Real-time ensemble-based tracker with Kalman filter

Pedro Senna†, Isabela Neves Drummond *, Guilherme Sousa Bastos †
† Instituto de engenharia de sistemas e tecnologia da informação
* Instituto de matemática e computação
Universidade Federal de Itajubá - UNIFEI
{pedrosennapsc, isadrummond, sousa} @unifei.edu.br

***Abstract*—This work presents an ensemble-based visual object tracker called KFebT. This method can fuse using a Kalman Filter the result of several out-of-the box trackers or specialist methods that solve parts of the problem, like methods that only estimate the target scale variation. Our purpose in joining multiple trackers is to take advantage of the different strengths and weaknesses of each approach. The proposed fusion method is simple and needs no training; it just needs the tracker output result and a confidence measure for the result of each tracker. We performed tests on the Visual Object Tracking Challenge (VOT) 2015 dataset and evaluated our tracker in terms of expected overlap, accuracy and robustness. We test our proposed method on combination of two and three tracking algorithms and the results demonstrate clear improvements over the trackers used in its composition.**

I. Introduction

Visual object tracking is a computer vision area that is receiving significant attention in the last fifteen years [1]. Visual tracking has several applications, such as: traffic control [2], robotics [3], sports analysis [4]–[6], and surveillance [7]. A popular variation of this problem is the model-free where there is no prior knowledge of the target object. In this case, the algorithm will receive the first frame of a video stream and a bound box defining the object. It will also need to learn online the representation of the object and report a bound box with the object posed in the upcoming frames.

Despite the large amount of tracking work, this problem still presenting challenges due to occlusion, changes in illumination, fast motion, background clutter, geometric transformations and appearance change. Some proposed trackers focus to solve individual problems, like scale changing [8], [9], occlusion handling [10], or tracking of non-rigid objects [11]. However, most of the trackers perform poorly when they are in a non designed situation [12]. In this scenario, some proposals attempt to join the results of several trackers with distinct tracking strategies [12]–[14] achieving better results than individual ones.

At the present time, most of the more robust tracking methods show a high computational cost. The fifteen best-ranked tracker methods compared on [1] do not achieve the real-time threshold, compromising the applicability on real-time applications. When a tracking method is used in real-time applications, it may be a part of a more complex system. It is also important that the tracker does not occupy all system resources. Considering these situations, we developed a tracking method that combines the result of fast techniques to produce a superior result, while still running in real time. Our proposal is to ensemble out-of-the box trackers using a Kalman Filter [15] to fuse the result multiple real time trackers.

The contribution of this work is a method to fuse multiple tracking methods using a Kalman filter. We also use an unexpected motion penalty to produce a continuous and smooth trajectory. It is a fast method that does not need training, being ideal for the fusion of fast trackers without overloading the performance. Our proposed approach was tested in Visual Object Tracking (VOT) 2015 dataset [1] on combinations of two and three trackers. The implementation of the method proposed in this work is public available [1].

This paper is organized as follows: Section II shows the related works. In Section III we formally present our approach describing the proposed method. Section IV presents the test methodology and experimental results, and finally, Section V states the conclusion and points to future research directions.

II. Related work

Given the visual tracking problem importance, there are numerous tracking algorithms proposed in the last years using the most diverse strategies, from part-based like CMT [16], which employs optical flow and feature matching, to techniques based on convolutional neural networks [17]. We restrict our review to ensemble-based trackers.

Considering the diversity of tracking algorithms proposed in the literature in the last years, proposals that use combinations of trackers are recently appearing [12]–[14], [18].

Santner et al. [18] combine the result of three tracking methods with different levels of model adaptations in order to reduce the drift due to inaccurate models updates. Trackers results are fused in a cascade-style using hard-coded rules.

Zhang et al. [19] use a multi-expert tracking framework learned from different time intervals to solve the drift due to undesirable model updates. The best expert tracker is selected in order to maximize the entropy-based cost function. This method does not use multiple trackers, but the same tracker with different time stamps; thus, it does not solve the failures inherent to the tracker.

