Online Distance Metric Learning for Object Tracking

Grigorios Tsagkatakis, *Student Member, IEEE* and Andreas Savakis, *Senior Member, IEEE*

***Abstract*— Tracking an object without any prior information regarding its appearance is a challenging problem. Modern tracking algorithms treat tracking as a binary classification problem between the object class and the background class. The binary classifier can be learned offline, if a specific object model is available, or online, if there is no prior information about the object's appearance. In this paper, we propose the use of online distance metric learning in combination with nearest neighbor classification for object tracking. We assume that the previous appearances of the object and the background are clustered so that a nearest neighbor classifier can be used to distinguish between the new appearance of the object and the appearance of the background. In order to support the classification, we employ a Distance Metric Learning (DML) algorithm that learns to separate the object from the background. We utilize the first few frames to build an initial model of the object and the background and subsequently update the model at every frame during the course of tracking, so that changes in the appearance of the object and the background are incorporated into the model. Furthermore, instead of using only the previous frame as the object's model, we utilize a collection of previous appearances encoded in a template library to estimate the similarity under variations in appearance. In addition to the utilization of the online DML algorithm for learning the object/background model, we propose a novel feature representation of image patches. This representation is based on the extraction of scale invariant features over a regular grid coupled with dimensionality reduction using Random Projections. This type of representation is both robust, capitalizing on the reproducibility of the scale invariant features, and fast, performing the tracking on a reduced dimensional space. The proposed tracking algorithm was tested under challenging conditions and achieved state-of-the art performance.**

***Index Terms*— object tracking, distance metric learning, online learning, nearest neighbor classification, random projections.**

I. Introduction

Object tracking is a vital part of many computer vision applications, including surveillance, human computer interaction, smart spaces and gaming, among others. Object tracking is a very challenging task due to appearance variations caused by occlusions and changes in illumination and pose. This task can become even more demanding when there is no prior information regarding the object's appearance. Traditional object tracking algorithms model the appearance of the object in a generative way, while modern techniques view tracking as a classification problem with temporal priors. In the latter context, the goal is to locate an image region that belongs to the same class as the tracked object, under the constraint that the new location is spatially adjacent to the previous one. This novel approach has received a lot of attention due to its ability to maintain accurate tracking despite severe conditions.

Based on the binary classification paradigm, a discriminative tracking algorithm can be trained either offline or online. If an object model is not available, modern online learning approaches for discriminative tracking utilize the boosting framework to incrementally train a binary classifier that will be able to separate the object from the background. These methods have solid theoretical background and have demonstrated very promising results. In similar spirit to the online trained discriminative tracking approaches, we propose a novel tracking method where the problem is treated as a nearest neighbor classification with online learned distance. The idea is based on the stipulation that different appearances of the object will be close to each other in a suitable feature space, i.e. the feature representation of the object's appearance in the current frame will be close to its feature representation in the previous frame. However, measuring similarity with Euclidean distance, as is typically done in nearest neighbor classification, is not adequate, since Euclidean distance does not encode any discriminative information. As a result, tracking is likely to fail due to misclassification under mild changes [3]. To overcome this impediment, we propose to learn an appropriate distance metric that will yield small distances for different appearances of the object and large distances for appearances of non-objects (background). This approach is formally known as distance metric learning (DML) and its goal is to discover a distance metric that will satisfy the constraints imposed by class labels, i.e. keep data points from the same class close to each other and map data points from different classes far apart.

In this work, we advocate that the utilization of an online learned distance can provide increased robustness compared to using a predefined or a pre-learned metric. Object localization

G. Tsagkatakis is with the Center for Imaging Science, Rochester Institute of Technology, Rochester, NY 14623, USA (corresponding author: gtsagkatakis@gmail.com, gxt6260@rit.edu).
A. Savakis is with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623, USA (andreas.savakis@rit.edu).

is facilitated by searching for the region that minimizes the learned distance between reference templates and the current appearance of the object. The learned distance is updated online in order to incorporate various appearances of the object as well as the background regions. In addition, instead of a single reference template, a collection of templates called the template library is used to capture the object's appearance at various stages of the tracking.

The use of nearest neighbor classification in combination with online distance metric learning offers several advantages in the context of object tracking:

1. It provides continuous values, rather than a binary output as is done in typical binary classifiers. The benefit of having continuous values for similarity is that they can be used as robust indicators of the changes in the appearance and assist in the detection of occlusions. Furthermore, the learned similarity can be used for estimating the correlation between the current appearance of the object and the examples in the template library, whereas in binary classification all previous templates are equally weighted.
2. It offers an efficient way to transition from completely supervised, to semi-supervised and unsupervised settings depending on the availability of prior information. For example, if an initial estimation of the object's appearance is available, a batch DML algorithm can be applied and used for tracking. On the other hand, when no prior information is available, as is the case of unsupervised tracking, the first few frames can be used for initial modeling and online metric learning will handle the subsequence changes.
3. Similarly to other detection based tracking algorithms, this method has the capability to learn both positive and negative examples, which offers increased discriminative power compared to generative approaches that model the object's appearance without any concern about the background.
4. There are numerous types of features that can be used for appearance encoding, ranging from edge responses to Scale Invariant Feature Transform (SIFT) features. In contrast, boosting based techniques are usually constrained to simple Haar-like features.

Although online distance metric learning is a powerful technique in the context of object tracking, the method used for the representation of the object's appearance is another critical component of the tracking algorithm. A novel approach in image representation that has gained interest is the extraction of SIFT features over a regular grid. This type of feature representation has been employed in state-of-the-art image classification schemes including [26] and [33]. The benefits of using SIFT features for object representation include robustness to small changes in appearance and illumination. However, the extraction of dense SIFT features generates a high dimensional representation of each region. Performing the required operations of the proposed tracking scheme on the high dimensional representations will severely degrade the execution speed of the system. We address this issue by utilizing the Random Projections (RPs) method for reducing the dimensionality of the region representation. RPs is a data-independent linear method for dimensionality reduction which offers performance acceleration without requiring prior knowledge on the object's appearance. The use of RPs in the context of object tracking was first proposed in [27], where the authors used a linear kernel and a template matching algorithm object tracking across a network of cameras.

The rest of the paper is organized as follows. Section 2 provides an overview of different approaches in object tracking and their association with our work. The processes of feature extraction using SIFT and RPs is discussed in Section 3. Distance metric learning is overviewed in Section 4, while the application of distance learning in object tracking is presented in Section 5. Experimental results are given in Section 6 and the paper concludes in Section 7.

II. Prior Work

Traditional template based tracking algorithms can be coarsely divided in two categories: offline and online. In the offline approaches, a model of the object is either learned offline, using similar visual examples, or is learned during the first few frames. In both cases once the object model is generated, a predefined metric is used to identify the locations in subsequent frames. Examples of this class of tracking algorithms include kernel based methods [1] and appearance models [6]. These methods suffer from the limitation that once the model is created, it is not updated and, as a result, tracking may fail due to unexpected object appearances. In addition, appearance based methods, such as [6], require training with all possible poses and illumination, a time consuming process. Furthermore, the use of a predefined metric such as the Bhattacharyya coefficient [1], the Kullback-Leibler divergence [2], the normalized cross correlation or the sum-of-absolute differences [3], may not be appropriate for representing the relationship between different appearances of the object that may be encountered during tracking.

The second line of thought utilizes online learning to learn the object's changing appearance during tracking. Online or incremental template update was first suggested by Jepson et al. in [15], where a mixture of three components was proposed for the representation of the object, namely the stable, the transient and the noise component. Other examples of online learning for tracking include the work of Matthews et al. [16], who suggested a template update strategy for maintaining an update appearance model, and the online subspace learning approaches by Lim et al. [17] and Kim et al. [18], that seek to incrementally update the object's appearance subspace. Although these methods attempt to maintain an accurate model for the object, they are vulnerable to drift, i.e. the slow degradation of the object's model and the adaptation to the background's appearance. A partial solution to the problem of drift was proposed in [19], where offline trained support vector machines were used for separating the appearance of the object to that of the background. Nevertheless, this approach is still an offline based method and thus faces similar problems to other offline based methods.

