An Agile Framework for Real-Time Motion Tracking

Saikat Basu, Robert DiBiano, Manohar Karki, Malcolm Stagg, Jerry Weltman and Supratik Mukhopadhyay
School of Electrical Engineering and Computer Science
Baton Rouge, LA, USA
e-mail: sbasu8@lsu.edu

Sangram Ganguly
Bay Area Environmental Research Institute/NASA
Ames Research Center
Moffett Field, CA, USA

***Abstract*— We present an agile framework for automated tracking of moving objects in full motion video (FMV). The framework is robust, being able to track multiple foreground objects of different types (e.g., person, vehicle) having disparate motion characteristics (like speed, uniformity) simultaneously in real time under changing lighting conditions, background, and disparate dynamics of the camera. It is able to start tracks automatically based on a confidence-based spatio-temporal filtering algorithm and is able to follow objects through occlusions. Unlike existing tracking algorithms, with high likelihood, it does not lose or switch tracks while following multiple similar closely-spaced objects. The framework is based on an ensemble of tracking algorithms that are switched automatically for optimal performance based on a performance measure without losing state. Only one of the algorithms, that has the best performance in a particular state is active at any time providing computational advantages over existing ensemble frameworks like boosting. A C++ implementation of the framework has outperformed existing visual tracking algorithms on most videos in the Video Image Retrieval and Analysis Tool (VIRAT: www.viratdata.org) and the Tracking-Learning-Detection data-sets.**

Full Motion Video; object tracking; Confidence-based spatio-temporal filtering; Agile tracking; Ensemble algorithm

I. INTRODUCTION

Automated tracking of moving objects in a video in real time is important for different applications such as video surveillance, activity recognition, etc. Existing visual tracking algorithms [8,11,12,13] cannot automatically adapt to changes in lighting conditions, background, types of sensors (e.g., EO vs IR) and their dynamics (zooming, panning, etc.) easily. They cannot gracefully handle data that simultaneously contains different types of motions such as both slow and fast moving objects, motion behind an occlusion, etc. Many of the existing tracking algorithms [8,11,12,13] cannot start the tracking process automatically; they require a user to draw a box on an object that needs to be tracked for the process to be initiated.

We present an agile framework for automated tracking of moving objects of full motion video (FMV). The framework is robust, being able to track multiple foreground objects of different types (e.g., person, vehicle) having disparate motion characteristics (like speed, uniformity) simultaneously in real time under changing lighting conditions, background, and disparate dynamics of the camera. It is able to start tracks automatically based on a spatio-temporal filtering algorithm and is able to gracefully handle objects in occluded surroundings. Unlike existing tracking algorithms [12], with high likelihood, it does not lose or switch tracks while following multiple similar closely-spaced objects. The framework is based on an ensemble of tracking algorithms that are switched automatically for optimal performance based on a performance measure without losing state. Only one of the algorithms, that has the best performance in a particular state is active at any time providing computational advantages over existing ensemble frameworks like boosting. We prove theoretically (lemmas 1 and 2) that the presented agile tracking framework is more accurate than existing individual/ensemble-based algorithms. A C++ implementation of the framework (for the purposes of this paper, we only consider two algorithms in our ensemble: Gaussian Mixture Background Subtraction (GM) and optical flow) has outperformed existing visual tracking algorithms on most videos in the Video Image Retrieval and Analysis Tool (VIRAT: www.viratdata.org) and the Tracking-Learning-Detection [12] data-sets.

II. RELATED WORK

A spatio-temporal tracking algorithm was proposed in [11] that involved tracking articulated objects in image sequences through self-occlusions and changes in viewpoint. However, they did not provide capabilities of automatic track starting or tracking multiple objects. The work in [13] combines background subtraction, feature tracking, and grouping algorithms. However, their work didn't have any suitable classification method based on the spatial features of the objects detected. A new particle filter, Kernel Particle Filter (KPF), was proposed in the [16] for visual tracking for multiple objects in image sequences. The idea proposed in [17] shows tracking using a single classification SVM. A boosting based approach was proposed in [20] that used a cascade of classifiers for object detection. However, it didn't address the problem of tracking objects through consecutive frames of a video sequence.

Among the existing tracking frameworks the one most relevant to our work is the TLD algorithm proposed in [12]. But, the main problem inherent in this algorithm is its inability to start tracks automatically as well as lacking a multi-object tracking feature. Also, TLD is based on template matching and hence fails for videos with multiple numbers of similar looking objects.

