Understanding and Diagnosing Visual Tracking Systems

Naiyan Wang[1] Jianping Shi[2,3] Dit-Yan Yeung[1] Jiaya Jia[3]
[1] Hong Kong University of Science and Technology [2] Linkface Group Limited
[3] Chinese University of Hong Kong
winsty@gmail.com jpshi@cse.cuhk.edu.hk dyyeung@cse.ust.hk leojia@cse.cuhk.edu.hk

Abstract

Several benchmark datasets for visual tracking research have been created in recent years. Despite their usefulness, whether they are sufficient for understanding and diagnosing the strengths and weaknesses of different trackers remains questionable. To address this issue, we propose a framework by breaking a tracker down into five constituent parts, namely, motion model, feature extractor, observation model, model updater, and ensemble post-processor. We then conduct ablative experiments on each component to study how it affects the overall result. Surprisingly, our findings are discrepant with some common beliefs in the visual tracking research community. We find that the feature extractor plays the most important role in a tracker. On the other hand, although the observation model is the focus of many studies, we find that it often brings no significant improvement. Moreover, the motion model and model updater contain many details that could affect the result. Also, the ensemble post-processor can improve the result substantially when the constituent trackers have high diversity. Based on our findings, we put together some very elementary building blocks to give a basic tracker which is competitive in performance to the state-of-the-art trackers. We believe our framework can provide a solid baseline when conducting controlled experiments for visual tracking research.

1. Introduction

Visual tracking is an essential building block of many advanced applications in the areas such as video surveillance and human-computer interaction. In this paper, we focus on the most general type of visual tracking problems, namely, short-term single-object model-free tracking [18]. We consider the most common setting for this problem: the tracker is given a bounding box to indicate the object to be tracked. The bounding box is either from human annotation or an automatic object detector. The tracker has no prior knowledge of the object to be tracked such as category and shape. Then in the following frames, the tracker needs to identify the object as it moves around in the video. Numerous such trackers have been proposed over the past few decades, ranging from the simple KLT tracker [20, 31] in the 1980s to the recent deep learning trackers [34, 16] which are a lot more complex.

Evaluating and comparing trackers has always been a nontrivial task. For a long time, researchers usually reported tracking results on a small number of videos based on specific model parameters manually tuned for each video. Since subjective bias [24] in the results can be caused by selection of videos, this practice makes it infeasible to give a fair comparison of different trackers. To address this fairness concern, several relatively large benchmarks [39, 18] and evaluation metrics [6] have been proposed recently. With the aid of these benchmarks, we have witnessed substantial advances in recent years. However, we would like to raise this question: *Is simply evaluating these trackers on the de facto benchmarks sufficient for understanding and diagnosing their strengths and weaknesses?*

We are afraid that the answer to the above question is not affirmative, for the following reason. Modern trackers are usually complicated systems made up of several separate components. When a tracker is evaluated as a whole, we cannot gain a detailed understanding of the effectiveness of each component. For illustration, suppose tracker A uses histograms of oriented gradients (HOG) [8] as features and the support vector machine (SVM) as the observation model, while tracker B uses raw pixels as features and logistic regression as the observation model. If tracker A outperforms tracker B in a benchmark, can we conclude that SVM is better than logistic regression for tracking? Obviously drawing such a conclusion would be arbitrary since HOG features have stronger representational power than raw pixels. This calls for a more carefully designed framework for the evaluation and comparison of trackers.

We propose in this paper a new way to understand and diagnose visual trackers. Note that our goal is not to create a new benchmark. Instead, our analysis will still be based on existing benchmarks. We first break a tracker

down into its constituent parts, namely, motion model, feature extractor, observation model, model updater, and ensemble post-processor. We note that most existing trackers can be viewed this way. Based on this framework, we conduct an ablative analysis on a tracker to identify the constituent part that is most crucial to the overall performance of the tracker. Contrary to popular belief, it turns out that the observation model (which is the focus of many papers on visual tracking) does not play the most important role in a tracker. Instead, we find that actually the feature extractor affects the performance most. Moreover, the ensemble post-processor is a simple yet effective way to achieve significant performance boost, but it is comparatively less studied. Furthermore, properly dealing with the details in motion model and model updater is also the key to good performance. By assembling the basic components properly, we can achieve results comparable with the state of the art without resorting to complicated techniques. We conclude this paper by highlighting some limitations of our proposed approach as well as some possible ways to address them in our future work.

2. Related Work

Significant advances in short-term single-object model-free tracking research have been made over the past few decades. It is impossible to review them all here due to space limitations. For a comprehensive survey, readers are referred to [28, 40].