A novel framework in visual tracking aims at alleviating the

problems caused by drift without requiring an explicit model of the object. According to this framework, tracking is treated as a binary classification problem where the objective is to both model the object's appearance and separate it from the appearance of the background during tracking instead of relying on offline training. In [20], Avidan proposed the use of the classification score of the Adaboost classifier as the object localization mechanism while replacing old or unreliable weak classifiers to cope with changes in the appearance of the object. In [12], Liu et al. proposed a semi-supervised ensemble tracking approach, where the particle filter framework was used for both object localization and collection of unlabeled data points used to train an Ensemble classifier. In [22], Grabner et al. proposed an online tracking method that employed an online Adaboost classifier to train a binary classifier as new samples were presented. The trained classifier was subsequently applied for object localization. The method was later extended in a semi-supervised setting in [13] using the SemiBoosting framework, where labeled examples were obtained from the first frame and subsequent examples were incorporated in a semi-supervised manner. In [21], Babenko et al. further extended the use of online learning for tracking by utilizing the Multiple Instance Learning framework, where instead of using a single positive example for the online Adaboost classifier, a set of image patches were introduced to capture the appearance of the object.

The utilization of online learned binary classifiers for inferring the presence or absence of the object in a candidate window has been shown to outperform traditional offline generative approaches in various challenging scenarios. The proposed tracking algorithm is motivated by the online discriminative framework, and employs an online learning approach for modeling the object's appearance that tackles the issues present in the offline approaches. In addition, a case-specific online learned distance metric is utilized to accurately model the association between the object's appearances at different time instances. The proposed tracking algorithm is similar to online learned binary classification approaches but does not impose the hard constraint of binary decisions. We argue that this type of hard decisions can lead to drift, since the binary classifier would be forced to accept and incorporate the appearance of the object even if it is not an optimal one, e.g. when the object is partially occluded or blurred due to fast motion. By utilizing a soft, distance-based classification metric, the proposed scheme can better model the changes in the object's appearance and continuously track the object, even if it is partially occluded, without adversely modifying the appearance model which can lead to drift. In addition to overcoming the drift problem, the learned distance can be used to compare the similarity between examples in the template library with the current object's appearance and update the library correspondingly. Boosting-like approaches, however, can utilize only the appearance of the object in the previous frame, which limits their ability to generate a robust and enduring object model. Furthermore, unlike the restriction of simple binary features used in boosting based approaches, our proposed algorithm can incorporate any form of vectorizable feature representation and thus offers a wider range of options depending on the computational constraints and the accuracy requirements.

III. Object representation

A key decision in the design of an object tracking algorithm is the type of representation to be used. Historically, various types of low level features have been considered. Object representation via raw pixel intensities is probably the simplest approach and the most efficient one in terms of computational complexity. However, changes in illumination can drastically change the representation of an object which may cause the tracker to fail. To resolve this problem, different approaches have been presented that encode image regions using color histograms [1], spatio-temporal appearance models [24] and part-based appearance models [25]. In this work we employ the Scale Invariant Feature Transform (SIFT) [23] over a regular grid to obtain the initial representation of each candidate window. Once the initial feature representation is obtained, we apply a linear dimensionality reduction transform to reduce the computational complexity of object localization and distance metric learning.

A. SIFT descriptors on a regular grid

The SIFT method is particularly successful for extracting features that are invariant to small changes in the object's appearance including scale and orientation and has been successfully applied in recent state-of-the-art systems for image classification [27, 33] etc. It has also been used as an extension of traditional color histogram representations for mean shift tracking in [29].

The first stage of SIFT is the keypoints localization. Keypoints correspond to maxima/minima of the Difference of Gaussians (DoG) occurring at multiple stages. After some intermediate steps, such as removing keypoints with low contrast, eliminating responses along edges and assigning the appropriate orientations, the selected region is described by a 128 dimensional vector corresponding to a histogram of oriented gradients. Even though experimental results suggest that the keypoints are stable, recent studies have shown that obtaining the SIFT points on a regular grid can outperform the keypoints obtained by the DoG [26]. In addition, the process of extracting the descriptors can be significantly accelerated by using a piecewise-flat weighting approach, rather than a Gaussian windowing function, as discussed in [30]. More specifically, instead of weighting the contribution of the extracted gradients based on a Gaussian windowing function, the gradients are all weighted equally. Once all the gradients have been accumulated into a spatial bin, the bin is reweighted by a Gaussian filter. This type of approximation incurs a minimum loss while being significantly faster. In our implementation, for each 50x50 pixel target window, 9 SIFT descriptors are extracted using the VLFeat feature extraction library [30], resulting in a 9 x 128=1152 dimensional

representation of the appearance of the window. The process of extracting the 1152 dimensional vector takes about 15msecs using the VLFeat library on a desktop computer.

B. Random Projections

The high dimensional representation of each image region that is generated by the dense SIFT descriptors can significantly reduce the processing speed of the system due to the complexity of object localization and distance learning. Dimensionality reduction may be used to reduce the computational load while keeping the most important information intact. The benefit of dimensionality reduction is that the execution of algorithms that depend on the dimensions of their input data, such as DML, can be significantly decreased.

In this work, we propose the use of a novel approach in dimensionality reduction, called Random Projections (RPs). RPs is a linear technique for dimensionality reduction based on the Johnson-Lindenstrauss (JL) lemma [35]. Formally, the JL lemma states that for a finite collection of points Q in R^d with fixed $0 < \epsilon < 1$ and $\beta > 0$, when

$$k \geq \left(\frac{4 + 2\beta}{\epsilon^2/2 - \epsilon^2/3} \right) \ln (\#Q) \approx O\left(\frac{\ln \#Q}{\epsilon^2} \right) \tag{1}$$

then a random matrix $\boldsymbol{R} \in \mathbb{R}^{k \times d}$, where $k \leq d$, whose elements are drown i.i.d. from a zero mean, bounded variance distribution satisfies

$$(1 - \varepsilon)\sqrt{k/d} \leq \frac{\|\boldsymbol{R}x - \boldsymbol{R}y\|_2}{\|x - y\|_2} \leq (1 + \varepsilon)\sqrt{k/d} \tag{2}$$

for every element $x, y \in Q$ with probability exceeding $1 - (\#Q)^{-\beta}$. An example of this type of distribution is the normal distribution where each elements r_{ij} of $\boldsymbol{R}$ follow $r_{ij} \sim N(O, 1)$. In addition, in [28] it was shown that the elements r_{ij} of $\boldsymbol{R}$ can be drawn i.i.d. from the following distribution

$$r_{ij} = \sqrt{3} \begin{cases} 1 \text{ with probability } 1/6 \\ 0 \text{ with probability } 2/3 \\ -1 \text{ with probability } 1/6 \end{cases} \tag{3}$$

The distribution in Eq. (3) is notably efficient, since it discards 2/3 of the data that correspond to multiplications by zero. Furthermore, it can be implemented using fixed point arithmetic operations consisting of only additions and subtractions if the scaling coefficient is factored out.

Compared to traditional dimensionality reduction approaches such as principal components analysis (PCA), the major benefit of RPs is the universality, i.e. the same RP matrix can be used without the need for training based on statistics, as is done is PCA. The fact that RPs do not require training is of particular importance, given that in most tracking scenarios a model of the object we wish to track may not be available beforehand. In the context of tracking, we apply the RPs method at each candidate window after the dense SIFT descriptors are extracted, thereby reducing its dimensionality from 1152 to 300 dimensions. This reduction is computationally efficient to apply, since it only requires a matrix multiplication but offers significant computational savings.