III. THE PROPOSED APPROACH

Figure 1 shows the schematic of our approach. First, a moving object must be automatically identified as part of the foreground. This involves starting tracks at particular pixels on the subsequent frames that have a higher probability of being part of the moving foreground object. This is achieved by 1) stabilizing the image and 2) feeding the stabilized image to the spatial and temporal filtering algorithms described below. Once the track starter algorithm has precisely marked the object coordinates, the objects must be tracked if any motion is to be identified. Issues such as camera instability (shaking, panning, rotating) come into play and require image stabilization for the tracking to be successful.

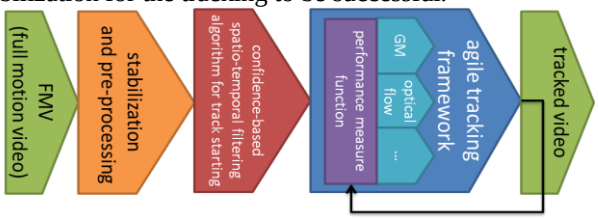

Figure 1. Schematic representation of our approach

A. Image Stabilization

An incoming video is first stabilized using an iterative algorithm:

1. Apply Shi and Tomasi's edge-finding algorithm to the first frame to identify significant feature points in the image.
2. For each subsequent frame, apply Lucas-Kanade optical flow to track the motion of the features identified by Shi and Tomasi's algorithm, refreshing the feature points when necessary.
3. With increasing precision for each iteration:

(a) For each angle of rotation in a certain range, determine the translation of each point.

(b) Find the most common (mode) translation/rotation pair (Θ, x) and (Θ, y) of all the features.

4. Warp the image to adjust for the total mode of the motion.

At present, our method stabilizes the image for small amounts of translational and rotational camera movement. Thus, for wide camera sweeps or changes in perspective or scale, our stabilization method is not, at present, appropriate.

B. Track Starting

The automated track starting algorithm based on a confidence-based spatio-temporal filtering algorithm first detects blobs using the GM Background Subtraction method [9]. This yields difference images, which are fed into the spatial filtering module below.

1) Opening or Closing images of Images via Image Morphing

The image obtained through the background subtraction algorithm is initially opened by a structuring element with diameter 3 pixels to filter out unnecessary noise. By opening, we mean the dilation of the erosion of a set A by a structuring element B. Then it is closed with k-means clustering [2]. This helps in detecting blobs over subsequent frames.

2) Spatial Filtering

Once blobs are detected in the difference images, they are filtered according to their spatial features. The pseudo code for the spatial filtering algorithm is provided below. Scale information available from the metadata accompanying the videos is used to filter blobs specifically based on their area and orientation. The filtered blobs are then passed as input to the temporal filtering algorithm below.

3) Temporal Filtering

To filter blobs in the temporal domain we use a *confidence measure*. Each blob has a confidence measure δ associated with it.

Initially the confidence value for each blob is zero. Confidence value for a blob increases as it is detected across successive frames In case a blob appears in consecutive frames, the confidence value increases according to a prior confidence measure. The confidence update equation is as follows:

Equation for confidence gain,

$$\delta = 0.5^{-n} \quad (1)$$

And, equation for confidence loss,

$$\delta = -0.5^{-n} \quad (2)$$

Where, n is the frame number.

The composite confidence update equation is as follows:

$$\delta = (0.5^{-n}) \, V \, (-0.5^{-n}) \quad (3)$$

So, the confidence update equation takes the form portrayed in Fig 2.

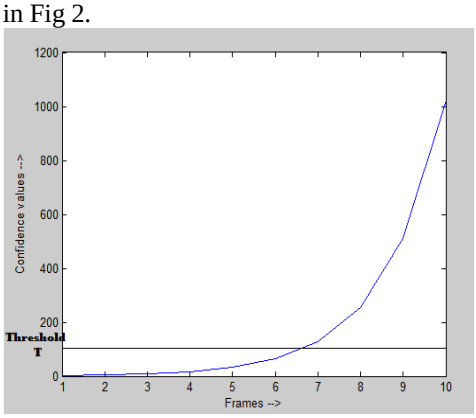

Figure 2. Confidence value update for the frames (for increasing confidence).

