Online Multiple Support Instance Tracking

Qiu-Hong Zhou Huchuan Lu
School of Electronic and Information Engineering
Dalian University of Technology
Liaoning, China

Ming-Hsuan Yang
Electrical Engineering and Computer Science
University of California at Merced
California, USA

***Abstract*— We propose an online tracking algorithm in which the support instances are selected adaptively within the multiple instance learning framework. The support instances are selected from training 1-norm support vector machines in a feature space, thereby learning large margin classifiers for visual tracking. An algorithm is presented to update the support instances by taking image data obtained previously and recently into account. In addition, a forgetting factor is introduced to weigh the contribution of support instances obtained at different time stamps. Experimental results demonstrate that our tracking algorithm is robust in handling occlusion, abrupt motion and illumination.**

I. INTRODUCTION

Visual tracking plays a crucial role in numerous vision applications including intelligent surveillance, human motion analysis, interaction video processing, to name a few [20]. Notwithstanding demonstrated success in the literature, visual tracking remains a challenging problem as effective and efficient algorithms entail the need to account for appearance variation caused by illumination change, varying pose, cluttered background, motion blur and occlusion.

Tracking algorithms can be categorized to into generative and discriminative approaches. Generative tracking methods model appearances of objects using intensity or features, and predict the object location by finding the image patch most similar to the target. Numerous algorithms have been proposed with demonstrated success, e.g., [5] [9] [15] [18] [14] [4], to name a few.

Discriminative algorithms formulate visual tracking as a classification problem with local search in which the optimal decision boundary for separating the target from the background is inferred. The support vector tracking algorithm [2] integrates a trained support vector machine (SVM) within the optical flow framework for predicting target locations. Although impressive empirical results have been demonstrated, it requires a large hand-labeled data set for training, and once trained, the SVM classifier is not updated. To adapt to object appearance change, discriminative trackers have been extended to update classifiers with online learning. In [8], a confidence map is constructed by finding the most discriminative color features in each frame for separating the target object from the background. In [3], an online method is proposed to learn an ensemble of weak classifiers for visual tracking. Grabber et al. [12] present a tracking method using the online AdaBoost algorithm, which achieves real-time performance. Nevertheless, the above-mentioned methods rely mostly on the recently arrived data and thus the classifier may not be correctly trained if the data is noisy due to factors such as occlusion and motion blur.

Multiple-instance learning (MIL) is introduced by Dietterich et al. [11] in the context of drug activity prediction. It provides a framework to handle scenarios where labels of training data are naturally represented by sets of samples, instead of individual ones. In recent years numerous learning problems, e.g., image database retrieval [16], image categorization [7], and object detection [19], have been successfully proposed within the multiple instance learning framework. With this general framework, generative approaches based on inductive learning [11], diversity density [16] [21] have been proposed. Furthermore, discriminative classifiers such as support vector machines [1] [7] [6], and AdaBoost [19] have been incorporated within the multiple instance learning framework.

Within the multiple instance leaning framework, the noisy-or model is with boosting methods to integrate the probability of each bag for face detection [19]. Babenko et al. [4] develop an online multiple instance learning algorithm for visual tracking Although their results show that MIL helps in reducing drifts for visual tracking, the instances in bags are not selected effectively due to the use of noisy-or model. While the noisy-or model is used to account for ambiguities in labeling positive instances (e.g., object location for visual tracking), it inevitably includes instances that are less effective for classification. To eliminate such unwanted instances, we propose an online algorithm that selects important instances that support the classifier, and call it the Online Multiple Support Instances Tracking (OMSIT) method.

In this paper, we use the 1-norm SVMs [22] for learning sparse and effective support instances in the presence of redundant noisy samples. These support instances provide a compact representation of the SVM classifier which is used for finding the most similar image patches for visual tracking. Our update algorithm takes into account the recently found support instances and those found in previous time stamps. A forgetting factor is introduced in order to emphasize the importance of the most recent support instances in the update process. Our tracker is bootstrapped with the IVT method [18] for collecting positive and negative samples in the first few frames. Some outputs with higher confidence value from the IVT are put in the positive bag, and image patches randomly sampled from an annular region of the target object are considered as negative instances to negative

bags. Next, the instances of each bag are mapped to a feature space based their similarity measure, and the 1-norm SVM is applied to select effective features for constructing the large margin classifier. The 1-norm SVMs are updated constantly (e.g., every 5 frames) to select the most effective support instances so as to account for appearance change. In addition, a forgetting factor is introduced to to avoid drifting problem by focusing on the most recent instances. Empirical evaluations against state-of-the-art algorithms show that our method is able to track objects effectively and efficiently.

II. Preliminaries

As the proposed algorithm is developed within the multiple instance learning framework with 1-norm support vector machines, we first describe the preliminaries of these algorithms and then our approach in the next section.