IV. Distance Metric Learning (DML)

The distance between two vectors is often measured using the Euclidean distance or the inner product. Recent research in the field of pattern classification has shown that using a more appropriate distance metric can significantly improve results. In supervised DML, the objective is to learn a new distance that will satisfy the pairwise constraints imposed by class label information. The new distance metric can be expressed as either a Mahalanobis-like distance or equivalently as a linear transformation of the input data. Formally, the distance between two data points x and $y \in \mathbb{R}^n$ is given by

$$d_{\mathrm{G}}(x, y) = \|x - y\|_{\mathrm{G}}^2 = (x - y)^T \mathrm{G}(x - y) \tag{4}$$

where $\mathrm{G} \in \mathbb{R}^{n \times n}$ is the distance metric. Alternatively, the matrix G can be decomposed as $\mathrm{G} = \mathrm{L}^{\mathrm{T}}\mathrm{L}$ in which case Eq. (4) becomes

$$d_{\mathrm{G}}(x, y) = \big(\mathrm{L}(\mathrm{x} - \mathrm{y})\big)^T \mathrm{L}(x - y). \tag{5}$$

The new distance is given by the Euclidean distance of the data projected into the L subspace. The matrix G is required to be positive semidefinite to guarantee that the new distance will satisfy the requirements for a metric, i.e. non-negativity, symmetry, and triangle inequality.

DML is closely related to subspace learning, as can be seen from Eq. (5). Subspace learning techniques have been extensively used in object tracking: for example, Eigentracking [6] applies principal components analysis (PCA) of the training data to identify a subspace that is subsequently utilized in tracking; linear discriminant analysis (LDA) [7] estimates a subspace that can both describe the data and provide discriminating information. However, eigentracking is an unsupervised method, i.e. it does not take class information into consideration, while the supervised LDA assumes that each class has a similar within-class distribution, which may not be true in all cases.

DML is primarily concerned with identifying a transformation that is optimized for classification. In general there are two non-exclusive approaches in DML, offline or batch processing and online. In offline processing, the underlying assumption is that all training data points and their corresponding labels are available during training. In online approaches, the assumption is that a single example is presented at each step, and the distance metric is modified so that it remains consistent with the previous examples and also satisfies the newly presented example.

One of the earliest approaches that explicitly discussed learning a distance function for classification in an offline fashion was proposed in [5], where distance metric learning was formulated as a constrained convex optimization problem. More recent approaches fall into the category of local DML and attempt to learn a distance metric that will satisfy the constraints in a local region around each data point instead of all pairwise constraints. Local DML approaches are easier to handle and are directly related to the local nature of the nearest neighbor classifier. Examples of local DML include the Neighborhood Component Analysis [8], the Multiple Collapsing Classes [9], and the Large Margin Nearest Neighbors [4].

In this paper, we utilize a recently proposed method for local DML called Information Theoretic Metric Learning (ITML) [10]. In ITML, given an initial estimate of the Mahalanobis distance matrix, G_0, the objective is to identify a new Mahalanobis matrix, G that will meet two constraints; it will be similar (in a KL divergence sense) to the original distance matrix and it will satisfy the class labels constraints. Assuming that each distance matrix G corresponds to the inverse of the covariance matrix of an unknown multivariate Gaussian distribution given by

$$\begin{aligned} p(x;\mathrm{G}) &= \frac{1}{Z}\exp(-\frac{1}{2}(x-y)^T\mathrm{G}(x-y)) \\ &= \frac{1}{Z}\exp(-\frac{1}{2}d_{\mathrm{G}}(x,\mu)) \end{aligned} \tag{6}$$

where μ is the mean and Z is a normalization constant, the Kullback–Leibler (KL) divergence can be employed as a robust metric of the correspondence between the two Gaussian distributions G and G_0, and is given by:

$$\begin{aligned} &KL\big(p(x;\mathrm{G}_0) \parallel p(x;\mathrm{G})\big) \\ &\qquad = \int p(x;\mathrm{G}_0)\log\frac{p(x;\mathrm{G}_0)}{p(x;\mathrm{G})}dx \end{aligned} \tag{7}$$

The second objective of ITML is to satisfy the class constraints, i.e. bring elements from the same class closer and move elements from different classes far apart. For example, let x_1, x_2, and x_3 be three vectors that belong to two classes C_1 and C_2 such that $\{x_1, x_2\} \in C_1$ and $\{x_3\} \in C_2$. The objective of a DML algorithm, and ITML in this case, is to identify a new distance matrix G such that the new distance metric d_{G} preserves the relationships $d_{\mathrm{G}}(x_1, x_2) < d_{\mathrm{G}}(x_1, x_3)$ and $d_{\mathrm{G}}(x_1, x_2) < d_{\mathrm{G}}(x_2, x_3)$.

Formally, the objective of ITML is to find the distance matrix G so that:

$$\min_{\mathrm{G}} KL(p(x;\mathrm{G}_0) \parallel p(x;\mathrm{G})) \tag{8}$$

subject to

$$\begin{aligned} d_{\mathrm{G}}(x,y) &\le l \ \text{ if } label(x) = label(y) \\ d_{\mathrm{G}}(x,y) &\ge u \ \text{ if } label(x) \ne label(y) \end{aligned} \tag{9}$$

The solution of the problem is found by a LogDet optimization. In [10], the authors introduced appropriate slack variables into the formulation for the case when the exact solution is not feasible due to unsatisfied constraints. A major benefit of the ITML method is that is does not require the expensive eigen-decomposition and thus is more computationally efficient.

In addition to offline algorithms, various online DML approaches have been presented. One of the earliest ones was the POLA algorithm [31] that optimize a large-margin objective. However, the algorithm required an eigenvector computation at each iteration, which can be slow in practice. Another approach is the recently proposed LogDet Exact Gradient Online (LEGO) [11] algorithm. Given two vectors u_t and v_t with distance $d_{\mathrm{G_t}}(u_t v_t) = \hat{y}_t$ and the target distance y_t, the LEGO algorithm updates the distance by minimizing

$$\mathrm{G}_{\mathrm{t+1}} = \arg\min_{\mathrm{G}>0} \mathrm{D}(\mathrm{G}, \mathrm{G}_{\mathrm{t}}) + \eta\ell(d_{\mathrm{G}}(u_t, v_t), y_t) \tag{10}$$

where $\mathrm{D}(\mathrm{G}, \mathrm{G}_{\mathrm{t}})$ is a regularization function, η is a regularization parameter and $\ell(\hat{y}_t, y_t)$ is the loss between the target distance $\hat{y}_t$ and the predicted distance y_t. The solution to the minimization problem is given by:

$$\mathrm{G}_{\mathrm{t+1}} = \mathrm{G}_{\mathrm{t}} - \frac{\eta(\bar{\mathrm{y}} - y_t)\mathrm{G}_{\mathrm{t}}\mathrm{z}_{\mathrm{t}}\mathrm{z}_{\mathrm{t}}^{\mathrm{T}}\mathrm{G}_{\mathrm{t}}}{1 + \eta(\bar{\mathrm{y}} - y_t)\mathrm{z}_{\mathrm{t}}^{\mathrm{T}}\mathrm{G}_{\mathrm{t}}\mathrm{z}_{\mathrm{t}}} \tag{11}$$

where $\mathrm{z}_{\mathrm{t}} = u_t - v_t$ and $\bar{y} = d_{\mathrm{G_{t+1}}}(u_t, v_t)$. The estimated distance $\bar{\mathrm{y}}$ is found by

$$\bar{\mathrm{y}} = \frac{\eta\hat{y}_t y_t - 1 + \sqrt{(\eta\hat{y}_t y_t - 1)^2 + 4\eta\hat{y}_t^2}}{2\eta\hat{y}_t} \tag{12}$$

V. Distance Metric Learning for Object Tracking

In this paper, we propose the use of batch and online DML for object tracking. To achieve this goal, we model the tracking problem as a binary classification problem. The first class includes visual examples of the object's appearance collected during tracking, while the second class includes visual examples of the background. The processing pipeline of the DML for tracking is shown in Figure 1, where the problem of tracking is translated to that of binary classification. The two classes are the object and the background shown in red and green boxes in the figure.

The background class is populated by regions that belong to the background and are spatially close to the object's location, while the object class is populated by regions containing the object at various time instances. Each region is presented as a point in some high dimensional feature space. Ideally, we would like the points corresponding to the background to be well separated from the points corresponding to the object. However, this may not be the case. We deal with this issue by applying a distance learning algorithm which produces a better separation between the background and the object. Regions

that belong to the object class collected from various time instances are grouped in the template library according to the template update strategy. Using the learned distance and the template library, the localization of the object in a new frame is achieved by locating the region that achieves the smallest distance with the template library.