4) Adaptive Thresholding

If the confidence value for a blob exceeds a specified upper threshold σ, a track is started on it. . The moment the confidence value for a blob falls beneath a lower threshold τ, the corresponding object is discarded. If the confidence value is between σ and τ, the corresponding blob is maintained in the list of prospective tracks. If the confidence measure increases to a value higher than the upper threshold σ, then a track is started at the pixel representing the object coordinates. For videos that have higher noise, clutter and random changes

in lighting conditions, as is often the case for outdoor videos taken from moving cameras, the upper threshold value σ is set higher. On the other hand, for videos with more stable conditions σ is set lower because of the lesser probability of encountering random classification noise.

The track-starting algorithm:

```
begin:
img ← getFrame(video);
img ← STABILIZE_IMAGE(img);
bw_img←GM_BACKGROUND_SUBTRACTION(img);
sl ← create_structuring_element(3);
        // here 3 is the diameter of the structuring element
img ← PERFORM_OPEN_ON_IMAGE(bw_img,sl);
        // performs morphological opening on the image
sl ← create_structuring_element(n);
        // n is chosen adaptively acc. to the image
img ← PERFORM_CLOSE_ON_IMAGE(img,sl);
         //performs morphological closing on the image
contour_img ← FIND_CONTOUR(img);
        // finds the boundaries on the image
count = 0;
while(contour != NULL)
prob_obj←GET_OBJ_FROM_CONTOUR(contour_img);
    /* prob_obj contains probable object*/
   count ← count + 1;
end while
for i ← 0 to count
    temp ← SPATIAL_FILTERING(prob_obj);
end for
while temp != NULL
    obj ← TEMPORAL_FILTERING(temp);
end while
end

SPATIAL_FILTERING(prob_obj)
begin:
    if (prob_obj.size < τ1 AND prob_obj.size > τ2 AND
    prob_obj.height/prob_obj.width < τ3 AND
    prob_obj.height/prob_obj.width > τ4)
     //Here τ1 ,τ2 ,τ3 and τ4 indicate the respective thresholds
        return prob_obj;
    else
        return NULL;
    endif
end

TEMPORAL_FILTERING(temp)
begin:
    for each prob_obj
        δprob_obj ← 0;  //init. weight of each object detected as 0.
    end for
    if for video.nextframe obj_detected = prob_obj
        δprob_obj ← δprob_obj + (0.5)^-n;
    else                                    // confidence update equationss
        δprob_obj ← δprob_obj - (0.5)^-n;
    end if
    if δprob_obj ≤ τ
        remove prob_obj from list of objects;
    else
        obj ← obj Φ prob_obj;  /* append prob_obj to the list of objects
        detected. Φ represents the append operator */
    end if
    for each obj, if δprob_obj ≥ σ
        start tracks on obj(x,y);   // start tracks on object centroids (x,y)
    end for
return obj;
end
```

C. The Agile Tracking Framework

Object tracking is a matter of determining the apparent motion of the target object, keeping track of its pixel coordinates. Many object tracking methods are based on optical flow. The fundamental assumption of any method used to compute optical flow is that the intensity of the target object moves with constant velocity across frames. Existing methods like Kalman Filter[8], based on a Bayesian model and TLD[11] based on Template Matching primarily use a single learner to perform the underlying computations. In the field of machine learning, ensemble methods consist of a collection of models that can be used to improve the classification accuracy as compared to any of the individual models [3,7,10]. It can be shown through the following lemma that an ensemble learner performs better than any of the constituent learners.

Lemma 1. Even a strong learner cannot endure situational variances, i.e., it cannot perform well at all situations.

Proof. The Boosting algorithm described by Schapire and subsequently proposed implementations like Adaboost use Convex Potantial Boosters. As shown in [19], for a wide range of convex potential functions, any boosting algorithm is bound to encounter random classification noise. They show that any such boosting algorithm is able to classify examples correctly in absence of noise but in the presence of noise the learner cannot learn to an accuracy better than 1/2. This holds even if the boosting algorithm stops early or the voting weights are bounded.

Consider two sets of disjoint concept classes C1 and C2 such that $C1 \Lambda C2 = \Phi$. Now, if we consider an instance space X containing elements from C1, then any $c \in C2$ can be classified as random noise in X. So, effectively at least two different learners L1 and L2 are needed for classifying the instances in X according to C1 and C2.