[1] github.com/psenna/KF-EBT

The methods closest to the one presented in this work are Wang et al. [13], Bailer et al. [14] and Vojir et al. [12]. They proposed a fusion of multiple and distinct out of the box trackers. Bailer et al. fuse the result of multiple trackers methods. The trackers run independently and the results are combined using an attraction field based strategy.

Vojir et al. propose an online training method for a Hidden Markov Model(HMM) using it to fuse the result of three trackers and a detector. Wang et al. propose a factorial HMM to aggregate the result of five state-of-the-art trackers and use a conditional particle filter to model inference and aggregate crowdsourced time series data. These ensemble-based techniques achieve better results than the individual trackers used in their composition.

III. Fusing trackers results with Kalman filter

Our proposed approach combines the result of multiple out-of-the box trackers or experts methods that solve parts of the problem, in a simple and fast way using Kalman Filter. This approach can fuse several trackers just needing a measure to evaluate their confidence results.

Two aspects of the fusion trackers process are vital: the data delivered from the tracker to the fusion algorithm and its feedback to the trackers. In order to simplify the tracker's integration, we use the tracker's result as an input to the fusion method. Using this strategy, we need to implement fewer modifications on the trackers.

The tracker fusion can be done with or without feedback [14]. When it is done without feedback, the tracker integration is easier, but if the tracker loses the right path, it can remain lost in the rest of the sequence, acting negatively in the final result. Our approach performs feedback informing the tracker's fusion result.

When a tracker presents some adaptation of its model, it should be done with the tracker own result instead of the fused result to facilitate the tracker integration.

A. Kalman filter

The Kalman filter was proposed by Kalman [15] and is used to filter and predict linear problems. It is composed of two processes, prediction, and correction. In the prediction step, the state at time t is estimated using the state at time $t-1$. This step also uses a model that approximates the system behavior(*e.g.* a physical model). In the correction step, the measured values are used to correct the predicted values.

The Kalman filter state is composed of two variables, the values x_t and the error covariance, which measures the reliable of the state values x_t.

The Kalman filter prediction process can be described by two steps, model update and error covariance estimation. The model update step is defined as follow:

$$x_t^- = Ax_{t-1} + Bu_t \tag{1}$$

where A is the system model, x_{t-1} is the last interaction state, and the term Bu_t is used to introduce some control or future knowledge in the prediction process.

The next step is the error covariance(P) estimation. This process is defined as 2

$$P_t^- = AP_{t-1}A^T + Q \tag{2}$$

where Q is the uncertainty of the prediction process.

The correction process happens when there is a new measure available. It is composed of three steps, the computation of the Kalman gain(K_t), the state correction, and the covariance estimation.

The Kalman gain defines how much the state must be changed by the measured values. It is computed as follow:

$$K_t = P_t^- H^T (HP_t^- H^T + R)^{-1} \tag{3}$$

where R is the uncertainty of the measure and H_t is the measure model.

The new state is obtained using the Kalman gain to define how much difference between the measure values z_t and the predicted values x_t^- will be added to the new state x_t (Eq. 4).

$$x_t = x_t^- + K_t(z_t - Hx_t^-) \tag{4}$$

The last step in the correction process is to estimate the resultant error covariance.

$$P_t = P_t^- - K_t * H * P_t^- \tag{5}$$

Each time the prediction process happens, the error covariance is added with the process uncertainty (Eq. 2) making the state less reliable. Otherwise, the correction process reduces the uncertainty.

B. Kalman Filter modeling

To create this Kalman filter modeling, we need to take into account the trackers limitations and features. In this work, we employ trackers that model their states as a bounding box that can be represented as its center position $C(x, y)$ and scale (S).

