2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)

Visual Tracking with Autoencoder-Based Maximum A Posteriori Data Fusion

Yevgeniy Reznichenko*, Enrico Prampolini*†, Abubakar Siddique*, Henry Medeiros* and Francesca Odone†
*Electrical and Computer Engineering
Marquette University
Milwaukee, USA
firstname.lastname@marquette.edu

†Computer Science
University of Genoa
Genoa, Italy
francesca.odone@unige.it

***Abstract*—In this paper, a novel method for tracker fusion is proposed and evaluated for vision-based object tracking. This work combines three distinct popular techniques into a recursive Bayesian estimation algorithm. First, a semi-supervised learning approach is used to train deep neural networks capable of detecting anomalous visual tracking behavior. Next, the network output is used to compute maximum a posteriori scores. Finally, these scores are integrated into the observation weighing mechanism of an existing data fusion algorithm. We evaluated the proposed algorithm on the OTB-100 benchmark dataset and compared its performance to the performance of the baseline fusion approach.**

I. Introduction

Designing vision-based tracking algorithms that are robust to a variety of challenging scenarios such as illumination changes and occlusions is still an elusive goal of computer vision research. While recent methods that use features based on convolutional neural networks in conjunction with sequential Monte Carlo methods have shown promising results [1], [2], [3], their robustness to substantial target appearance changes is generally dependent upon specialized mechanisms to detect that the algorithm has lost track of the target [4]. Unfortunately, these mechanisms are not algorithm-agnostic and hence must be re-designed if the underlying tracking approach changes.

Tracker ensemble methods are an effective and generally applicable tool for the design of fault-tolerant object tracking algorithms [5], [6], [7], [8]. The Hierarchical Adaptive Bayesian Data Fusion (HABDF) algorithm [9] is one such approach that uses a weighting mechanism based on the Mahalanobis distances of the predicted observations of multiple trackers to determine the level of confidence of individual trackers. However, its performance is limited by is its high susceptibility to outliers. To address this issue, we propose substituting the Mahalanobis-based weighing approach with a mechanism based on a deep convolutional autoencoder.

Autoencoders have shown great potential as an anomaly detection tool [10], [11], [12]. However, to the best of our knowledge no work has explored their use as a weighing mechanism for tracker ensembles. The methods proposed in [13], [14] are perhaps the closest to our work, albeit they were applied to the different problems of monitoring wind turbines and electrocardiograms. Our method builds on these works by utilizing multi-frame features produced by multiple Kalman filters and then integrating the outputs of the individual trackers using a weighing mechanism based on a maximum a posteriori confidence estimation method.

We use a convolutional autoencoder to learn a model that represents a tracker's normal operating conditions. The reconstruction errors generated by the network during anomalous tracking conditions is used to penalize tracking mistakes accordingly. We acquire our training data by first running a tracker ensemble on various video sequences and then partitioning the tracking results into "normal" and "anomalous" data, and use only the "normal" data for training.

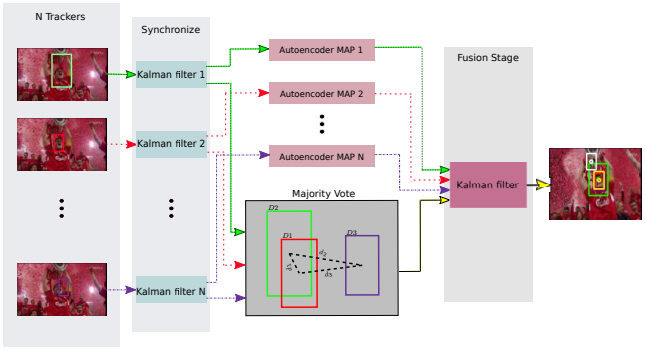


Fig. 1: Schematic representation of the proposed framework. Best viewed in color.