In the light of this pre-defined notion, we present a new agile learning based tracker, that uses a combination of two methods for computing optical flow: Gaussian Mixture (GM) background subtraction [9] for quick-moving and the Lucas-Kanade method for slow-moving [1] objects in order to account both for fast and slow velocities. By the agile learning based tracker, we imply that our tracker can adaptively switch between the constituent learners at runtime based on the object velocities.

We present a new agile tracking framework that uses an ensemble of k individual trackers. The framework allows adaptive switching between the constituent trackers dynamically based on a performance measure. The algorithm for adaptive switching is described below.

The switching algorithm:

```
SWITCH():
j ← 1;
active_tracker ← Tj
// Tj is the jth tracker compute the performance measure λ
    if λ ≥ threshold Φ
        CHECKPOINT_CURRENT_STATE();
        //saves the current state
        active_tracker ← CALL_TRACKER_SELECTOR();
        //calls a new tracker
```

```
        state ← GET_CHECKPOINTED_STATE();
        //returns the currently checkpointed state
        state ← active_tracker(state);
    else
        continue;
    endif
    if performance measure λ is minimized
        i ← i+1
    endif
```

The switching module is called by the agile tracking algorithm.

The tracking algorithm:

```
AGILE_TRACKER(freq):
for each frame i,
    if frame_number % freq = 0
        call SWITCH();
    endif
endfor
```

In the above algorithm, state refers to the set of tuples (x,y,n,I), where x and y are the pixel coordinates, n is the frame number and I is the intensity. The agile tracker calls the switching algorithm at a user specified frequency. The switching algorithm computes the performance measure at the current state. If it exceeds a threshold the current tracker is substituted with a new one obtained from an ensemble through a pre-defined policy in such a way that the application of the new tracker to the current state results in one whose performance measure value is below the threshold. While switching, the current state is checkpointed so that it can be accessed by the new tracker. We use the linear function given below as the performance measure

$P = k_1$* stabilization_error + k_2* track_overlap_amount + k_3* probability_jump_detected + k_4* probability_drift_detected + k_5* track_speed

where $k_1, k_2, \dots, k_5$ are constants whose sum is 1 and whose values depend on the constituent trackers in the ensemble. The individual components of the equation are discussed further in the extended version of the paper. The performance measure quantifies the tracking error at the current state.

The next lemma shows that dynamic switching between individual trackers yields more accurate results.

Lemma 2. Switching between individual trackers dynamically can decrease the upper bound for error up to a certain pre-defined value.

Proof. Suppose c(v) is the correct classification for v and $h_1(v)$, $h_2(v)$ etc. are the classifications produced by the trackers T_1, T_2, etc respectively. h(v) is the estimate produced by the effective composite tracker T.

Here, $T = T_1 \Delta T_2 \Delta \dots \Delta T_n$, where, T_1, T_2, etc indicates the trackers and Δ indicates the switch operator on the trackers.

Also, let a_1, a_2, etc be the respective probabilities of error or misclassifications. Also, for switching between trackers dynamically at runtime we incorporate the idea of defining adaptive thresholds τ_1, τ_2 etc. So, we define the set $\tau = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \dots, \tau_n\}$ as the threshold for the number of misclassifications. If the number of misclassifications for a particular tracker T_i exceeds the corresponding threshold τ_i we switch the learner.

Suppose for the i^{th} tracker, the no. of misclassifications become (τ_i+1) at the $(n_i+1)^{th}$ instance. So, upto the n_i^{th} instance, probability of error or misclassification.

Also, let $\dot{\alpha}$ be the upper bound of error on any of the individual trackers. Hence, for the total tracking process, the composite probability of misclassification is given by

$$\Pr(h(v) \neq c(v)) = \Pr((h_1(v) \neq c(v)) \Lambda (h_2(v) \neq c(v)) \Lambda \dots \Lambda (h_n(v) \neq c(v)))$$

$$= \left(\frac{(n1)}{\tau 1}\right) X\, a_1^{\tau 1} \times \left(\frac{(n2)}{\tau 2}\right) \times a_2^{\tau 2} \times \left(\frac{(n3)}{\tau 3}\right) \times a_3^{\tau 3} \times \dots \times \left(\frac{(nN)}{\tau N}\right) \times a_N^{\tau N}$$

$$\leq \left(\frac{(n1)}{\tau 1}\right) \times \dot{\alpha}^{\tau 1} \times \left(\frac{(n2)}{\tau 2}\right) \times \dot{\alpha}^{\tau 2} \times \left(\frac{(n3)}{\tau 3}\right) \times \dot{\alpha}^{\tau 3} \times \dots \times \left(\frac{(nN)}{\tau N}\right) \times \dot{\alpha}^{\tau N}$$