The Kalman filter state in time t (x_t in the matrix 6) is composed of the three variables from the tracker's states (x, y, and s), and the first and second derivations of this variables. The transition matrix A is defined below and, to simplify the problem, we consider that the time between the frames acquisition (dt) is constant, that way the problem can be modeled linearly. In this implementation, the time dt will be determined by the camera (when working with 20 frames per second camera, dt is about 50 milliseconds).

$$A = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 & \frac{dt^2}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 & 0 & \frac{dt^2}{2} & 0 \\ 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 & \frac{dt^2}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad x_t = \begin{bmatrix} x \\ y \\ s \\ \dot{x} \\ \dot{y} \\ \dot{s} \\ \ddot{x} \\ \ddot{y} \\ \ddot{s} \end{bmatrix} \tag{6}$$

An important observation is that we are dealing with random motion and have no information or control over the object trajectory. Thus, the estimation process can get poor results if the object suddenly changes its motion pattern. For this reason, the prediction uncertainty, Q in equation 2, is fixed and tends to be higher when compared with trackers uncertainty. In the case when the application uses some control or action system with the tracker, the control information can be used to improve the Kalman filter prediction process.

Considering that the target object trajectory is continuous, we employ an unexpected motion penalization (p) to produce a smooth path, penalizing the tracker that estimates the object location unlike the predicted by the Kalman filter. The penalization is computed individually for each tracker and increases the tracker uncertainty. The penalization is defined as:

$$p = ((bb_x - KFpredict_x)/bb_{width})^2 + ((bb_y - KFpredict_y)/bb_{height})^2 \quad (7)$$

where bb is the result bounding box reported by the tracker and $KFpredict$ is the bounding box predicted by Kalman filter, x_t^- in equation 1.

The Kalman filter needs a measure to define how reliable is each result during the fusion process. Thus, we use the tracker uncertainty(u), which is an exponential function that depends on tracker result confidence and on motion penalization (Eq. 8). Trackers confidence is a measure that evaluates the confidence over the result presented by the tracker. In section III-D we present the parameters used as confidence by the trackers.

$$u = e^{-(\alpha\ confidence - \beta\ p)} \quad (8)$$

Whereas the tracker's confidence can have different meanings, we use two scale parameters α and β to adjust the confidence and the penalization to reasonable values and to control the influence of each tracker. If a tracker presents better results, we can make its α slightly higher resulting in more influence in the final result. We set the same β value to all trackers to reduce adjustment in the number of parameters.

For each tracker result, the variable (x,y,s) may have different uncertainties. This modeling allows the fusion of trackers that do not work with all variables, reporting the not worked variables with the last fused result with an uncertainty higher than the tracker computed uncertainty. This strategy may be used to add more trackers that solve other problems, such as rotation, or to include expert methods on parts of the problem, as the CMT method [16] to estimating scale, while preventing results with low confidence mess the final result.

The uncertainty is used in the variable R in Eq. 3 and is represented in the matrix 9, where x_{tnx} is the uncertainty of the tracker tn for the variable x.

$$R_t = \begin{bmatrix} u_{t1x} & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & u_{t1y} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & u_{t1s} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & u_{tnx} & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & u_{tny} & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & u_{tns} \end{bmatrix} \quad (9)$$

Estimated locations are used in the z_t matrix (Eq. 4) and are structured as in Eq. 10, where tn_x is x coordinate of the object center estimated by tracker n. The measure matrix H is used to link the trackers estimations z_t with the Kalman state x_t.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} z_t = \begin{bmatrix} t1_x \\ t1_y \\ t1_s \\ \dots \\ tn_x \\ tn_y \\ tn_s \end{bmatrix} \quad (10)$$

The results reported by the trackers are fused using the Kalman filter correction process. The influence for each tracker and for the Kalman prediction in the fused result are defined during the computation of Kalman gain (Eq. 3) and is based on their uncertainty. The fused result is computed in Eq. 4 and its uncertainty is computed with Eq. 5.

C. Proposed method

The proposed Kalman filter ensemble-based tracker (KFebT) is presented in Algorithm 1.

Our method starts receiving the first frame of the sequence and the target bounding box, line 2. Then, the trackers and the Kalman filter are created and initialized, line 3 to 7 in the algorithm.

