Sketch Metric Learning

Yuting Mai†, Weihong Li†, Yongyi Tang†, Xixi Bi‡ and Wei-Shi Zheng†*
†School of Data and Computer Science, Sun Yat-sen University, China
‡Station of Automation, 95826 Force, China
*wszheng@ieee.org

***Abstract*—The main theme of this paper is to develop a systematic framework to learn a Mahalanobis distance metric based on matrix sketching. Within this framework, we present a novel sketch metric learning algorithm which sequentially sketches the received samples from training dataset and formulates a new kind of constraint for metric learning. This is in contrast to the traditional constraints that are only consisted of data from training dataset. In this paper, one training instance in the constraint is replaced by a pseudo center, which is generated during the sketching stage. Due to this change, our learning algorithm can focus on pushing every received sample to its corresponding similar pseudo center closer and pulling it far away from the dissimilar one. In addition, it can further achieve better performance of some kinds of time-varying process (e.g. on object tracking) than the compared related competitors. We demonstrate how to implement other methods in our algorithm framework and experiment to show that our method outperforms the competitors and relevant baselines on multiple datasets.**

I. Introduction

Distance metric learning is a fundamental topic in computer vision and pattern recognition. It is critical to many real-world applications, such as object tracking, classification, and clustering [1]–[7]. Numerous algorithms have been proposed and examined for distance metric learning. In particular, the learning methods can be generally classified into two categories: supervised learning and unsupervised learning. We refer the readers to [8], [9] for a comprehensive survey on metric learning. In this work, we focus on supervised distance metric learning.

In the typical supervised learning, where each training sample is associated with a label, the training samples are cast into pairwise constraints [10]–[13]: including 1) the intra-class constraint, $i.e.$ pairs of data samples belong to the same class labels should be close together, and 2) inter-class constraint, $i.e.$ pairs of data samples belong to different class labels should not be near in the metric they learned. Most successful results are relied on the assumption that all constraints are accessed at the onset of the metric learning. This can be called as offline or batch learning. However, in many real-world applications, the desired distance function may need to be changed gradually over time as additional information of input samples are received. For instance, object tracking is a time-varying process which deals with a dynamic stream data in an online manner.

To address the need of metric learning, plenty of works on online metric learning attempt to handle tuples that are received sequentially [4], [10], [13]. Essentially, these algorithms predict whether the two samples of the tuple have the same class label whenever a new tuple is observed. After the prediction, an instantaneous loss is computed to measure the similarity between the prediction and the true class label. At the end of each step, some parameters should be updated in order to improve the prediction performance of the algorithms. These online learning algorithms, however, are not in a real online manner due to the need of receiving pairs of samples in a batch setting. In other words, they need to collect a certain number of samples to formulate tuples and then a small random subset of them are used to update the metric. Moreover, random selection of the tuples could lead to an incomplete and indeterministic learning.

To deal with these problems, we redefine the constraint for a more efficient online learning method. Inspired by the Matrix Sketching in [14], we propose to sketch the information of each class, which is able to represent the principal structures within each class. This sketched information is stored in a matrix which will be updated every time when it is full. A pseudo center based metric learning framework is formulated after we extracted the pseudo center and constructed tuples from the sketched matrix of each class. Since the sketched matrix will be updated if new samples are arrived, our framework is online. We call the proposed framework *sketch metric learning* (SML). Moreover, SML can be extended so that it can be combined with most of the developed metric learning methods to improve their performances.

The rest of this paper is organized as follows. In section II, we will give a review of the online metric learning. The proposed sketch metric learning framework will be discussed in Section III. To demonstrate the compatibility and efficiency of our framework while incorporating the existing state-of-the-art metric learning algorithms, we separate our experiments into two main parts in Section IV. First, we evaluate our learning algorithm for the classification task using several public datasets. Second, we apply the learning algorithms to object tracking problem on several challenging videos. Experiments show that our method outperforms the existing approaches in general. Finally, we will conclude this paper in Section V.

II. A Review of Online Metric Learning

To our best knowledge, the first online metric learning method can be dated back to Shalev-Shwartz $et\ al.$'s work [10] where the online learning is developed for learning