Briefly speaking, there are two major categories of trackers: generative trackers and discriminative trackers. Generative trackers typically assume a generative process of the appearance of the target and search for the most similar candidate in the video. Some representative methods are (robust) PCA [26, 33], sparse coding [23], and dictionary learning [35]. On the other hand, discriminative trackers take a different approach. They usually train a classifier to separate the target from the background. Thanks to advances made by machine learning researchers, many sophisticated techniques have been applied to visual tracking, including boosting [12, 13], multiple-instance learning [3], structured output SVM [14], Gaussian process regression [11], and deep learning [36, 34, 16]. Recent benchmarking studies show that the top-performing trackers are usually discriminative trackers [9, 15] or hybrid ones [43] mainly because purely generative trackers cannot handle complicated background well, making it easy to drift away from the target.

As for tracker evaluation, we have witnessed an exploding trend in building datasets and the corresponding benchmarks for visual tracking. A milestone is the recent contribution made by a benchmark [39] which consists of 50 videos with full annotations. The authors also proposed a novel performance metric which uses the area under curve (AUC) of the overlap rate curve or the central pixel distance curve for evaluation. Recently this benchmark has been extended to an even larger one [40]. Another representative work is the Visual Object Tracking (VOT) challenge [18] which has been held annually since 2013. The key difference with the benchmark above lies in the evaluation metric. To characterize better the properties of short-term tracking, evaluation is based on two independent metrics: accuracy and robustness. While accuracy is measured in terms of the overlap rate between the prediction and ground truth when the tracker does not drift away, robustness is measured according to the frequency of tracking failure which happens when the overlap rate is zero. Whenever such failure occurs, the tracker is reset to the correct bounding box to continue tracking. Readers are referred to [6] for more details. Other benchmark datasets include the Princeton tracking benchmark [29], NUS-PRO [19] and ALOV++ [28]. We tabulate them in Table 1 for easy comparison.

Another related work is [24]. For fair evaluation of the trackers, the authors first collected evaluation results from the published papers and then removed the results of the proposed method in each paper to reduce subjective bias, because the authors tend to select videos or tune parameters specifically to demonstrate the advantages of the proposed tracker. On the other hand, the authors are usually fair to the other trackers compared. They then used several rank aggregation methods to rank the trackers. The results are basically consistent with those run directly on the benchmark.

Dataset	Year	#Videos
VTB1.0 [39]	2013	50
PTB [29]	2013	100
ALOV++ [28]	2013	314
VOT2014 [18]	2014	25
VTB2.0 [40]	2015	100
NUS-PRO [19]	2015	365

Table 1. Summary of some visual tracking benchmark datasets.

3. Our Proposed Framework

We present our proposed framework in this section. As mentioned above, we break a tracking system into multiple constituent parts. Their functions are summarized below:

1. **Motion Model:** Based on the estimation from the previous frame, the motion model generates a set of candidate regions or bounding boxes which may contain the target in the current frame.

2. **Feature Extractor:** The feature extractor represents each candidate in the candidate set using some features.

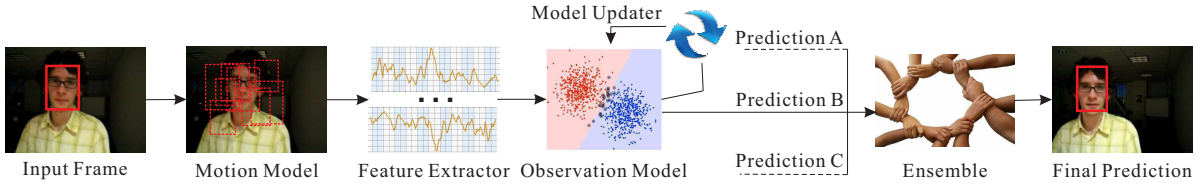


Figure 1. Pipeline of the proposed framework of a visual tracking system.

3. **Observation Model:** The observation model judges whether a candidate is the target based on the features extracted from the candidate.

4. **Model Updater:** The model updater controls the strategy and frequency of updating the observation model. It has to strike a balance between model adaptation and drift.

5. **Ensemble Post-processor:** When a tracking system consists of multiple trackers, the ensemble post-processor takes the outputs of the constituent trackers and uses the ensemble learning approach to combine them into the final result.

A tracking system usually works by initializing the observation model with the given bounding box of the target in the first frame. In each of the following frames, the motion model first generates candidate regions or proposals for testing based on the estimation from the previous frame. The candidate regions or proposals are fed into the observation model to compute their probability of being the target. The one with the highest probability is then selected as the estimation result of the current frame. Based on the output of the observation model, the model updater decides whether the observation model needs any update and, if needed, the update frequency. Finally, if there are multiple trackers, the bounding boxes returned by the trackers will be combined by the ensemble post-processor to obtain a more accurate estimate. This pipeline is illustrated in Fig. 1.