We build features consisting of several estimates of the positions and velocities of the target, which are generated by Kalman filters that use the outputs of the individual trackers as observations. These features also incorporate consecutive image frames, thereby integrating the temporal relationship between the outputs generated by the trackers. We use the frames from properly functioning trackers to train a separate autoencoder neural network for each of the trackers. Next, the networks are deployed and the reconstruction error for various frames is used to generate a maximum a posteriori estimate of whether the tracker result is an inlier or outlier. This score is used within an ensemble of trackers to improve upon the baseline algorithm in [9] by approximately 3%.

978-1-7281-2607-4/19/$31.00 ©2019 IEEE
DOI 10.1109/COMPSAC.2019.00077

II. Tracker Architecture

Our proposed tracker is based on the HABDF algorithm [9]. As Fig. 1 illustrates, similar to HABDF, our method fuses the outputs of N trackers (e.g., [15], [16], [17], [18]). The bounding boxes generated by each of these trackers are used as inputs to separate Kalman filters. The filter outputs are then processed by two confidence estimation methods, one based on a majority voting process and the other based on autoencoder reconstruction errors. Finally, the tracker outputs are combined by a fusion stage that uses another Kalman filter to incorporate the confidence weights determined on the previous step.

A. Individual Kalman Filters

The motion and observation models for the individual Kalman filters are given by

$$x_m = Ax_{m-1} + \nu_m \tag{1}$$

$$y_m = Cx_m + \eta_m, \tag{2}$$

where x_m is the state vector and y_m is the observation vector at frame m. Eq. (1) represents the system dynamics with $A \in \mathbb{R}^{D\times D}$ corresponding to the transition matrix and $\nu_m \in \mathbb{R}^D$ modeling the process noise. We define $x_m \in \mathbb{R}^D$ as the concatenation of b_m and $\dot{b}_m$, where $b_m = [u, v, h, w]$ consists of the vertical and horizontal coordinates of the target, u and v, and its height and width, h and w, and $\dot{b}_m = \left[\dot{u}, \dot{v}, \dot{h}, \dot{w}\right]$ is its corresponding velocity. That is, for each tracker, the dimension of the state vector $D = 8$. Although all the components of b_m as well as their velocities vary with time, we drop that dependence to simplify the notation. In Eq. (2) $C \in \mathbb{R}^{\frac{D}{2}\times D}$ is the observation matrix (i.e., only the bounding boxes are observed), and $\eta_m \in \mathbb{R}^{\frac{D}{2}}$ is the measurement noise. Both noises are assumed to be white and Gaussian with variances R_{ww} and R_{vv}.

B. Tracker Confidence Estimation

As in HABDF, our tracker uses two sources of information to determine how to combine the outputs of the individual trackers into a global estimate. The first mechanism is based on a majority voting scheme based on the pairwise Euclidean distances among the trackers, which is given by

$$m_d^{(i)} = \min_{\substack{j=1,2,\ldots,N \\ j\neq i}} (||b^{(i)} - b^{(j)}||), \tag{3}$$

where $b^{(i)}$ and $b^{(j)}$ represent the bounding boxes corresponding to trackers i and j. The majority voting weight for the i-th tracker is then computed by normalizing the distances using a hyperbolic tangent function

$$w_d^{(i)} = \omega_0 + \tanh\left(m_d^{(i)} - \lambda^{(i)}\right), \tag{4}$$

where ω_0 determines the vertical displacement of the weighting function, and λ represents a penalization offset.

Whereas HABDF's second mechanism to determine the confidence of each tracker is based on the Mahalanobis distances [19] of the observations, we generate weights for our trackers using a mechanism that reflects the probability that the tracker's output corresponds to its "normal" operation. Our approach, which we call Bayesian Autoencoder Maximum A Posteriori Data Fusion (AMAP), uses an autoencoder-based maximum a posteriori score to weigh the trackers. As explained in detail in Subection III-D, we estimate the probability $\mathcal{P}^{(i)}$ that each tracker $i = 1, \ldots, N$ is lost along with a correction offset ψ. We then employ a normalization mechanism similar to the one used in Eq. (4) for each of the probability scores $\mathcal{P}^{(i)}$

$$w_a^{(i)} = \rho_0 + \tanh\left(\kappa(\mathcal{P}^{(i)} - \psi)\right), \tag{5}$$

where ρ_0 represents the vertical displacement, and κ is the slope with which the penalization takes place. The higher the value of κ, the more abrupt the transitions. These parameters are determined heuristically based on the performance of the trackers on the visual tracking benchmark.