* Corresponding author

978-1-5090-0620-5/16/$31.00

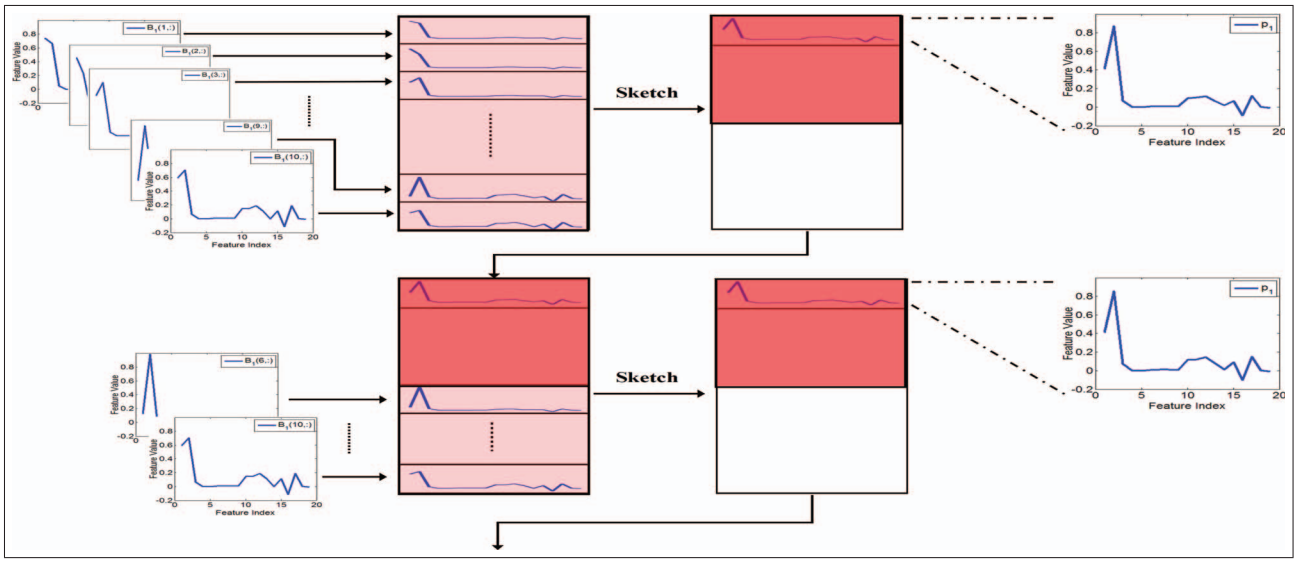

Fig. 1: An illustration of how to generate the pseudo center, where the dimension $d = 19$ and the sketch length $\ell = 10$. Each plot in the generating process represents an instance $\mathbf{x}$ or $\mathbf{p}$ which have been normalized by $\frac{\mathbf{x}}{\|\mathbf{x}\|_2}$ or $\frac{\mathbf{p}}{\|\mathbf{p}\|_2}$. The x-axis and y-axis in each graph represent feature index and feature value respectively.

a Mahalanobis distance metric. At each time, given a pair of samples, the metric matrix is updated by successive projections onto the positive semi-definite cone and the threshold parameter is updated by a hinge loss function simultaneously. Li *et al*. [4] incorporated an online metric learning method into a tracking algorithm to solve the optimization problem for obtaining the linear representation. All above online metric learning models are inspired by the passive-aggressive algorithm mechanism used in [15]. At each online step, a tuple is received and the metric matrix will be updated when the constraint is violated. Other passive-aggressive based algorithms are presented in [2], [16]. Furthermore, Davis *et al*. [11] presented an online version of information-theoretic metric learning (ITML) approach using LogDet divergence as the regularizer. LEGO [12] is an improved version of ITML-ONLINE [11]. In LEGO [12], Jain *et al*. developed an online metric learning algorithm that uses LogDet regularization and exact gradient descent. All the methods mentioned above tried to learn a measure of similarity between pairs of instances which can be incorporated into our framework.

III. Sketch Metric Learning

In this section, we begin with describing the generation of pseudo center. Next, we demonstrate how to learn the metric with the new kind of constraint in an online manner. Finally, we introduce more implementations of sketch metric learning.

A. Sketch the Pseudo Center

We start with the basic idea of Matrix Sketching in [14]. The algorithm receives m rows of a large matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$ one after the other, in a streaming fashion. It maintains a sketch matrix $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ containing only $\ell \ll m$ rows but still guarantees that $\mathbf{A}^T\mathbf{A} \approx \mathbf{B}^T\mathbf{B}$. The detail is described in Algorithm 1.

Algorithm 1 Matrix Sketching

Input: $\mathbf{A} \in \mathbb{R}^{m \times d}$
$\mathbf{B} \leftarrow$ all zeros matrix $\in \mathbb{R}^{\ell \times d}$
for all i in $1...m$ **do**
  Insert $\mathbf{A}_i$ into a zero valued row of $\mathbf{B}$
  if $\mathbf{B}$ has no zero valued rows **then**
    $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] \leftarrow$SVD($\mathbf{B}$)
    $\delta \leftarrow \sigma^2_{\ell/2}$
    $\check{\mathbf{\Sigma}} \leftarrow \sqrt{\max(\mathbf{\Sigma}^2 - \mathbf{I}_\ell \delta, 0)}$
    $\mathbf{B} \leftarrow \check{\mathbf{\Sigma}}\mathbf{V}^T$
  end if
end for

Inspired by Matrix Sketching in [14], we sketch a matrix, which to represent the principal structures within each class, also denoted as $\mathbf{B} \in \mathbb{R}^{\ell \times d}$, where ℓ is the sketch length and d is the feature dimension. Under this setting, we only need to maintain the sketched matrix and update it whenever it is full. After the sketching procedure finished, we extract the first row of $\mathbf{B}$ as pseudo center $\mathbf{p} \in \mathbb{R}^{1 \times d}$ in order to formulate tuples and $\mathbf{p}$ will be updated by the newest one if $\mathbf{p}$ already exists. The procedure is shown in Figure 1. Note that the **Sketch** operation in Figure 1 is based on Algorithm 1.

