Visual Object Tracking using Adaptive Correlation Filters

David S. Bolme J. Ross Beveridge Bruce A. Draper Yui Man Lui
Computer Science Department
Colorado State University
Fort Collins, CO 80521, USA
bolme@cs.colostate.edu

Abstract

Although not commonly used, correlation filters can track complex objects through rotations, occlusions and other distractions at over 20 times the rate of current state-of-the-art techniques. The oldest and simplest correlation filters use simple templates and generally fail when applied to tracking. More modern approaches such as ASEF and UMACE perform better, but their training needs are poorly suited to tracking. Visual tracking requires robust filters to be trained from a single frame and dynamically adapted as the appearance of the target object changes.

This paper presents a new type of correlation filter, a Minimum Output Sum of Squared Error (MOSSE) filter, which produces stable correlation filters when initialized using a single frame. A tracker based upon MOSSE filters is robust to variations in lighting, scale, pose, and non-rigid deformations while operating at 669 frames per second. Occlusion is detected based upon the peak-to-sidelobe ratio, which enables the tracker to pause and resume where it left off when the object reappears.

Note: This paper contains additional figures and content that was excluded from CVPR 2010 to meet length requirements.

1 Introduction

Visual tracking has many practical applications in video processing. When a target is located in one frame of a video, it is often useful to track that object in subsequent frames. Every frame in which the target is successfully tracked provides more information about the identity and the activity of the target. Because tracking is easier than detection, tracking algorithms can use fewer computational resources than running an object detector on every frame.

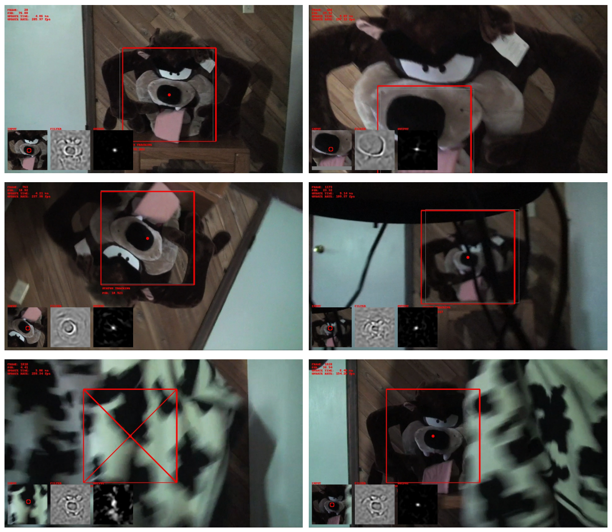


Figure 1: This figure shows the results of the MOSSE filter based tracker on a challenging video sequence. This tracker has the ability to quickly adapt to scale and rotation changes. It is also capable of detecting tracking failure and recovering from occlusion.

Visual tracking has received much attention in recent years. A number of robust tracking strategies have been proposed that tolerate changes in target appearance and track targets through complex motions. Recent examples include: Incremental Visual Tracking (IVT) [17], Robust Fragments-based Tracking (FragTrack) [1], Graph Based Discriminative Learning (GBDL) [19], and Multiple Instance Learning (MILTrack) [2]. Although effective, these techniques are not simple; they often include complex appearance models and/or optimization algorithms, and as result struggle to keep up with the 25 to 30 frames per second produced by many modern cameras (See Table 1).

In this paper we investigate a simpler tracking strategy. The targets appearance is modeled by adaptive correlation filters, and tracking is performed via convolution. Naive methods for creating filters, such as cropping a template from an image, produce strong peaks for the target but also falsely respond to background. As a result they are not particularly robust to variations in target appearance and fail on challenging tracking problems. Average of Synthetic Exact Filters (ASEF), Unconstrained Minimum Average Correlation Energy (UMACE), and Minimum Output Sum of Squared Error (MOSSE) (introduced in this paper) produce filters that are more robust to appearance changes and are better at discriminating between targets and background. As shown in Figure 2, the result is a much stronger peak which translates into less drift and fewer dropped tracks. Traditionally, ASEF and UMACE filters have been trained offline and are used for object detection or target identification. In this research, we have modified these techniques to be trained online and in an adaptive manor for visual tracking. The result is tracking with state of the art performance that retains much of the speed and simplicity of the underlying correlation based approach.

Despite the simplicity of the approach, tracking based on modified ASEF, UMACE, or MOSSE filters performs well under changes in rotation, scale, lighting, and partial occlusion (See Figure 1). The Peak-to-Sidelobe Ratio (PSR), which measures the strength of a correlation peak, can be used to detect occlusions or tracking failure, to stop the online update, and to reacquire the track if the object reappears with a similar appearance. More generally, these advanced correlation filters achieve performance consistent with the more complex trackers mentioned earlier; however, the filter based approach is over 20 times faster and can process 669 frames per second (See Table 1).

Table 1: This table compares the frame rates of the MOSSE tracker to published results for other tracking systems.

Algorithm	Frame Rate	CPU
FragTrack[1]	realtime	Unknown
GBDL[19]	realtime	3.4 Ghz Pent. 4
IVT [17]	7.5fps	2.8Ghz CPU
MILTrack[2]	25 fps	Core 2 Quad
MOSSE Filters	669fps	2.4Ghz Core 2 Duo

The rest of this paper is organized as follows. Section 2 reviews related correlation filter techniques. Section 3 introduces the MOSSE filter and how it can be used to create a robust filter based tracker. Section 4 presents experimental results on seven video sequences from [17]. Finally, Section 5 will revisit the major findings of this paper.