For each frame in the sequence, the process starts with the Kalman filter prediction process. Then, the trackers receive the new frame and run independently, returning their results and confidence parameters. During the correction, the Kalman filter prediction bounding box is compared with each tracker result to compute the distance penalization.

The uncertainty result of each tracker is computed using its confidence and distance penalization, and R and z_t matrices are created using the tracker's results and uncertainties which will fuse them and predict the bounding box for the next frame.

The last step is the feedback, where trackers receive the fused result from the previous frame as feedback and use it as the start point in the search for the target object in the next frame.

D. Used trackers

In this work, we use three trackers and an expert method that estimates the target object scale variation. These trackers are briefly described in this section.

Algorithm 1: KFebT - Kalman filter ensemble-based tracker

```
begin
    img, region ← getFirstFrame()
    trackersVector ← createTrackers()
    KF.create();
    foreach Tracker in trackersVector do
        tracker.init(img, region)
    end
    foreach Frame in sequence do
        KF.predict()
        img ← getNextFrame()
        foreach Tracker in trackersVector do
            tracker.track(img)
            resultsVector ← tracker.result()
            confidencesVector ← tracker.confidence()
        end
        fusedResult ← KF.correct(resultsVector,
         confidencesVector)
        foreach Tracker in trackersVector do
            tracker.feedBack(fusedResult)
        end
    end
end
```

The first tracker is the ASMS (scale adaptive mean-shift tracker), which was proposed by Vojir et al. [8] and use the mean shift algorithm over a color histogram. The ASMS uses a color histogram weighting that enhances the central object pixels and a forward-backward consistency check. As confidence measure, we use the Bhattacharyya coefficient between the target model and the target candidate histogram.

The Kernelized correlation filter (KCF) [20] uses a structure of circulant patch, which enhances its discriminative ability by working with all patch cyclic shifts, and multichannel features instead of raw pixels. We use the same parameters of the original work. As confidence measure, we use the result of the correlation process. Since KCF uses a filter with fixed size and does not estimate scale, we need to rescale the patch to fit the filter size and the uncertainty.

The last tracker is a simplification of the CMT (clustering of static-adaptive correspondences for deformable object tracking) [16], that uses keypoints matching and tracking with a consensus based voting scheme. To make this method faster and test the benefit of using a simple tracker with poor results in the fusion process, we removed the matching part of CMT. We call this simplification as the Consensus Based Tracker(CBT). The CBT has no model and does not store the keypoint descriptors like the CMT. It uses the last image and the last bounding box to find keypoints with the good features to track [21] and the same optical flow as CMT. The scale variation is computed as in CMT and the keypoints are used in the CMT consensus voting scheme. Since this simple method does not store a model of the target object, we created a target color histogram measuring the result quality computing the Bhattacharyya coefficient between this and the result found by CBT. This coefficient was multiplied by the ratio between the number of keypoints (that was assigned as belonging to the target object in the voting process) and the number of keypoints found by the feature detector in the start of the process, being used as the confidence of CBT.

We also employed the CMT method [16] to estimate scale. This method uses the median distance variation between pairs of keypoints. In our implementation, we used the good features to track algorithm [21] as feature detector. As confidence measure, we used the ratio of the points that last after the forward-backward validation. Since this method only estimates the scale variation, the target object center reported is the position received as feedback in the last frame, and the x and y variables uncertainty are seven times higher than the computed uncertainty.

We tested three combinations of these methods: ASMS and KCF (KFebT-AK), ASMS, CBT and KCF (KFebT-ACK), and ASMS, KCF and CMT scale estimation method (KFebT-AKS). α and β parameters (Eq 8) (for each tracker and for the Kalman filter uncertainty – prediction process) are presented in Table I.

TABLE I
PARAMETER USED IN THE TEST.