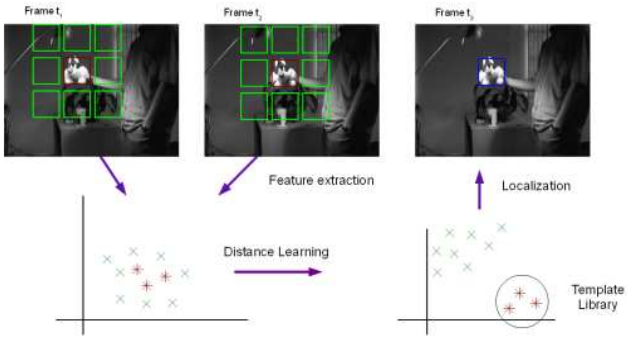

Figure 1: Example demonstrating distance learning for object tracking

The proposed tracking algorithm, termed DMLTracking, utilizes both offline and online distance metric learning to learn the appearance of the object and differentiate it with the appearance of the background. During initialization, a batch DML approach, the ITML, is employed to get a distance metric that is consistent with the initial appearance of the object and the background. In other words, during bootstrapping, the objective is to learn a distance metric that will produce small distances (more similar) between the appearances of the object in the first few frames, while it will create large distances (less similar) between the appearances of the object and the background. Once the initialization is complete, we utilize an online DML algorithm, the LEGO, to incrementally update the distance so that it remains consistent with the evolving appearance of both the object and the background.

The use of the ITML and the LEGO algorithms for distance learning offers three significant benefits with respect to the requirements of tracking. First, both algorithms introduce a term that enforces the *smoothness* in the variability of the learned distance matrix. This is especially important for online learning since we expect that the variation in the object's appearance will be smooth and therefore the learned distance would not significantly change in successive frames. In addition, the ITML and the LEGO algorithms are designed based on the *large margin property*. This property is well suited for tracking, especially in the application of ITML as bootstrapping, since we expect changes in the object's and background's appearance to take place. Introducing a large margin reduces the probability of misclassifying the background as part of the object or vice versa. Last, the LEGO has *low complexity*, a vital trade for a real-time tracking algorithm.

A. *Object Localization*

Under the smooth motion assumption, the object's location in the new frame will be near its location in the previous frame, so that, in most cases, only a small region has to be searched. This is fortunate, as exhaustively searching the full image for a region that exhibits high similarity with the tracked object is computationally demanding, and is not suitable for a resource constrained system.

In this work, we employ an iterative search method using the learned distance to localize the object in a new frame. The search pattern is initialized at the location of the object in the previous frame. For each region in the search pattern, the distance between the template library and the representation of the region is estimated and the region corresponding to the smallest distance is found. If this region is at the centre, it means the best possible location is found. Otherwise, the search pattern is realigned and the search continues. Figure 2 displays the pattern of locations that are searched at each iteration. Each point corresponds to the centre of a candidate search region with the same size as the target region. We selected this type of search pattern because higher sampling rate near that centre can lead to more accurate localization, while lower sampling rate further away from the centre can support faster object motion with lower complexity.

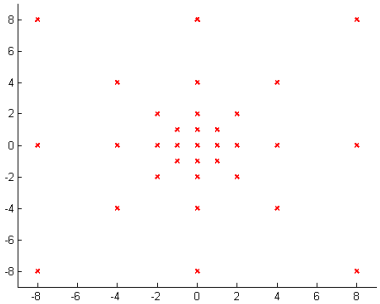

Figure 2: Region search pattern

Given the object's appearance at time $\mathrm{t}-1$ and spatial location X, the object's representation (SIFT over regular grid and random projections) is given by $\hat{\mathrm{I}}(\mathrm{t}-1,\mathrm{X})$. The objective of the template matching mechanism is to estimate the new object location $\tilde{X}$ given by

$$\tilde{X} = X + \Delta X = \min_{X+\Delta X \in S} \min_{\mathrm{i}=\{1\ldots\mathrm{p}\}} d_{\mathrm{G}}(\hat{\mathrm{I}}(\mathrm{t},\mathrm{X}+\Delta X),\mathrm{M}_i) \quad (13)$$

where M_i for $i=\{1\ldots p\}$ is the template library, d_{G} is the learned distance and S is the search pattern. We discuss the design of the template library in Section 5.3.

B. *Distance Metric Update*

In this work we employ a learned distance to estimate the new location of the object in Eq. (13). We consider the scenario where the object's characteristics are not known a-priori. As such, we cannot train a DML algorithm offline. However, relying solely on incrementally learning the appropriate distance can lead to drift. We tackle this problem by following a hybrid approach. We use Eq. (13) without any

learned distance metric (i.e. matrix G corresponds to the identity matrix) for the first few frames (4 in this case) to collect a baseline appearance of the object and the background. Once the baseline representations are collected, we run the ITML algorithm to learn the object-specific distance metric as well as the corresponding thresholds, as shown in Eq. (8) and Eq. (9). This type of bootstrap initialization offers two major benefits.

First, during the bootstrapping process, we assume that the appearance of the object and the appearance of the background remain relatively stationary for the first few frames and therefore we can apply a batch DML algorithm to obtain a reliable estimate of the object's and background's appearance. Second, in addition to modeling the appearance of the object and the background, we also obtain the distances between all examples from the same class and the distances between all the examples from different classes. We denote as l in Eq. (9), the maximum allowable distance between elements from the same class and u in Eq. (9) the minimum allowable distance between elements from different classes. In our experiments, we selected the threshold l to be 95% of the maximum distance between examples from the same class and u to be 5% of the minimum distance between examples from different classes. These values are used as thresholds and can subsequently be used to identify occlusions and guide the distance metric update strategy. Alternatively, if an estimate of the object's appearance is available beforehand, such as the case of face tracking, we can obtain initial estimates of the learned distance and threshold by training the DML algorithm offline using numerous visual examples.

In both cases, the offline and the bootstrap, the learned distance should be updated to incorporate changes in the object's appearance. Following the previous notation, $\hat{\mathrm{I}}(\mathrm{X};\mathrm{t})$ is the representation of the image region that contains the object, $\mathbf{M} = \{M_i \mid i = 1 \dots m\}$ is the template library and $\hat{\mathrm{J}}(\mathrm{X_j};\mathrm{t})$, where $j = \{1, \dots c\}$, are representations of the image regions that are close to the previous location X but do not contain the object (background regions). The objective of online distance metric learning is to update the distance matrix $\mathrm{G_t}$ to the new distance matrix $\mathrm{G_{t+1}}$ so that

$$d_{\mathrm{G_{t+1}}}\left(\hat{\mathrm{I}}(\mathrm{X};\mathrm{t}), \mathbf{M}\right) \ll d_{\mathrm{G_t}}\left(\hat{\mathrm{I}}(\mathrm{X};\mathrm{t}), \mathbf{M}\right) \tag{14}$$

and

$$d_{\mathrm{G_{t+1}}}\left(\hat{\mathrm{I}}(\mathrm{X};\mathrm{t}), \hat{\mathrm{J}}(\mathrm{X_j};\mathrm{t})\right) \gg d_{\mathrm{G_t}}\left(\hat{\mathrm{I}}(\mathrm{X};\mathrm{t}), \hat{\mathrm{J}}(\mathrm{X_j};\mathrm{t})\right) \tag{15}$$

In the context of tracking, we identified two types of distances that have to be updated. The first one is the distance between the current object appearance and the elements of the template library. This distance should be as small as possible, i.e. $d_{\mathrm{G_{t+1}}}\left(\hat{\mathrm{I}}(\mathrm{X};\mathrm{t}), \mathbf{M}\right) \approx \epsilon$. The second type is the distance between the current object appearance and the neighboring background regions $\hat{\mathrm{J}}(\mathrm{X_j};\mathrm{t})$ that do not contain the object. This distance should be large, i.e. $d_{\mathrm{G_{t+1}}}\left(\hat{\mathrm{I}}(\mathrm{X};\mathrm{t}), \hat{\mathrm{J}}(\mathrm{X_j};\mathrm{t})\right) = u + \delta$, where u is the threshold in Eq. (9) and δ is an arbitrary high valued constant. In this work, the distance thresholds are learned during bootstrapping and remain static during tracking.

