REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
	Technical Report	-

4. TITLE AND SUBTITLE	
AN AGILE FRAMEWORK FOR REAL-TIME VISUAL TRACKING IN VIDEOS	5a. CONTRACT NUMBER W911NF-10-1-0495
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 0G10BC
6. AUTHORS Saikat Basu, Malcolm Stagg, Robert DiBiano, Manohar Karki, Supratik Mukhopadhyay, Jerry Weltman	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES	8. PERFORMING ORGANIZATION REPORT NUMBER
Louisiana State University and A&M College Office of Sponsored Programs Louisiana State University and A&M College Baton Rouge, LA 70803 -0000	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	ARO
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) 58840-CS-DRP.3

12. DISTRIBUTION AVAILIBILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. 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

15. SUBJECT TERMS

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

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Supratik Mukhopadhyay
UU	UU	UU	UU		19b. TELEPHONE NUMBER 225-578-1496

Standard Form 298 (Rev 8/98)
Prescribed by ANSI Std. Z39.18

Report Title

AN AGILE FRAMEWORK FOR REAL-TIME VISUAL TRACKING IN VIDEOS

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

AN AGILE FRAMEWORK FOR REAL-TIME VISUAL TRACKING IN VIDEOS

Saikat Basu[1], Malcolm Stagg, Robert DiBiano, Manohar Karki, Supratik Mukhopadhyay, and Jerry Weltman

Louisiana State University, Baton Rouge{[1]sbasu8@lsu.edu}

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 [12] data-sets.

***Keywords* —** Full Motion Video, object tracking, Confidence-based spatio-temporal filtering, Agile tracking, Ensemble algorithm

1. 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 [8,11,12,13,21,22,23,24] 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 [8,12] 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 provides 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 spatial classification algorithm based on blob sizes and aspect ratio allows our framework to distinguish vehicles from humans. A C++ implementation of the framework (for the purposes of this paper, we consider three algorithms in our ensemble: Gaussian Mixture Background Subtraction (GM), a color histogram approach, 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.

1.1. RELATED WORK

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.

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. Also unlike our framework, the approach in [11] does not involve adapting to changing (environmental condition/data distribution) through agile dynamic switching of trackers based on a performance measure. The work in [13] combines background subtraction, feature tracking, and grouping algorithms. However, their work doesn't have any suitable classification method based on the spatial features of the objects detected.

Among the existing tracking frameworks, the one most relevant to our work is the TLD algorithm proposed in [12]. But, a 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 (e.g., in the Indian driving scene video, Figure 4).

The approach proposed in [22] uses color histograms as the only feature. They use a cascade composition of a particle filter and mean shift. The approach of [22] does not adaptively switch between multiple trackers at runtime based on a performance measure, unlike our framework. Also, the approach is limited to two fixed algorithms (particle filter and mean shift) whereas in our framework an ensemble consisting of a plurality of algorithms can be used providing more flexibility. For example, for the embodiment of the framework used for the experiments in Section 4, we used Gaussian mixture background subtraction method, color histogram and flow, as well as stateful switching between an ensemble of trackers. The method proposed in [23] uses an incremental update function to learn the object model. It uses principal component analysis to update the sample mean and uses a forgetting factor for older observations. We use a spatial classification algorithm based on blob sizes and aspect ratio allows our framework to distinguish vehicles from humans. The work in [23] does not provide insights on the way the track is started, manually or automatically. The method proposed in [24] is similar to the approach proposed in TLD. The difference between this and TLD is that they use multiple instances as the positive examples in each frame. However, like TLD, they lack the ability to start tracks automatically as marking the location of the object initially is a pre-requisite.

An approach on multi-target object detection is proposed in [30] while [28, 31] enumerate approaches to target tracking based on Markov models and Gait recognition respectively. Another method for detecting event sequences in surveillance videos is proposed in [29] but it is applicable only to videos at very low frame rate.

None of these approaches are based on stateful dynamic switching between an ensemble of trackers based on a performance measure. Our tracking architecture is also parallelized resulting in an efficient implementation for real time visual tracking.

2. THE PROPOSED APPROACH

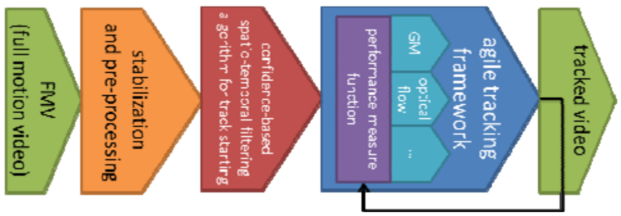

Figure 1: Schematic representation of our approach

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.