	AKT	ACK	AK
β	0.35	0.35	0.30
α ASMS	1	1	1.1
α KCF	1.1	1.1	1
α CBT	-	0.45	-
α CMT Scale	0.6	-	-
KF uncertainty	$0.8x10^{-4}$	$0.8x10^{-4}$	$0.8x10^{-4}$

KFebT-AK (the main combination of this work) fuses two fast trackers with good results. KFebT-ACK tests the combination of a weak tracker (CBT) with ASMS and KCF. KFebT-AKS tests the inclusion of a scale estimation method (CMT) to evaluate the fusion with methods that are not trackers.

IV. EXPERIMENTAL RESULTS

We executed three experiments using the same dataset. In the first one, the algorithm was initialized with the ground truth bound box from the first frame needing to report the estimated pose of the object for each sequential frame on the video stream. The test was run until the end of the sequence or the object lost by the tracker (the overlap between the reported bound box and the ground truth is zero). When the target was lost, the tracker was reinitialized five frames after. The results of our proposed tracker were compared against their bases trackers and the trackers presented in Visual Object Tracking challenge 2015 (VOT2015) report [1].

The second experiment (noise test) tested the behavior of the tracker when initialized with noise data. The tests were executed in the same dataset and with the same methodology, but the initialization bound box had a random perturbation in the order of ten percent of its size. Since this experiment had a

TABLE II
COMPARISON WITH TRACKERS USED IN THE COMBINATIONS USING THE VOT2015 DATASET.

	Expected overlap	Acc.	Fail.	fps
KFebT-AKS	**0.2663**	0.5185	**1.2190**	59
KFebT-ACK	0.2642	**0.5224**	1.3592	53
KFebT-AK	0.2610	**0.5224**	1.2583	109
ASMS	0.2125	0.5112	1.9736	**198**
KCF	0.1364	0.4397	3.5252	130
CBT	0.0548	0.4000	6.72	96

stochastic component, it was performed fifteen times for each sequence. The results were compared with the baseline.

The third experiment (unsupervised test) evaluated the trackers in a scenario where they do not have a supervision system to reinitialization when a failure happened. The experiment was executed only once for each tracker.

We selected the VOT2015 dataset [1] to run all our test. It is composed of 60 video sequences. All the sequences are composed of colored images with size varying from 320x180 to 1280x720 pixels. The dataset was annotated with rotated bounding box. The sequences length varies from 41 to 1500. Because of this disparity in sequences lengths, the presented results are a weighted mean from the sequences individual results (same methodology used in [1]).

We used the same performance measures as [1]: accuracy, robustness, and expected overlap. Accuracy measures the overlap between the bounding boxes estimated by the tracker and by the ground truth. Robustness measures how many times the tracker lost the target in the image sequence. Expected average overlap is a measure that represents the tracker expected average overlap on a n frames sequence.

All the following experiments were performed on a mobile computer with an Intel Core i7 5500u processor, 8 GB of RAM and a solid state disk. The implementation was made in C++ and each tracker runs in a separated threads, taking advantage of multi-core processors.

A. Comparison with base

We present bellow the experiment results for the baseline VOT2015 test. Table II displays the results for the three tested combinations of KFebT and its based trackers (ASMS, KCF, and CBT). For expected overlap, KFebT-AKS, KFebT-ACK, and KFebT-AK obtained a gain of 25%, 24% and 22% over ASMS and 95%, 93% and 91% over KCF, respectively. Comparing robustness, KFebT-AKS, KFebT-ACK, and KFebT-AK fail 38%, 31% and 36% less than ASMS and 65%, 61% and 64% less than KCF, respectively. Our proposed method also got a better accuracy than their bases, while all combinations run over 50 frames per second.

The inclusion of CBT contributes to a slight improvement in the results. Considering that CBT results are considerably weaker than ASMS and KCF, our method succeeds to fuse CBT results without disturbing the final results, making KFebT-ACK obtain a small advantage against KFebT-AK.

TABLE III
COMPARISON WITH VOT 2015 TRACKERS THAT RUN IN REAL TIME.