As shown in Figure 1, the generation takes place in a sequence manner. On each round the algorithm observes some input samples and stores them in the corresponding class matrix $\mathbf{B}$. Initially, the matrix $\mathbf{B}$ is empty with ℓ rows space. Once the matrix is full of input data, we sketch the class

information by compressing the data to half of their original size in order to free up space immediately so that the newly arrived subsequent samples can be added in. In the next step, we extract the pseudo-center $\mathbf{p}$ in order to formulate our tuples.

B. Learning the Metric

Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ denotes the training dataset with n samples $\mathbf{x}_i \in \mathbb{R}^d$, each associated with a discrete (but not necessarily binary) class label $y_i \in \{1, 2, ..., C\}$. We also denote pseudo centers as $\{(\mathbf{p}_j, y_j)\}_{j=1}^C$ with $y_j \in \{1, 2, ..., C\}$. Throughout the sketching stage, each class space has its own sketched matrix, denoted by $\{\mathbf{B}_i\}_{i=1}^C \in \mathbb{R}^{\ell \times d}$. Based on the pseudo centers, given a new input sample, we construct a set of tuples for pairs $(\mathbf{x}_i, \mathbf{p}_j)$. Among all these pairs, let $\mathcal{S}$ indexes the similar pairs and $\mathcal{D}$ indexes the dissimilar pairs. For instance, $(\mathbf{x}_i, \mathbf{p}_j)$ is a similar pair if y_i is equal to y_j, otherwise, $(\mathbf{x}_i, \mathbf{p}_j)$ is a dissimilar pair. The space of symmetric d times d matrices is denoted by $\mathbb{S}^d$ and the cone of positive semi-definite matrices is denoted by $\mathbb{S}_+^d$.

Our goal is to learn a distance function $D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{p}_j) = (\mathbf{x}_i - \mathbf{p}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{p}_j)$ parameterized by $\mathbf{M} \in \mathbb{S}_+^d$ that minimizes the distance of similar pairs and maximizes the distance of dissimilar pairs. To achieve this purpose, we need to predict if the two samples of tuple have the same class label or not. After the prediction, the algorithm may suffer from an instantaneous loss which reflects how much the prediction fails. At the end of each step, some parameters should be updated to improve its prediction rule for the following steps.

Many state-of-the-art learning algorithms can be incorporated into the above framework. In the following, we first demonstrate how to implement a well-known pseudo-metric online learning algorithm (POLA) [10] in our framework.

The main idea of POLA [10] is an update rule which is based on successive projections onto the positive semi-definite cone and onto the half-space constraint. Intuitively, they look for a new pseudo-metric and threshold that, on the one hand, will predict correctly the last sample they have just received and, on the other hand, will be as close as possible to the previous pseudo-metric and threshold. The constraint inherited by POLA [10] can be presented as follows,

$$\mathbf{y}(b - D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{p}_j)) \geq 1, \tag{1}$$

where $b \in \mathbb{R}$ is a bias term, and $\mathbf{y} = 1$ if $y_i = y_j$ and -1 otherwise.

To learn a distance function that satisfies the constraint in Eq. (1), we define a global loss $\mathcal{L}_{\mathbf{M},b}$ that accumulates hinge losses over all possible tuples: $\mathcal{L}_{\mathbf{M},b} = \sum_{(i,j)\in\mathcal{D}\cup\mathcal{S}} l_{\mathbf{M},b}(\mathbf{x}_i, \mathbf{p}_j)$, with the loss for a single tuple being,

$$l_{\mathbf{M},b}(\mathbf{x}_i, \mathbf{p}_j) = \max\{0, \mathbf{y}(D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{p}_j) - b) + 1\}, \tag{2}$$

In the online setting, we first observe a tuple $(\mathbf{x}_i, \mathbf{p}_j)$ in a sequential manner and then calculate $D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{p}_j)$. If $D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{p}_j)$ is greater than the learned bias, the pair is recognized as dissimilar one. Otherwise, we say that the pair is similar. After the prediction, we receive the true label $\mathbf{y}$ and may suffer a (hinge) loss in Eq. (2) if there is a discrepancy between our prediction and $\mathbf{y}$. The goal of the online algorithm is to minimize the cumulative loss. So we iteratively update the matrix $\mathbf{M}$ and the threshold b. The update rule is composed of two successive projections.

First, in step $\boldsymbol{\tau}$, we project $(\mathbf{M}_{\boldsymbol{\tau}}, b_{\boldsymbol{\tau}})$ onto $(\mathbf{M}_{\hat{\boldsymbol{\tau}}}, b_{\hat{\boldsymbol{\tau}}})$ to satisfy $l_{\mathbf{M}_{\hat{\boldsymbol{\tau}}}, b_{\hat{\boldsymbol{\tau}}}}(\mathbf{x}_i, \mathbf{p}_j) = 0$ by,

$$\mathbf{M}_{\hat{\boldsymbol{\tau}}} = \mathbf{M}_{\boldsymbol{\tau}} - \mathbf{y}\alpha_{\boldsymbol{\tau}}\mathbb{Z}_{\boldsymbol{\tau}}, \quad b_{\hat{\boldsymbol{\tau}}} = b_{\boldsymbol{\tau}} + \mathbf{y}\alpha_{\boldsymbol{\tau}}, \tag{3}$$

where $\alpha_{\boldsymbol{\tau}} = l_{\mathbf{M}_{\boldsymbol{\tau}}, b_{\boldsymbol{\tau}}}(\mathbf{x}_i, \mathbf{p}_j)/(\|\mathbf{x}_i - \mathbf{p}_j\|_2^4 + 1)$ and $\mathbb{Z}_{\boldsymbol{\tau}} = (\mathbf{x}_i - \mathbf{p}_j)(\mathbf{x}_i - \mathbf{p}_j)^T$.