C. Data Fusion

In the data fusion step, the estimated state vectors $x_m^{(i)}$, are provided as inputs to another Kalman filter, which acts as the fusion center. The filter used in the fusion center is essentially identical to those applied to the individual trackers (Eqs. 1, 2) with the exception that the observation matrix C reflects the fact that the observations are given by the concatenated outputs of the N trackers. That is, $C \in \mathbb{R}^{\left(\frac{D\cdot N}{2}\right)\times D}$, where N is the number of trackers. The fusion center adapts itself to changes in the performance of individual trackers by updating its measurement noise covariance according to

$$R_{\sigma\sigma}(w_d, w_a) = W_d + W_a, \tag{6}$$

where $W_d = \text{diag}(\gamma^{(1)}w_d^{(1)}, \gamma^{(2)}w_d^{(2)}, \cdots, \gamma^{(N)}w_d^{(N)})$, $W_a = \text{diag}(\delta^{(1)}w_a^{(1)}, \delta^{(2)}w_a^{(2)}, \cdots, \delta^{(N)}w_a^{(N)})$, and diag(.) represents a diagonal matrix whose elements are the function parameters. $\gamma^{(i)}$ and $\delta^{(i)}$ are penalization weights that can be adjusted individually based on the expected performance of each tracker. That is, the majority voting weights $w_d^{(i)}$ and the reconstruction error weights $w_a^{(i)}$ are used by the global tracker to update $R_{\sigma\sigma}$, which is then used in the global correction stage of the Kalman filter to generate the global state x_f. This process allows the Kalman filter to assign lower confidence to trackers that have lower weights.

III. Anomaly Detection

We use the outputs produced by each tracker under normal operation to train N distinct autoencoders. These autoencoders are then used to estimate the confidence of the individual trackers.

A. Network Topology

Fig. 2 illustrates our network model. The network first learns to generate a reconstruction of the input and then uses a dense layer to predict the current frame based on this reconstruction. To allow the networks to differentiate between normal and anomalous scenarios, we find the frames in which a given tracker is operating normally (i.e., it has not lost track of

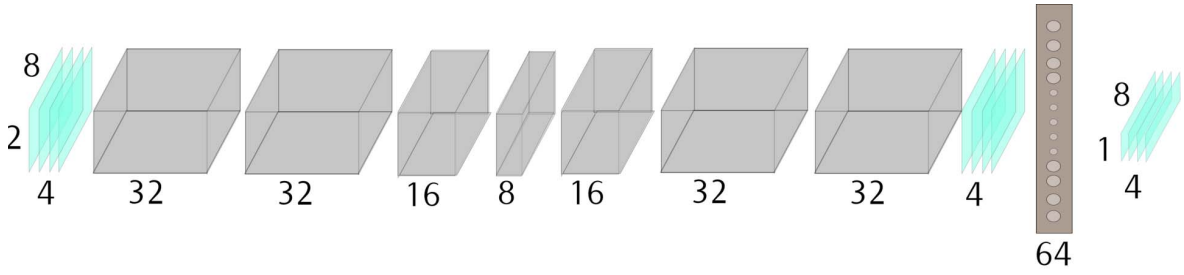


Fig. 2: Network topology. Gray boxes represent feature maps, while the dark rectangle on the right is a dense layer. The light blue slices are respectively the input, the learned reconstruction, and the current frame feature vector prediction.

the target), and use the outputs of all the trackers in the two previous frames to predict it.

