Recommended Keypoint-Aware Tracker: Adaptive Real-time Visual Tracking Using Consensus Feature Prior Ranking

Ran Duan[1,2], Changhong Fu[1,2], Erdal Kayacan[1]
[1] School of Mechanical and Aerospace Engineering
[2] Singapore Technologies Engineering-NTU Corp Lab
Nanyang Technological University
50 Nanyang Avenue, 639798 Singapore
Email: {duanran, changhongfu, erdal}@ntu.edu.sg

Danda Pani Paudel
ICube-AVR
University of Strasbourg
67412 Illkirch Cedex France
Email: dppaudel055081@gmail.com

***Abstract*—This paper deals with the problem of historical feature selection for appearance model update in feature-based tracking. In particular, we convert the feature selection procedure into a ranking process where the top-N keypoint features are ranked based on the tracking histories. To the best of our knowledge, for the first time in this paper, a consensus feature prior (CFP) recommendation system is proposed that allows us to learn and update the appearance model online within a limited model size. Furthermore, the ranking scores obtained from the proposed recommendation system also provide a conviction of recovering the tracking after its failure. Extensive experiments (more than 600,000 frames) have been done by strictly following the Visual Tracking Benchmark v1.0 protocol. The results demonstrate that our method outperforms most of the state-of-art trackers both in terms of speed and accuracy.**

I. Introduction

Object tracking has been one of the biggest challenges in computer vision due to appearance change. In literature, static appearance models are considered to be robust and efficient towards a limited extent of appearance variations [1], [2], while tracking under larger changes in appearance and local deformations strictly demand an online learning of the appearance model [3]–[7]. When a target object goes under change in illumination, occlusion, and rotation, its appearance may change significantly such that the static appearance assumption becomes invalid. Likewise, if appearance model is updated using only an immediately previous frame, tracking failure in any frame causes a failure for the whole sequence. In the case of online learning, the feature-based appearance models are widely used due to their tractability for non-rigid objects, machine learning frameworks, and large image displacements. Therefore, online learning of the appearance model – selecting a good set of features from tracking history in our case – is a critical task for robust and accurate tracking. This paper investigates the problem of appearance modeling using a feature-based online learning method. We propose a method for online update of appearance model, represented by features' dictionary, that benefits from the consistency of the tracked features over the entire history.

Based on appearance modeling, tracking algorithms can be broadly categorized into two main classes: generative and discriminative. Generative methods search the best matched image region between candidate model and appearance model [8], [9]. On the other hand, discriminative methods learn differences between target and background [10], [11]. Recent methods use wide varieties of machine learning techniques, ranging from simple classification approach to advanced methods such as convolutional neural network (CNN) [12], spatio-temporal context learning [13], compressive sensing (CS) [14], and sparse representation [3]. In the same context, a notable work by Viola *et al*. [15] and its variations [6], [16], [17] use multiple instance learning (MIL) to learn the appearance model online. Although, machine learning-based methods have demonstrated their ability of successfully separating object features from background features, they still suffer from the so-called positive examples selection problem. As positive examples are taken from the current tracker location, with inevitable sub-optimality due to the imprecise tracker location, the selection procedure introduces the degradation of appearance model over time.

Alternatively, other learning-based tracking methods focus on using an efficient data structure to represent the appearance model. In this regard, tree-based approach is used for online random forests in [18], mixture of trees in [19], and tree-structured graphical models in [20]. A graph-based representation introduced by Chen *et al*. [21] formulates the tracking process as a ranking problem which is then solved by using the Google PageRank algorithm [22]. Although this algorithm, in its original form, is computationally expensive, it provides an insight towards the possibility of appearance modeling using a graph structure, thus posing the tracking as a ranking problem.

The importance of feature selection for online learning-based tracking is discussed in [23], [24]. With the recent development on tracking-by-detection [25], [26], an alternative adaptive appearance model is highly desired. Therefore, the appearance model proposed in this paper is represented by a set of historical features (*i.e*. a feature dictionary) using the consensus-based prior ranking of the keypoints. Then, the proposed recommendation system uses only the top-N keypoint features to limit the size of feature dictionary, serving as the appearance model. Once the dictionary is built, we use tracking-by-detection technique which is run in real-time, and it outperforms most of the state-of-the-art trackers in terms of both robustness and accuracy. More importantly, the proposed method demonstrates a preferable ability of handling

occlusion, rotation, and scale change of the tracked object.