Second, we project $(\mathbf{M}_{\hat{\boldsymbol{\tau}}}, b_{\hat{\boldsymbol{\tau}}})$ onto $(\mathbf{M}_{\boldsymbol{\tau}+1}, b_{\boldsymbol{\tau}+1})$ to satisfy $\mathbf{M}_{\boldsymbol{\tau}+1} \succeq 0$, $b_{\boldsymbol{\tau}+1} \geq 1$ by,

$$\mathbf{M}_{\boldsymbol{\tau}+1} = \mathbf{M}_{\hat{\boldsymbol{\tau}}} - \lambda_d \mathbf{u}_d \mathbf{u}_d^T, \quad b_{\boldsymbol{\tau}+1} = \max\{1, b_{\hat{\boldsymbol{\tau}}}\}, \tag{4}$$

where λ_d is the minimal eigenvalue of $\mathbf{M}_{\hat{\boldsymbol{\tau}}}$ and $\mathbf{u}_d$ is the corresponding eigenvector. These two parameters can be calculated efficiently by using the Lanczos method [17]. More analysis about loss bounds can be found in Sec. 4 of [10].

We summarize the procedure of SML in Algorithm 2 and call it as **SML-POLA**. In Algorithm 2, we receive an input sample along with its class label for sketching and then learn a distance metric one by one in a steaming manner. We utilize

Algorithm 2 Sketch metric learning with POLA

Input: $xTr \in \mathbb{R}^{n\times d}$, $yTr \in \mathbb{R}^{n\times 1}$, $\ell \in \mathbb{R}$
$\mathbf{M}_0 \leftarrow$ all zeros matrix $\in \mathbb{R}^{d\times d}$
$\mathbf{B}_{1,...,C} \leftarrow$ all C zeros matrices $\in \mathbb{R}^{\ell\times d}$
for $i = 1 : n$ **do**
  $y_i \leftarrow yTr_i$, $x_i \leftarrow xTr_i$
  add $\mathbf{x}_i^T$ to one all-zero valued row of $\mathbf{B}_{y_i}$
  if $\mathbf{B}_{y_i}$ is full **then**
    $(\mathbf{p}_{y_i}, \mathbf{B}_{y_i}) = sketch(\mathbf{B}_{y_i}, \ell)$
  end if
  for $j = 1 : C$ **do**
    Get true $\mathbf{y}$ and calculate $l_{\mathbf{M}_{\boldsymbol{\tau}}}(\mathbf{x}_i, \mathbf{p}_j)$
    if $l_{\mathbf{M}_{\boldsymbol{\tau}}}(\mathbf{x}_i, \mathbf{p}_j) > 0$ **then**
      $\alpha_{\boldsymbol{\tau}} = l_{\mathbf{M}_{\boldsymbol{\tau}}, b_{\boldsymbol{\tau}}}(\mathbf{x}_i, \mathbf{p}_j)/(\|\mathbf{x}_i - \mathbf{p}_j\|_2^4 + 1)$
      $\mathbf{M}_{\hat{\boldsymbol{\tau}}} = \mathbf{M}_{\boldsymbol{\tau}} - \mathbf{y}\alpha_{\boldsymbol{\tau}}\mathbb{Z}_{\boldsymbol{\tau}}$, $b_{\hat{\boldsymbol{\tau}}} = b_{\boldsymbol{\tau}} + \mathbf{y}\alpha_{\boldsymbol{\tau}}$
      if $(\mathbf{y} == 1)$ **then**
        $b_{\boldsymbol{\tau}+1} = b_{\hat{\boldsymbol{\tau}}}$
        calculate $(\lambda_d, \mathbf{u}_d)$
        if $(\lambda_d < 0)$ **then**
          $\mathbf{M}_{\boldsymbol{\tau}+1} = \mathbf{M}_{\hat{\boldsymbol{\tau}}}$ -$\lambda_d \mathbf{u}_d \mathbf{u}_d^T$
        else
          $\mathbf{M}_{\boldsymbol{\tau}+1} = \mathbf{M}_{\hat{\boldsymbol{\tau}}}$
        end if
      else
        $\mathbf{M}_{\boldsymbol{\tau}+1} = \mathbf{M}_{\hat{\boldsymbol{\tau}}}, b_{\boldsymbol{\tau}+1} = \max\{1, b_{\hat{\boldsymbol{\tau}}}\}$
      end if
    else
      $\mathbf{M}_{\boldsymbol{\tau}+1} = \mathbf{M}_{\boldsymbol{\tau}}, b_{\boldsymbol{\tau}+1} = b_{\boldsymbol{\tau}}$
    end if
  end for
end for