	Expected overlap	Acc.	Fail.	Speed (EFO)
KFebT-AKS	**0.2663**	0.5185	**1.2190**	55.45
KFebT-ACK	0.2642	**0.5224**	1.3592	49.25
KFebT-AK	0.2610	**0.5224**	1.2583	102.65
ASMS (VOT)	0.2117	0.5066	1.8464	115.09
sKCF [9]	0.1620	0.4847	2.6811	66.22
bdf [23]	0.1526	0.4014	3.1058	**200.24**
PKLTF [24]	0.1524	0.4527	2.7210	29.93
FCT [25]	0.1512	0.4314	3.3383	83.37
matflow [26]	0.1500	0.4199	3.1213	81.34
FoT [27]	0.1385	0.4316	4.3605	143.62
NCC [28]	0.0795	0.5000	11.3448	172.85

The inclusion of CBT scale estimation method in the fusion also improved tracker results. This result shows that our method can fuse incomplete trackers or just parts of trackers, making possible the use of methods that only solve part of the problem (like the scale or rotation of the target object), without the need to change the other methods used in the fusion.

B. Comparison with VOT2015

The VOT challenge has its own speed measure unit called Equivalent Filter Operations (EFO) [22], that is computed relative to a maximum filter operation, being less sensitive to changes in hardware compared to frames per second.

VOT2015 report [1] establishes the threshold of 20 EFO to consider that the tracker run in real time. We compare our proposed tracker with the trackers that overcome the real time threshold in Table III. All combinations of KFebT achieve a higher expected overlap and accuracy, and a lesser failure rate than all compared trackers. It is important to note that our proposed tracker, even if not being the faster, overcome the real time threshold by a comfortable margin while getting an expressive advantage in expected overlap compared to the other real time trackers.

The ASMS results presented in Table III is reported on VOT challenge, which probably has some minor implementation difference when compared with the ASMS used in this work.

We also compare our proposed tracker with the ten best-ranked trackers in the VOT2015 report. The trackers in VOT report are ranked in terms of expected overlap. We can consider (by the number of trackers submitted and by the fact that the challenge is open to accept any submission that exceeds a minimum quality threshold) that the best-ranked trackers are closer to the state of the art.

we present in table IV the comparison with the best-ranked trackers in VOT2015. It is important to point out that the three combinations of our tracker are the only ones to achieve the real time threshold, and none of the others trackers in this comparison get two digits in the speed measurement.

C. Noise test

In real word applications, is hard to find an initialization bounding box with the same quality to the one found in VOT

TABLE IV
COMPARISON WITH VOT 2015 BEST 10 TRACKERS.

	Expected overlap	Acc.	Fail.	Speed (EFO)
MDNet [29]	**0.3783**	**0.6033**	**0.6936**	0.87
DeepSRDCF [30]	0.3181	0.5637	1.0457	0.38
EBT [31]	0.3130	0.4732	1.0213	1.76
srdcf [30]	0.2877	0.5592	1.2417	1.99
LDP [32]	0.2785	0.4890	1.3332	5.19
sPST [33]	0.2767	0.5473	1.4796	1.03
KFebT-AKS	0.2663	0.5185	1.2190	55.45
KFebT-ACK	0.2642	0.5224	1.3592	49.25
KFebT-AK	0.2610	0.5224	1.2583	**102.65**
SC-EBT [13]	0.2548	0.5529	1.8587	0.79
NSAMF [34]	0.2536	0.5305	1.2921	5.58
STRUCK [35]	0.2458	0.4712	1.6097	2.44
RAJSSC [36]	0.2420	0.5659	1.6296	2.12

TABLE V
VOT 2015 RESULTS IN THE TEST WITH NOISE IN THE INITIATION AREA.

	Expected overlap	Accuracy	Failures
KFebT-AKS	**0.2443**	**0.4988**	**1.3699**
KFebT-ACK	0.2392	0.4950	1.4705
KFebT-AK	0.2359	0.4959	1.5292
ASMS	0.2083	0.4899	1.9500
KCF	0.1309	0.4125	3.3278
CBT	0.0543	0.2833	9.3629

ground-truth. In this scenario, we performed a test with the addition of noise in the initialization bounding box.