Learning algorithms for binary classification problems are often posed within a probabilistic framework in which $p(y|x)$ is estimated from a set of training data, $\{(x_1,y_1),\ldots,(x_n,y_n)\}$, where each x_i is an instance, and $y_i \in \{0,1\}$ is a binary label. In the multiple instance learning framework the training data is formed as $\{(X_1,y_1),\ldots,(X_n,y_n)\}$ where $X_i = \{x_{i1},\ldots,x_{im}\}$ is a bag (or set) of instances and y_i is a bag label. The main difference is that examples are presented in sets ("bags"), and labels are provided for the bags rather than individual instances. The essence of MIL is that a bag is labeled positive if it contains at least one positive instance. On the other hand, a bag is labeled negative if all the instances are negative [11].

In this work, a positive bag i is denoted as B_i^+ and the j-th instance in that bag as x_{ij}^+. The bag B_i^+ consists of n_i^+ instances x_{ij}^+, $j = 1,\ldots,n_i^+$. Similarly, B_i^-, x_{ij}^-, and n_i^- represent a negative bag, the j-th instance in bag i, and the number of instances in bag i, respectively. When the label of the bag does not matter in description, it is referred to as B_i with instances as x_{ij}. The number of positive (negative) bags is denoted as $l^+(l^-)$. The diverse density framework, derived in [16], is based on the assumption that there exists a single target concept from which individual instances can be labeled correctly. Let a given concept class be c, the diverse density D of a concept $t \in c$ is defined as the probability that the concept t is the target concept given the training bags [16]:

$$D(t) = \Pr(t/B_1^+,\ldots,B_{l^+}^+,B_1^-,\ldots,B_{l^-}^-). \tag{1}$$

Applying Bayes' rule to (1) and further assuming that all bags are conditionally independent given the target concept, it can be shown that [16]:

$$\begin{aligned} D(t) &= \frac{\Pr(t)\prod_{i=1}^{l^+}\Pr(B_i^+/t)\prod_{i=1}^{l^-}\Pr(B_i^-/t)}{\Pr(t/B_1^+,\ldots,B_{l^+}^+,B_1^-,\ldots,B_{l^-}^-)} \\ &= \left[\frac{\prod_{i=1}^{l^+}\Pr(B_i^+)\prod_{i=1}^{l^-}\Pr(B_i^-)}{\Pr(B_1^+,\ldots,B_{l^+}^+,B_1^-,\ldots,B_{l^-}^-)\Pr(t)^{l^++l^--1}}\right] \\ &\quad \times \left[\prod_{i=1}^{l^+}\Pr(t/B_i^+)\prod_{i=1}^{l^-}\Pr(t/B_i^-)\right]. \end{aligned} \tag{2}$$

Assuming a uniform prior on t, maximizing $D(t)$ is equivalent to maximizing $\prod_{i=1}^{l^+}\Pr(t/B_i^+)\prod_{i=1}^{l^-}\Pr(t/B_i^-)$, Maron [16] proposed that if c is a single point concept class, where every concept corresponds to a single point in a big set, the maximum likely estimates can be computed by

$$\begin{aligned} \Pr(t/B_i^+) &\to \max_j \exp\left(-\frac{\left\|x_{ij}^+ - t\right\|^2}{\sigma^2}\right) \\ \Pr(t/B_i^-) &\to 1 - \max_j \exp\left(-\frac{\left\|x_{ij}^- - t\right\|^2}{\sigma^2}\right), \end{aligned} \tag{3}$$

where σ is a predefined factor and the maximum likelihood of D can be estimated. Although this algorithm is extended with the EM algorithm [21], neither of them guarantee the global optimality. In order to obtain the optimal estimates, multiple runs with different initializations are necessary. Consequently, the process of such maximum likelihood estimates is often very time consuming.

The similarity measure between two instances, x_i and x_j, is defined by $s(x_i,x_j) = \exp\left(-\frac{1}{\sigma^2}\left\|x_i - x_j\right\|_2^2\right)$. Based on this, the similarity between an instance x_k and a bag B_i is determined by x_k and the closest instance in the bag as:

$$s(x_k,B_i) = \max_j \exp\left(-\frac{\left\|x_{ij} - x_k\right\|^2}{\sigma^2}\right). \tag{4}$$

Using all instances in the training bags $\{x_1,\ldots,x_k,\ldots,x_N\}$, and a bag B_i is mapped to feature vector: $f(B_i) = [s(x_1,B_i),s(x_2,B_i),\ldots,s(x_n,B_i)]^\top$, the mapping is described by

$$\begin{aligned} &\left[f_1^+,\ldots,f_{l^+}^+,f_1^-,\ldots,f_{l^-}^-\right] \\ &= \left[f(B_1^+),\ldots,f(B_{l^+}^+),f(B_1^+),\ldots,f(B_{l^-}^-)\right] \\ &= \begin{bmatrix} s(x_1,B_1^+) & \ldots & s(x_1,B_{l^-}^-) \\ s(x_2,B_1^+) & \ldots & s(x_2,B_{l^-}^-) \\ \ldots & & \\ s(x_n,B_1^+) & \ldots & s(x_n,B_{l^-}^-) \end{bmatrix}. \end{aligned} \tag{5}$$

where each column represents a bag, and the k-th feature in the mapped feature space is the k-th row in the above matrix,

$$s(x_k,\cdot) = \left[s(x_k,B_1^+),\ldots,s(x_k,B_{l^+}^+),(x_k,B_1^-),\ldots,(x_k,B_{l^-}^-)\right]. \tag{6}$$

If x_k has high similarity score to some positive bags and low similarity score to some negative bags, $s(x_k,\cdot)$ provides useful information in determining the decision boundary, and is considered as a support instance, which will be derived in the following section.