C. *Template Library Update*

To keep the template library up-to-date with the appearances of the object, it should be updated by replacing old templates with new ones. One of the benefits of using DML is that distances corresponding to similar template elements and can be used to predict if and how the template library should be updated. When a new frame is presented, the element with the highest distance is replaced with the new one. Formally, let

$$s_t = \min_i d_G(\hat{\mathrm{I}}(\tilde{X}), \mathrm{M_i}) \tag{16}$$

be the minimum distance between the object's appearance at the estimated location $\tilde{X}$ and template library elements $\mathbf{M} = \{M_i \mid i = 1 \dots m\}$. We can identify occlusion by comparing this distance with the threshold for the minimum allowable distance u between the object and the background. If $s_t > u$, then we can infer that the new location contains something that is more similar to the background than the object, which is indicative of occlusion. If $s_t \leq u$, then the window probably contains the object and therefore it can be used as an example for the template library.

To update the template library, we first have to decide if the new appearance is informative enough to be included in the library. A new appearance is considered a candidate for the template library if $s_t > s_{t-1}$, i.e. if the similarity between the appearance of the object in the current frame and the examples in the template library is less than the similarity achieved in the previous frame. If this assertion holds, then the template element that achieves the smallest distance (most similar) to the object's current appearance is replaced with the new appearance. The proposed update mechanism is motivated by the fact that a template element that is very similar to the object's current appearance carries little information, whereas a template element with higher distance is more informative. By maintaining a small number of template examples, instead of a single previous one, we ensure that if the appearance of the object suddenly changes in a frame and then returns to the original one, at least one template element will be consistent with the original object appearance and therefore will not be evicted from the template library during update.

D. *Overview of the proposed tracking algorithm*

In this section we provide an algorithmic description of the proposed tracking method. The algorithm consists of two parts; the bootstrapping process and the tracking process. Bootstrapping is the initialization of the object's model given its appearance in the first few frames and is described by the following steps:

Algorithm 1: Bootstrapping process

Input: Initial object location from first n frames

1. Collect image windows from the same location $I^+ = \{I_1^+, .., I_m^+\}$ (positive samples) and windows around the location $I^- = \{I_1^-, .., I_n^-\}$ (negative samples) from first n frames using Eq. (13) with $G = I$, where I is the identity matrix.
2. Perform feature extraction on the collected samples to obtain $\hat{I}^+$ and $\hat{I}^-$
3. Run batch distance metric learning (ITML) such that $d_G(\hat{I}^+, \hat{I}^-) \geq d_G(\hat{I}^+, \hat{I}^+)$ for all examples positive and negative examples

Output: Distance matrix G, thresholds l and u

During the bootstrapping process, it is assumed that the object's appearance will remain relatively the same for the first few frames (4 in our experiments), given its location in the first frame and therefore applying Eq. (13) with the identity matrix will not result in misclassification. In our experimental results, this assumption did not cause any instability to the ITML. During runtime, for each new frame, object localization and model update are achieved according to the following process:

Algorithm 2: Tracking process

Input: Object location, distance metric G_t, thresholds l and u, template library $\mathbf{M}_t$

1. Collect candidate windows using the search pattern from previous location $(t, X + \Delta X)$
   1.1. Perform feature extraction to get $\hat{I}(t, X + \Delta X)$
   1.2. Measure the distance to all the elements from template library $\mathbf{M}_t$and return the one that achieved the smallest distance s_t
2. If the window that achieves the distance s_t does not correspond to the central region of the search pattern, update location and return to step 1, otherwise
3. Set new object location equal to the location of the central region and calculate the distance with the template library examples using Eq. (16)
4. If distance $s_t < u$, where u is a threshold
   4.1. Collect negative examples I^- (around object's location) and positive example I^+ (the window containing the object)
   4.2. Update distance metric matrix G_{t+1} using Eq. (10)
   4.3. If $s_t > s_{t-1}$, update template library to $\mathbf{M}_{t+1}$ by removing the most similar element (smallest distance) according to the updated distance metric G_{t+1}.
5. If distance $s_t > u$
   5.1. Set the object as lost/occluded and stop the updates

Output: Object location $\tilde{X}$, updated distance metric G_{t+1}, updated template library $\mathbf{M}_{t+1}$

VI. Experimental Results

A series of experiments were conducted to evaluate the tracking accuracy of the proposed system. For all test sequences, the same set of parameters was used, i.e. the system parameters were not adjusted for each video sequence, which corresponds to a more realistic scenario and is more challenging. Regarding object localization, we utilized the search pattern shown in Figure 2 in an iterative localization process. Each candidate image region was resized to 50x50 pixels and 9 SIFT descriptors were extracted. The 1152 dimensional representation of the window was reduced to 300 dimensions using the RPs methods. In order to obtain an initial model of the object and the background, the first 4 frames were collected and the object's location was extracted using an identity matrix as the distance matrix. In addition, for each frame eight regions of background were collected around the estimated location of the object in eight spatially adjacent locations at the following orientations: 0°, 45°, 90°, 135°, 180°, 225°, 270° and 315°. For the bootstrapping process, the ITML was applied with a gamma value set to 10 and the number of neighbours set to 5. To maintain a rich representation of the object's appearance, 4 templates were used from previous time instances. These templates were updated only if the newly obtained object representation was below the appropriate threshold. The same threshold was used to update the learned distances using the LEGO algorithm. The learning parameter η of the LEGO algorithm was set to 0.6 for learning the object's appearance and 0.1 for learning the background's appearance. Updating the template library and the learned distance on every frame, our Matlab implementation currently operates at around 5 fps in a dual core desktop computer.

The objective of the experiments was to compare the performance of the proposed DMLTracking algorithm with state-of-the-art object tracking algorithms in challenging scenarios. The experiments were carried out using publically available video sequences. The video sequences shown in Figures 3-10 are called "Coke Can", "Coupon", "Sylvester", "Tiger 2", "David", "Girl", "Face occluded" and "Face occluded 2" and where download from [32], while the sequence "Car chasing" in Figure 11 was downloaded from Google videos. This set of video sequences represent challenging scenarios for an object tracking algorithm due to the significant variations of the object's and the background's appearance. The sequences in Figures 6, 9, 10 and 11 investigate the trackers' ability to maintain accurate tracking when severe occlusions take place. The occlusions shown in these sequences can confuse online trackers so that the occluding object, instead of the actual object, is modeled and tracked, which causes drift. The sequences shown in Figures 3, 5, 6, 7, 8, 10 and 11 evaluate the ability of the tracker to

maintain an accurate object model despite temporary and permanent changes in appearance. For example, the sequence in Figure 7 shows a person walking while changing his pose, removing his glasses and changing expression. Another type of obstacle that a tracker must overcome is that of similarly looking objects that are part of the background. Characteristic examples of this type of challenge are shown in Figures 4 and 11. Finally, dramatic changes in illumination, such as the ones shown in Figures 5 and 11, can also create confusion to a tracker. We note that we explored generic object tracking, as opposed to specific objects such as human faces. Tracking a generic object is far more challenging than a specific, predefined one, since the lack of a prior model makes the identification of occlusions and changes in appearance very challenging. The performance of the DMLTracking algorithm (shown using red solid lines) was compared with two state-of-the-art online tracking algorithms, the Semi-supervised On-line boosting (SemiBoost) [13] (shown using green dot-dashed lines) and the Multiple Instance Learning tracking (MILTrack) [21] (shown using blue double-dashed lines).

The first sequence, called “Coke Can” (Figure 3), examines the case of tracking a rigid object that undergoes changes in appearance and illumination in a cluttered scene. The changes in appearance are due to the rotation of the object and the illumination changes are caused by moving under a direct light source. We observe that the DMLTracker is able to maintain an accurate tracking in contrast to the Semiboost tracking that looses the object in several frames. The MILTracker achieves similar performance.