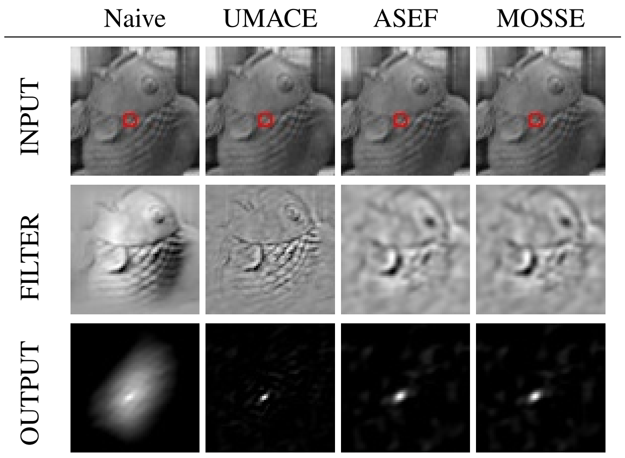


Figure 2: This figure shows the input, filters, and correlation output for Frame 25 of the **fish** test sequence. The three correlation filters produce peaks that are much more compact than the one produced by the Naive filter.

2 Background

In the 1980's and 1990's, many variants of correlation filters, including Synthetic Discriminant Functions (SDF) [7, 6], Minimum Variance Synthetic Discriminant Functions (MVSDF) [9], Minimum Average Correlation Energy (MACE) [11], Optimal Tradeoff Filters (OTF) [16], and Minimum Squared Error Synthetic Discriminant Functions (MSESDF) [10]. These filters are trained on examples of target objects with varying appearance and with enforced hard constraints such that the filters would always produce peaks of the same height. Most relevant is MACE which produces sharp peaks and high PSRs.

In [12], it was found that the hard constraints of SDF based filters like MACE caused issues with distortion tolerance. The solution was to eliminate the hard constraints and instead to require the filter to produce a high average correlation response. This new type of "Unconstrained" correlation filter called Maximum Average Correlation Height (MACH) led to a variant of MACE called UMACE.

A newer type of correlation filter called ASEF [3] introduced a method of tuning filters for particular tasks. Where earlier methods just specify a single peak value, ASEF specifies the entire correlation output for each training image. ASEF has performed well at both eye localization [3] and pedestrian detection [4]. Unfortunately in both studies ASEF required a large number of training images, which made it too slow for visual tracking. This paper reduces this data requirement by introducing a regularized variant of ASEF that is suitable for visual tracking.

3 Correlation Filter Based Tracking

Filter based trackers model the appearance of objects using filters trained on example images. The target is initially selected based on a small tracking window centered on the object in the first frame. From this point on, tracking and filter training work together. The target is tracked by correlating the filter over a search window in next frame; the location corresponding to the maximum value in the correlation output indicates the new position of the target. An online update is then performed based on that new location.

To create a fast tracker, correlation is computed in the Fourier domain Fast Fourier Transform (FFT) [15]. First, the 2D Fourier transform of the input image: $F = \mathcal{F}(f)$, and of the filter: $H = \mathcal{F}(h)$ are computed. The Convolution Theorem states that correlation becomes an element-wise multiplication in the Fourier domain. Using the $\odot$ symbol to explicitly denote element-wise multiplication and $*$ to indicate the complex conjugate, correlation takes the form:

$$G = F \odot H^* \qquad (1)$$

The correlation output is transformed back into the spatial domain using the inverse FFT. The bottleneck in this process is computing the forward and inverse FFTs so that the entire process has an upper bound time of $O(P \log P)$ where P is the number of pixels in the tracking window.

In this section, we discuss the components of filter based trackers. Section 3.1 discusses preprocessing performed on the tracking window. Section 3.2 introduces MOSSE filters which are an improved way to construct a stable correlation filter from a small number of images. Section 3.3 shows how regularization can be used to produce more stable UMACE and ASEF filters. Section 3.4 discusses the simple strategy used for the online update of the filters.

3.1 Preprocessing

One issue with the FFT convolution algorithm is that the image and the filter are mapped to the topological structure of a torus. In other words, it connects the left edge of the image to the right edge, and the top to the bottom. During convolution, the images rotate through the toroidal space instead of translating as they would in the spatial domain. Artificially connecting the boundaries of the image introduces an artifact which effects the correlation output.

This effect is reduced by following the preprocessing steps outlined in [3]. First, the pixel values are transformed using a $\log$ function which helps with low contrast lighting situations. The pixel values are normalized to have a mean value of 0.0 and a norm of 1.0. Finally, the image is multiplied by a cosine window which gradually reduces the pixel values near the edge to zero. This also has the benefit that it puts more emphasis near the center of the target.

3.2 MOSSE Filters

MOSSE is an algorithm for producing ASEF-like filters from fewer training images. To start, it needs a set of training images f_i and training outputs g_i. Generally, g_i can take any shape. In this case, g_i is generated from ground truth such that it has a compact ($\sigma = 2.0$) 2D Gaussian shaped peak centered on the target in training image f_i. Training is conducted in the Fourier domain to take advantage of the simple element-wise relationship between the input and the output. As in the previous section, we define the upper case variables F_i, G_i and the filter H to be the Fourier transform of their lower case counterparts.

$$H_i^* = \frac{G_i}{F_i} \qquad (2)$$

where the division is performed element-wise.