III. Proposed Algorithm

In our algorithm, each instance is mapped to a feature space where its similarity score is computed with respect to each bag. The support instances are then extracted via the 1-norm SVM [22] and used for locating objects in visual tracking. A forgetting factor is introduced in order to weigh more on the most recent support instances. We propose an effective update method using the newly arrived support instances. Figure 1 shows the flow chart of our tracking algorithm and the major steps are summarized in Figure 2.

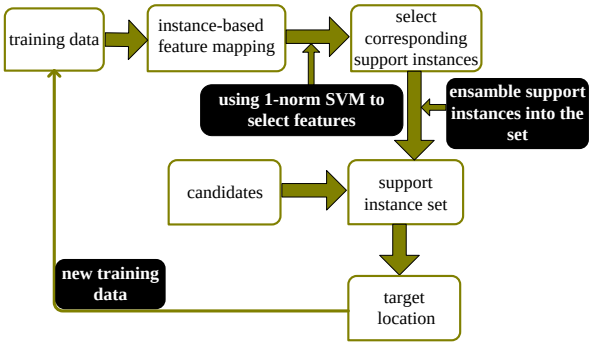

Fig. 1. Flowchart of the proposed OMSIT algorithm.

A. Initialization

Most tracking algorithms are initialized manually or with object detectors. For efficiency, our algorithm is bootstrapped with some outputs from the IVT where only a small number of particles are drawn (e.g., 300 in our experiments). For each particle, we extract its corresponding image patch, and calculate its confidence value (i.e., likelihood) using the current observation model. In the initialization process, the IVT is used to track the target object in the first few frames for collecting samples to form positive and negative bags. The samples with higher confidence values from the IVT model are used to form positive bags. The patches randomly sampled from an annular region of the target object are considered as negative instances of the negative bags. The samples collected in the initialization process are then used to train a 1-norm SVM for classification.

B. Feature Extraction and Feature Mapping

We use the histogram of oriented gradients (HOG) [10] to represent objects. As described above, image patches for positive and negative instances are sampled to form a few bags. In this work, we use a small set of positive and negative instances as well as bags to form a feature matrix for training 1-norm SVM. For example, we typically use 5 positive and 5 negative bags with a total of 500 instances for initialization. Thus, we have 500×10 feature matrix as follows:

$$\begin{bmatrix} s(x_1,B_1^+) & \dots & s(x_1,B_{l^-}^-) \\ s(x_2,B_1^+) & \dots & s(x_2,B_{l^-}^-) \\ \dots & & \\ s(x_n,B_1^+) & \dots & s(x_n,B_{l^-}^-) \end{bmatrix}. \tag{7}$$

As this feature matrix is rather small, this 1-norm SVM can be trained efficiently.

C. 1-norm SVM and Support Instance Selection

The 1-norm SVM exploits the ℓ_1-norm (i.e., Lasso) penalty term in deriving a sparse large margin classifier [22], which has been shown to be effective in the presence of redundant noisy features [22] [6]. Within the multiple instance learning framework, the 1-norm SVM can be trained with features mapped from positive and negative bags as

$$y = \text{sign}\left(w^\top f + b\right), \tag{8}$$

where w and b are model parameters and f corresponds to the mapped feature of a bag in the feature space. While conventional SVMs entail the need of solving a quadratic program and 1-norm SVMs are trained via a linear program, the MIL 1-norm SVM is formulated as the following optimization problem [1] [6]:

$$\begin{aligned} \min_{w,b,\xi,\eta} \ & \lambda \sum_{k=1}^{n} |w_k| + C_1 \sum_{i=1}^{l^+} \xi_i + C_2 \sum_{j=1}^{l^-} \eta_j \\ s.t. \ & \left(w^\top f_i^+ + b\right) + \xi_i \geqslant 1, i = 1,\dots,l^+, \\ & -\left(w^\top f_i^- + b\right) + \eta_i \geqslant 1, i = 1,\dots,l^-, \\ & \xi_i, \eta_j \geqslant 0, i = 1,\dots,l^+, j = 1,\dots,l^-, \end{aligned} \tag{9}$$

where ξ_i and η_i are slack variables for positive and negative bags, and C_1 as well as C_2 are penalty weights for false positives and negatives. This optimization problem can be posed and solved as a linear program in a way similar to [22] [6]. Assume that w^* and b^* are the optimal parameters of (9), the magnitude w^* determines the weight of the k-th feature whereas most elements of the w^* are zero as a result of using 1-norm penalty in deriving a sparse SVM. That is, we only need to retain a set of indexes for nonzero elements of w^*:

$$\Lambda = \{k : |w_k^*| > 0\}. \tag{10}$$

Consequently, x_k can be classified using MIL 1-norm SVM with respect to bag B_i by

$$y = \text{sign}\left(\sum_{k\in\Lambda} w_k^* s(x_k,B_i) + b^*\right). \tag{11}$$