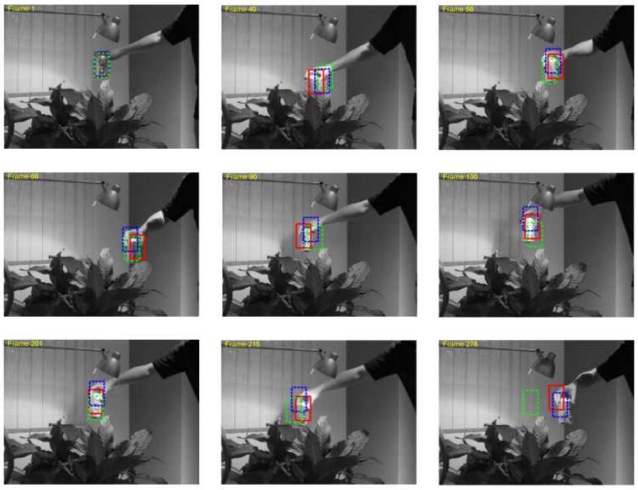

Figure 3: “Coke Can” Sequence. A Coke can is shown in a cluttered background while it undergoes changes in appearance due to out-of-plane rotations and changes in illumination caused by the direct light source

The second sequence, called the “Coupon Book” (Figure 4), examines the case where the object undergoes a permanent change in appearance while a similarly looking object is also present in the scene. This example illustrates the ability of the tracker to update the appearance model of the object and separate the tracked object from another object with similar initial appearance.

We observe that the DMLTracker correctly updates the appearance of the object and does not become confused by the similarly looking object. Similar results are evident for the MILTracker. The Semiboost tracker on the other hand, builds an initial model from the first frame and is not able to adapt to the new appearance with causes the tracker to lock on the similar object and not the tracked object

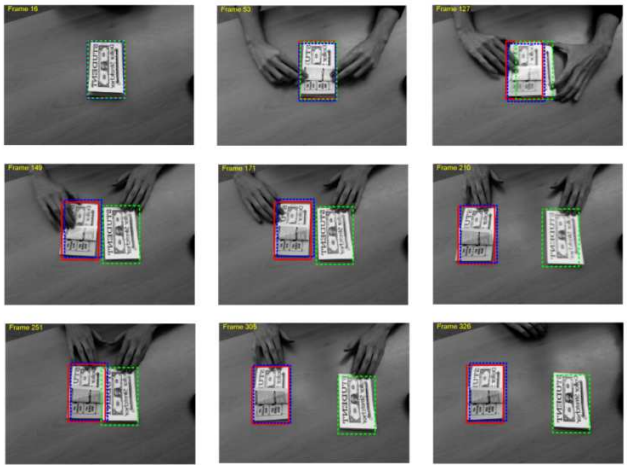

Figure 4: “Coupon” Sequence. A booklet is tracked while it changes its appearance (folding) and a new object with similar appearance is introduced in the scene.

The “Sylvester” sequence (Figure 5) examines the ability of the tracker to maintain accurate tracking in long sequences (1300 frames) while going through changes in appearance due to pose and illumination. The object is a toy animal that is initially presented under a direct light source.

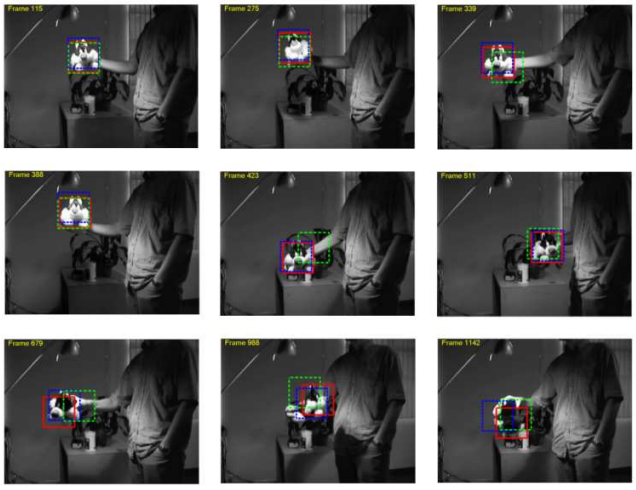

Figure 5: “Sylvester” Sequence. A toy animal is shown moving under significant illumination changes caused by direct light source while significantly changing appearance due to out-of-plane rotation.

We observe that during the duration of the tracking, the object changes appearance due to rotation while moving in and out of the direct light source, which causes significant changes in appearance. Furthermore, the object is moving in a cluttered scene, and this makes the tracking even more challenging. We observe that the DMLTracker correctly tracks the object despite the challenging conditions and achieves

similar performance with the MILTracker. The Semiboost tracker also maintains tracking; however it provides lower localization accuracy.

The “Tiger 2” (Figure 6) sequence presents the case where the object suffers severe occlusions while changing appearance. More specifically, a toy animal is shown moving behind a plant. The cluttered foregrounds, in addition to changes in appearance of the object, create very challenging conditions. We observe that the Semiboost Tracker loses the object in many frames and locks on to a region similar to the region that causes the occlusion. The DMLTracker achieves higher accuracy, but it eventually suffers from drift at the end of the sequence. The drift is caused by the inability of the DMLTracker to accurately model the object under all possible occlusion conditions. In this scenario, the MILTracker achieves the best results.

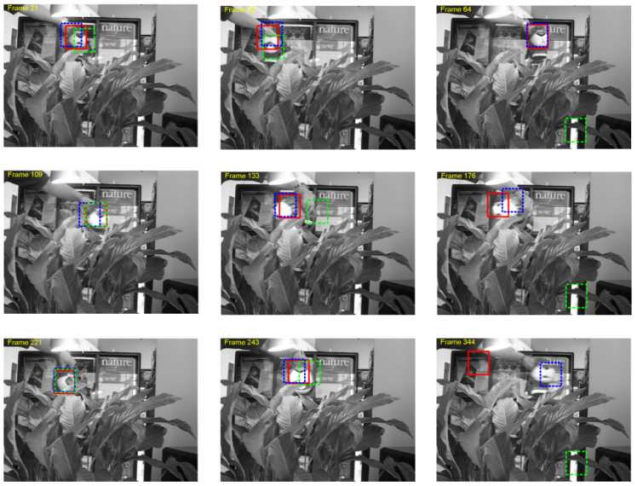

Figure 6: “Tiger 2” Sequence. A toy animal is tracked while it is partially occluded by a plant and changing appearance.

The next sequence, called “David Indoor” (Figure 7), studies the case of realistic face tracking. The face undergoes several changes in appearance due to illumination, changes in pose, and permanent appearance changes (glasses). More specifically, the person enters the scene from a dark room which immediately creates a challenge with respect to illumination robustness. Subsequently the face undergoes changes in size, viewpoint and appearance (glasses).

In this sequence, the DMLTracker achieves the best results with much higher localization accuracy compared to both the MILTracker and the Semiboost Tracker. The increased performance is a consequence of the double model update mechanism via the updates in the distance metric and the updates in the template library. The Semiboost tracker often fails to update the appearance and drifts, while the MILTracker is more stable but less accurate than DMLTracker.

A similar scenario is examined in the “Girl” (Figure 8) sequence with more challenging changes in the appearance of the face due to the out-of-plane rotation, changes in scale and the presence of another similarly looking object (the man’s face). The DMLTracker is able to update the object’s appearance model without suffering drift. The MILTracker and the Semiboost tracker maintain tracking, however they exhibit lower accuracy and in one instance (frame 423) the Semiboost tracker is confused by the other face.

Figure 7: “David Indoor” Sequence. A face tracking sequence is presented where the face is shown in different scales, from different viewpoints and under different appearances e.g. expressions and glasses.