[Since, for all i, $a_i \leq \dot{\alpha}$]

$$= \left(\frac{(n1)}{\tau 1}\right) \left(\frac{(n2)}{\tau 2}\right) \left(\frac{(n3)}{\tau 3}\right) \dots \left(\frac{(nN)}{\tau N}\right) \dot{\alpha}^{(\tau 1+\tau 2+\tau 3+\dots+\tau N)} \leq \dot{\alpha} \qquad (5)$$

Here, N is the number of switches performed at runtime.

Observations:

1) Inequation (5) holds because each of the terms $\left(\frac{(ni)}{\tau i}\right) \leq 1$ as well as $\dot{\alpha}^{(\tau 1+\tau 2+\tau 3+\dots+\tau N)} \leq \dot{\alpha}$, since, $\dot{\alpha} \leq 1$.
2) So, the overall upper bound for the error of the composite tracker is reduced owing to switching at runtime.
3) Inequation (5) proves that the effective composite error bound of the *agile tracker* T is less than any of the individual trackers T_i.

1, 2 and 3 justify our argument that using switching reduces the overall error bound.

Threshold value selection is a very important criterion in optimizing the agile tracker. In order to evaluate the threshold selection criteria, let us concentrate on the simplified version of the equation presented in (5).

So, we have, Classification error

$$\Pr(h(v) \neq c(v)) \leq \prod_{i=1}^{N} \left(\frac{(ni)}{\tau i}\right) \dot{\alpha}^{\tau i} = \prod_{i=1}^{N} \left(\frac{1}{\tau i!(ni-\tau i)!}\right) \dot{\alpha}^{\tau i} \qquad (6)$$

The error bound can be minimized by increasing τ_i until $\tau_i = \lceil n_i/2 \rceil$.

In a typical video scenario, most features are stationary from frame to frame with only a few objects moving. The stationary features are considered to be in the background, and the moving objects are foreground. The GM background subtraction method described in [9] efficiently segments foreground and background objects in real time, allowing for effective object tracking. However, as is typical of background segmentation methods, it becomes less effective when there is camera instability. Even with a stable camera, this method tends to lose foreground objects if there is relatively small movement in the foreground. To compensate for these deficiencies, we also use a more traditional and robust optical flow method for object tracking.

The Lucas-Kanade method, like many algorithms used to compute optical flow, imposes a constraint on the optical flow problem: the displacement $(\delta x, \delta y)$ of the image intensity from a pixel (x,y) to a pixel $(x+\delta x, y+\delta y)$ in the subsequent frame is small and constant over time. That is, it must satisfy for all pixels p the equation:

$$I_x(p)V_x + I_y(p)V_x = -I_t(p), \qquad (7)$$

where Ix, Iy and It are the partial derivatives of the image intensity with respect to x, y and t, and Vx and Vy are the velocity vectors. This usually results in an over-determined system and uses least-squares to find a solution. Due to the constraint imposed by the method, it is best suited for a object moving slowly with constant velocity. We use pyramidal Lucas-Kanade. That is, we compute Lucas-Kanade at the lowest-resolution image I0; then, having obtained this lower-resolution result, we compute Lucas-Kanade incrementally for the next lowest resolution I1. Similarly, we obtain I2 from I1, and so forth until reaching the full resolution.

Combined, the Lucas-Kanade method and GM background tracking ensure motion-tracking performance superior that of either method used alone.

When used on Unmanned Aerial Vehicle (UAV) videos, object tracking presents an array of challenges. One is camera instability; often, during recording, the camera shakes, pans, or rotates, which causes background objects to appear to move. A second is poor image quality due to low-definition recording equipment or long distance; this obscures images and interferes with the tracking process. A third is the need for real-time tracking, which requires simple, efficient methods to keep up with the pace of real-time input.

1) Agile Tracking vs. Other Ensemble Based Trackers

A tracker based on an ensemble machine learning technique like boosting will create, based on training data an optimal tracker of the form:

$$T = \sum_{p=1}^{P} \alpha_p t_p \tag{8}$$

where P is the number of rounds, tp is a tracker in the ensemble, and α_p are weights such that $\sum_{p=1}^{P} \alpha_p = 1$

While running on actual data T will need to run all the P trackers on each datapoint (i.e., frame) and compute a weighted sum of the outputs. In our case only one tracker is active at any particular time, i.e., only one tracker is run on each datapoint. This is crucial for real time performance.