2.1. IMAGE STABILIZATION

In nearly all Full Motion Video (FMV), there is at least slight camera motion. Aerial videos in particular typically contain jitter as well as significant rotational and translational camera motion. For good quality tracking, a tracker must be robust to this significant camera motion. If the camera moves even slightly, the GM background subtraction algorithm, an algorithm used to detect motion and determine the target object's location, will incorrectly detect stationary objects as moving.

In order to stabilize an incoming streaming video, we use the following iterative algorithm which attempts to hold each background pixel in the same position regardless of lateral and rotational camera motion:

1. Apply Shi and Tomasi's [4] 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 [1] 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 translational and rotational motion.

Before we can adjust for background motion, we must identify features of the frame; to do so, we use the Shi-Tomasi method [4]. The Shi-Tomasi method detects features such as corners and edges by approximating the weighted sum of squares of image patches shifted by certain values. The approximation results in the vector (x, y) multiplied by the structure tensor, for which there are two eigenvalues λ_1 and λ_2; if either or both is large and positive, an edge or corner is found.

Next, we apply a pyramidal Lucas-Kanade method for determining optical flow at each point of interest. We then find the mode of the resulting flow value pairs, including rotation, by placing the pairs in bins. Each iteration, the bin widths are decreased, yielding an increasingly accurate estimate of the motion. The image is then adjusted to account for the determined background movement. When the image is stabilized in this manner, not only are fewer false foreground objects detected, but the correct coordinates of objects are also maintained.

If a stabilization failure is detected from Lucas-Kanade (LK) flow having many points with a large mean-square error distance (due probably to video corruption, or a perspective motion for which we do not compensate), stabilization transforms of nearby frames are interpolated, and GM background is considered unreliable so LK flow and the color histogram model are used exclusively for these frames.

At present, our method stabilizes the videos 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 appropriate. Scale compensation, however, may be integrated similarly to rotation.

2.2. 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.

2.2.1. OPENING OR CLOSING 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.

2.2.2. 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.

2.2.3. 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,

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

And, equation for confidence loss,

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

Where, n is the frame number.

The composite confidence update equation is as follows:

$$\delta = (0.5^{-n}) \vee (-0.5^{-n}) \quad \ldots (3)$$

Where, *0.5* indicates the increase in confidence, *-0.5* the decrease in confidence and *n* is the frame number.
So $-n$ denotes that as frame number increases, the confidence keeps increasing either in positive or negative direction.
The confidence update equation takes the form portrayed in fig 2.

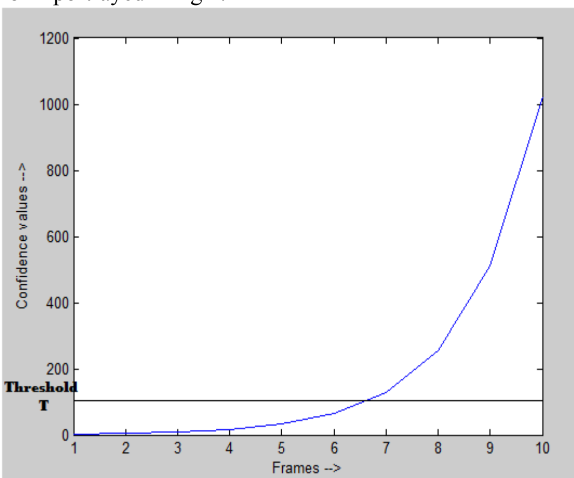

Figure 2: Confidence value update for the frames (for increasing confidence).