Data	SML-POLA	POLA	SML-ITML-OL	ITML-Online	SML-LEGO	LEGO	SML-Tri	Tri	Euc
Climate	**9.37±0.52**	9.94±0.73	**11.24±0.73**	12.06±0.63	**10.44±0.69**	11.65±0.77	8.72±0.82	**8.28±0.70**	12.35±0.69
Diabetes	**38.93±0.75**	39.45±1.40	**39.71±1.24**	39.75±1.10	**39.50±1.51**	39.57±1.08	40.09±1.28	**39.87±1.51**	39.63±0.98
Iris	**2.27±0.34**	2.40±0.34	**3.53±0.55**	3.53±0.55	**3.47±0.53**	3.47±0.53	**3.13±0.63**	3.13±0.77	3.53±0.55
Waveform	**20.87±0.37**	25.89±0.56	**22.79±0.24**	25.44±0.35	**20.95±0.52**	25.29±0.43	**19.60±0.36**	21.56±0.61	23.30±0.17
Waveform-Noise	**20.32±0.21**	26.81±0.84	**23.42±0.28**	26.98±0.47	**20.40±0.29**	27.09±0.60	**20.76±0.39**	22.11±0.29	25.71±0.30
MNIST	3.10±0.08	**2.82±0.13**	**2.69±0.11**	2.71 ±0.11	**2.89±0.14**	3.02± 0.15	3.49± 0.09	**3.23±0.12**	2.79±0.11
ImageNet	**51.66±0.32**	53.49± 0.34	**57.33±0.31**	57.73± 0.41	**52.20±0.35**	55.28± 0.35	**51.84±0.61**	52.89± 0.22	59.04± 0.24

TABLE I: Average testing error (%). The better performance is denoted in bold type.

dataset	samples	dimensions	classes
Iris	150	4	3
Climate	540	18	2
Diabetes	768	8	2
Waveform	5000	21	3
Waveform-Noise	5000	40	3
MNIST	70000	400	10
ImageNet	1261406	1000	1000

TABLE II: Description of datasets, we note that only 10 classes of data was selected from ImageNet dataset due to the time-consuming methods such as ITML-Online [11].

the first few samples to generate the first pseudo center so that the learning stage can be started. We note that the sketching stage would be activated when the input samples fulfill the sketched matrix, while the learning stage is activated every time a new sample is arriving.

C. More Implementations of Sketch Metric Learning

Except POLA [10], other state-of-the-art algorithms such as ITML-Online [11], LEGO [12] and triple tuple based algorithm [4] can also be implemented in our sketch learning framework seamlessly. We respectively denote them as **SML-ITML-OL**, **SML-LEGO** and **SML-TRI**.

To our best knowledge, SML-ITML-OL and SML-LEGO have the same type of tuple with SML-POLA while SML-TRI has a triple type. By incorporating TRI into our framework, we denote each triple tuple in the constraint by $(\mathbf{x}_i, \mathbf{p}_j, \mathbf{p}_k)$ where $y_i = y_j$ and $y_i \neq y_k$. Except the definition of a new kind of constraint, the rest part of metric learning is remained unchanged and is not reviewed here due to the page limitation.

IV. Experiment

We compare the proposed methods SML-POLA, SML-ITML-OL, SML-LEGO, SML-Tri with the related competitors: POLA [10], ITML-Online [11], LEGO [12], Tri [4] and a baseline algorithm that uses the standard Euclidean distance denoted by **Euc**. In SML-POLA and POLA [10], we set the threshold $b = 1.5$. In SML-ITML-OL, ITML-Online [11], SML-LEGO and LEGO [12], we set η to be 1/8 and we also generated the target distances as in [11]. For same-class pairs, the target distance was set to be equal to the 5th percentile of all distances in the data, while for different-class pairs, the 95th percentile was used. The trade-off parameter was set to 0.5 in SML-Tri and Tri [4]. In the sketching stage, we set the sketch length $\ell = \min\{200, d/2\}$ where d is the dimension of the input data. The number of tuples was set to be Cn.

The experiments were split into two main parts: Classification and Object Tracking. In Classification, we evaluated metric learning for k-NN classification error rates via five-fold cross validation with k = 1. In Object Tracking, positive samples were sampled from a region with a radius of 4 pixels centered at the target of current frame. Negative samples were sampled from a region with an inner radius of 8 pixels and an outer radius of 30 pixels centered at the target. And candidate samples were sampled from a region with a shifting step of 4 pixels and a radius of 20 pixels centered at the target of last frame. All the parameters remained unchanged during the experiments.

A. Classification on Benchmark Datasets

First, we evaluated the compatibility and effectiveness of our framework on seven benchmark datasets, that is, 1) Iris, 2) Climate, 3) Diabetes, 4) Waveform, 5) Waveform-Noise, 6) MNIST, and 7) ImageNet. The statistics of datasets summarized in Table II and Table I summarizes the main results. We note that our experiment goal is to verify the improvement after incorporating matrix sketching in metric learning. The results show that our methods almost outperform the competitors on Iris, Waveform, Waveform-Noise and ImageNet dataset. While on the Climate, Diabetes and MNIST dataset, we have a nearly equal result with the them.