Moreover, in boosting, the weights α_p are fixed once the training is over. This can create problems if the character of the data changes drastically from the examples on which the training is performed due to changes in background, lighting conditions, etc. This can be avoided in the agile framework by having multiple boosted trackers in the ensemble and switching them accordingly using the SWITCH() method (of course increasing the computational cost) but definitely yielding higher performance.

2) Image Quality and Real-time Tracking

How do we accommodate both poor image quality and the need for real-time tracking? The combination of GM background subtraction and the Lucas-Kanade method ensures a better result than either one alone; Lucas-Kanade tends to succeed where GM background subtraction fails, and vice versa. For a blurry, low-quality, quickly-moving object, GM background subtraction works well as long as the image is stable so that background and foreground objects can be distinguished. If a failure, defined as a large jump or the object is not moving quickly enough to show up in the GM background image, is detected, we track the object's movement according to the output of the Lucas-Kanade flow field.

Fortunately for the sake of efficiency, the incremental cost to stabilize the image is small, since Shi and Tomasi's algorithm need only run once and the Lucas-Kanade flow field is already being computed to track foreground objects.

3) Object Passing

One problem with GM background subtraction is when two moving objects are nearby or occluded, it becomes difficult to separate them. Likewise, with Lucas-Kanade, the boundaries of the tracked objects must be approximately known. To account for this, we create a probability image when two objects are nearby, consisting of a two Gaussians. The first object cannot move to where the probability is 0 (e.g. at the center of the second object), and likewise for the second object. This, along with preventing large jumps, usually solves the problem with two objects passing each other in the near vicinity.

IV. IMPLEMENTATION OF OUR APPROACH

We implemented tracking in C++ using the OpenCV library for real-time computer vision. The ensemble in our case consisted of two individual algorithms: Gaussian Mixture Background Subtraction and Lucas-Kanade optical flow (LK). The GM algorithm works well at high speeds while the LK performs well at lower speeds. The parameters k1, k2, k3, and k4 are currently equally weighted, with the exception of k5 which has been determined by the ability of LK to obtain certain track speeds. Also the switching algorithm was called by the agile tracker every frame.

TABLE I. COMPARISON OF THE VARIOUS TRACKERS

	Tot. no. of frames	BeyondSemi-Boost	coGD	CVPR	MIL
Jumping	313	14	1	96	313
Car	945	28	34	29	220
Motocross	2665	6	1	59	63
Car chase	9928	66	1	334	321
Panda	3000	130	1	358	992

V. RESULTS AND COMPARITIVE STUDIES

We compare the results from our tracker against seven existing trackers whose outputs are available at the publicly available TLD dataset [12]. Table 1 and Table 2 show the number of frames after which the trackers lost track for the first time. The measure proves to be effective in the absence of a track merging algorithm. The agile tracker performs significantly well in most of the cases. Fig. 3 shows the outputs of the agile tracker on the TLD dataset. Also TLD is based on template matching and hence fails for videos with multiple numbers of similar looking objects. This is illustrated in Fig. 4 where TLD switches tracks arbitrarily between

similar looking foreground objects whereas the agile tracker keeps tracking a particular object for the entire time frame of its visibility. The full length tracked videos along with further results on VIRAT data are available at [15].

TABLE II. COMPARISON OF THE VARIOUS TRACKERS (CONTD.)

	Tot. no. of frames	Online Boost	Semi Boost	TLD	Agile Tracker
Jumping	313	26	21	313	313
Car	945	545	652	802	581
Motocross	2665	15	59	173	110
Car chase	9928	316	190	244	402
Panda	3000	1004	83	277	2568