4. Validation Setup

In this section, we will first introduce our experimental settings which include the dataset and the evaluation metric. A basic model will then be used as the starting point for illustration. This is followed by findings in each component, leading to gradual improvement of our model. The source codes for the validation are provided in `http://winsty.net/tracker_diagnose.html`.

4.1. Settings

Due to space limitations, we cannot provide in the paper the detailed parameter settings for each component. Instead, we leave them to the supplemental material. We determine the parameters of each component using five videos outside the benchmark and then fix the parameters afterwards throughout the evaluation unless specified otherwise. For this paper, we use the most common dataset, VTB1.0 [39], as our benchmark. However, the evaluation approach demonstrated in this paper can be readily applied to other benchmarks as well.

Following the convention of [39], we use two metrics for evaluation. The first one is the AUC of the overlap rate curve. In each frame, the performance of a tracker can be measured by the overlap rate between the ground-truth and predicted bounding boxes, where the overlap rate is defined as the area of intersection of the two bounding boxes over the area of their union. With a given threshold for the overlap rate, we can calculate the success rate of the tracker over all the video frames. By varying the threshold from 0 gradually to 1, it will yield a curve which varies from its maximum success rate to success rate 0 accordingly. A larger AUC of this curve indicates a higher accuracy of the tracker. The second metric is the precision at threshold 20 for the central pixel error curve. The curve is generated in a way similar to that for the overlap rate. The central pixel error is defined as the distance between the centers of the two bounding boxes in pixels. This metric is useful for the cases that the scale of the object changes but the tracker does not support scale variation, since using only the scale of the first frame will definitely give a low overlap rate which will make the results indistinguishable.

4.2. Basic Model

We need a basic model to start our analysis. As a starting point, we use a very simple one which adopts the particle filter framework as the motion model, raw pixels of grayscale images as features, and logistic regression as the observation model. For the model updater, we use a simple rule that if the highest score among the candidates tested is below a threshold, the model will be updated. Moreover, we only consider a single tracker in this basic model and hence no ensemble post-processor will be used. Details of all these components will be provided in the next section. For illustration, we show in Fig. 2 the performance of this basic model along with some popular trackers. We can see that even this very simple model can obtain moderate results when compared to some competitive methods in [39].

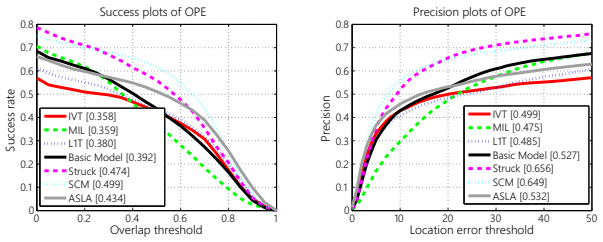


Figure 2. One Pass Evaluation (OPE) plots on VTB1.0 [39]. The performance score for each tracker is shown in the legend. For the success plots of overlap rate, the score is the AUC value. While for precision plots of central pixel error, the score is the precision at threshold 20.

5. Validation and Analysis

We now conduct an ablative analysis to see how each component of a tracker affects its final tracking performance. We present our analysis of different components in the order of their importance and necessity.

5.1. Feature Extractor

The feature extractor converts the raw image data into some (usually) more informative representation. Five feature representations are commonly used for object detection and tracking:

1. **Raw Grayscale:** It simply resizes the image into a fixed size, converts it to grayscale, and then uses the pixel values as features.

2. **Raw Color:** It is the same as raw grayscale features except that the image is represented in the CIE Lab color space instead of grayscale.

3. **Haar-like Features:** We consider the simplest form, rectangular Haar-like features, which was first introduced in 2001 [32].

4. **HOG:** It is a good shape detector widely used for object detection. It was first proposed in 2005 [8].

5. **HOG + Raw Color:** This feature representation simply concatenates the HOG and raw color features.

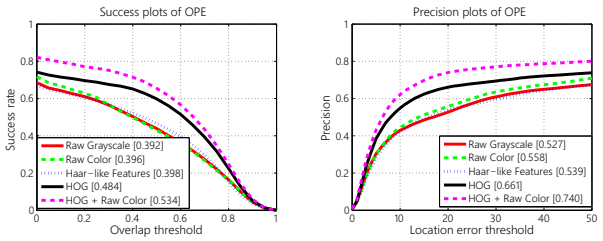


Figure 3. Results of different feature representations.