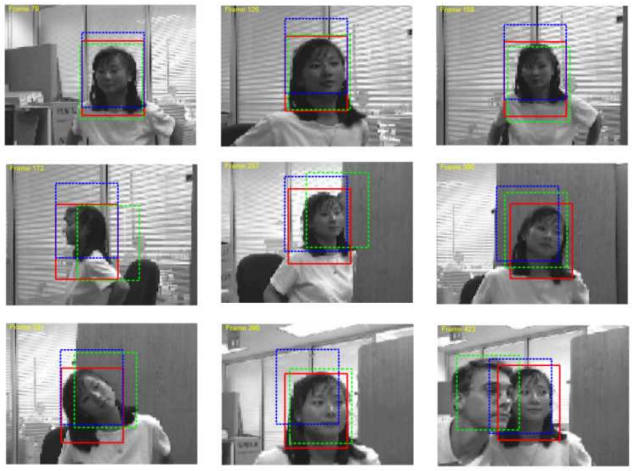

Figure 8: “Girl” Sequence. A face is tracked while changing appearance and being occluded by another face

The previous sequences examined the behavior of the proposed tracking algorithm, as well as state-of-the-art tracking algorithm in scenarios where the object undergoes severe changes in appearance, usually due to out-of-plane rotations and illumination changes. In the next two sequences, we examine the behavior of these algorithms when significant occlusions take place. The “Face occluded” (Figure 9) and “Face occluded 2” (Figure 10) sequences present situations of face tracking under partial and complete occlusions. The “Face occluded” sequence is less challenging than the “Face occluded 2”, since the appearance of the face does not change during the sequence.

We observe that all three algorithms correctly maintain the tracking without drifting. We note, however, that the

difference in localization accuracy presented in Table 1 is due to the unclear nature of ground truth in this sequence. In other words, the tracking window may move to cover most of the object that is visible or it may wait until the object is visible again. The DMLTracker and the MILTracker follow the former approach, while the Semiboost follows the latter approach which causes the apparent inconsistency in the localization results.

Figure 9: "Face Occluded" Sequence. An occluded face is tracked.

In addition to overcoming the drift problem, the DMLTracker is able to report significant. This is achieved because the current view of the object starts to become more similar to the background and thus its distance with the rest of the template library examples starts to exceed the same class threshold.

The problems caused by occlusion are more evident in the second sequence shown in Figure 10, called "Face occluded 2", where the object undergoes changes in appearance in addition to undergoing occlusions.

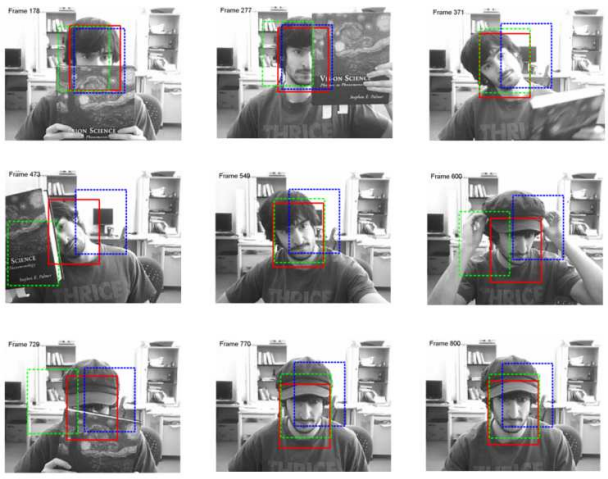

Figure 10: "Face Occluded 2" Sequence. A face is tracked while undergoing occlusions with significant variations in appearance.

In this scenario, the optimum behavior for a tracker is to update the model when the face changes appearance (moving the head, adding the hat) while being able to identify occlusions and stop updating the model before it causes drift. In this scenario the DMLTracker achieves superior accuracy compared to both the MILTracker and the Semiboost Tracker.

The last case we explore involves tracking an object from a moving camera under challenging illumination changes and occlusion. This scenario is presented in Figure 11, where a car is followed by a helicopter mounted camera while it is moving on a highway.

As we can see, the car undergoes severe illumination conditions, as shown in the second image on the top row, as well as occlusions. We observe that the proposed tracking scheme (shown in red) is able to handle the demanding requirements and maintain accurate tracking. On the other hand, both the MILtracker (in green) and the SemiBoost (in blue) fail to maintain tracking. The MILtracker fails to follow the car when it goes under the traffic signs and completely loses the object afterwards. Semiboost is more robust, since it handles the first complete occlusion, but suffers from drift that eventually causes complete failure of the tracker during the second occlusion.

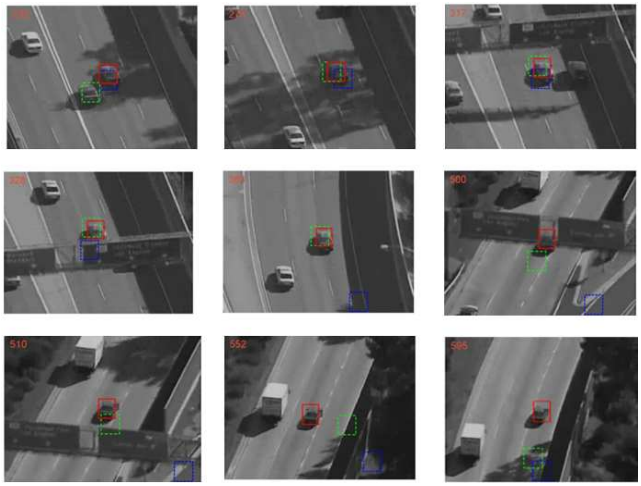

Figure 11: "Car chasing" Sequence. A car is tracked while moving at high speed close to other similarly looking cars and suffering occlusions by the traffic signs.

To supplement the results shown in Figures 3-11 we provide Table 1 with the localization accuracy of four state-of-the-art object tracking algorithms as well as the proposed one. These algorithms include the Online AdaBoost (OAB) tracker [22], the SemiBoost tracker [13], the Fragments based tracker (FragTrack) [34] and the MILTracker [21]. The results in the table correspond to mean distance in pixels between the center of the tracking window and the center of the ground truth window after 5 independent trials. The best performance is indicated by a smaller number corresponding to closer match between the predicted center and the ground truth center. The results marked in red indicate the best performance and the ones in green correspond to the second best. We observe that the proposed algorithm, the DMLTracking, achieves the best

performance in six out of nine sequences and the second best in two others. The accuracy in the "Coke Can" is comparable to the SemiBoost tracker and significantly better than the MILTracker. In the sequences "Coupon", "David" and "Occluded Face 2", the DMLTracking algorithm outperforms all other algorithms by a large margin, while in the "Car chasing" sequence, the DMLTracking algorithm is the only one that correctly maintains the tracking through the sequence and achieves the best localization results. The only case where the proposed algorithm does not achieve top performance is in the case of "Occluded Face". The lower performance is most likely due to the fact that the proposed DMLTracking algorithm, as well as the MILTracker, try to select the largest portion of the visible object, while methods such as SemiBoost and the FragTrack try to estimate where the object might be behind the occlusion and select that region.

Table 1: Localization Accuracy on Generic Object Tracking

Sequence/ Method	OAB	Semi Boost	Frag Track	MILtracking	DMLTracking
Coke Can	24.80	13.09	63.44	20.13	**12.84**
Coupon Book	24.93	66.59	55.88	14.74	**5.68**
Sylvester	25.21	15.84	11.12	10.82	**9.79**
Tiger2	33.41	61.20	36.64	**17.85**	31.39
David Indoor	49.23	38.87	46.27	23.12	**8.82**
Girl	**27.12**	48.32	68.01	52.21	33.95
Occluded Face	43.50	6.98	**6.34**	27.23	19.29
Occluded Face 2	21.46	22.86	45.19	20.19	**14.97**
Car chasing	79.09	19.78	95.17	82.07	**6.65**

VII. Conclusions

In this paper, we propose the use of an online discriminative learning mechanism for robust tracking of objects without any prior model regarding their appearance. The proposed scheme employs distance metric learning to reliably represent the similarity between different appearances of the object as well as the difference in appearance between the object and the background. The object's location in a new frame is found by selecting the region that minimizes the distance relative to a library of templates. Both the distance metric and the template library are updated online in order to adapt to the object's appearance as well as to changes in illumination and pose. We employ a bootstrapping process for the initial estimation of the object's appearance. The representation of each image window is based on a combination of SIFT features extracted over a regular grid and Random Projections for dimensionality reduction. Experimental results suggest that the proposed algorithm is robust to changes in pose and illuminations and occlusions and achieves performance comparable to state-of-the-art tracking algorithms.