Reproducing the experiment used to test POLA in [10] and LEGO in [12], we posed a binary classification problem between each pair of handwritten digits in MNIST dataset (45 problems in all). Figure 2 shows the test errors between the four pairwise competitors, our algorithms match and even outperform the competitors on 29/45, 37/45, 33/45 and 22/45 respectively. We note that our algorithms outperform the related competitors except SML-Tri (nearly equals to). Based on the additional baselines provided in [10], [12], we show that our approach also fares well compared to other metric learners on this dataset.

B. Object Tracking

Many studies show that using distance metric learning can improve the discriminative capability in real-time object tracking [1], [2], [4]. In this subsection, we will demonstrate how the distance metric learning can further improve the tracking performance after being deployed into our sketch learning framework while comparing with the related competitors. The experiment was taken on twelve public challenging video sequences (shown in Figure 3), while only gray scale information was used for our experiment. Our methods

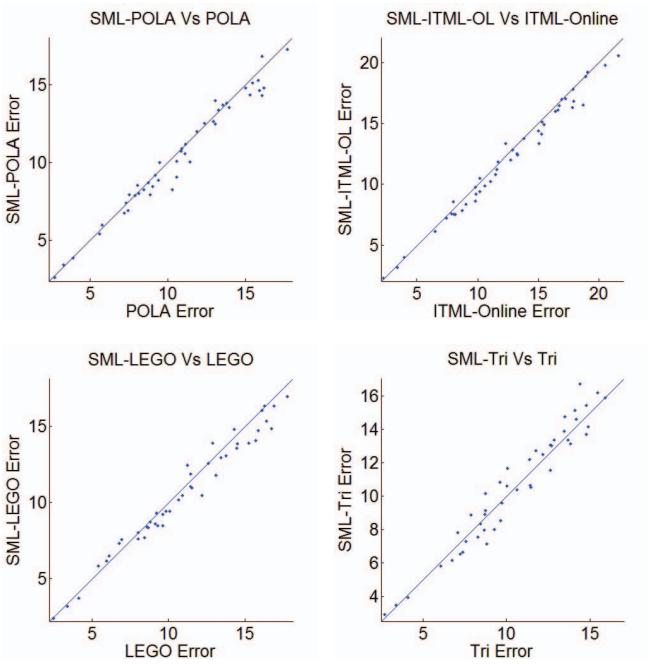

Fig. 2: Comparison of error for the four pairwise competitors on 45 binary classification problems using the MNIST dataset.

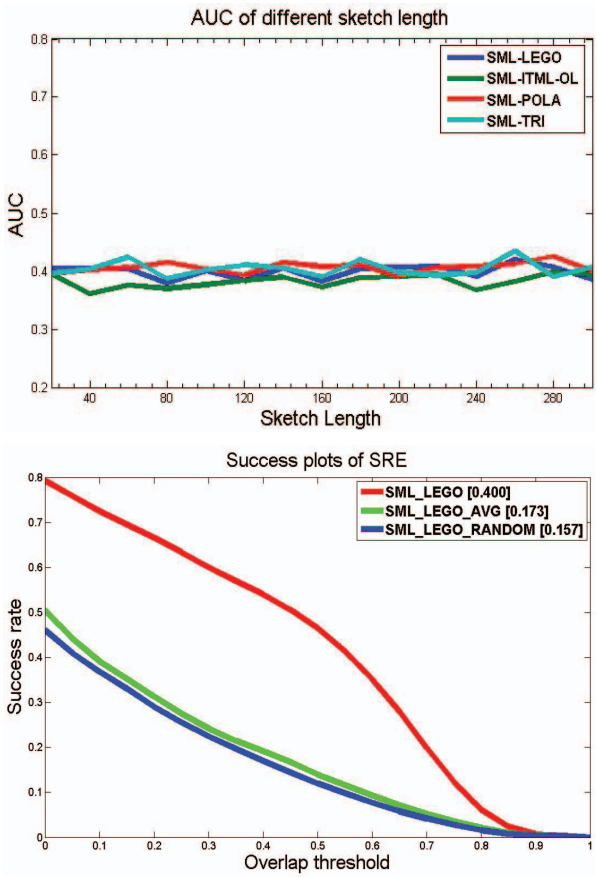

Fig. 4: Top: Performance with different sketch lengths. Down: Effectiveness in different pseudo centers.

were also compared against seven state-of-the-art tracking algorithms denoted as Struck (structured output tracking with kernels [18]), OAB (online AdaBoost [19]), SemiT (semi-supervised tracking [20]), BSBT (beyond semi-supervised tracking [21]), MIL (multiple instance learning [22]), STC (spatio-temporal context [23]) and FCT (fast compressive tracking [24]). For fair comparison, we directly used the twelve video sequences accompany with the ground truth files and the tracking result of the five trackers (Struck [18], OAB [19], SemiT [20], BSBT [21] and MIL [22]) to implement the performance evaluation, which are all available at $http : //www.visual-tracking.net$. The experiment is conducted in MATLAB R2012a with a 3.30 GHZ Intel Core i3-3220 CPU and 12 GB memory. Our proposed algorithms, running 40 frames per second (FPS) on average, can achieve a real-time tracking.