D. Locate Object Using Support Instances

An instance in bag B_i is assigned to the positive class (negative class) if its contribution to $\sum_{k\in\Lambda} w_k^* s(x_k,B_i)$ of (11) is greater (less) than or equal to a threshold. For efficient visual tracking, we aim to find a minimal set of support instance from the ones extracted from 1-norm SVMs. That is, we aim to reselect the most important instances from the ones on the decision boundaries. Such approaches have also been exploited for fast object detection [17] and image categorization [6]. We define an index set:

$$\Psi = \left\{ j^* : j^* = \arg\max_j \exp\left(-\frac{\left\|x_{ij} - x_k\right\|^2}{\sigma^2}\right), k \in \Lambda, x_{ij} \in B_i \right\}. \tag{12}$$

From this set, for bag B_i the instances in the set Ψ are effective, and other instances x_{ij} $(j^* \notin \Psi)$ that do not affect the value of $\sum_{k\in\Lambda} w_k^* s(x_k,B_i)$ can thus be discarded. Furthermore, x_{ij} $(j^* \in \Psi)$ can be a maximizer of (12) for different x_k, $k \in \Lambda$, and then for each $(j^* \in \Psi)$, we can find a smaller set of instances as follows:

$$\Lambda_{j*} = \left\{ k : k \in \Lambda, j^* = \arg\max_j \exp\left(-\frac{\left\|x_{ij} - x_k\right\|^2}{\sigma^2}\right) \right\}. \tag{13}$$

Initialization
Initialize parameters including a region of interest and parameters of IVT
for i = 1 to n (n is usually set to 5)
1) Rank the tracking outputs of IVT, and those with higher probability are considered as positive instances to form positive bags. Randomly sample patches from an annular region of the current target object and use them as negative instances to form negative bags.
2) Extract HOG features from positive and negative instances.
3) Construct feature matrix (7) and put the instances into a training set.

end
Train 1-norm SVMs using (9), and select support instances.

Online tracking
for $i = n+1$ to the end of the sequence
1) Sample candidates and extract HOG features.
2) Determine the index set using (13) and the corresponding similarity score using (14).
3) Determine the object location in current frame by weighting the support instances found in the current and previous frames with (15), (16) and (17).
4) Extract positive and negative instance to accumulate training data into training data set.

when periodic update takes place
1) Update the 1-norm SVMs and select new support instances by (9).
2) Update the set of support instances.
3) Empty the current training data set.

end
end

Fig. 2. Proposed Online Multiple Support Instance Tracking algorithm.

After retaining only the most important and effective instances, we can compute $s(x_k, B_i)$ efficiently using $s\left(x_k, \left\{x_{ij*}\right\}\right)$ for $(k \in \Lambda_{j^*})$, i.e.,

$$s(x_k, B_i) = s\left(x_k, \left\{x_{ij*}\right\}\right). \tag{14}$$

For efficient visual tracking, our goal is to find the most important support instances, and thus we use (13) to reselect support instances rather than every instance in Λ. The contribution of a support instance x_s is computed by

$$h_f(x_s) = \sum_{k \in \Lambda_{j^*}} \frac{w_k^* s(x_k, x_s)}{m_k}, \tag{15}$$

where m_k represents the number of maximizers for the x_k. That is, the contribution of x_s is computed by considering both its similarity to existing support instances x_k and the number of times x_k is selected. As $h_f(x_s)$ is computed by the current support instances whereas we have support instances from previous frames, it is important to retain only the most relevant ones in the context of visual tracking. If we retain all previous training data in 1-norm SVMs training process, the time complexity grows significantly. For visual tracking, we use an update method for selecting support instances (described in Section III-E) that takes both performance and efficiency into account. The weighted MIL 1-norm SVM is computed by

$$H(x_s) = \sum_f e^{-\frac{1}{f}} h_f(x_s), \tag{16}$$

where f is the number of frames corresponding to support instances, and we select support instances periodically (e.g., f is a multiple of 5 in our experiments). The term, $e^{-\frac{1}{f}}$, plays the role as a forgetting factor. It puts more weight on the newly selected support instances and retains the contribution of support instances found in previous frames.

E. Updating 1-norm SVMs and Renew Support Instances Set

We update the 1-norm SVM periodically and obtain new support instances to account for appearance variation. This is of great importance as the object appearance may undergo drastic change due to variation in illumination, pose, occlusion and background scene. The weights of new support instances are determined with (15) and the weighted SVM is computed by (16).