The main contributions of this paper can be summarized in four folds: (i) a recommendation system for tracking-by-detection in the presence of appearance variation, scale change, rotation, and occlusion; (ii) consensus-based prior ranking of the keypoints for top-N recorded features detection; (iii) accurate estimation of tracker location using ranking score based weighted averaging; (iv) three layer matching strategies to enhance the tracking robustness.

II. Overview Of The Proposed Recommender

The main strategy of our recommendation system is shown in Fig. 1. Our appearance model falls under the generative method category which builds a dictionary of the selected features from the tracking history. Let $\mathbf{D}(d_1, d_2, ...)$ be the dynamic dictionary of features (feature vector) that keeps on updating based on the features extracted from the newly tracked windows. We use this dictionary in a feature-based tracking framework which involves three major steps: feature matching, feature ranking, and dictionary update. Let $\mathbf{F}(f_1.f_2, ...)$ be a set of features extracted from a search region, the matching step performs the matching between $\mathbf{D}$ and $\mathbf{F}$, resulting in a set of matched features, say $\mathbf{R}$, such that $\mathbf{R} \subset \mathbf{F}$. We denote $\mathbf{R}$ as the recommended set of feature points, which may contain mismatched correspondences (referred to as outliers hereafter). In this work, we identify inliers $\mathbf{C}$ and outliers $\mathbf{W}$ ($\mathbf{C}, \mathbf{W} \subset \mathbf{F}$) using the ranking weighted shift method (the details are discussed in Section III-A). During the ranking step, the inlier features are proved with their consensus priors followed by the ranking of both new inlier and dictionary features based on their voting scores. The dictionary update step constructs a new dictionary by selecting the top-N ranked features. Note that every update is likely to improve the consistency of the features in the dictionary, hence making the more suitable for matching in the next frame.

A. Consensus Feature Prior Voting

A temporal new dictionary is simply updated by $\mathbf{D}' = \mathbf{D} \cup \mathbf{F}$. To obtain optimal $\mathbf{D}$, we propose a recommender based on consensus feature prior voting. This recommendation system aims to rank the consensus features to the top. In this context,

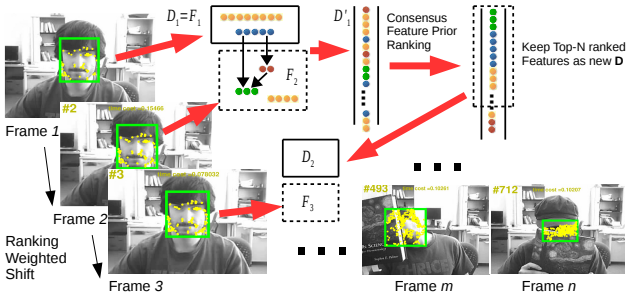

Fig. 1: Recommendation system framework: matched features, inliers, and outliers are marked by blue, green, and red colors, respectively.

each feature contributes to the voting score using the propose voting matrix. Our voting matrix relies on a "hyperlink" table $\mathbf{G}(g_{11}, g_{12}, ...)$ (referred as voting table hereafter) , inspired by the PageRank algorithm [22]. The entries g_{ij} of table $\mathbf{G}$ are binary variables, where $g_{ij} = 1$ means that the feature j votes for the feature i. We propose the following three simple rules for the voting strategies:

- All candidate features vote for themselves.
- Dictionary features vote for their correspondences.
- Outliers vote for the inliers in each frame.

More formally, table $\mathbf{G}$ is initialized as identity matrix $I^{m \times m}$ (m is number of features in $\mathbf{D}$). To obtain the voting table of $\mathbf{D}'$, *i.e.*, establish the link between $\mathbf{D}$ and $\mathbf{F}$ (contain n features), we define an *AddLink* function which is used in Algorithm 1:

$$AddLink(\mathbf{G}, \mathbf{D}, \mathbf{F}, \mathbf{R}, \mathbf{C}, \mathbf{W}) = \begin{bmatrix} G_{DF}^{m \times m} & G_{DF}^{m \times n} \\ G_{FD}^{n \times m} & G_{FF}^{n \times n} \end{bmatrix} \tag{1}$$

where G_{DF}, G_{FD} and G_{FF} consist of g_{ij},

$$g_{ij} = \begin{cases} 1 & \text{if } i = j \text{ and } i, j \in \mathbf{F} \\ 1 & \text{if } (i, j) \text{ is correspondences; } i \in \mathbf{R} \text{ and } j \in \mathbf{D} \\ 1 & \text{if } i \in \mathbf{C} \text{ and } j \in \mathbf{W} \\ 0 & \text{otherwise} \end{cases}$$