We compare the performance of these feature representations in Fig. 3. Note that the performance gaps between features can be quite large. For example, the best scheme (HOG + raw color) outperforms the basic model (raw grayscale) by more than 20%. In fact, the best result is even beyond the best performance reported in [39]. Although there exist even more powerful features such as those extracted by the convolutional neural network (CNN) and they indeed can yield state-of-the-art performance [34, 16], naïve application of this approach will incur high computational cost which is highly undesirable for tracking applications. For efficiency consideration, some special designs as in [34] are needed. Another interesting direction is to exploit the color information. Some recent methods [10, 25] demonstrated notable performance with carefully designed color features. Not only are these features lightweight, but they are also suitable for deformable objects. We believe that finding good features for object tracking is still a research direction that is worth pursuing.

Our Findings: *The feature extractor is the most important component of a tracker. Using proper features can dramatically improve the tracking performance. Developing a good and effective feature representation for tracking is still an open problem.*

5.2. Observation Model

The observation model returns the confidence of a given candidate being the target, so it is usually believed to be the key component of a tracker. Since the top-performing trackers in recent benchmarking studies are exclusively discriminative trackers, we do not include generative observation models in our analysis. We consider the following observation models:

1. **Logistic Regression:** Logistic regression with l_2 regularization is used. Online update is achieved by simply using gradient descent.

2. **Ridge Regression:** Least squares regression with l_2 regularization is used. The targets for positive examples are set to one while those for negative examples are set to zero. Online update is achieved by aggregating sufficient statistics, a scheme originated from [21] for online dictionary learning.

3. **SVM:** Standard SVM with hinge loss and l_2 regularization is used. The online update method is from [38].

4. **Structured Output SVM (SO-SVM):** The optimization target of the structured output SVM is the overlap rate instead of the class label. This method is from [14].

We test these four classifiers using two feature representations, a weak one (raw grayscale) and a strong one (HOG

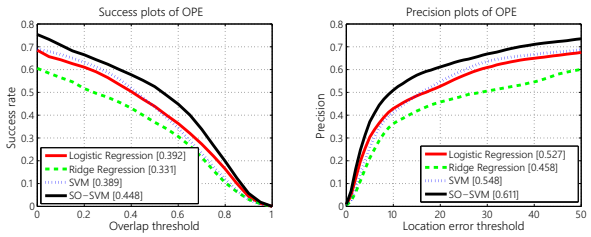


Figure 4. Results of different observation models with weak features.

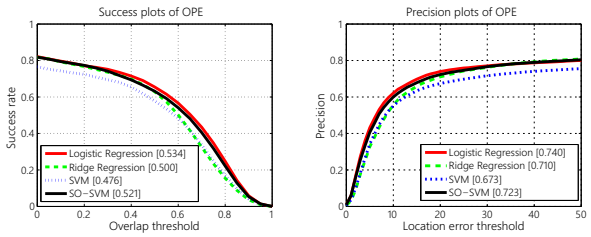


Figure 5. Results of different motion models with strong features.

+ raw color). The results are shown in Fig. 4 and Fig. 5, respectively.

When weak features are used, a powerful classifier such as SO-SVM can indeed improve the performance of the basic model by about 10%. However, when strong features are used, surprisingly the results are reversed. Logistic regression becomes the best-performing observation model. Similar observation was also reported in [15]: when raw pixels are used as features, a kernelized classifier beats a simple linear one by a large margin; however, when HOG features are used, the performance gap reduces to almost zero. We believe that our finding is by no means just coincidence.

Our Findings: *Different observation models indeed affect the performance when the features are weak. However, the performance gaps diminish when the features are strong enough. Consequently, satisfactory results can be obtained even using simple classifiers from textbooks.*

5.3. Motion Model

In each frame, based on the estimation from the previous frame, the motion model generates a set of candidates for the target. We consider three commonly used motion models:

1. **Particle Filter:** Particle filter is a sequential Bayesian estimation approach which recursively infers the hidden state of the target. For a complete tutorial, we refer the readers to [2] for details.

2. **Sliding Window:** The sliding window approach is an exhaustive search scheme which simply considers all possible candidates within a square neighborhood.

3. **Radius Sliding Window:** It is a simple modification of the previous approach which considers a circular region instead. It was first considered in [14].

The key differences between the particle filter and sliding window approaches lie in the following two aspects. First, the particle filter approach can maintain a probabilistic estimation for each frame. Thus when several candidates have high probability of being the target, they will all be kept for the next frames. As a result, it can help to recover from tracker failure. In contrast, the sliding window approach only chooses the candidate with the highest probability and prune all others. Second, the particle filter framework can easily incorporate changes in scale, aspect ratio, and even rotation and skewness. Due to the high computational cost induced by exhaustive search, however, the sliding window approach can hardly pursue it. Results of the comparison are shown in Fig. 6.