At each frame, the tracker states are arranged into the array $f_m \in \mathbb{R}^{D\times N}$ according to

$$f_m = \left[x_m^{(1)^T} \dots x_m^{(N)^T}\right]^T. \tag{7}$$

The arrays f_{m-1} and f_{m-2} at the two previous frames are then arranged as the two channels of a $2 \times D \times N$ tensor, which is used as the input of the network. The autoencoder output $\tilde{f}_m \in \mathbb{R}^{D\times N}$ is then given by

$$\tilde{f}_m^{(i)} = h^{(i)}(f_{m-1}, f_{m-2}), \tag{8}$$

where $h^{(i)} : \mathbb{R}^{\mathbf{2\times D\times N}} \rightarrow \mathbb{R}^{\mathbf{D\times N}}$ is the autoencoder function (including the fully-connected prediction layer) for the network trained based on the i-th tracker data. At test time, the autoencoder reconstruction error is given by the norm of the difference between the predicted and the observed feature arrays, i.e.,

$$\varrho^{(i)} = \left\| \tilde{f}_m^{(i)} - f_m \right\|. \tag{9}$$

B. Data Partitioning

We define "normal" and "anomalous" tracker behavior according to the Jaccard index of the tracker output with respect to the ground truth annotations. That is, whenever the Jaccard index for a given tracker output is less than a threshold, that tracker's behavior is considered "anomalous". Otherwise, it is considered "normal". That is, let $\mathcal{F}$ be the dataset corresponding to the tracker outputs f_m, $m = 1, ..., M$ generated over the M frames of a set of video sequences. For each of the $i = 1, ..., N$ trackers, we create two datasets by partitioning $\mathcal{F}$ according to

$$\mathcal{F}_{O^{(i)}} = \left\{ f_m \in \mathcal{F} | J(b_m^{(i)}) < \tau \right\}, \tag{10}$$

$$\mathcal{F}_{I^{(i)}} = \left\{ f_m \in \mathcal{F} | J(b_m^{(i)}) \geq \tau \right\}, \tag{11}$$

where $J(b_m^{(i)})$ is the Jaccard index of the bounding box generated by tracker i at frame m with respect to the ground truth. $\mathcal{F}_{I^{(i)}}$ represents the set of "normal" samples for tracker i and $\mathcal{F}_{O^{(i)}}$ is its complement, i.e., the set of "anomalous" samples.

C. Network Training

We train our networks using 51 video sequences from the OTB-100 dataset and use the remaining 49 sequences as our test set. We refer to our training set as $\mathcal{F}^{(1)}$ and to our test set as $\mathcal{F}^{(2)}$. We train a separate network for each of the N trackers using the frames in $\mathcal{F}^{(1)}_{I^{(i)}} = \mathcal{F}_{I^{(i)}} \cap \mathcal{F}^{(1)}$, that is, the frames in $\mathcal{F}^{(1)}$ for which tracker i has $J \geq \tau$. Each frame is prescaled so that $-1 \leq f_m \leq 1$. The networks are trained with the RMSProp [20] optimizer for 20 epochs and with uniform Gaussian noise applied to the input.

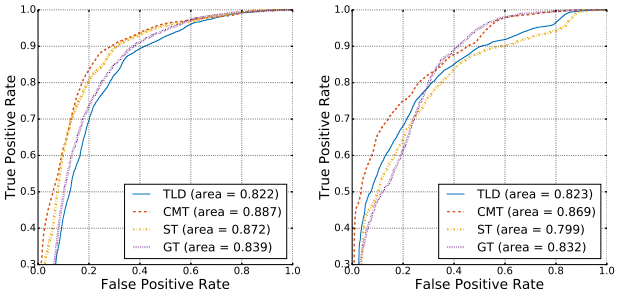


Fig. 3: ROC curves comparing the performance of the N outlier detection networks with $\tau = .3$. The left graph demonstrates performance on the training set $\mathcal{F}^{(1)}$ and the graph on the right corresponds to the test set $\mathcal{F}^{(2)}$.

