 | 0.125 | 0.7 | 1 | 49 | 1467 | 0.488 |
| **10** | **0.125** | **0.7** | **1.1** | **45** | **1353** | **0.5** |
| 10 | 0.2 | 0.6 | 1 | 56 | 1537 | 0.495 |
| 10 | 0.2 | 0.6 | 1.1 | 49 | 1333 | 0.497 |
| 10 | 0.2 | 0.7 | 1 | 55 | 1337 | 0.491 |
| 10 | 0.2 | 0.7 | 1.1 | 51 | 1285 | 0.499 |

Table V: Gridsearch for each parameter for two best performing individual values.

## VI. Tracking speed

The tracking speed in most cases is very high, reachin speeds of about 1300 FPS on average. Since the filter only uses element-wise multiplication and inverse FFT the only parameter that affects the speed is the size of the template. All other parameters only change the values in filter $H$ and thus do not affect the speed. The speed was greatly affected by the size of the template, with a window size multiplier of 2.0 the speed dropped to 524 FPS which is a 63 % decrease from the baseline of no window size multiplier. Using the filter on higher resolution images would also increase the computation time as more element-wise operations are introduced.

## VII. Conclusion

Correlation filters are fast and robust trackers that can be used in real-time tracking applications. The drawback of the filter is the invariability to scale and rotation as the filter does not take them into account. The filter is also very sensitive to the initialization and starting parameters. At the cost of speed, some of these drawbacks can be mitigated by scale pyramids and rotations of the target.

## 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*. IEEE, 2010, pp. 2544–2550.