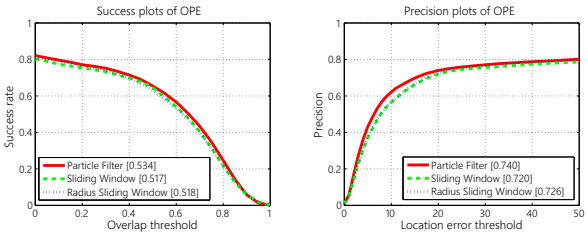


Figure 6. Results of different motion models.

We note that the three motion models show no significant difference on the benchmark. Although particle filter has the two advantages mentioned above, they do not translate into performance gain in the evaluation. Nevertheless, we should note that this observation is valid only when performing object tracking under normal scenarios. In case there is severe camera shake such as in egocentric videos, more sophisticated motion models specially designed for a purpose are definitely worth trying.

A closer look at the subcategory results of the benchmark in Fig. 7 reveals some interesting observations. Not surprisingly, particle filter is much better than the sliding window approach when scale variation exists, but it is much worse for the fast motion sub-category. So, can we perform well in both subcategories simultaneously?

To answer this question, we first examine the role of the translation parameters in a particle filter: They control the search region of the tracker. When the search region is too small, the tracker is likely to lose the target when it is in fast motion. On the other hand, having a large search region will make the tracker prone to drift due to distractors in the background. We have noticed an improper practice in setting the parameters, which is often to use the number of pixels as unit. However, different videos may have very different resolution. Using an absolute number of pixels to set the parameters will actually result in different search re-

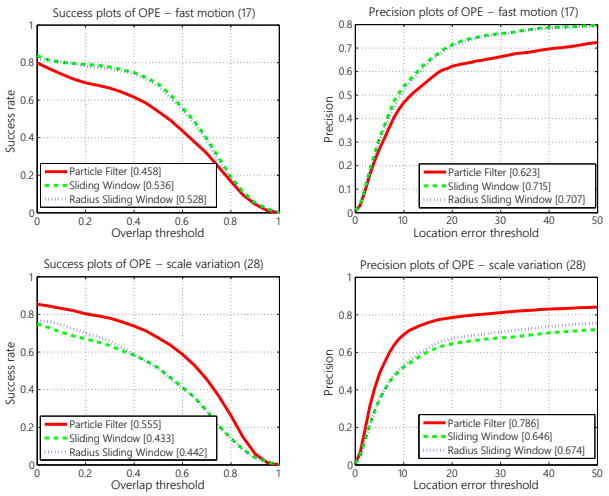


Figure 7. Results of different motion models with fast motion and scale variation.

gions. A simple solution is to scale the parameters by the video resolution which, equivalently, resizes the video to some fixed scale. We adopt the latter approach and report the results in Fig. 8.

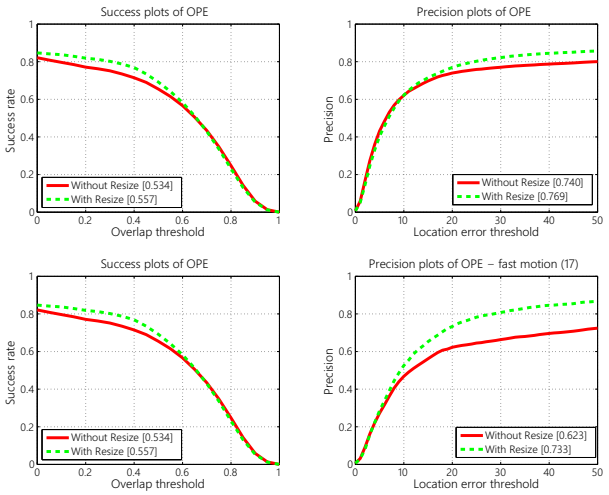


Figure 8. Results comparing the settings with and without resizing the input video to a fixed size.

We find that even such a simple normalization step can improve the performance significantly especially when there exists fast motion. By applying this simple normalization step, particle filter could handle both scale variation and fast motion well. This experiment thus validates our hypothesis that the parameters of the motion model should be adaptive to video resolution.