To validate our method's ability to distinguish outliers from inliers, each network was tested on the "normal" and "anomalous" data in both the $\mathcal{F}^{(1)}$ and $\mathcal{F}^{(2)}$. Figure 3 shows that our networks can successfully detect outliers for the four trackers used in our experimental evaluation: TLD [16], CMT [17], Struck (ST) [18], and Goturn (GT) [15]. Because the network is relatively shallow, the time to perform each prediction of the autoencoder is less than 30ms, which can be considered real-time for most tracking applications. Further computational performance improvements should be possible by optimizing and paralellizing our implementation.

D. Autoencoder-Based Maximum a Posteriori Estimator

Using Bayes rule, we formulate the probability that the i-th tracker output corresponds to an outlier given a reconstruction

error as

$$P(O^{(i)}|\varrho^{(i)}) = \frac{P(\varrho^{(i)}|O^{(i)})P(O^{(i)})}{P(\varrho^{(i)})}, \tag{12}$$

where $P(\varrho^{(i)}|O^{(i)})$ is the probability of a certain reconstruction error given that the i-th tracker is in an anomalous state, $P(O^{(i)})$ is a static probability indicating the overall chance that tracker i is incorrect, and $P(\varrho^{(i)})$ is the probability of a given reconstruction error.

Similarly, the probability that a tracker is operating normally is then defined as

$$P(I^{(i)}|\varrho^{(i)}) = \frac{P(\varrho^{(i)}|I^{(i)})P(I^{(i)})}{P(\varrho^{(i)})}, \tag{13}$$

where $P(\varrho^{(i)}|I^{(i)})$ is the probability of a certain reconstruction error given that the i-th tracker is operating normally, and $P(I^{(i)})$ is a static probability indicating the overall chance that tracker i is correct.

The conditional reconstruction error probabilities $P(\varrho^{(i)}|O^{(i)})$ and $P(\varrho^{(i)}|I^{(i)})$ are computed based on the histograms of the reconstruction errors in the training set [21]. That is, for each tracker, we generate a discrete distribution of the reconstruction errors in $\mathcal{F}^{(1)}_{O^{(i)}}$ and $\mathcal{F}^{(1)}_{I^{(i)}}$ and use the normalized value at a given reconstruction error value $\varrho^{(i)}$ as the conditional reconstruction probability. Figure 4 shows the reconstruction error histograms corresponding to the four trackers used in our experimental evaluation. In our application, we used histograms of 30 bins in the interval between 0 and 3.

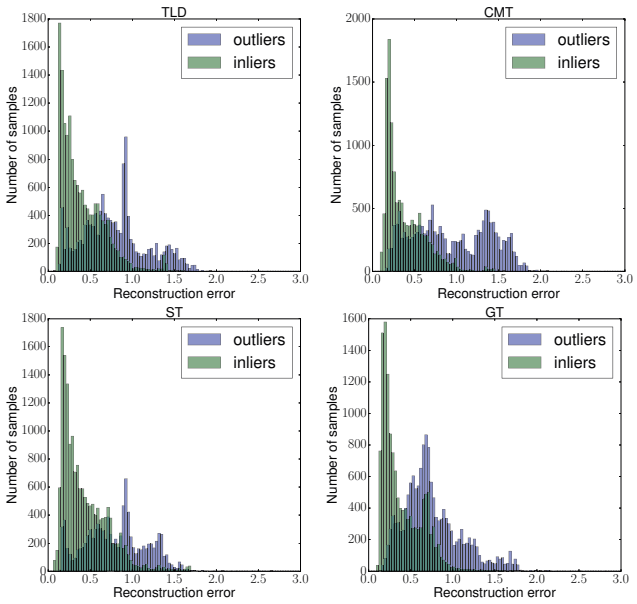


Fig. 4: Reconstruction error histograms for the four trackers for $\tau = .3$. The outliers $\mathcal{F}^{(1)}_{O^{(i)}}$ are shown in blue and inliers $\mathcal{F}^{(1)}_{I^{(i)}}$ are in green. Best viewed in color.