To find a filter that maps training inputs to the desired training outputs, MOSSE finds a filter H that minimizes the sum of squared error between the *actual* output of the convolution and the *desired* output of the convolution. This minimization problem takes the form:

$$\min_{H^*} \sum_i |F_i \odot H^* - G_i|^2 \qquad (3)$$

The idea of minimizing Sum of Squared Error (SSE) over the output is not new. In fact, the optimization problem in Equation 3 is almost identical to optimization problems presented in [10] and [12]. The difference is that in those works it was assumed that the target was always carefully centered in f_i and that the output (g_i) was fixed for the entire training set, whereas customizing every g_i is a fundamental idea behind ASEF and MOSSE. In the tracking problem the target is not always centered, and the peak in g_i moves to follow the target in f_i. In a more general case g_i can have any shape. For example, in [4] f_i contains multiple targets and g_i has multiple corresponding peaks.

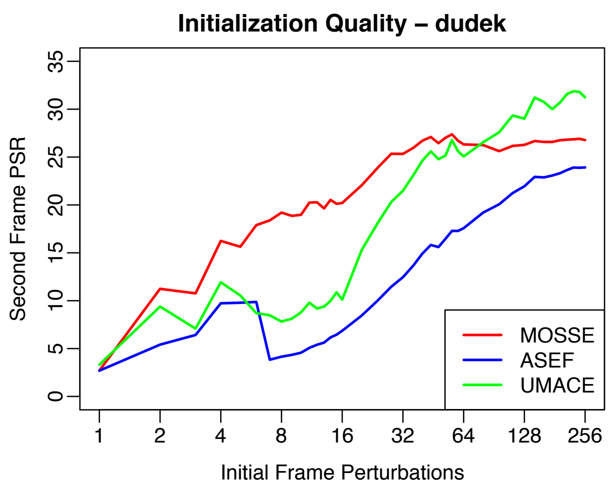


Figure 3: Results shown without regularization.

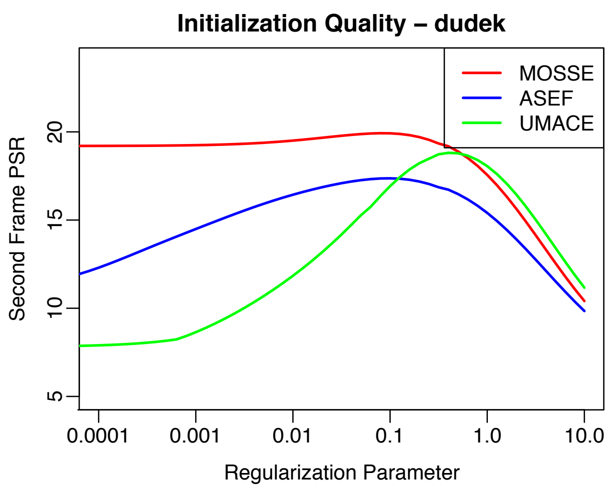


Figure 4: In this figure all three filters were initialized using the same eight images while adjusting the regularization parameter. At $\epsilon \approx 0.1$ all three filters have a high PSR.

Solving this optimization problem is not particularly difficult but does require some care because the function being optimized is a real valued function of a complex variable. First, each element of H (indexed by ω and ν) can be solved for independently because all operations in the Fourier domain are performed element-wise. This involves rewriting the function in terms of both $H_{\omega\nu}$ and $H^*_{\omega\nu}$. Then, the partial W.R.T. $H^*_{\omega\nu}$ is set equal to zero, while treating $H_{\omega\nu}$ as an independent variable [13].

$$0 = \frac{\partial}{\partial H^*_{\omega\nu}} \sum_i |F_{i\omega\nu} H^*_{\omega\nu} - G_{i\omega\nu}|^2 \tag{4}$$

By solving for H^* a closed form expression for the MOSSE filter is found:

$$H^* = \frac{\sum_i G_i \odot F_i^*}{\sum_i F_i \odot F_i^*} \tag{5}$$

A complete derivation is in Appendix A. The terms in Equation 5 have an interesting interpretation. The numerator is the correlation between the input and the desired output and the denominator is the energy spectrum of the input.

From Equation 5, we can easily show that UMACE is a special case of MOSSE. UMACE is defined as $H^* = D^{-1}m^*$ where m is a vector containing the FFT of the average centered cropped training images, and D is a diagonal matrix containing the average energy spectrum of the training images [18]. Because D is a diagonal matrix, multiplication by its inverse essentially performs an element-wise division. When rewritten with the current notation, UMACE takes the form:

$$H^* = \frac{\sum_i F_i^*}{\sum_i F_i \odot F_i^*} \tag{6}$$

however, UMACE requires that the target is centered in F_i. Recentering can be performed using correlation. If we define g_i to be a Kronecker delta (with a peak of one at the target center and zero elsewhere) this will essentially recenter the target and compute a UMACE filter. The difference between this and the traditional implementation is that here we crop and then translate, where the traditional method translates and then crops.

To show that MOSSE produces better filters than ASEF, an experiment was conducted which varied the number of images used to train the filters. The filters were initialized by applying random small affine perturbations to the tracking window for the first frame of the video. The PSR on the second frame was used as a measure of filter quality. Figure 3 shows that MOSSE produces better filters when trained on a small number of image windows. The reason will be discussed in the next section.