Acknowledgment

The authors would like to thank the anonymous reviews for their comments. This research was supported in part by the Eastman Kodak Company and the Center for Emerging and Innovative Sciences (CEIS), a NYSTAR-designated Center for Advanced Technology in New York State.

References

[1] D. Commaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 25, no. 5, pp. 564–577, 2003.

[2] A. Elgammal, R. Duraiswami, and L. Davis, "Probabilistic tracking in joint feature-spatial spaces," IEEE Int. Conf. On Computer Vision and Pattern Recognition, vol. 1 2003.

[3] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," ACM Computing Surveys, vol. 38, no. 4, pp. 1-45, 2006.

[4] K.Q., Weinberger, and L.K. Saul, "Distance metric learning for large margin nearest neighbor classification," The Journal of Machine Learning Research, vol. 10, pp. 207-244 2009.

[5] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning with application to clustering with side-information," Advances in Neural Information Processing Systems, 2003.

[6] 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, vol. 26, no. 1, pp. 63-84, 1998.

[7] R.T. Collins, Y. Liu, and M. Leordeanu, "Online Selection of Discriminative Tracking Features," IEEE Trans. Pattern Analysis and Machine Intelligence, 27(10): 1631-1643 2005.

[8] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighborhood component analysis," Advances in Neural Information Processing Systems, 2004.

[9] A. Globerson, and S. Roweis, "Metric learning by collapsing classes," Advances in Neural Information Processing Systems, 2006.

[10] J.V. Davis, B. Kulis, P. Jain, S. Sra, and I.S Dhillon, "Information-Theoretic Metric Learning," International Conference on Machine Learning, pp. 209-216, 2007.

[11] P. Jain, B. Kulis, I.S. Dhillon, and K. Grauman, "Online metric learning and fast similarity search," Advances in Neural Information Processing Systems, 2008.

[12] H. Liu, and F. Sun, "Semi-supervised ensemble tracking," Proc. International Conference on Acoustics, Speech and Signal Processing, pp. 1648-1648, 2009

[13] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," European Conference on Computer Vision, pp. 234-247, 2008.

[14] M. Isard, and A. Blake, "CONDENSATION - Conditional Density Propagation for Visual Tracking," International Journal of Computer Vision, vol. 29, no. 1, pp. 5-28, 1998.

[15] A.D. Jepson, D.J. Fleet, and T.F. El-Maraghi, "Robust Online Appearance Models for Visual Tracking," IEEE Trans. on Pattern Recognition and Machine Intelligence, vol. 25, no. 10, pp. 1296-1311, 2003.

[16] L. Matthews, T. Ishikawa, and S. Baker, "The Template Update Problem," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 26, no. 6, pp. 810-815, 2004.

[17] J. Lim, D. Ross, R.S. Lin, and M.H. Yang, "Incremental Learning for Robust Visual Tracking," International Journal of Computer Vision, vol. 77 no.1-3, pp. 125-141, 2008.

[18] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley, "Face Tracking and Recognition with Visual Constraints in Real-World Videos," IEEE Int. Conf. On Computer Vision and Pattern Recognition, pp. 1-8, 2008.

[19] S. Avidan, "Support vector tracking," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 26, no. 8, pp. 1064 – 1072, 2004.

[20] S. Avidan, “Ensemble tracking,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 2, pp. 261-271, 2007.
[21] B. Babenko, M.H. Yang, and S. Belongie, “Visual Tracking with Online Multiple Instance Learning,” IEEE International Conference on Computer Vision and Pattern Recognition Workshops, pp. 983-990, 2009.
[22] H. Grabner, M. Grabner, and H. Bischof, “Real-time tracking via on-line boosting,” Proc BMVC, vol. 1, pp 47-56, 2006.
[23] D.G. Lowe, “Object recognition from local scale-invariant features,” International Conference on Computer Vision, vol. 2, pp. 1150-1157, 1999.
[24] N. Gheissari, T. Sebastian, P. Tu, J. Rittscher, and R. Hartley, “Person reidentification using spatiotemporal appearance,” IEEE Int. Conf. On Computer Vision and Pattern Recognition, 2006.
[25] J. Li, S.K. Zhou, R. Chellappa, “Appearance context modeling under geometric context,” International Conference on Computer Vision, 2005.
[26] J. Yang, K. Yu, Y. Gong, and T. Huang, “Linear spatial pyramid matching using sparse coding for image classification,” IEEE Int. Conf. On Computer Vision and Pattern Recognition, pp.1794-1801, 2009.
[27] G. Tsagkatakis and A. Savakis, “A random projections model for object tracking under variable pose and multi-camera views,” International Conference on Distributed Smart Cameras, pp. 1-7, 2009.
[28] D. Achlioptas, “Database-friendly random projections: Johnson-Lindenstrauss with binary coins,” Journal of Computer and System Sciences, vol. 66, no. 4, pp. 671-687, 2003.
[29] H. Zhou, Y. Yuan, and C. Shi, “Object tracking using SIFT features and mean shift,” Computer Vision and Image Understanding, vol. 113, no. 3, pp 345-352, 2009.
[30] A. Vedaldi, B. Fulkerson, “Vlfeat: Feature extraction library” http://www.vlfeat.org/ (last accessed on 6/2010)
[31] S. Shalev-Shwartz, Y. Singer, and A.Y. Ng, Online and batch learning of pseudo-metrics,” Proceedings International Conference on Machine learning, 94, 2004.
[32] http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml (last accessed on 6/2010)
[33] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” IEEE Int. Conf. On Computer Vision and Pattern Recognition, vol. 2, 2006.
[34] A. Adam, E. Rivlin and I. Shimshoni, “Robust Fragments-based Tracking using the Integral Histogram,” IEEE Int. Conf. On Computer Vision and Pattern Recognition, pp. 798-805, 2006.
[35] W. Johnson, and J. Lindenstrauss, “Extensions of Lipschitz mapping into a Hilbert space,” Proc. Conference on Modern Analysis and Probability, pp 189-206, 1984.

Grigorios Tsagkatakis, (S’08) received his B.S. and M.S. degrees in electronics and computer engineering from Technical University of Crete, Greece in 2005 and 2007 respectively. He is currently working towards his Ph.D. in imaging science at the Center for Imaging Science, Rochester Institute of Technology, USA.

He was a teaching and research assistant with the Department of Electronics and Computer Engineering and has worked on various European funded projects from 2003 to 2007. His main research interests include computer vision and machine learning. He is presently teaching assistant with the department of Computer Engineering at RIT and research assistant with at RIT’s Real Time Computer Vision Lab working on human computer interaction and computer vision for smartphones. He was awarded the best paper award in Western New York Image Processing Workshop in 2010 for his paper “A Framework for Object Class Recognition with No Visual Examples”.

Andreas Savakis, (M’91, SM’97) received the B.S. (Summa Cum Laude) and M.S. degrees from Old Dominion University and the Ph.D. from North Carolina State University, all in Electrical Engineering. He is currently servicing as a professor and Department Head of Computer Engineering and member of the Ph.D. faculty in Imaging Science and Computing and Information Sciences at the Rochester Institute of Technology. Before joining RIT he was with the Eastman Kodak Company.

His research interests include computer vision, image processing and medical imaging algorithms and their implementation on mobile systems, multi-camera environments and high performance platforms. His research has generated 11 patents and over 80 publications in journals, conferences and book chapters. Dr. Savakis also serves as ABET evaluator for Electrical Engineering and Computer Engineering Programs. His activities were recognized by the IEEE Third Millennium Medal from the IEEE Rochester Section in 2000, and the NYSTAR Technology Transfer Award for Economic Impact in 2006.