2.2.4. 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); /* GET_OBJ_FROM_CONTOUR finds each element from the list of contours */
                                               /* 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
        δ_prob_obj ← 0;                        /* intialize weight of each object detected as 0. */
end for
if for video.nextframe obj_detected = prob_obj
        δ_prob_obj ← δ_prob_obj + (0.5)^-n;
Else                                           /* confidence update equations */
        δ_prob_obj ← δ_prob_obj - (0.5)^-n;
end if
if δ_prob_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 δ_prob_obj ≥ σ
       start tracks on obj_(x,y);    /* start tracks on the pixel(x,y) representing the centroids of objects */
end for
return obj;
end
```

2.3. 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 statistics and machine learning, *ensemble methods* use multiple models to obtain better predictive performance than could be obtained from any of the constituent 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 in all *situations*.

Proof. The Boosting algorithm described by Schapire and subsequently proposed implementations like Adaboost use Convex Potential 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* C_1 and C_2 such that $C_1 \cap C_2 = \Phi$. Now, if we consider an instance space X containing elements from C_1, then any $\square \in C_2$ can be classified as random noise in X. So, effectively at least two different learners L_1 and L_2 are needed for classifying the instances in X according to C_1 and C_2.

In the light of the above lemma, we present a new *agile learning* based tracking framework that dynamically switches between an ensemble of classifiers based on a performance measure while preserving state to deal with unforeseen situational variances. An embodiment of the framework with which experiments in Section 4 were conducted uses a combination of three methods for tracking object motion: Gaussian Mixture (GM) background subtraction [9] with *mean-shift*, Lucas-Kanade (LK) optical flow [1], and a color-histogram [32] approach also utilizing *mean-shift*. A combination of these algorithms allows our tracker to track fast, slow, stopped, and partially-occluded objects. By an agile learning based tracker, we imply that our tracker can adaptively switch dynamically between the constituent learners at runtime based on velocities and certain measures of track quality while preserving state. The next lemma proves that dynamic switching between the learners at runtime yields more accurate results.

The new agile tracking framework uses an ensemble of *k* individual trackers. It 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                               // Note: 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 below:

The tracking algorithm:

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

In the above, 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 then 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 a state 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. For the current embodiment of the framework, 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 * high_track_speed + k_6 * low_track_speed

where $k_1, k_2, \ldots, k_6$ are constants whose sum is 1 and whose values depend on the constituent trackers in the ensemble. Drift is defined as a lack of movement of the track while there is foreground motion present which would cause the track to continue to move.

GM background performs poorly during stabilization failure, moderately well during track overlaps (when combined with the object passing algorithm described in 2.3.4), sometimes jumps to background noise, tends not to drift, and works best for fast moving objects. Thus, for GM, k_1 is large, k_2 is slightly smaller, k_3 is large, k_4 and k_5 are 0, and k_6 is large. For simplicity, assume $k_1=k_3=k_6=0.3$, $k_2=0.1$, $k_4=k_5=0$.

The color histogram tracker performs well during stabilization failures, moderately well during track overlaps, rarely jumps, occasionally drifts, and performs well for fast or slow moving objects, though is especially good for slow or stopped objects. For this tracker, reasonable parameters are $k_1=k_6=0$, $k_2=k_3=0.25$, $k_4=0.4$, $k_5=0.1$.

LK flow performs quite well during stabilization failure, very well during object passing, typically does not jump, though often drifts, and performs best for slow, but not stopped, objects. Assuming high_track_speed and low_track_speed are both small for objects moving at such a moderate speed, reasonable values are $k_1=k_2=k_3=0$, $k_4=0.5$, $k_5=k_6=0.25$.

Parameters such as k_4-k_6 may be determined experimentally, since the ability to track at specific speeds is not prior knowledge, and may also vary based on the type of video to be tracked. The performance measure quantifies the tracking error at the current state. If more information regarding the video characteristics is known, it may be beneficial to experimentally adjust the performance measure based on those characteristics.

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, $T = T_1 \Delta T_2 \Delta \ldots \Delta T_n$, 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 $\tau = \{ \tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \ldots, \tau_n \}$ 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, up to the $n_i^{\ th}$ instance, probability of error or misclassification

$$\Pr(h_i(v) \neq c(v)) = \left(\frac{\binom{\square\square}{\tau i}}{ni!}\right) \text{ X } a_i \qquad \ldots (4)$$

Also, let $\square$ 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

$$\Pr(h(v) \neq c(v)) = \Pr((h_1(v) \neq c(v)) \wedge (h_2(v) \neq c(v)) \wedge \ldots \wedge (h_n(v) \neq c(v)))$$

$$= \left(\frac{\binom{\square 1}{\tau 1}}{n1!}\right) \text{X } a_1^{\tau 1} \text{ X } \left(\frac{\binom{\square 2}{\tau 2}}{n2!}\right) \text{X } a_2^{\tau 2} \text{ X } \left(\frac{\binom{\square 3}{\tau 3}}{n3!}\right) \text{X } a_3^{\tau 3} \text{ X } \ldots \text{ X } \left(\frac{\binom{\square\square}{\tau N}}{nN!}\right) \text{X } a_N^{\tau N}$$