B. Updating Appearance Model

To express the consistency of feature points over time, we define a voting matrix $\mathbf{V}$ with the help of voting table $\mathbf{G}$. If v_{ij} is an entry of matrix $\mathbf{V}$, it is defined as follows:

$$v_{ij} = \alpha \frac{g_{ij}}{\sum_{i=1}^{n} g_{ij}} + (1 - \alpha) \frac{g_{ij}}{\sum_{j=1}^{n} g_{ij}} \tag{2}$$

Note that $\sum_{i=1}^{n} \sum_{j=1}^{n} v_{ij} = n$, *i.e.*, the total number of votes is constant, and the sum along the rows of $\mathbf{V}$ is given by:

$$\sum_{j=1}^{n} v_{ij} = \alpha \frac{\sum_{j=1}^{n} g_{ij}}{\sum_{i=1}^{n} g_{ij}} + (1 - \alpha) \tag{3}$$

where the scalar $\alpha \in [0, 1]$ is the confidence weight defined by the previous voting results (as described in Algorithm 1 below). When the confidence weight is maximum ($\alpha = 1$), feature i gets votes directly from its voters. In the case of no confidence, ($\alpha = 0$), the voter j distributes its vote equally to all the features that it is voting for. Consequently, the consideration of matching results (in the current frame) depends upon the value of α. A higher value recommends for higher consideration, and vice versa.

In each new frame, we initialize the voting score as 1 for all features. With the recommendation relationship (given by $\mathbf{G}$) and the vote passing rules (given by $\mathbf{V}$), the finial votes of feature can be obtained by voting iterations. To accumulate the vote, we assume each feature should have at least one vote in each voting iteration. Let $\mathbf{x}(x_1, x_2, ..., x_n)$ being the current vote vector of features in $\mathbf{D}'$, $x_i' = max(\sum_{j=1}^{n} v_{ij} x_j, 1)$. The voting iteration is shown in Algorithm 2. To guarantee a proper passing of votes, the number of iteration usually is equal to n: the dimension of $\mathbf{x}$.

Algorithm 1: AppearanceUpdating

Data: $\mathbf{D},\mathbf{F},\mathbf{R},\mathbf{C},\mathbf{W},\mathbf{G},\alpha$
Result: $\mathbf{D},\mathbf{G},\mathbf{x},\alpha$
initialization: $n = size(\mathbf{D}) + size(\mathbf{F})$;
$\mathbf{D}' = \mathbf{D} \cup \mathbf{F}$
$\mathbf{G}' = AddLink(\mathbf{G},\mathbf{D},\mathbf{F},\mathbf{R},\mathbf{C},\mathbf{W})$ % Equation 1
$\mathbf{x}$ = VotingProcess($\mathbf{G}', \alpha, n$) % Algorithm 2
$\mathbf{D} = \mathbf{D}'(k)$, $\mathbf{G} = \mathbf{G}'(k)$, where $x_k \in$ top-N ranked $\mathbf{x}$
$\alpha = \frac{\sum_{i\in\mathbf{C}} x_i}{\sum_{j\in\mathbf{R}} x_j}$

Algorithm 2: VotingProcess

Data: $\mathbf{G}, \alpha, n$
Result: $\mathbf{x}$
initialization: $x_i = 1$, $i = 1,2,...,n$;
$c_{col} = sum(\mathbf{G}, col)$; % sum of column of $\mathbf{G}$
$c_{row} = sum(\mathbf{G}, row)$;
$\mathbf{A} = diag(1/c_{col})$; % diagonal element $a_{ii} = 1/c_{col_i}$
$\mathbf{B} = diag(1/c_{row})$;
$\mathbf{V} = \alpha\mathbf{G}\mathbf{A} + (1-\alpha)\mathbf{B}\mathbf{G}$
while $iter < n$ **do**
  $\mathbf{x}' = \mathbf{V}\mathbf{x}$
  $x_i = max(x'_i, 1), i = 1,2,...,n$
  $iter = iter + 1$
end

III. Tracking Control

Once the appearance model is built, the next task is to estimate the object position in the next frame. In this section, we propose a simply yet very fast and accurate position estimation method, i.e., ranking weighted shift. Furthermore, we also present a three layer matching strategy to enhance the tracking robustness.