The new support instances are inferred from a training set collected using a sampling strategy similar to [4]. For each new frame we crop out a set of image patches $X^s = \left\{x \,|\, s > \left\|l(x) - l_{f-1}^*\right\|\right\}$ as candidates where l_{f-1}^* denotes the target location in the previous frame. We compute $H(x)$ for all $x \in X^s$, then update the current location l_f^* by

$$l_f^* = l\left(\arg\max_{x \in X^s} H(x)\right). \tag{17}$$

Once the target location is, we crop out a set of patches $X^r = \left\{x \,|\, r > \left\|l(x) - l_f^*\right\|\right\}$ for positive instances to form a positive bag, and patches from an annular region $X^{r,\beta} = \left\{x \,|\, \beta > \left\|l(x) - l_f^*\right\| > r\right\}$ as negative instances. Each negative is put into its own negative bag. The radius r is the same as before and β is another scalar $(\beta > r)$.

During each update, we accumulate a number of bags (e.g., 5 positive and 5 negative bags in our experiments) to train 1-norm SVMs and select new support instances. To ensure the efficiency and performance, we only use new bags obtained in the recent frames, but retain all the support instances found in previous frames. We employ (15) and (16) to select the support instances. Figure 3 illustrates support instances that are updated using test sequence D. Note that the support instances change significantly as there are drastic scene variations in the image sequence. This also demonstrates the importance of online update of support instances.

IV. Experiments

We present the implementation details, experimental results, and discussion in this section.

A. Implementation Details

In our experiments, we use HOG features to represent the tracking targets. The number of particles is set to 300 for initialization with IVT. Each positive bag contains 50 instances among which at least of them is a positive one, and each negative bag contains 50 negative instances. Both

Fig. 3. Online support instances set. We update the 1-norm SVMs and select support instances into the set periodically.

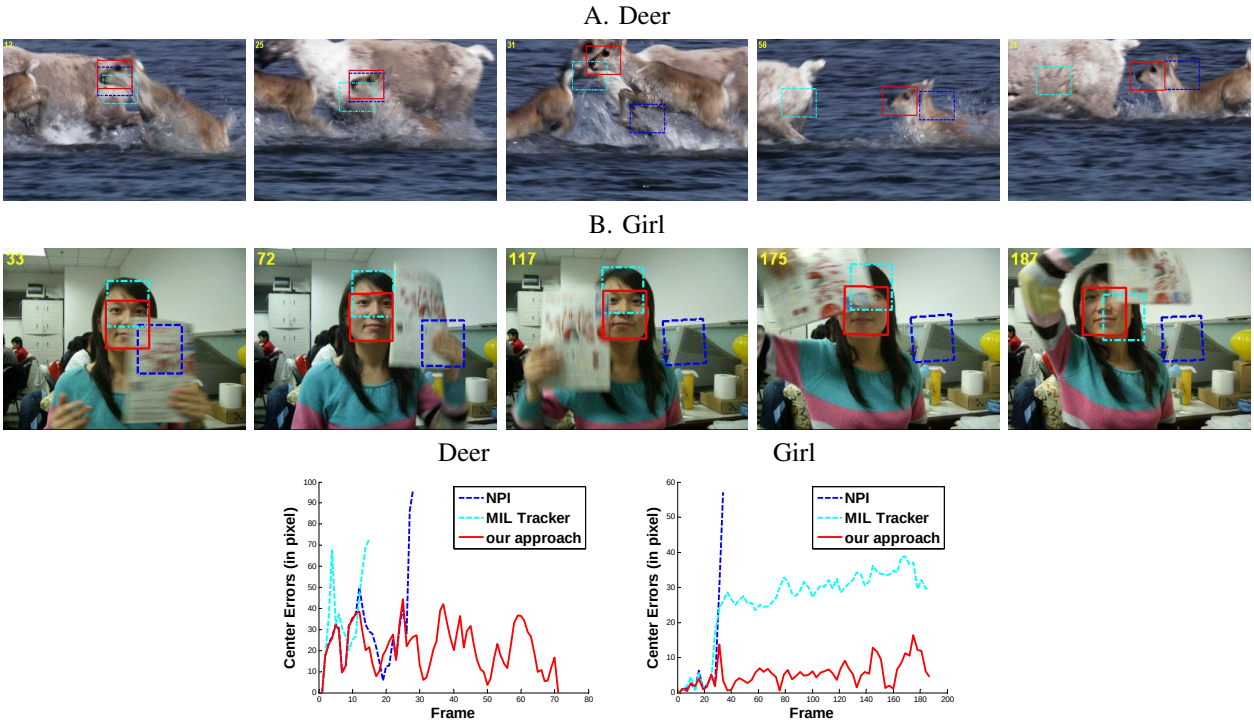

Fig. 4. Tracking results when the objects undergo pose variation with heavy occlusion (red: OMSIT, blue: NPI, light blue: MILTrack).

penalty weights C_1 and C_2 of (16) are set to 0.5. The parameters for sampling instances, s, r and β, are set to 20, 8, and 40, respectively. We utilize the CPLEX toolbox for solving linear programming problems with 1-norm SVMs where σ is set to 10^3. In order to validate that the proposed algorithm can be applied to different sequences without tuning, all these parameters are fixed in the experiments. In addition, we evaluate our algorithm with other methods using the videos available in the public domain except sequence B. To demonstrate the necessity and merits of the our update method, we carry out a series of experiments with no prior information, i.e., the support instances obtained in previous frames. Namely, (16) is reduced to $H(x_s) = h_f(x_s)$ in the NIP tracker. Our MATLAB implementation currently runs on a Pentium 4 PC with 2GB memory at 2 to 5 frames per second. As mentioned above, by solving linear programs of 1-norm SVMs instead of quadratic programs of standard SVMs, our method reduces computation complexity significantly. We implement the proposed online multiple support instance tracking (OMSIT) algorithm, and its variant with no previous information (NPI). The MATLAB codes and data sets are available at our web sites.