Table V shows the results from KFebT and the base trackers in the noise test. KFebT-AK achieved an expected overlap 13% higher than ASMS and 80% higher than KCF. Analyzing the robustness, KFebT-AK failed 21% less than ASMS and 54% less than KCF. It also archived a higher accuracy than their bases.

D. Unsupervised test

As in noisy test, the unsupervised test aims to evaluate the tracker's behavior in a scenario closer to a real world application when compared to the standard VOT test. In this test, we used the VOT2015 dataset.

Unlike the standard VOT test, in the unsupervised test the tracker is not reinitialized when it lost the target. The tracker must be capable to recover from a failure by itself, or it will continue to fail until the end of the sequence. This methodology is closer to a real world application because the tracker is unlikely to run with another system that monitors its results frame by frame and restarts or reports the correct location of the object if it fails.

TABLE VI
VOT 2015 DATASET RESULTS IN UNSUPERVISED TEST.

	Expected overlap	Accuracy
KFebT-ACK	**0.4527**	**0.3683**
KFebT-AKS	0.4507	0.3682
KFebT-AK	0.4446	0.3599
ASMS	0.4004	0.3283
KCF	0.3922	0.2537
CBT	0.1687	0.0897

The Table VI shows the results from KFebT and the base trackers in the unsupervised test. In this test, the failures are not computed and the tracker is evaluated by its accuracy and expected overlap. Analyzing the best combination, KFebT-ACK achieved an expected overlap 13% higher than ASMS and 15% higher than KCF. In terms of accuracy, KFebT-ACK gets an advantage of 12% and 45% from ASMS and KCF respectively.

V. CONCLUSION

We presented a simple and fast ensemble-based tracker using a linear Kalman filter, which can fuse the result of several out-of-the box trackers or specialist methods.

The proposed approach was tested on the VOT2015 60 video sequence dataset. The results show that our approach presents a higher expect overlap and accuracy and fail less than the methods that it is composed of. Comparing with the real-time trackers presented in VOT 2015, our approach also reaches a smaller failure rate and a higher expected overlap and accuracy. It also has a mean speed above 100 fps in the VOT2015 database (in a mobile computer), which is an appropriate value for real-time applications.

In future works, we intend to test nonlinear Kalman filters variation, like the extended [37] and the unscented [38]. We also plan to investigate a reinitiation strategy for the trackers model when it presents a high uncertainty for a long period.

ACKNOWLEDGMENT

The authors are thankful to CAPES, Brazilian funding agency for the support to this work.

REFERENCES

[1] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, "The visual object tracking vot2015 challenge results," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 1–23.

[2] I. Hwang, H. Balakrishnan, K. Roy, and C. Tomlin, "Multiple-target tracking and identity management in clutter, with application to aircraft tracking," in *American Control Conference, 2004. Proceedings of the 2004*, vol. 4. IEEE, 2004, pp. 3422–3428.

[3] H. Wang, "Adaptive visual tracking for robotic systems without image-space velocity measurement," *Automatica*, vol. 55, pp. 294–301, 2015.

[4] F. Yan, A. Kostin, W. Christmas, and J. Kittler, "A novel data association algorithm for object tracking in clutter with application to tennis video analysis," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 634–641.

[5] X. Yu, C. Xu, H. W. Leong, Q. Tian, Q. Tang, and K. W. Wan, "Trajectory-based ball detection and tracking with applications to semantic analysis of broadcast soccer video," in *Proceedings of the eleventh ACM international conference on Multimedia*. ACM, 2003, pp. 11–20.

[6] A. Ekin, A. M. Tekalp, and R. Mehrotra, "Automatic soccer video analysis and summarization," *IEEE Transactions on Image processing*, vol. 12, no. 7, pp. 796–807, 2003.

[7] I. Haritaoglu, D. Harwood, and L. S. Davis, "W 4: Real-time surveillance of people and their activities," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 809–830, 2000.

[8] T. Vojir, J. Noskova, and J. Matas, "Robust scale-adaptive mean-shift for tracking," *Pattern Recognition Letters*, vol. 49, pp. 250–258, 2014.