$$\leq \left(\frac{\binom{\square 1}{\tau 1}}{n1!}\right) \text{X } \square^{\tau 1} \text{ X } \left(\frac{\binom{\square 2}{\tau 2}}{n2!}\right) \text{X } \square^{\tau 2} \text{ X } \left(\frac{\binom{\square 3}{\tau 3}}{n3!}\right) \text{X } \square^{\tau 3} \text{X } \ldots \text{ X } \left(\frac{\binom{\square\square}{\tau N}}{nN!}\right) \text{X } \square^{\tau N} \qquad [\text{Since, for all i, } a_i \leq \square]$$

$$= \left(\frac{\binom{\square 1}{\tau 1}}{n1!}\right) \left(\frac{\binom{\square 2}{\tau 2}}{n2!}\right) \left(\frac{\binom{\square 3}{\tau 3}}{n3!}\right) \ldots \left(\frac{\binom{\square\square}{\tau N}}{nN!}\right) \square^{(\tau 1+\tau 2+\tau 3+\ldots+\tau N)} \leq \square \qquad \ldots (5)$$

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

Observations:

1) Inequation (5) holds because each of the terms $\left(\frac{\binom{\square\square}{\tau i}}{ni!}\right) \leq 1$ as well as $\square^{(\tau 1+\tau 2+\tau 3+\ldots+\tau N)} \leq \square$, since, $\square \leq 1$.
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 that 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

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

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 with the mean-shift algorithm. However, as is typical of background segmentation methods, it becomes less effective when there is uncompensated 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 (LK) method, like many algorithms used to compute optical flow, imposes a constraint on the optical flow problem: the displacement (δx, δy) 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:

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

where I_x, I_y and I_t are the partial derivatives of the image intensity with respect to x, y and t, and V_x and V_y 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 an object moving slowly with constant velocity. We use pyramidal LK. That is, we compute LK at the lowest-resolution image I_0; then, having obtained this lower-resolution result, we compute LK incrementally for the next lowest resolution I_1. Similarly, we obtain I_2 from I_1, and so forth until reaching the full resolution.

Combined, the LK method and GM background tracking ensure motion-tracking performance superior that of either method used alone. However, neither method performs well on objects which are stopped; GM tends to jump to nearby moving objects and noise while LK drifts significantly over time. To track these objects, we introduce a third method based on a color histogram of the object being tracked. We create a model of the object based on the frequency of red, green and blue intensities in the foreground and background, as obtained from GM.

This model slowly updates over time. A probability image is created, which is an image where each pixel value corresponds to the predicted probability of the object existing at that point. Each pixel probability is computed from the color histograms of the region of interest using the equations $P(x,y) = P_f(x,y)*P_f(x,y)/(P_f(x,y)+P_b(x,y))$, where P_f is the normalized frequency of each RGB value on the foreground histogram and P_b is the frequency on the background histogram. Mean-shift is then used on the probability image to track the object by re-centering the region of interest on the center of mass of the probability image.

When used on full motion 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.

2.3.2. AGILE TRACKING VS. OTHER ENSEMBLE BASED TRACKERS

A tracker based on an ensemble machine learning technique like boosting (e.g., Adaboost) will create, based on training data an optimal tracker of the form:

$$T = \sum_{p=1}^{P} \alpha_p t_p \qquad \ldots (8)$$

where P is the number of rounds, t_p 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 data point (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 data point. 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.

2.3.3. IMAGE QUALITY AND REAL-TIME TRACKING

In the current embodiment of the framework, handling poor image quality with real-time tracking is primarily handled through the use of the three tracking algorithms. Utilizing all these methods ensures a better result than any one alone; LK flow succeeds 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 well-stabilized. LK flow can track slower objects well, and works without stabilization information. The color histogram also works during stabilization failure, and can track stopped objects better than the other algorithms. If a failure is detected in any algorithm, defined as the performance measure exceeding a certain threshold, we can simply switch to another algorithm and continue tracking.

LK flow is used for stabilizing the image, so the marginal cost to use it as a tracking algorithm is relatively low. Using the stated algorithms together is an efficient choice to achieve real-time tracking.

2.3.4. 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. Even the color histogram model will not always separate two similarly colored cars. To account for this, we create a *probability image* when two objects are nearby, consisting of two Gaussians. This probability image contains pixel values equal to the expected probability of the object being centered at each pixel location, based on the expected motion of each object. 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 when two objects pass each other in the near vicinity.

2.3.5. DISTINGUISHING BETWEEN HUMANS AND CARS

A track is classified as a human or car based on the blob size and aspect ratio. A confidence measure is built up over time, and, if necessary, a correction may be made. If either the area or aspect ratio alone is a strong indicator of the presence of a car, then only this metric is used to devise the classifier. Otherwise, both area and aspect ratio must be used. σ_{car} and σ_{human} are initially set to zero, so an estimate may be immediately obtained. They should then be changed to nonzero values to prevent fluctuations between human and car detections due to inaccurate blobs. The size and aspect ratio of the region of interest used depends on the human or car classification.

The human/car classification algorithm:

```
begin:
sz ← getBlobSize(track);
if sz.width*sz.height>τcar_area1 OR sz.width*sz.height>τcar_area2 AND sz.width/sz.height>τcar_aspect2 OR  sz.width/sz.height>τcar_aspect1
          /* τcar_area1 and τcar_aspect1 are thresholds where it is almost certain that the tracked object is a car. τcar_area2 and τcar_aspect2 have a
          lower confidence */
            probableCar ← 1;
            probableHuman  ← 0;
else
```

```
        probableCar ← 0;
        probableHuman ← 1;
endif
carConfidence ← carConfidence + probableCar - probableHuman;
if carConfidence> σ_car
        track is a car            /* σ_car is generally > 0 and σ_human < 0 */
else if carConfidence< σ_human
        track is a human
endif
end
```

3. 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 three individual algorithms: Gaussian Mixture Background Subtraction with *mean-shift*, Lucas-Kanade optical flow, and the color histogram model with *mean-shift*. The selection of k_1 to k_6 is explained above in section 2.3, and may vary based on known tracking algorithm characteristics. The switching algorithm is called by the agile tracker every frame.

4. 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 shows the number of frames after which the trackers lost track for the first time. Table 2 gives the number of frames up to the first track loss for the TUD dataset [21]. 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].

We also compare our tracker against the TUD Pedestrian Detector for multi-object tracking. A measure of the total number of correct and false object detections is used.

Algorithms	**Jumping** Total number of frames=313	**Car** Total number of frames=945	**Motocross** Total number of frames=2665	**Car chase** Total number of frames=9928	**Panda** Total number of frames=3000
Beyond semi-supervised tracking [12]	14	28	6	66	130
Co-trained Generative-Discriminative tracking [12]	11	34	1	1	1
"CVPR" results as given in [12]	96	29	59	334	358
Online Multiple Instance Learning [12]	313	220	63	321	992
On-line Boosting [12]	26	545	15	316	1004
Semi-Supervised On-line Boosting [12]	21	652	59	190	83
TLD [12]	313	802	173	244	277
Agile Tracker	313	581	110	402	2568

Table 1. Comparison of the various single-object trackers

	Campus Correct (False)	Crossing Correct (False)
Expected Detections	303	1008
TUD Pedestrian Detector [21]	227 (0)	692 (7)
Agile Tracker	222 (0)	541 (28)

Table 2. Comparison of the multi-object trackers

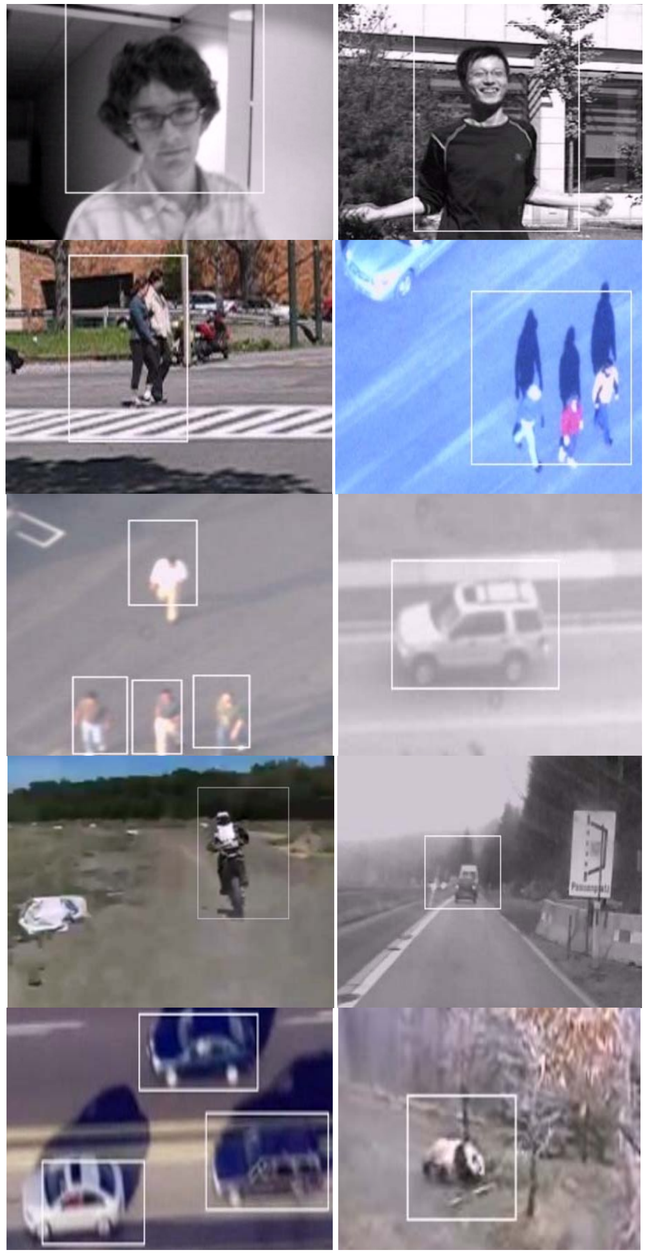

Figure 3: Results from the agile tracker

Figure 4: The top one represents the output from the agile tracker and the bottom one represents that from TLD.