A. Ranking Weighted Shift

We use the ranking weighted shift vector to estimate the tracker position for the tracking purpose. When correspondences (recommend set $\mathbf{R}$) between dictionary $\mathbf{D}$ and candidate features $\mathbf{F}$ are established, the normalized weight w_i of correspondences (d_i, f_i) (where d_i and f_i are coordinates of matched feature pairs) is defined as $w_i = \frac{x_i}{\sum_{j\in\mathbf{R}} x_j}$. Here, x_i is the voting score of feature d_i. Let $m_i = f_i - d_i$ be the location shift vector, the location shift of the tracker is given by $M = \sum_{i\in\mathbf{R}} w_i m_i$. As the voting score represents the trust level of a feature, shift vectors of more trustworthy features also receive the higher weights. Denote that the shift error $e_i = \| m_i - M \|$ and their mean standard deviations σ_e, then we define $f_i \in \mathbf{C}$, if $e_i < 2\sigma_e$ and vise versa for $\mathbf{W}$.

B. Scale Adaptation

The scale can be computed by using the geometric relationships such as affine transformation. However, in the presence outliers, these methods strictly require classification of inlier/outlier correspondences. As the computation cost of the classification methods is usually expensive, we propose to estimate the scale by computing the appearance-to-candidate feature distribution ratio. If σ_f and σ_d are the mean standard deviations of the recommended feature points (candidate) and their corresponding dictionary features, respectively, the scale is updated as being σ_f/σ_d.

C. Three Layer Matching

Although the proposed method handles the appearance model very efficiently, the dynamic dictionary $\mathbf{D}$ may sometimes get updated to complete new set of feature points. This is usually not a problem. For the cases when the object sufficiently changes its appearance and recovers it back, we also keep the track of the original appearance model as a static dictionary $\mathbf{S}$. Unfortunately, original appearance recovery is very rare in practice, therefore we also build a third dictionary appearance model (confidence dictionary) $\mathbf{Q}$ which is a copy of the most recent and trustworthy $\mathbf{D}$. The candidate features are matching with these three appearance models $\mathbf{S},\mathbf{Q}$ and $\mathbf{D}$ with decreasing priority in order. In other words, $\mathbf{F}$ will match with $\mathbf{S}$ first. We assume a good match should have more than 6 matched features. If the matching results are not satisfactory, $\mathbf{F}$ will match with $\mathbf{Q}$. The dynamic dictionary $\mathbf{D}$ is used as the last option. This is very important because it dramatically decreases the tracking error caused by the accumulation of imprecise estimations. We use the voting scores to measure the matching confidence ζ of appearance model which is given as $\zeta = \frac{\sum_{i\in\mathbf{R}} x_i}{\sum_{j\in\mathbf{F}} x_j}$.

IV. Experiments

We evaluate the proposed tracker while strictly following the Visual Tracker Benchmark v1.0 protocol proposed by Y. Wu, *et al* [27] using the algorithm implemented in MATLAB 2015a. The FAST features [28] extracted in gray color space are used for the dictionary and matching. To evaluate the results, we employ two most common evaluation measures: center location error (CLE) and success rate (SR). CLE measures the distance between ground truth and estimated centers of the tracker providing the tracking precision. On the other hand, SR measures the bounding box overlap which is related to scale adaptation. For tracked bounding box be r_t and ground-truth r_a, the overlap score is defined as $score = \frac{r_t \cap r_a}{r_t \cup r_a}$. Typically, $score > 0.5$ is considered as a successful tracking. The reported measure SR is the percentage of successfully tracked frames in a sequence.