$P(O^{(i)})$ and $P(I^{(i)})$ are static terms that indicate the overall probability that tracker i generates an outlier or an inlier, respectively. Since the success rate of the tracker in the OTB benchmark reflects its expected performance, we use it to determine $P(I^{(i)})$ and make $P(O^{(i)}) = 1 - P(I^{(i)})$. In scenarios where no prior information about the performance of the tracker is available, it is possible to simply use $P(O^{(i)}) = P(I^{(i)}) = 0.5$.

To determine the probability that a tracker output corresponds to an outlier, we use a log-maximum a posteriori (MAP) similar the approach proposed in [22]

$$\mathcal{P}^{(i)} = \ln \frac{P(O^{(i)}|\varrho^{(i)})}{P(I^{(i)}|\varrho^{(i)})}, \tag{14}$$

which, using Eqs. (12) and (13), becomes

$$\mathcal{P}^{(i)} = \ln \frac{P(\varrho^{(i)}|O^{(i)})P(O^{(i)})}{P(\varrho^{(i)}|I^{(i)})P(I^{(i)})}, \tag{15}$$

$$\mathcal{P}^{(i)} = \ln P(\varrho^{(i)}|O^{(i)}) - \ln P(\varrho^{(i)}|I^{(i)}) + \ln P(O^{(i)}) - \ln P(I^{(i)}). \tag{16}$$

We then use $\mathcal{P}^{(i)}$ to determine the confidence of tracker i as explained in Section II-B.

E. Offset Compensation

Finally, to account for reconstruction error drift over a video sequence, we update our penalization offset parameter ψ_m at each frame using a moving average of the reconstruction errors. That is, the average reconstruction error for tracker i over the past λ frames is given by

$$\bar{\varrho}^{(i)}_m = \frac{1}{\lambda} \sum_{j=m-\lambda}^{m} \varrho^{(i)}_j. \tag{17}$$

The penalization offset is then given by a fraction $0 < \alpha \leq 1$ of the maximum reconstruction error, i.e.,

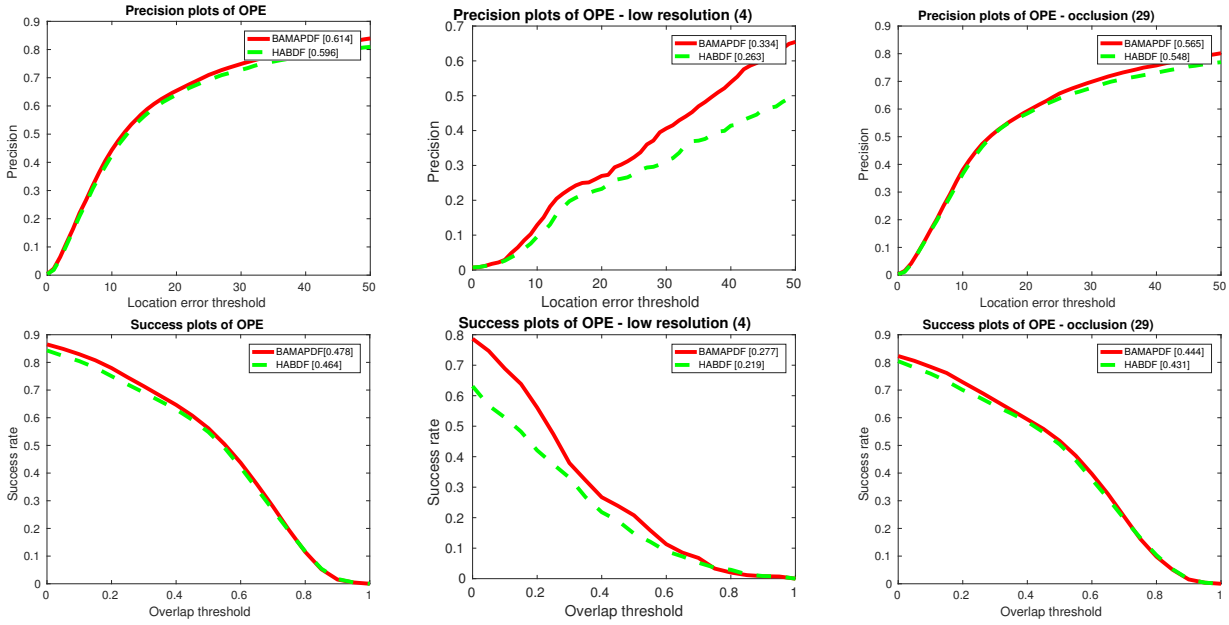
$$\psi_m = \alpha \max_{i=1,\ldots,N} \bar{\varrho}^{(i)}_m. \tag{18}$$