3.3 Regularization of ASEF

ASEF takes a slightly different approach to minimizing error in the correlation transformation. It turns out that when there is only one training image F_i and one output image G_i, there is a filter that produces zero error. That filter is called an exact filter and can be found by solving

Equation 1:

$$H_i^* = \frac{G_i}{F_i} = \frac{G_i \odot F_i^*}{F_i \odot F_i^*} \quad (7)$$

An exact filter trained on one image will almost always overfit that image. When applied to a new image, that filter will often fail. Averaging is used to produce a filter that is more general. Motivation for averaging comes from Bootstrap Aggregation [5] where the output of weak classifiers can be averaged to produce a much stronger classifier. With some manipulation, the equation for an ASEF filter can be shown to be:

$$H^* = \frac{1}{N} \sum_i \frac{G_i \odot F_i^*}{F_i \odot F_i^*} \quad (8)$$

If only one image is used for training, MOSSE and ASEF both produce an exact filter.

ASEF filters are unstable when trained on small numbers of images because the element-wise division in Equation 8 becomes unstable when a frequency in the training image contains very little energy (or the denominator is close to zero). Averaging large numbers of exact filters compensates for this problem and produces robust ASEF filters. Because the denominator for MOSSE is the sum of the energies over more images, it will rarely produce small numbers and is therefore more stable.

Alternatively, regularization can be used to correct for low-energy frequencies and produce more stable ASEF filters. This is performed by adding a small value to each element in the energy spectrum. The $F_i \odot F_i^*$ is replaced with $F_i \odot F_i^* + \epsilon$ where ϵ is the regularization parameter.

Regularization is similar to a result that came from OTF theory which is typically used in conjunction with UMACE filters. This result suggests that adding the energy spectrum of the background noise to that of the training imagery will produce a filter with better in noise tolerance [16]. Here we have essentially added white noise.

Figure 4 shows the effect of adjusting ϵ. With proper regularization all of the filters are producing good peaks and should be stable enough to produce a good track.

3.4 Filter Initialization and Online Updates

Equations 8 and 5 describe how filters are constructed during initialization. The training set is constructed using random affine transformations to generate eight small perturbations (f_i) of the tracking window in the initial frame. Training outputs (g_i) are also generated with their peaks corresponding to the target center.

During tracking, a target can often change appearance by changing its rotation, scale, pose, by moving through different lighting conditions, or even by undergoing non-rigid deformation. Therefore, filters need to quickly adapt in order to follow objects. Running average is used for this purpose. For example, the ASEF filter learned from Frame i is computed as:

$$H_i^* = \eta \frac{G_i \odot F_i^*}{F_i \odot F_i^*} + (1 - \eta) H_{i-1}^* \quad (9)$$

and the the MOSSE filter as:

$$\begin{aligned} H_i^* &= \frac{A_i}{B_i} && (10) \\ A_i &= \eta G_i \odot F_i^* + (1-\eta) A_{i-1} && (11) \\ B_i &= \eta F_i \odot F_i^* + (1-\eta) B_{i-1} && (12) \end{aligned}$$

where η is the learning rate. This puts more weight on recent frames and lets the effect of previous frames decay exponentially over time. In practice we have found that $\eta = 0.125$ allows the filter to quickly adapt to appearance changes while still maintaining a robust filter.

3.5 Failure Detection and PSR

As mentioned before a simple measurement of peak strength is called the Peak to Sidelobe Ratio (PSR). To compute the PSR the correlation output g is split into the peak which is the maximum value and the sidelobe which is the rest of the pixels excluding an 11×11 window around the peak. The PSR is then defined as $\frac{g_{\max} - \mu_{s1}}{\sigma_{s1}}$ where $g_{\max}$ is the peak values and μ_{s1} and σ_{s1} are the mean and standard deviation of the sidelobe.

In our experience, PSR for UMACE, ASEF, and MOSSE under normal tracking conditions typically ranges between 20.0 and 60.0 which indicates very strong peaks. We have found that when PSR drops to around 7.0 it is an indication that the object is occluded or tracking has failed. For the Naive implementation PSR ranges between 3.0 and 10.0 and is not useful for predicting track quality.

4 Evaluation

Initially a realtime MOSSE based tracking system was created and evaluated on live video from a webcam. Realtime feedback makes it easy to test small changes to the tracker configuration and to perform qualitative analyses of the tracker performance for various targets and tracking conditions. These tests provided valuable insights into the operation of the tracker and helped produce the fast and robust tracker that is presented in this paper.

A more controlled evaluation was performed on seven commonly used test videos which can be freely downloaded from `http://www.cs.toronto.edu/~dross/ivt/`. The test videos are all in grayscale and include challenging variations in lighting, pose, and appearance. The camera itself is moving in all the videos which adds to the erratic motion of the targets. The seven sequences include two vehicle tracking scenarios (**car4**, **car11**), two toy tracking scenarios (**fish**, **sylv**), and three face tracking scenarios (**davidin300**, **dudek**, and **trellis70**).

4.1 Filter Comparison

This section evaluates tracking quality of the UMACE, ASEF, and MOSSE filters. These are compared with a Naive filter which was based on an average preprocessed tracking window with online updates. The tracking output was manually labeled as good tracking, tracking which had drifted off center, or a lost track (See Figure 5).