1) Evaluation Criterion: We evaluated the performance of tracking algorithms in two aspects [25]: precision plot and success plot. Precision plot is used to measure the percentage of frames whose center location error is within the given threshold distance. And the center location error, is defined as the average Euclidean distance between the center locations of the tracked targets and the manually labeled ground truth. The threshold is set to be 20. In addition, the success plot shows the ratios of successful frames at the thresholds varied from 0 to 1. The so called successful frame is the one that its overlap score is greater than a specific threshold such as 0.5. The overlap score is defined as $score = \frac{area(ROI_T \bigcap ROI_{GT})}{area(ROI_T \bigcup ROI_{GT})}$, where ROI_T and ROI_{GT} are target region and ground truth region respectively. We used the area under curve (AUC) of each success plot to rank the tracking algorithms. Furthermore, to further improve the robustness of the evaluation, we used three ways for the initialization of the first frame: one-pass evaluation (OPE), temporal robustness evaluation (TRE) and spatial robustness evaluation (SRE). We refer the readers to [25] for more details.

Performance with different sketch lengths. We evaluated the tracking performance by computing the AUC score of the four proposed algorithms with different sketch lengths from 20 to 300 with a interval of 20. The result is given at the top graph of Figure 4. With the increase sketch length of the sketched matrix, the AUC score had no obvious correlation with it. This is because the matrix sketching algorithm in [14] is based on Singular Value Decomposition (SVD). In the other words, the singular vectors, corresponding to several highest singular values, are retained during sketching. So that in our framework, only a small matrix is enough to sketch the principal structure of all the input samples.

Effectiveness in different pseudo centers. In order to verify the effectiveness of our sketched mode to generate pseudo center, we compared ours with two other different generation

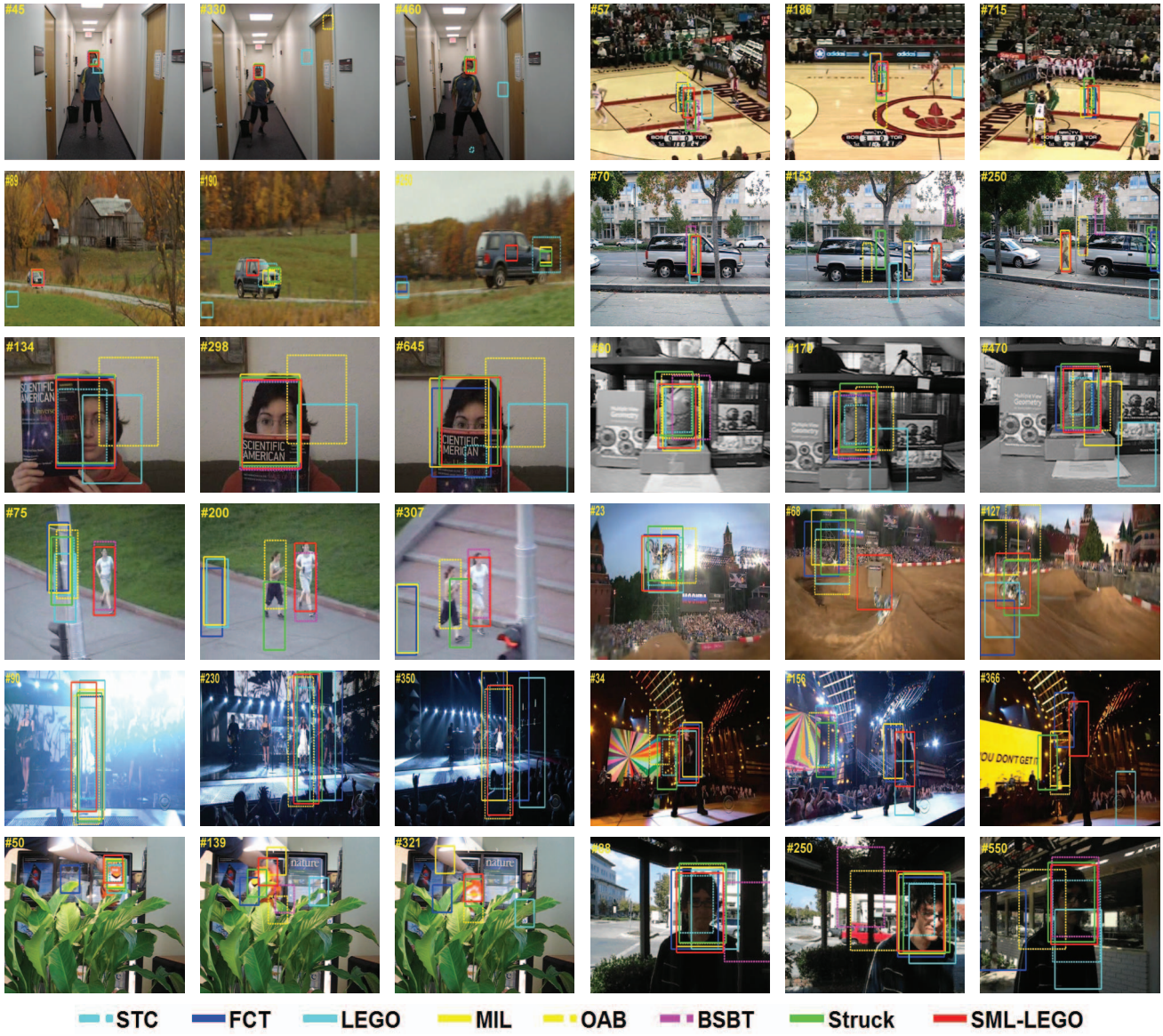

Fig. 3: Tracking results of selected algorithms in representative frames of twelve challenging sequences (from left to right and top to down are Boy, Basketball, CarScale, David-3, Faceocc-1, Fish, Jogging-2, MotorRolling, Singer-1, Singer-2, Tiger-1, and Trellis, respectively).