Our Findings: *When compared to the feature extractor and observation model components, in general the motion model only has minor effects on the performance. However, under scale variation and fast motion, setting the parameters properly is still crucial to obtaining good performance. Furthermore, for some specific scenarios such as egocentric video, it is beneficial to design the motion model carefully. Due to its ability to adapt to scale changes which are not uncommon in practice, we will still take the particle filter approach with resized input as the default motion model in the sequel.*

5.4. Model Updater

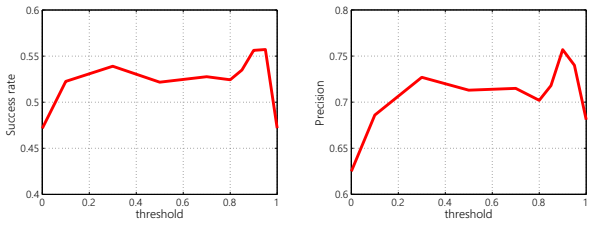
The model updater determines both the strategy and frequency of model update. Since the update of each observation model is different, the model updater often specifies when model update should be done and its frequency. As under our tracking setting there is only one reliable example, the tracker must maintain a tradeoff between adapting to new but possibly noisy examples collected during tracking and preventing the tracker from drifting to the background.

When the model needs update, we first collect some positive examples whose centers are within 5 pixels from the target and some negative examples within 100 pixels but with overlap rate less than 0.3. We consider two model update methods:

1. The first method is to update the model whenever the confidence of the target falls below a threshold. Doing so ensures that the target always has high confidence. This is the default updater used in our basic model.

2. The second method is to update the model whenever the difference between the confidence of the target and that of the background examples is below a threshold. This strategy simply maintains a sufficiently large margin between the positive and negative examples instead of forcing the target to have high confidence. It is potentially helpful when the target is occluded or disappears. This method was proposed and evaluated in [30].

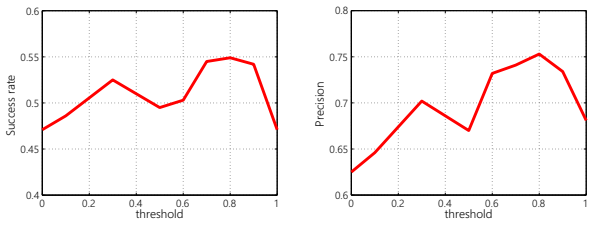
We show the results of these two methods in Fig. 9 and Fig. 10.



(a) AUC of overlap rate (b) Precision@20 for central pixel error curve

Figure 9. Results of varying the threshold for the first model update method.

Varying the threshold can indeed affect the results by more than 10%. The best results for both methods are very



(a) AUC of overlap rate (b) Precision@20 for central pixel error curve

Figure 10. Results of varying the threshold for the second model update method.

similar, although the second method seems to give satisfactory results over a broader range of parameters.

Most research effort in this area focuses on generative trackers. In [22], Matthews *et al.* first empirically compared the effect of different template update strategies. Following this work, Ross *et al.* proposed to use incremental PCA [26] for template update, Wang *et al.* showed the importance of sparsity and robustness [35] for this problem, and Xing *et al.* proposed to maintain three dictionaries of different lifespans [41]. However, the model updater is less studied in discriminative trackers. Santner *et al.* first noticed this issue, and then proposed a simple yet effective method [27] to balance the stability and plasticity by combining the results of template matching, random forest and optical flow. Recently, Zhang *et al.* proposed a more principled method for model update in [42]. It uses entropy minimization to identify reliable model update and discard the incorrect ones.

Our Findings: *Although implementation of the model updater is often treated as engineering tricks in papers especially for discriminative trackers, their impact on performance is usually very significant and hence is worth studying. Unfortunately, very few work focuses on this component.*

5.5. Ensemble Post-processor

From the analysis above, we can see that the result of a single tracker can sometimes be very unstable in that the performance can vary a lot even under small perturbation of the parameters. The purpose of taking the ensemble approach is to overcome this limitation. We regard the ensemble as a post-processing component which treats the constituent trackers as blackboxes and takes only the bounding boxes returned by them as input. This rationale is quite different from ensemble tracking [12, 13] which uses boosting to build a better observation model. Our ensemble includes six trackers, with four of them corresponding to four different observation models in our framework and the other two are DSST [9] and TGPR [11]. We choose these two trackers because they are among the best-performing trackers, and their techniques are complementary to ours. We show the performance of individual trackers in Fig. 11. Their results are very competitive. For the ensemble, we consider two recent methods:

1. The first one is from [4]. This paper first proposed a loss function for bounding box majority voting and then extended it to incorporate tracker weights, trajectory continuity and removal of bad trackers. We adopt two methods from the paper: the basic model and online trajectory optimization.

2. The second one is from [37]. The authors formulated the ensemble learning problem as a structured crowdsourcing problem which treats the reliability of each tracker as a hidden variable to be inferred. Then they proposed a factorial hidden Markov model that considers the temporal smoothness between frames. We adopt the basic model called ensemble based tracking (EBT) without self-correction.