Qualitatively, all of the filters including the Naive filter were able to track objects with very little drift through the full range of scale, rotation, and illumination changes found in the test set. Most drifting and failures occur when the target undergoes a large out-of-plane rotation. See Figure 6 for an example from the **davidin300** sequence. The filters tend to track a point in the center of the target. As the target is rotated that point moves towards the target boundary, and the tracker ends up in a state where much of the tracking window is covered by background. The filters adapt to this half background window and when the target rotates back to a frontal pose, the filters sometimes shift to a new location or they may loose the target and track the background.

These results show that the advanced correlation filters track targets longer than the Naive method. The sharp peaks also have the benefit that the PSR is a good predictor of track quality, while PSR is not particularly informative for the Naive filter. For the advanced filters drifting and failures are always associated with low PSRs. This is shown in Figure 7 which shows that the MOSSE PSR can locate the most challenging sections of that video.

For the filter based trackers it is difficult to claim that any one filter type clearly out performs another. On four of the seven video sequences the correlation filters perform perfectly. On **davidin300** all the filters drift from the center of the face to the eye during the same out-of-plane rotation and in **sylv** the filters drift during the same difficult section of that sequence. These two sequences suggest that the choice of filter type is not particularly important because the filters are failing in exactly the same way.

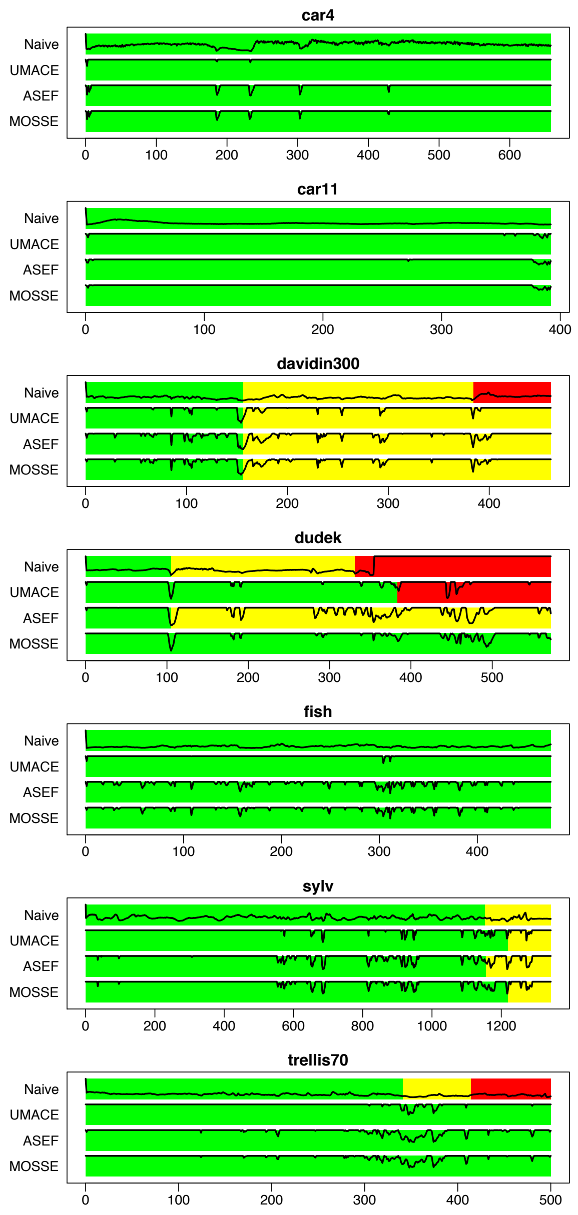


Figure 5: This figure shows the performance of the filter based trackers on all seven of the video sequences. Each output video was hand annotated where Green indicates a good track, Yellow indicates the track drifted off center, and Red indicates tracking failure. The black lines shows PSR that was clipped to the range [0,20] and indicates the quality of the track for each frame of the video.

Only on the **dudek** sequence is there a notable difference among the three filters. While MOSSE completed

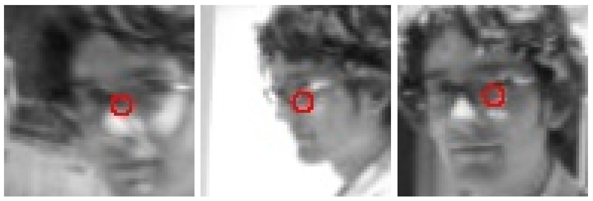


Figure 6: An example of drifting caused by out-of-plain rotation.

the sequence perfectly, UMACE and ASEF had issues at challenging parts of the video. Even though evidence presented in Section 3 indicates MOSSE may be the best filter for this task, single failures on one video sequence are not enough to support a strong claim; more research is needed.

4.2 Comparison to Other Trackers

To evaluate the ability of the algorithm to maintain tracks we compared our output to videos posted by the authors of IVT [17] and MILTrack [2] (see Section 4.1). Those videos also contain sample results for Robust Online Appearance Models (ROAM) [8], Online Ada-Boost (OAB) [14], and FragTrack[1]. We considered downloading code for these other algorithm but we opted instead to study the authors own videos which represent the best performance of those algorithms and which also mitigates arguments that we failed to properly implement or tune those algorithms. In those comparisons our method was able to maintain tracks as well as or better than those algorithms. In this spirit, we have also posted our results to our website/YouTube (`http://youtube.com/users/bolme2008`) so that others can conduct the same comparison. Figure 8 describes format and annotations in the videos.