In our experiments, we use $\alpha = 0.5$ and $\lambda = 2$.

IV. Experimental Results

By evaluating our approach on the OTB-100 benchmark, we observe an overall relative improvement of approximately 3% with respect to the baseline method on both precision and success rates as seen in Fig. 5a. It is noteworthy that the HABDF baseline already provided a 5% increase over the best tracker in the ensemble. The increase in certain key scenarios was substantial and demonstrates that our method overcomes the main limitations of the baseline method.

The largest performance improvement was observed in low resolution sequences, where the precision rate increased by 27% and the success rate increased by 26% as shown in Fig. 5b. In this scenario, HABDF showed a performance degradation with respect to the best tracker alone. We also observe that in all the scenarios considered in the OTB-100 dataset, our approach provides results that are better than or comparable to the baseline. In Fig. 5c, which presents the results for the occlusion scenario, we notice the least improvement. In that scenario, our results are approximately



(a) (b) (c)

Fig. 5: Results of our tracker (BAMAPDF) on the One Pass Evaluation (OPE) of the OTB-100 dataset compared to the baseline method. The top row shows the precision plots and the bottom row shows the success plots. a) Overall OPE results. b) Results for the Low Resolution case, which represents the largest improvement relative to the baseline method. c) Results for the Occlusion scenario, which represents the smallest improvement.

1% better than those of the baseline. The results for all the scenarios are shown in Table I.

Fig. 6: Scenario illustrating the case when only the two worst-performing trackers in the ensembles are correct, whereas the generally superior TLD and ST fail. The blue, green, white and purple boxes correspond to the outputs of TLD, CMT, ST and GT, respectively. The yellow box is the output of the fused approach. Best viewed in color.

Fig. 6 shows one example that illustrates how our method is able to handle scenarios that used to pose a challenge for HABDF. This result is particularly important because it shows that our method is able to integrate generally weaker trackers. Since these trackers rely on alternative visual features than those used by the better performing trackers, there are cases (such as the one illustrated in Fig. 6) where they can improve the overall tracking performance. Our method is also capable of handling scenarios in which only one tracker in the ensemble is correct, as shown in Fig. 7.

Fig. 7: Success shown when only the GT tracker is correct. See the caption of Fig. 6 for a description of the colors. Best viewed in color.

V. Conclusion

We introduced the Bayesian Autoencoder Maximum Likelihood Data Fusion (AMAP) algorithm. Our method inte-

TABLE I: Summary of results on OPE.

Scenario	AMAP Score (Precision /Success)	HABDF Score (Precision /Success)	Percent Change Relative to Baseline
Total	0.614/0.478	0.596/0.464	3.02%/3.02%
illumination	0.558/0.461	0.524/0.427	5.68%/7.68%
out-of -plane rotation	0.589/0.460	0.575/0.448	2.43%/2.68%
scale variation	0.601/0.455	0.574/0.432	4.70%/5.32%
occlusion	0.565/0.444	0.548/0.431	3.10%/3.10%
deformation	0.574/0.453	0.568/0.450	1.06%/0.67%
motion blur	0.489/0.408	.450/0.366	8.67%/11.48%
fast motion	0.493/0.413	.455/0.377	8.35%/9.50%
in-plane rotation	0.588/0.462	0.556/0.439	5.76%/5.24%
out of view	0.500/0.479	0.465/0.441	7.53%/8.62%
background clutter	0.570/0.458	0.547/0.437	4.20%/4.81%
low resolution	0.334/0.277	0.263/0.219	27.00%/26.85%