Figure 5: Agile Tracker results for the TUD campus and crossing videos

5. CONCLUSIONS

Our novel algorithm for starting tracks using confidence measure and adaptive thresholding not only performs in real time but is also accurate. The *agile tracking framework* allows dynamic adaptive 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.

6. ACKNOWLEDGEMENTS

This work has been partially supported by Army Research Office (ARO) under grant number W911NF-10-1-0495. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the ARO or the United States Government.

7. 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.
[21] M. Andriluka, S. Roth, B. Schiele, “People-Tracking-by-Detection and People-Detection-by-Tracking” IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'08), Anchorage, USA, June 2008.
[22] E. Maggio and A. Cavallaro, “Hybrid particle filter and mean shift tracker with adaptive transition model,” in Proc. of IEEE International Conf. on Acoustics, Speech, and Signal Processing, Philadelphia, 2005.
[23] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. IJCV, 77(1):125–141, May 2008.
[24] B. Babenko, M. Yang, and S. Belongie. “Visual Tracking with Online Multiple Instance Learning”, CVPR, 2009.
[25] Z. Kalal, J. Matas, and K. Mikolajczyk, “Online learning of robust object detectors during unstable tracking,” *On-line Learning for Computer Vision Workshop*, 2009.
[26] Z. Kalal, K. Mikolajczyk, and J. Matas, “Forward-Backward Error: Automatic Detection of Tracking Failures,” *International Conference on Pattern Recognition*, 2010, pp. 23-26.
[27] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-Learning-Detection,” *Pattern Analysis and Machine Intelligence*, 2011.
[28] Tomás Crivelli, Patrick Bouthemy, Bruno Cernuschi Frías, and Jian-feng Yao, "Mixed-State Markov Models in Image Motion Analysis", *Book Chapter in Machine Learning for Vision-Based Motion Analysis*, Springer, 2011.
[29] Paolo Lombardi and Cristina Versino, "Learning to Detect Event Sequences in Surveillance Streams at Very Low Frame Rate", *Book Chapter in Machine Learning for Vision-Based Motion Analysis*, Springer, 2011.
[30] Xiaoyu Wang, Gang Hua, and Tony X. Han, "Discriminative Multiple Target Tracking", *Book Chapter in Machine Learning for Vision-Based Motion Analysis*, Springer, 2011.
[31] Guoliang Fan and Xin Zhang, "Video-Based Human Motion Estimation by Part-Whole Gait Manifold Learning", *Book Chapter in Machine Learning for Vision-Based Motion Analysis*, Springer, 2011.
[32] D. Comaniciu, V. Ramesh, and P. Meer “Real-Time Tracking of Non-Rigid Objects Using Mean-Shift,” *Proc. 2000 IEEE Conf. Computer Vision and Pattern Recognition,* 2000.