In [17], IVT [17] and ROAM [8] were compared on the four sequences in Figure 5. Of those, the **davidin300** and **dudek** sequences completed successfully. IVT failed near frame 620 for **sylv** and near frame 330 for **trellis70**. In the published video sequences, the ROAM tracker performed perfectly; however the videos for **sylv** and **trellis70** stopped shortly after IVT failure, and it is unclear if ROAM successfully completed those sequences. One feature of these two trackers that the filter based tracker lacks, is that they estimate scale and orientation of the target which provides more information about its position in space.

In [2], MILTrack [2], OAB [14], and FragTrack [1] were compared on the **davidin300** and **sylv** sequences.

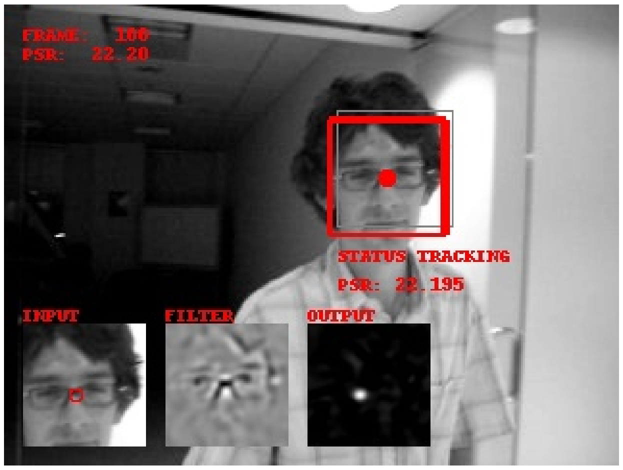


Figure 8: This shows a sample from the **davidin300** sequence for the MOSSE filter. The frame number and PSR are shown in the upper left hand corner. Thin gray box around the face shows starting location of the tracking window for each frame. The thick red box shows the updated location of the track window. The red dot shows the center point of the tracking window and helps to determine if the windows has drifted off the original location. In the Taz video, failure detection is enabled and a red X is drawn in the window to indicate that a failure or occlusion has been detected. Across the bottom of the video are images are the input (cropped from the grey rectangle), the updated filter from this frame, and the correlation output.

All the trackers showed significant drift with OAB and FragTrack failing on **davidin300**. The drifting for these trackers was very different than that seen with the filters. In these videos, the tracking windows wandered back and forth across the targets. When the filters drifted, they tended to shift off center when the target underwent pose changes, and then they locked on to a new center point.

4.3 Real Time Performance

Tests were performed on an a single processor of a 2.4Ghz Core 2 Duo MacBook Pro. The tracker tested in this paper was written in Python using the PyVision library, OpenCV, and SciPy.[1] The original Python implementation averages approximately 250 track updates per second when using a 64×64 tracking window. To better benchmark runtime performance of the tracker, some of the slower parts of the code were reimplemented in C which included better memory management and more efficient

[1]`http://pyvision.sourceforge.net`, `http://opencv.willowgarage.com`, and `http://www.scipy.org`

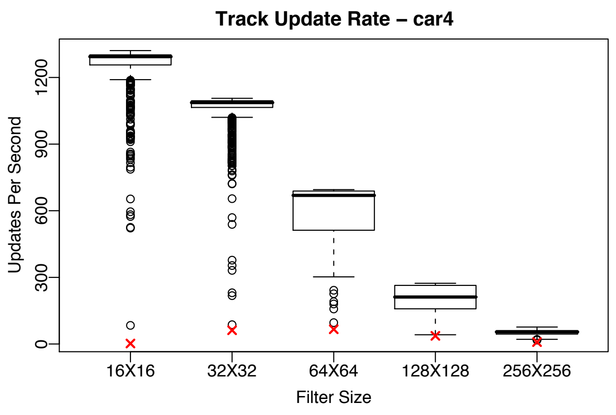


Figure 9: This figure shows the track update rates for various filter sizes. The red x's indicates the rate during initialization.

computation of time consuming tasks such as normalization, FFTs, and PSRs. These optimizations result in a median frame rate of 669 updates per second as shown in Figure 9.

The computational complexity of filter based tracking is $O(P \log P)$, where P is the number of pixels in the filter. This comes from the FFTs used during the correlation operation and the online update. Track initialization incurs a one time cost of $O(NP \log P)$ where N is the number of affine perturbations that are used to initialize the first filter. While this is many times slower than online updates, initialization is still faster than realtime at 66.32 updates per second.

5 Conclusions

This paper has shown that the visual tracking problem, which is traditionally solved using heavy weight classifiers, complex appearance models, and stochastic search techniques can be replaced by efficient and simpler MOSSE correlation filters. The result is an algorithm that is easy to implement, can be just as accurate, and is much faster.