grates the outputs of multiple tracking algorithms using an autoencoder-based maximum a posteriori confidence estimator. We evaluated our approach on the OTB-100 benchmark dataset, and showed that it provided performance improvements with respect to a benchmark tracker that uses a Mahalanobis distance-based tracker confidence estimation mechanism.

REFERENCES

[1] R. J. Mozhdehi, Y. Reznichenko, A. Siddique, and H. Medeiros, "Deep convolutional particle filter with adaptive correlation maps for visual tracking," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 798–802, IEEE, 2018. 1

[2] R. J. Mozhdehi and H. Medeiros, "Deep convolutional particle filter for visual tracking," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3650–3654, Sep. 2017. 1

[3] R. J. Mozhdehi, Y. Reznichenko, A. Siddique, and H. Medeiros, "Convolutional adaptive particle filter with multiple models for visual tracking," in *Advances in Visual Computing*, pp. 474–486, Springer International Publishing, 2018. 1

[4] R. Walsh and H. Medeiros, "Detecting tracking failures from correlation response maps," in *International Symposium on Visual Computing*, pp. 125–135, 2016. 1

[5] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: robust tracking via multiple experts using entropy minimization," in *European Conference on Computer Vision*, pp. 188–203, Springer, 2014. 1

[6] T. A. Biresaw, A. Cavallaro, and C. S. Regazzoni, "Tracker-level fusion for robust bayesian visual tracking.," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 25, no. 5, pp. 776–789, 2015. 1

[7] W. S. Chaer, R. H. Bishop, and J. Ghosh, "A mixture-of-experts framework for adaptive Kalman filtering," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 27, no. 3, pp. 452–464, 1997. 1

[8] C. Bailer, A. Pagani, and D. Stricker, "A superior tracking approach: Building a strong tracker through fusion," in *European Conference on Computer Vision*, pp. 170–185, 2014. 1

[9] Y. Reznichenko and H. Medeiros, "Improving target tracking robustness with bayesian data fusion," in *British Machine Vision Conference*, September 2017. 1, 2

[10] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 665–674, 2017. 1

[11] M. Lopez-Martin, B. Carro, A. Sánchez-Esguevillas, and J. R. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in IoT," in *Sensors*, 2017. 1

[12] H. Tran and D. Hogg, "Anomaly detection using a convolutional winner-take-all autoencoder," in *British Machine Vision Conference*, September 2017. 1

[13] G. Jiang, P. Xie, H. He, and J. Yan, "Wind turbine fault detection using denoising autoencoder with temporal information," *IEEE/ASME Transactions on Mechatronics*, vol. PP, no. 99, pp. 1–1, 2017. 1

[14] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-time patient-specific ECG classification by 1-D convolutional neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 63, pp. 664–675, March 2016. 1

[15] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *European Conference on Computer Vision (ECCV)*, 2016. 2, 3

[16] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012. 2, 3

[17] G. Nebehay and R. Pflugfelder, "Consensus-based matching and tracking of keypoints for object tracking," in *Winter Conference on Applications of Computer Vision*, pp. 862–869, IEEE, Mar. 2014. 2, 3

[18] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels.," in *ICCV* (D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, eds.), pp. 263–270, IEEE Computer Society, 2011. 2, 3

[19] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936. 2

[20] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016. 3

[21] R. Chalapathy, A. Krishna Menon, and S. Chawla, "Anomaly Detection using One-Class Neural Networks," *ArXiv e-prints*, Feb. 2018. 4

[22] V. A. Siris and F. Papagalou, "Application of anomaly detection algorithms for detecting SYN flooding attacks," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 4, pp. 2050–2054, IEEE, 2004. 4