A. Tracking Results

Figure 2 shows some of the tracking results obtained using our algorithm and other trackers, i.e., Frag [1], OAB [29], MIL [6], CT [14], TLD [7], Struck [30] and LOT [31], on benchmark v1.0 datasets. Figure 3 shows the CLE comparison between our method and 6 other trackers tested on 8 different datasets. In this test, all the trackers are initialized by the ground-truth bounding box of target in first frame. Table I and II show the statistical tracking results, i.e., CLE and SR, of the proposed tracker and 7 other trackers tested on

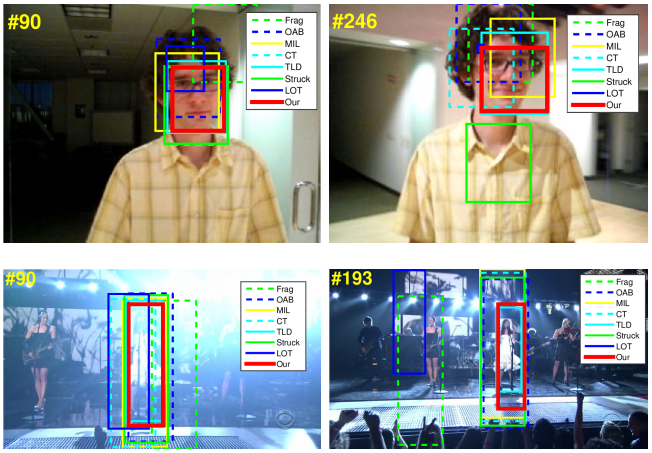

Fig. 2: Tracking result examples (Our tracking result is marked by red bounding box, the video can be found at https://youtu.be/JcAOlbnb8uo, source code is located at: https://github.com/randuan/RKA_tracker_demo.git)

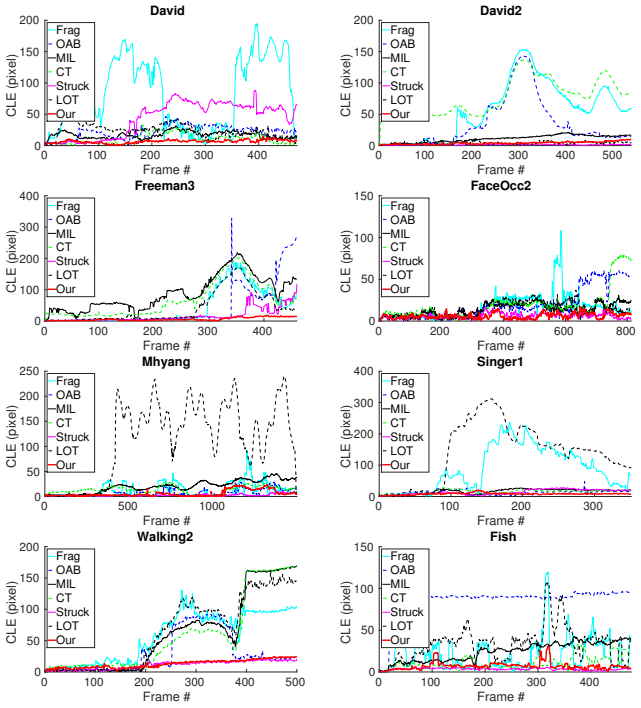

Fig. 3: Error plots for eight full sequences in benchmark

12 datasets. (Dataset *Suv* is tested only until the frame 506 since we do not consider the cases of fully occluded targets). The reader is referred to benchmark for temporal robustness evaluation (TRE) [27]. Initial bounding boxes (required for all the trackers and experiments) are obtained from ground-truth tracker positions and each dataset is tested 20 times with different starting frames.

B. Analysis

Figure 2, Table I and II illustrate the performance of the proposed method. Our method is No.2 ranked by CLE and

TABLE I: Center location error (CLE) and Frame Per Second (FPS) by TRE. Red fonts: best performance, green : second best ones,Dark Forest blue fonts: third best ones. C: C/C++, M: Matlab, MC: Mixture of Matlab and C/C++. The total number of evaluated frames was 600280.