B. Empirical Results

We evaluate the empirical tracking performance of our method with the MILTtrack algorithm [4], the online AdaBoost tacking method [12] (OAB), and the SemiBoost tracker [13] using the code available on the web for fair evaluation. The tracking results of our algorithm, NPI tracker and MILTrack are shown with solid red, dashed blue, and dashed light blue boxes, respectively. Table I lists the sequences evaluated in our experiments and their characteristics. The empirical results are presented in the following sections, and videos can be found on our web sites.

TABLE I

TRACKING SEQUENCES IN OUR EXPERIMENTS.

Sequence	Frames	Main Challenges
A. Deer	72	pose variation, background clutter
B. Girl	187	occlusions
C. Shopper	320	abrupt motions
D. Bono	182	illumination changes
E. Woman	103	occlusions
F. Man	500	occlusions, similar objects
G. Santana	181	varying illumination
H. Female	898	occlusions

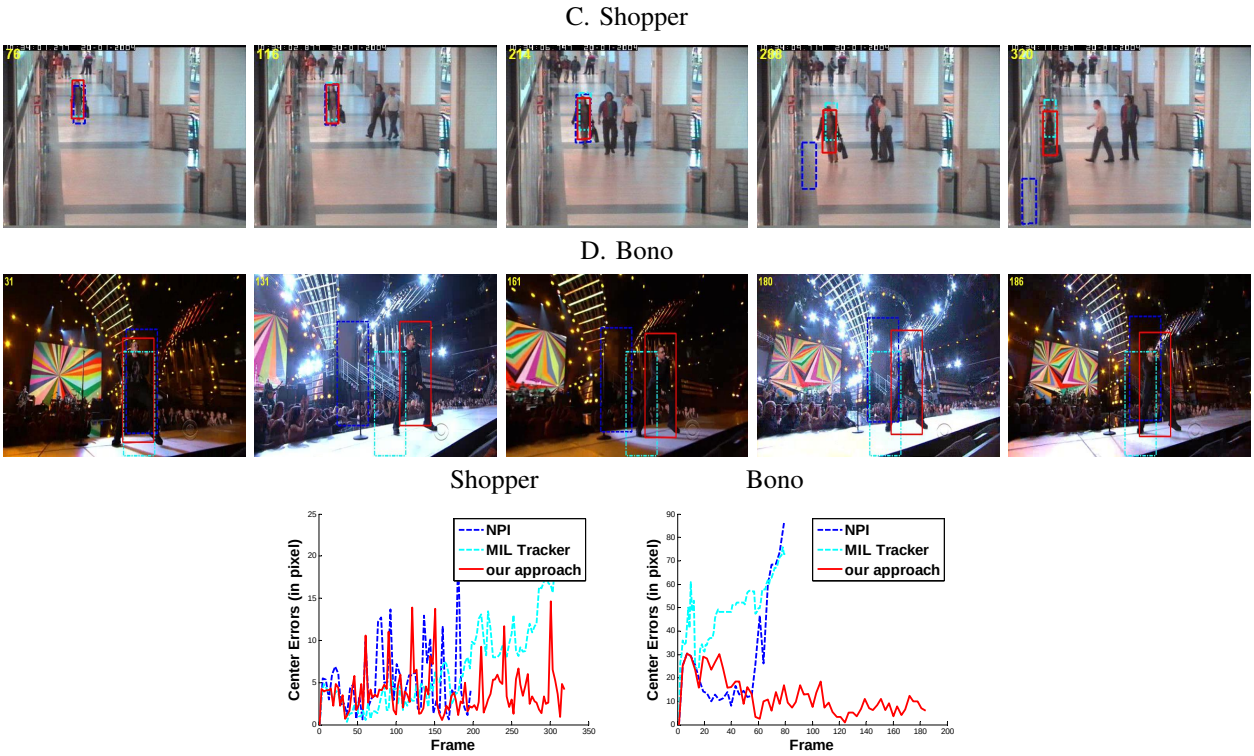

Fig. 5. Tracking results when the objects undergo abrupt motion and illumination (red: OMSIT, blue: NPI, light blue: MILTrack).