In this paper the tracker was kept simple to evaluate the filter's ability to follow and adapt to difficult tracking scenarios. There are a number of simple ways that this tracker can be improved. For example, if the appearance of the target is relatively stable, drifting could be mitigated by occasionally recentering the filter based on the initial frame. The tracker can also be extended to estimate changes in scale and rotation by filtering the log-polar transform of the tracking window after an update.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, 2006. 1, 2, 7

[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual Tracking with Online Multiple Instance Learning. In *CVPR*, 2009. 1, 2, 7

[3] D. S. Bolme, B. A. Draper, and J. R. Beveridge. Average of synthetic exact filters. In *CVPR*, 2009. 2, 3

[4] D. S. Bolme, Y. M. Lui, B. A. Draper, and J. R. Beveridge. Simple real-time human detection using a single correlation filter. In *PETS*, 2009. 2, 3

[5] L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996. 5

[6] D. Casasent. Unified synthetic discriminant function computational formulation. *Appl. Opt*, 23(10):1620–1627, 1984. 2

[7] C. Hester and D. Casasent. Multivariant technique for multiclass pattern recognition. *Appl. Opt.*, 19(11):1758–1761, 1980. 2

[8] A. Jepson, D. Fleet, and T. El-Maraghi. Robust online appearance models for visual tracking. *T-PAMI*, 25(10):1296–1311, 2003. 7

[9] B. Kumar. Minimum-variance synthetic discriminant functions. *J. Opt. Soc. of America.*, 3(10):1579–1584, 1986. 2

[10] B. Kumar, A. Mahalanobis, S. Song, S. Sims, and J. Epperson. Minimum squared error synthetic discriminant functions. *Optical Engineering*, 31:915, 1992. 2, 3

[11] A. Mahalanobis, B. V. K. V. Kumar, and D. Casasent. Minimum average correlation energy filters. *Appl. Opt.*, 26(17):3633, 1987. 2

[12] A. Mahalanobis, B. Vijaya Kumar, S. Song, S. Sims, and J. Epperson. Unconstrained correlation filters. *Applied Optics*, 33(17):3751–3759, 1994. 2, 3

[13] D. Messerschmitt. Stationary points of a real-valued function of a complex variable. Technical report, EECS, U.C. Berkeley, 2006. 4, 10

[14] N. C. Oza. *Online Ensemble Learning*. PhD thesis, U.C. Berkeley, 2001. 7

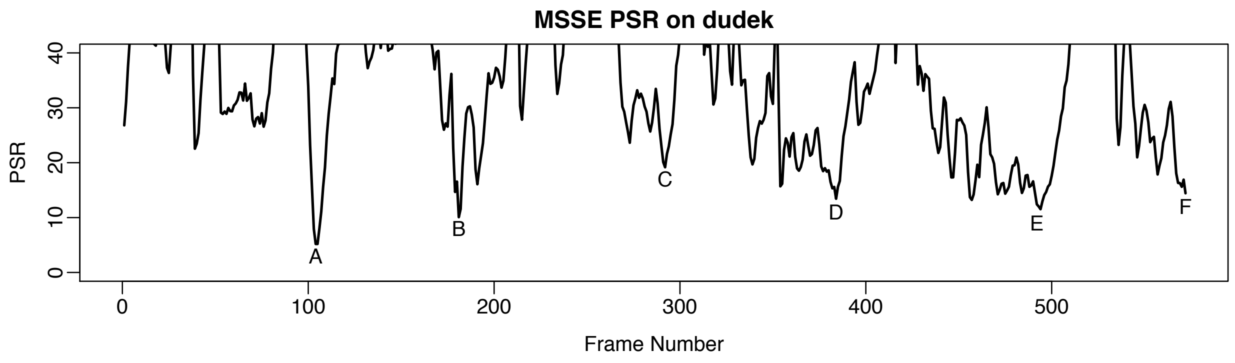
[15] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge Univ. Press, 1988. 3

[16] P. Refregier. Optimal trade-off filters for noise robustness, sharpness of the correlation peak, and Horner efficiency. *Optics Letters*, 16:829–832, June 1991. 2, 5

[17] D. Ross, J. Lim, R. Lin, and M. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1):125–141, 2008. 1, 2, 7

[18] M. Savvides, B. Kumar, and P. Khosla. Face verification using correlation filters. In *AIAT*, 2002. 4

[19] X. Zhang, W. Hu, S. Maybank, and X. Li. Graph based discriminative learning for robust and efficient object tracking. In *ICCV*, 2007. 1, 2