	Frag	OAB	MIL	CT	TLD	Struck	LOT	Our
David	37.55	41.66	**18.91**	22.21	–	**17.27**	37.27	**15.24**
David2	13.98	9.48	10.57	58.73	**5.35**	**1.98**	7.81	**4.50**
FaceOcc1	**7.18**	19.58	27.85	25.12	23.78	**12.16**	28.61	**14.75**
FaceOcc2	41.10	18.21	15.99	20.11	–	**7.90**	25.52	**9.65**
Fish	20.15	45.83	24.51	**14.67**	–	**4.93**	20.67	**18.04**
FleetFace	67.64	**54.20**	63.86	73.66	–	**46.05**	**61.56**	61.61
Freeman3	**27.83**	38.53	28.77	61.30	–	29.54	**27.54**	**14.93**
Mhyang	18.21	8.49	15.12	15.35	**8.39**	**3.90**	36.04	**5.21**
Singer1	54.87	**11.81**	17.56	13.25	–	**10.26**	60.20	**6.52**
Suv*	**3.54**	6.76	31.33	14.16	**3.45**	5.09	**3.49**	4.52
Walking	22.62	**3.91**	**5.27**	6.91	8.31	**4.07**	18.52	13.02
Walking2	44.99	**18.72**	46.59	55.25	–	**6.58**	39.41	**13.81**
Average CLE	29.97	**23.10**	25.53	31.73	–	**12.48**	30.55	**15.15**
Average FPS	5.09	5.46	25.99	36.38	24.18	14.16	0.52	12.79
Code	C	C	C	MC	MC	C	M	M

TABLE II: Success rate (SR) by TRE

	Frag	OAB	MIL	CT	TLD	Struck	LOT	Our
David	36.76	24.46	50.49	53.87	**85.25**	**57.07**	35.62	**59.80**
David2	80.22	77.32	44.00	0.47	**82.55**	**98.56**	66.34	**81.90**
FaceOcc1	**99.96**	87.63	80.18	**87.96**	80.18	**100.00**	35.60	44.09
FaceOcc2	50.26	**79.87**	**86.94**	79.11	76.94	**99.12**	28.35	67.32
Fish	67.48	36.88	46.77	**76.81**	**91.85**	**96.65**	41.01	60.32
FleetFace	**46.63**	46.42	46.04	**48.75**	44.00	**47.73**	41.40	40.77
Freeman3	28.25	25.96	18.17	7.21	**56.54**	25.35	**43.58**	**37.48**
Mhyang	68.83	**91.54**	64.94	67.32	88.99	**98.65**	52.40	**91.97**
Singer1	31.13	42.13	41.99	42.70	**77.98**	42.51	**48.92**	**63.96**
Suv*	**99.08**	97.32	44.23	81.52	**100.00**	96.26	**99.65**	84.83
Walking	69.95	**75.32**	71.88	74.26	58.66	**75.44**	**90.75**	69.22
Walking2	43.64	**55.65**	44.99	37.58	44.43	**67.72**	45.62	**51.24**
Average SR	60.18	61.71	53.39	54.80	**73.95**	**75.42**	52.44	**62.74**

No.3 ranked by SR, but not as good as Struck (Struck has been evaluated as the best tracker by [27]). Our FPS processing reached 12.79 with MATLAB code (without optimization), which is expected to be a fully real-time tracking if implemented in C/C++ code.

Feature-based algorithms usually have an intrinsic drawback. When the target is blurred or mostly occluded, poor feature information is inevitable resulting in a poor tracking uncertainty. Moreover, decreasing the number of features causes an imprecise estimation of the size of bounding box. Therefore, the CLE and SR measures of our method are lower for the datasets with heavy occlusion and/or blur.

V. Conclusion

We present a new recommendation system for feature-based visual tracking. The appearance model consists of historical features of target recommended by consensus feature prior ranking process. Thus, the detected features in the next frame are more likely to find their correspondences which enhances the detection ability of the tracking-by-detection approach. The recommendation system also evaluates the confidence of tracking to support recover strategy. The proposed tracking method is fast enough to be implemented in a real-time process. The results demonstrate that our method outperforms most of the state-of-art trackers both in terms of speed and accuracy.

Acknowledgement

The research was partially supported by the ST Engineering - NTU Corporate Lab through the NRF corporate lab@university scheme.

REFERENCES

[1] A. Adam, E. Rivlin, and I. Shimshoni, "Robust Fragments-based Tracking using the Integral Histogram," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, June 2006, pp. 798–805.

[2] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2, 2000, pp. 142–149 vol.2.

[3] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust L1 tracker using accelerated proximal gradient approach," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 1830–1837.

[4] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 1838–1845.

[5] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 2042–2049.