Pose Variation and Occlusion

Figure 4 shows the tracking results of a deer galloping through water (sequence A) and a girl whose face is partially occluded (sequence B). The target object undergoes rapid motion with similar deers in the scene. The proposed algorithm is able to track the deer throughout the entire sequence whereas the NPI tracker fails at frame 25 and the MILTrack method loses track of the target at frame 40. The sequence B contains a girl whose face is constantly occluded by a magazine. From frame 160 to 180, the face of this girl is almost fully occluded. In the presence of heavy occlusion, our OMSIT algorithm is still able to track the target in all frames. At frame 33, the NPI tracker mistakenly tracks the magazine cover when the girl reappears from the occlusion. It can be explained that the NPI tracker mistakenly updates its model with most recent instances from image patches of the magazine cover without using the instances learned in previous frames (i.e., without using (15) and (16)). Consequently, the tracker ends up tracking the image patch of the magazine cover. We note that the MILTrack performs well in this sequence due to the use of Haar-like features and online update.

Abrupt Motion and Illumination Change

Figure 5 shows how the proposed method performs when the targets undergo abrupt motions and illumination change. At the end of the sequence C, an abrupt motion occurs and the MILTrack method drifts off the target. Nevertheless, our method is able to handle abrupt motion and achieve good performance. The sequence D consists of a singer performing on stage with drastic lighting change as a result of neon and spot lights. It is worth noticing that the images are acquired with flying camera shots, thereby causing significant appearance change of the signer. Our tracker is able to track the singer reasonably well in this challenging sequence whereas the NPI tracker fails at frame 60 and the MILTrack method gradually loses track of the target as shown in Figure 5.

Scale Change and Severe Occlusion

We evaluate three trackers using sequences from the CAVIAR data set and present the results in Figure 6. As the scenes in sequence E are rather static without significant occlusion, all the trackers perform reasonably well. On the other hand, sequence F is rather challenging as it contains significant occlusion and scale change. Our method tracks the target person well until the end of this sequence. The MILTrack method loses the target when another person occludes the target subject at frame 221, and the NPI misses the target at the same frame. The reason that our tracker performs well can be explained as follows. First, our tracker is able to reduce drifts with the use of multiple support instances. Second, the proposed algorithm uses 1-norm SVM to update support instances that account for large and drastic appearance change. In addition, our method introduces a forgetting factor for retaining previous support instances, thereby reducing the effects of noisy data during update. Finally, Figure 7 shows the results from sequences G and F. The proposed algorithm is able to track faces undergoing large lighting variation and heavy occlusion.

E. Woman

F. Man

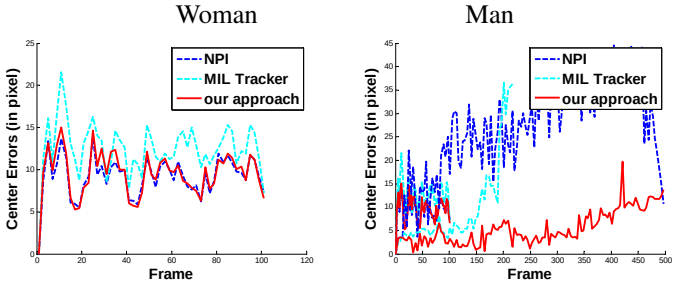

Fig. 6. Tracking results when the objects undergo scale change and heavy occlusion (red: OMSIT, blue: NPI, light blue: MILTrack).

G. Santana

H. Female

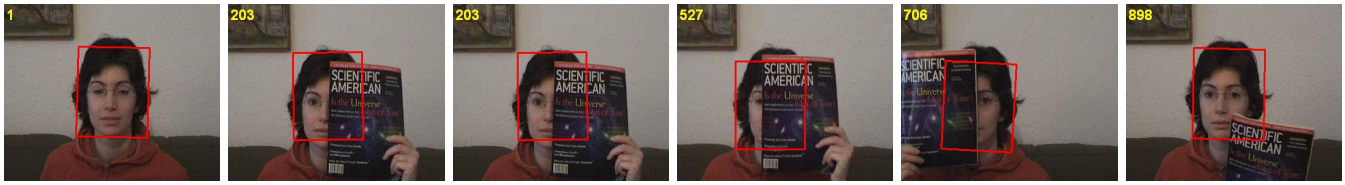

Fig. 7. Tracking results of the proposed OMSIT method (red: OMSIT, blue: NPI, light blue: MILTrack).