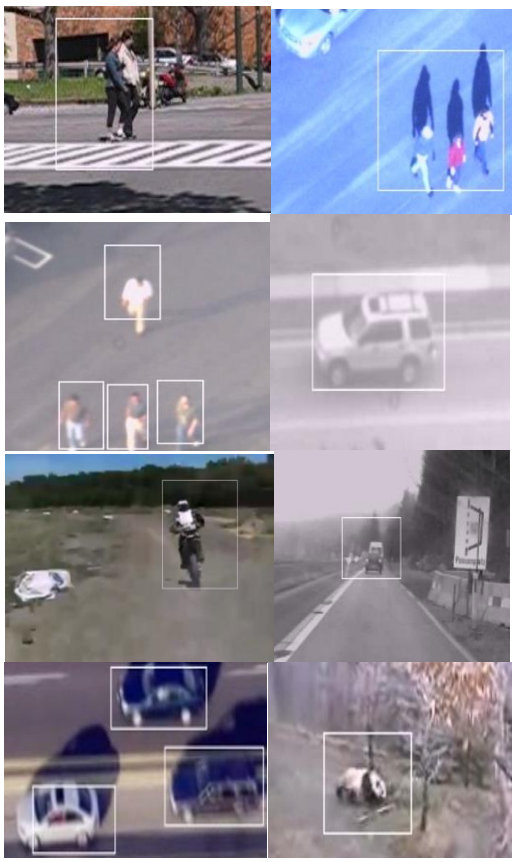

Figure 3. Results from the agile tracker

Figure 4. The left one represents the output from the agile tracker and the right one represents that from TLD.

VI. CONCLUSIONS

Our novel approach to track starting using confidence measure and adaptive thresholding not only performs in real time but is also accurate. The *agile tracking framework* allows dynamic switching within an ensemble of tracking algorithms based on a performance measure while preserving state providing more accuracy than any of the individual algorithms. We believe that the presented framework provides the foundation for real time video activity recognition.

REFERENCES

[1] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique With An Application To Stereo Vision", Proc. Seventh International Joint Conference on Artificial Intelligence (IJCAI-81), Vancouver. pages 674-679, 1981.

[2] Chris Ding and Xiaofeng He, "K-means Clustering via Principal Component Analysis", Proc. of Int'l Conf. Machine Learning (ICML 2004), pp 225-232. July 2004.

[3] D. Opitz, R. Maclin, "Popular ensemble methods: An empirical study", Journal of Artificial Intelligence Research, **11**: 169–198, 1999.

[4] J. Shi and C. Tomasi, "Good Features To Track", 9th IEEE Conference on Computer Vision and Pattern Recognition, pages- 593-600, 1994.

[5] J.Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker : description of the algorithm," OpenCV Document, Intel, Microprocessor Research Labs, 2000.

[6] Kaiki Huang and Tieniu Tan, "Vs-star: A Visual Interpretation System for Visual Surveillance", Pattern Recognition Letters, 31(14):2265-2285, 2010.

[7] L. Rokach, "Ensemble-based classifiers". Artificial Intelligence Review **33** (1-2): 1–39, 2010.

[8] N. Funk, "A study of the Kalman filter applied to visual tracking." Technical report, University of Alberta, 2003.

[9] P. Kaewtrakulpong and R. Bowden, "An Improved Adaptive Background Mixture Model For Real-Time Tracking With Shadow Detection", Proc. European Workshop Advanced Video Based Surveillance Systems, 2001.

[10] R. Polikar, "Ensemble based systems in decision making". IEEE Circuits and Systems Magazine **6** (3): 21–45, 2006.

[11] X. Lan and D. Huttenlocher. A unified spatio-temporal articulated model for tracking.CVPR, vol. 1, pp. 722–729, 2004.

[12] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N Learning: Bootstrapping Binary Classifiers from Unlabeled Data by Structural Constraints. In Conference on Computer Vision and Pattern Recognition, 2010.

[13] Z. Kim. Real time object tracking based on dynamic feature grouping with background subtraction. In CVPR, 2008.

[14] VIRAT video dataset, web source: viratdata.org

[15] Website: https://xythos.lsu.edu/users/mstagg3/web/tracker/

[16] C. Chang, R. Ansari, and A. Khokhar, "Multiple object tracking with kernel particle filter", in Proc. CVPR, 2005, pp. 566-573.

[17] O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In Proc. Int'l Conf. Computer Vision, pages 353–360, Nice, France, 2003.

[18] R. Schapire. The boosting approach to machine learning: An overview. In MSRI Workshop on Nonlinear Estimation and Classification, 2001.

[19] P. M. Long and R. A. Servedio, "Random classification noise defeats all convex potential boosters", In International Conference on Machine Learning, pages 608–615, 2008.

[20] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", In Computer Vision and Pattern Recognition, volume 1, pages 511-518, 2001.