[6] B. Babenko, M.-H. Yang, and S. Belongie, "Robust Object Tracking with Online Multiple Instance Learning," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 8, pp. 1619–1632, Aug 2011.

[7] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-Learning-Detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 7, pp. 1409–1422, July 2012.

[8] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental Learning for Robust Visual Tracking," *Int. J. Comput. Vision*, vol. 77, no. 1-3, pp. 125–141, May 2008.

[9] X. Mei and H. Ling, "Robust Visual Tracking and Vehicle Classification via Sparse Representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 11, pp. 2259–2272, Nov 2011.

[10] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised On-Line Boosting for Robust Tracking," in *Computer Vision ECCV 2008*, ser. Lecture Notes in Computer Science, D. Forsyth, P. Torr, and A. Zisserman, Eds. Springer Berlin Heidelberg, 2008, vol. 5302, pp. 234–247.

[11] C. Fu, R. A. Suarez Fernandez, M. Olivares-Mendez, and P. Campoy, "Real-Time Adaptive Multi-Classifier Multi-Resolution Visual Tracking Framework for Unmanned Aerial Vehicles," in *Research, Education and Development of Unmanned Aerial Systems (RED-UAS), 2013 2nd IFAC Workshop on*, 2013, pp. 99–106.

[12] S. Hong, T. You, S. Kwak, and B. Han, "Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network," in *Proceedings of the 32nd International Conference on Machine Learning, 2015, Lille, France, 6-11 July 2015*, 2015.

[13] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M.-H. Yang, "Fast Visual Tracking via Dense Spatio-temporal Context Learning," in *Computer Vision - ECCV 2014*, ser. Lecture Notes in Computer Science, 2014, vol. 8693, pp. 127–141.

[14] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*, ser. ECCV'12, 2012, pp. 864–877.

[15] P. Viola, J. C. Platt, and C. Zhang, "Multiple Instance Boosting for Object Detection," in *Advances in Neural Information Processing Systems 18*, January 2007, pp. 1417–1426.

[16] C. Fu, A. Carrio, M. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy, "Robust real-time vision-based aircraft tracking from Unmanned Aerial Vehicles," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 5441–5446.

[17] Z. Ni, S. Sunderrajan, A. Rahimi, and B. Manjunath, "Particle Filter Tracking with Online Multiple Instance Learning," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, Aug 2010, pp. 2616–2619.

[18] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof, "PROST: Parallel robust online simple tracking," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 723–730.

[19] F. Dellaert, V. Kwatra, and S. M. Oh, "Mixture trees for modeling and fast conditional sampling with applications in vision and graphics," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Conference on*, vol. 1, June 2005, pp. 619–624 vol. 1.

[20] S. Hong and B. Han, "Visual Tracking by Sampling Tree-Structured Graphical Models," in *Computer Vision - ECCV 2014*, ser. Lecture Notes in Computer Science, 2014, vol. 8689, pp. 1–16.

[21] C. Gong, K. Fu, A. Loza, Q. Wu, J. Liu, and J. Yang, "PageRank Tracker: From Ranking to Tracking," *Cybernetics, IEEE Transactions on*, vol. 44, no. 6, pp. 882–893, June 2014.

[22] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web." Stanford InfoLab, Technical Report 1999-66, November 1999, previous number = SIDL-WP-1999-0120.

[23] X. Liu and T. Yu, "Gradient Feature Selection for Online Boosting," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, Oct 2007, pp. 1–8.

[24] J. Yuan and F. Bastani, "Robust object tracking via online informative feature selection," in *Image Processing (ICIP), 2014 IEEE International Conference on*, Oct 2014, pp. 471–475.

[25] S. Avidan, "Ensemble Tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 2, pp. 261–271, Feb 2007.

[26] H. Grabner and H. Bischof, "On-line Boosting and Vision," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, June 2006, pp. 260–267.

[27] Y. Wu, J. Lim, and M. Yang, "Object Tracking Benchmark," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 37, no. 9, pp. 1834–1848, Sept 2015.

[28] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 1, pp. 105–119, Jan 2010.

[29] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *Proc. BMVC*, 2006, pp. 1–10.

[30] S. Hare, A. Saffari, and P. H. S. Torr, "Struck: Structured output tracking with kernels," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 263–270.

[31] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," *International Journal of Computer Vision*, 2014.