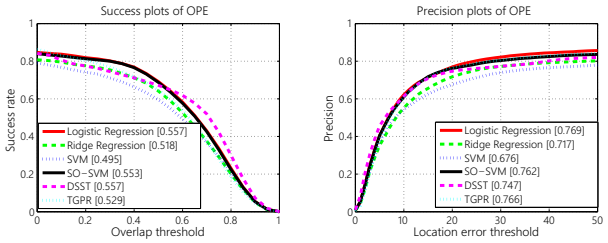


Figure 11. Results of individual trackers used in ensemble.

Since the four trackers from our framework are all using the same features and motion model, their diversity is somewhat limited. A main reason of including the last two trackers into the ensemble is to increase the diversity of the trackers, because diversity often plays an important role in increasing the effectiveness of an ensemble. To investigate how diversity can affect the ensemble performance, we report two sets of results: with and without DSST and TGPR. Their results are shown in Fig. 12 and Fig. 13, respectively.

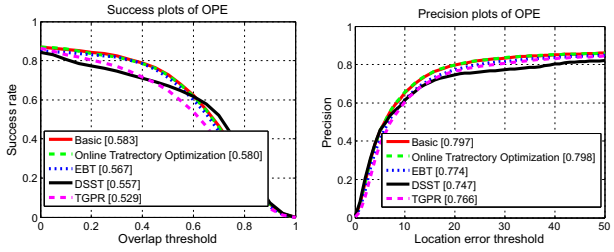


Figure 12. Results of ensemble when the individual trackers are of low diversity (the four different observation models from our framework). *Basic* and *Online Trajectory Optimization* methods are from [4] and *EBT* is from [37].

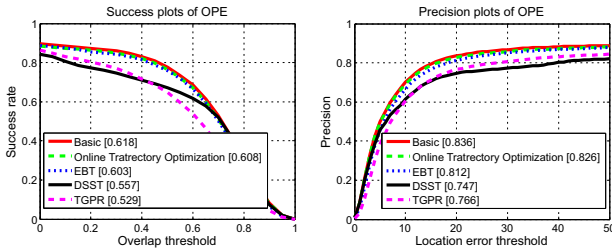


Figure 13. Results of ensemble when the individual trackers are of high diversity (all the six trackers). *Basic* and *Online Trajectory Optimization* methods are from [4] and *EBT* is from [37].

We can see that diversity in the ensemble helps to achieve good results. Both ensemble methods can significantly improve the results when the trackers have high diversity. Even when the diversity is low, the ensemble does not impair the performance but still slightly outperforms the best single tracker.

Our Findings: *The ensemble post-processor can improve the performance substantially especially when the trackers have high diversity. This component is universal and effective yet it is least explored.*

6. Limitations of Current Framework

The primary goal of this work is to gain a deeper understanding into the different components of a visual tracking system, rather than trying to include all existing trackers into our framework. Thus, inevitably, some excellent trackers are not represented in the current framework. We list and discuss some of them here.

First, in some methods, several components are tightly coupled. For example, in the classical mean-shift tracker [7], the observation model must be paired with a probabilistic map as output; in some part-based methods, such as [1, 17], the observation model must be designed in such a way to take the part information into consideration; and in the latest deep learning trackers [36, 34], the feature extractor and observation model are combined into a unified deep learning framework for end-to-end learning.

Second, while accuracy is an important factor in visual tracking systems, it is certainly not the only one. Speed is another important factor to consider in practice. Since our framework is designed to be as universal and generic as possible to accommodate more, though not all, algorithms, we have not put much effort on optimizing the speed on purpose. Our best combination runs about 10fps in MATLAB. There exist some recent attempts that focus on developing fast tracking models. For example, fast Fourier transform (FFT) [5] and circular matrices [15, 9] are used to accelerate dense (kernelized) ridge regression. In their work, the motion model and observation model are coupled. Although we could approximate their methods in our framework using sliding windows and ridge regression, such implementation would be much slower than that in the original paper.

7. Conclusion and Future Work

"God is in the details."

— Ludwig Mies van der Rohe

In this paper, we have analyzed and identified some important factors for a good visual tracking system. We show that if we design each component carefully, even some very elementary building blocks from textbooks can result in a tracker that is as competitive as state-of-the-art trackers. By breaking a visual tracking system down into its constituent parts and analyzing each of them carefully, we have arrived at some interesting conclusions. First, the feature extractor is the most important part of a tracker. Second, the observation model is not that important if the features are good enough. Third, the model updater can affect the result significantly, but currently there are not many principled ways for realizing this component. Lastly, the ensemble post-processor is quite universal and effective. Besides, we demonstrate that paying attention to some details of the motion model and model updater can significantly improve the performance.

Our work enlightens several interesting directions to pursue, including the development of lightweight and effective feature representations, principled ways of model update, and advanced ensemble methods. It is our hope that, besides the observation model which has been the focus of many studies, other equally important components in tracking systems will attract more research attention as a consequence of our findings.