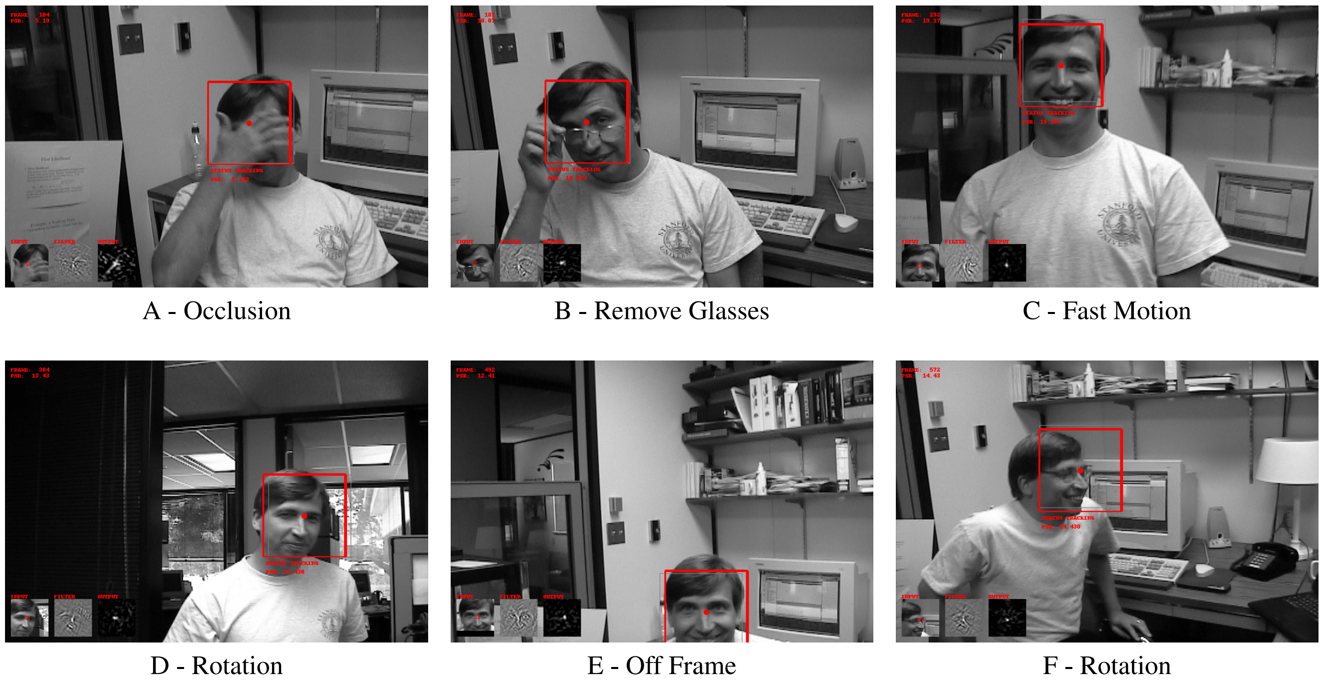


Figure 7: This figure shows that the most challenging section of a video can be located by finding low points in the PSR.

A Minimizing the Output Sum of Squared Error

Here we include a more detailed derivation of the MOSSE process. The paper covers the major steps, and the final result. The explanation here shows most of the intermediate steps. The first step in finding a MOSSE filter is to set up the Minimization problem that will be optimized:

$$H = \min_{H} \sum_i |F_i \odot H^* - G_i|^2 \tag{13}$$

Where F_i and G_i are the input images and the corresponding desired outputs in the Fourier domain and goal is to find a filter H that minimizes the sum of squared output error. Because correlation in the Fourier domain is an element-wise multiplication, each element of the filter H can be optimized independently. The optimization problem can therefore be transformed from a multivariate optimization problem to a problem that optimizes each element of H independently.

$$H_{\omega\nu} = \min_{H_{\omega\nu}} \sum_i |F_{i\omega\nu} H^*_{\omega\nu} - G_{i\omega\nu}|^2 \tag{14}$$

where ω and ν index the elements of H.

This function is real valued, positive, and convex so it will have only a single optima. Normally to find the optima of a function, the stable points are found by setting the derivative equal to zero and then solving for the for the variable of interest. Finding the stable point for this function is different because it is a real valued function of a complex variable. Care needs to be taken to solve this problem correctly. It turns out that finding the stable points of such a function is not that difficult. To summarize, it involves rewriting the function in terms of both $H_{\omega\nu}$ and $H^*_{\omega\nu}$. Then, the partial W.R.T. $H^*_{\omega\nu}$ is set equal to zero, while treating H as an independent variable.

$$0 = \frac{\partial}{\partial H^*_{\omega\nu}} \sum_i |F_{i\omega\nu} H^*_{\omega\nu} - G_{i\omega\nu}|^2 \tag{15}$$

It can be shown that any $H_{\omega\nu}$ which satisfies this equation is a stable point. A short tutorial on this technique can be found in [13]. Transforming this equation leads to:

$$0 = \frac{\partial}{\partial H^*_{\omega\nu}} \sum_i (F_{i\omega\nu} H^*_{\omega\nu} - G_{i\omega\nu})(F_{i\omega\nu} H^*_{\omega\nu} - G_{i\omega\nu})^* \tag{16}$$

$$0 = \frac{\partial}{\partial H^*_{\omega\nu}} \sum_i [(F_{i\omega\nu} H^*_{\omega\nu})(F_{i\omega\nu} H^*_{\omega\nu})^* - (F_{i\omega\nu} H^*_{\omega\nu}) G^*_{i\omega\nu} - G_{i\omega\nu}(F_{i\omega\nu} H^*_{\omega\nu})^* + G_{i\omega\nu} G^*_{i\omega\nu}] \tag{17}$$

$$0 = \frac{\partial}{\partial H^*_{\omega\nu}} \sum_i F_{i\omega\nu} F^*_{i\omega\nu} H_{\omega\nu} H^*_{\omega\nu} - F_{i\omega\nu} G^*_{i\omega\nu} H^*_{\omega\nu} - F^*_{i\omega\nu} G_{i\omega\nu} H_{\omega\nu} + G_{i\omega\nu} G^*_{i\omega\nu} \tag{18}$$

Computing the partial derivative leads to:

$$0 = \sum_i [F_{i\omega\nu} F^*_{i\omega\nu} H_{\omega\nu} - F_{i\omega\nu} G^*_{i\omega\nu}] \tag{19}$$

We can then distribute the summation and solve for $H_{\omega\nu}$.

$$H_{\omega\nu} = \frac{\sum_i F_{i\omega\nu} G^*_{i\omega\nu}}{\sum_i F_{i\omega\nu} F^*_{i\omega\nu}} \tag{20}$$

Finally, we rewrite this expression in our original array notation as:

$$H = \frac{\sum_i F_i \odot G^*_i}{\sum_i F_i \odot F^*_i} \tag{21}$$