[9] A. Montero, J. Lang, and R. Laganiere, "Scalable kernel correlation filter with sparse feature integration," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 24–31.

[10] H. Grabner, J. Matas, L. Van Gool, and P. Cattin, "Tracking the invisible: Learning where the object might be," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1285–1292.

[11] J. Kwon and K. M. Lee, "Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1208–1215.

[12] T. Vojir, J. Matas, and J. Noskova, "Online adaptive hidden markov model for multi-tracker fusion," *arXiv preprint arXiv:1504.06103*, 2015.

[13] N. Wang and D.-Y. Yeung, "Ensemble-based tracking: Aggregating crowdsourced structured time series data," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1107–1115.

[14] C. Bailer, A. Pagani, and D. Stricker, "A superior tracking approach: Building a strong tracker through fusion," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 170–185.

[15] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[16] G. Nebehay and R. Pflugfelder, "Clustering of Static-Adaptive correspondences for deformable object tracking," in *Computer Vision and Pattern Recognition*. IEEE, Jun. 2015.

[17] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," *arXiv preprint arXiv:1510.07945*, 2015.

[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*. IEEE, 2010, pp. 723–730.

[19] J. Zhang, S. Ma, and S. Sclaroff, "Meem: Robust tracking via multiple experts using entropy minimization," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 188–203.

[20] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 37, no. 3, pp. 583–596, 2015.

[21] J. Shi *et al.*, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[22] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Cehovin, G. Nebehay, T. Vojir, and G. Fernandez, "The visual object tracking vot2014 challenge results," in *ECCV Workshop on the VOT2014 Visual Object Tracking Challenge*. Springer, 2014, pp. 1–27.

[23] M. E. Maresca and A. Petrosino, "Clustering local motion estimates for robust and efficient object tracking," in *Computer Vision-ECCV 2014 Workshops*. Springer, 2014, pp. 244–253.

[24] A. González, R. Martín-Nieto, J. Bescós, and J. M. Martínez, "Single object long-term tracker for smart control of a ptz camera," in *Proceedings of the International Conference on Distributed Smart Cameras*. ACM, 2014, p. 39.

[25] A. Varfolomieiev and O. Lysenko, "An improved algorithm of median flow for visual object tracking and its implementation on arm platform," *Journal of Real-Time Image Processing*, vol. 11, no. 3, pp. 527–534, 2016.

[26] M. E. Maresca and A. Petrosino, "Matrioska: A multi-level approach to fast tracking by learning," in *Image Analysis and Processing–ICIAP 2013*. Springer, 2013, pp. 419–428.

[27] T. Vojíř and J. Matas, "The enhanced flock of trackers," in *Registration and Recognition in Images and Videos*. Springer, 2014, pp. 113–136.

[28] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Aerospace/Defense Sensing, Simulation, and Controls*. International Society for Optics and Photonics, 2001, pp. 95–102.

[29] Y. Pang and H. Ling, "Finding the best from the second bests-inhibiting subjective bias in evaluation of visual tracking algorithms," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2784–2791.

[30] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4310–4318.

[31] G. Zhu, F. Porikli, and H. Li, "Tracking randomly moving objects on edge box proposals," *arXiv preprint arXiv:1507.08085*, 2015.

[32] A. Lukežič, L. Čehovin, and M. Kristan, "Deformable parts correlation filters for robust visual tracking," *arXiv preprint arXiv:1605.03720*, 2016.

[33] Y. Hua, K. Alahari, and C. Schmid, "Online object tracking with proposal selection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3092–3100.

[34] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *European Conference on Computer Vision*. Springer, 2014, pp. 254–265.

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

[36] M. Zhang, J. Xing, J. Gao, and W. Hu, "Robust visual tracking using joint scale-spatial correlation filters," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1468–1472.

[37] A. H. Jazwinski, *Stochastic processes and filtering theory*. Courier Corporation, 2007.

[38] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.