Acknowledgement

This research has been partially supported by Faculty Research Award Z0400-D granted to Dit-Yan Yeung.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, pages 798–805, 2006. 8

[2] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002. 5

[3] B. Babenko, M. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011. 2

[4] C. Bailer, A. Pagani, and D. Stricker. A superior tracking approach: Building a strong tracker through fusion. In *ECCV*, pages 170–185. 2014. 7, 8

[5] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *CVPR*, pages 2544–2550, 2010. 8

[6] L. Čehovin, A. Leonardis, and M. Kristan. Visual object tracking performance measures revisited. *arXiv preprint arXiv:1502.05803*, 2015. 1, 2

[7] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, pages 142–149, 2000. 8

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005. 1, 4

[9] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Accurate scale estimation for robust visual tracking. In *BMVC*, 2014. 2, 7, 8

[10] M. Danelljan, F. S. Khan, M. Felsberg, and J. v. d. Weijer. Adaptive color attributes for real-time visual tracking. In *CVPR*, pages 1090–1097, 2014. 4

[11] J. Gao, H. Ling, W. Hu, and J. Xing. Transfer learning based visual tracking with Gaussian processes regression. In *ECCV*, pages 188–203. 2014. 2, 7

[12] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, pages 47–56, 2006. 2, 7

[13] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, pages 234–247, 2008. 2, 7

[14] S. Hare, A. Saffari, and P. H. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263–270, 2011. 2, 4, 5

[15] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *arXiv preprint arXiv:1404.7584*, 2014. 2, 5, 8

[16] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. *arXiv preprint arXiv:1502.06796*, 2015. 1, 2, 4

[17] X. Jia, H. Lu, and M. Yang. Visual tracking via adaptive structural local sparse appearance model. In *CVPR*, pages 1822–1829, 2012. 8

[18] M. Kristan and *et al.*. The visual object tracking VOT2014 challenge results. In *ECCV Workshop*, 2014. 1, 2

[19] A. Li, M. Lin, Y. Wu, M.-H. Yang, and S. Yan. NUS-PRO: A new visual tracking challenge. *To Appear in IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015. 2

[20] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981. 1

[21] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(1):19–60, 2010. 4

[22] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815, 2004. 7

[23] X. Mei and H. Ling. Robust visual tracking using l_1 minimization. In *ICCV*, pages 1436–1443, 2009. 2

[24] Y. Pang and H. Ling. Finding the best from the second bests-inhibiting subjective bias in evaluation of visual tracking algorithms. In *ICCV*, pages 2784–2791, 2013. 1, 2

[25] H. Possegger, T. Mauthner, and H. Bischof. In defense of color-based model-free tracking. In *CVPR*, 2015. 4

[26] D. Ross, J. Lim, R. Lin, and M. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1):125–141, 2008. 2, 7

[27] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST: Parallel robust online simple tracking. In *CVPR*, pages 723–730, 2010. 7

[28] A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7), 2014. 2

[29] S. Song and J. Xiao. Tracking revisited using RGBD camera: Baseline and benchmark. In *ICCV*, pages 233–240, 2013. 2

[30] J. Supancic and D. Ramanan. Self-paced learning for long-term tracking. In *CVPR*, pages 2379–2386, 2013. 6

[31] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991. 1

[32] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001. 4

[33] D. Wang, H. Lu, and M.-H. Yang. Least soft-threshold squares tracking. In *CVPR*, pages 2371–2378, 2013. 2

[34] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung. Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*, 2015. 1, 2, 4, 8

[35] N. Wang, J. Wang, and D.-Y. Yeung. Online robust non-negative dictionary learning for visual tracking. In *ICCV*, pages 657–664, 2013. 2, 7

[36] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, pages 809–817, 2013. 2, 8

[37] N. Wang and D.-Y. Yeung. Ensemble-based tracking: Aggregating crowdsourced structured time series data. In *ICML*, pages 1107–1115, 2014. 7, 8

[38] Z. Wang and S. Vucetic. Online training on a budget of support vector machines using twin prototypes. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3(3):149–169, 2010. 4

[39] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013. 1, 2, 3, 4

[40] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *To Appear in IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015. 2

[41] J. Xing, J. Gao, B. Li, W. Hu, and S. Yan. Robust object tracking with online multi-lifespan dictionary learning. In *ICCV*, pages 665–672, 2013. 7

[42] J. Zhang, S. Ma, and S. Sclaroff. MEEM: Robust tracking via multiple experts using entropy minimization. In *ECCV*, pages 188–203. 2014. 7

[43] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *CVPR*, pages 1838–1845, 2012. 2