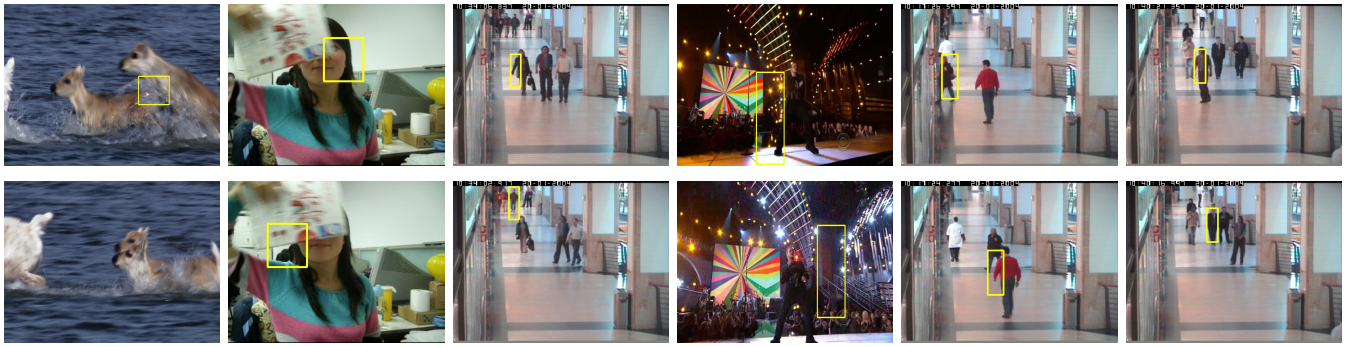

Fig. 8. Tracking results using Online-AdaBoost Tracker [12] (OAB) on the first row, and SemiBoost Tracker [13] on the second row.

C. Discussion

We evaluate the performance of the proposed OMSIT algorithm against the tracker with no previous instances Tracker (NPI), MILTrack [4], online AdaBoost tracker [12] (OAB), and SemiBoost method [13] using publicly available data sets. The ground truth center location of the target object in each sequence are manually labeled. The average tracking error are presented in Table II where the best and results are shown with bold red fonts, and the second best ones are shown with blue fonts.

TABLE II

AVERAGE CENTER LOCATION ERRORS (PIXELS).

Image sequence	OMSIT	NPI	MIL	OAB	Semi
A. Deer	**7.7**	117	189	49	68
B. Girl	**5**	79	26	9	6
C. Shopper	**4**	21	7	8	50
D. Bono	**12**	79	66	174	165
E. Woman	**8**	12	**8**	12	19
F. Man	**6**	28	12	73	14

In all sequences our OMSIT algorithm outperforms the Online AdaBoost and SemiBoost Trackers, and in most cases outperforms or ties with the MILTrack method. Figure 8 illustrates some failure results of the OAB and SemiBoost algorithms using the same sequences in our experiments (shown in Figure 4-6). The MILTrack method updates weak classifiers with all instances including redundant and irrelevant ones, and consequently the discriminative power of the classifier may gradually degrade. Similarly, the OAB and SemiBoost algorithms rely mostly on all the samples obtained online, and consequently these methods do not work well when there is an abrupt motion or heavy occlusion. One contribution of this paper is the use of support instances to represent the target object. As the support instances are the ones that reside on decision boundaries, they provide more critical and representative information than other ones. The second contribution of this work is we propose an update method that takes all previous and most recent support instances into account. A forgetting factor is used to properly weigh the contribution of previous support instances, which is important in the presence of occlusion. Otherwise, the occluded image patches may be mistakenly considered as the "correct" and "recent" instances for updating the target object, thereby resulting in drifts when the target reappears in the scene without occlusion.

V. CONCLUSIONS

In this paper, we propose an online multiple support instance tracking algorithm in which instances are mapped into features for training 1-norm SVMs and selecting a compact set of instances. These support instances are used to locate the target in each frame. The update method includes training 1-norm SVMs and reselecting support instances in order to account for appearance change for robust visual tracking. A forgetting factor is used to weigh the contribution of previous support instances, which facilitates in handling occlusion. Our future work will extend this algorithm in several aspects. First, we aim to derive more robust and informative features for object representation. Second, we will extend the proposed algorithm for tracking multiple objects. Finally, we will apply the proposed multiple support instance algorithm to other problems including image classification, image retrieval and object detection.

REFERENCES

[1] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems*, pages 561–568, 2002.

[2] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004.

[3] S. Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, 2007.

[4] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 983–990, 2009.

[5] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[6] Y. Chen, J. Bi, and J. Z. Wang. Miles: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1931–1947, 2006.

[7] Y. Chen and J. Z. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5:913–939, 2004.

[8] R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1631–1643, 2005.

[9] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.

[11] T. G. Dietterich, R. H. Lathrop, and L. T. Perez. Solving the multiple-instance problem with axis parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[12] H. Grabner and H. Bischof. On-line boosting and vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 260–267, 2006.

[13] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *Proceedings of European Conference on Computer Vision*, pages 234–247, 2008.

[14] B. Han, Y. Zhu, D. Comaniciu, and L. S. Davis. Visual tracking by continuous density propagation in sequential Bayesian filtering framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):919–930, 2009.

[15] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1296–1311, 2003.

[16] O. Maron. *Learning from ambiguity*. PhD thesis, Massachusetts Institute of Technology, 1998.

[17] S. Romdhani, P. H. S. Torr, B. Schölkopf, and A. Blake. Computationally efficient face detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 695–700, 2001.

[18] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

[19] P. Viola, J. C. Platt, and C. Zhang. Multiple instance boosting for object detection. In *Advances in Neural Information Processing Systems*, pages 1417–1426, 2005.

[20] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):1–45, 2006.

[21] Q. Zhang and S. A. Goldman. Em-dd: An improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems*, pages 1073–1080, 2001.

[22] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Advances in Neural Information Processing Systems*, pages 49–56, 2003.