modes: including 1) selecting one sample in the training set as a pseudo center randomly, and 2) calculating the average of the input samples as the pseudo center. The result is shown in the second graph of Figure 4. Our proposed sketched pseudo center, which is corresponding to the highest singular value of the singular vector, performs the best. And the random and average methods, however, failed to capture the main information and ultimately led to a poor result.

Performance with and without sketching. To demonstrate the improvement by incorporating matrix sketching into metric learning, we compared our sketch based algorithms with four related competitors. And the EUC algorithm was considered as baseline. Figure 5 shows the result that all the online metric learning methods surpass the baseline algorithm except LEGO [12] in the plots of OPE. And the performance score of the methods without sketching is only about one-third of ours on average. As we defined above, our learning framework is based on a new kind of constraint that all the input samples are learned to update a metric that minimizes the distance between the samples and pseudo center with similar class and maximizes the distance between the two with dissimilar classes. However, in the typical metric learning methods, the tuples are just consisted of input samples and the learning stage is proceeded on a random set of tuples to avoid overload computing, which will finally incur the lack of robustness.

In addition, it is known that visual trackers are sensitive to initialization. From the success plots of Figure 5, we found that the experiment results of SRE in seven of the 9 trackers are worse than that of OPE. Especially the SML-POLA tracker, the performance score fell to 0.357 at 29% and the ranking dropped from 1 to 4, while the performance score of LEGO [12] tracker rose to 0.129 at 14% and finally the ranking went up from the last place to 6. It seems that LEGO tracker [12] is the most robustness tracker to initialization.

Comparison with state-of-the-art trackers. The overall performance for all the seven state-of-the-art trackers and ours is shown in Figure 6. Except the success plots of TRE, our algorithms perform the best. In the success plots of TRE, the AUC score of Struck [18] outperforms ours. This is because the Struck tracker [18] is in favor of short sequences. However, it is easy to drift away after a portion of frames passed. That is why we surpass it in the plots of OPE and SRE.

In [25], the sequences are annotated with 11 different attributes, including: illumination variation, scale variation, occlusion, deformation, motion blur, fast motion, in-plane rotation, out-of-plane rotation, out-of-view, background clutters and low resolution. We performed a comparison with seven trackers on all the sequences annotated with these 11 attributes. The result shows that our algorithms perform favorably on most of these attributes. Especially for occlusion, deformation, motion blur, illumination variation and out-of-plane rotation attributes, our algorithms outperform the Struck tracker [18]. One reason is that the sketched matrix can maintain the history samples structures so that a transient appearance change of the target could not incur drift easily. Figure 7 shows the precision plots of SRE on deformation, occlusion and out-of-plane rotation attributes.

Qualitative evaluation. We compared our SML-LEGO algorithm with other seven trackers (LEGO [12], Struck [18], OAB [19], BSBT [21], MIL [22], STC [23] and FCT [24]) on twelve challenging sequences in Figure 3. The LEGO [12] algorithm fails to track the target objects on all the twelve sequences due to the lack of robustness to handle the appearance variations. The Struck tracker [18] performs well on most of sequences except David-3, Jogging-2, and Singer-2 sequences due to the heavy occlusion and illumination variation. Overall, the proposed SML-LEGO algorithm performs well on all the twelve sequences. Especially, when heavy occlusion happens on David-3 and Jogging-2 sequences, only BSBT [21], STC [23] and ours are able to track the object. On the CarScale and Singer-2 sequences, only our method tracks the objects successfully in all the three frames.

V. Conclusion

We developed a sketch metric learning framework to learn a Mahalanobis distance metric based on matrix sketching. In this framework, we sketched the received training samples and generated a pseudo center to formulate a new kind of constraint for metric learning. Our framework is well suited for time-varying process such object tracking and it is also deterministic, simple to implement. The experiment results demonstrate the improvement by incorporating matrix sketching into metric learning and prove the efficiency and practicability of our method.

Acknowledgment

This work is supported by the Natural Science Foundation of China (under Grant 61472456) and Guangzhou Pearl River Science and Technology Rising Star Project (under Grant 2013J2200068).

References

[1] N. Jiang, W. Liu, and Y. Wu, "Learning adaptive metric for robust visual tracking," *Image Processing*, vol. 20, no. 8, pp. 2288–2300, 2011.
[2] G. Tsagkatakis and A. Savakis, "Online distance metric learning for object tracking," *IEEE TCSVT*, vol. 21, no. 12, pp. 1810–1821, 2011.
[3] J. Wang, H. Wang, and Y. Yan, "Robust visual tracking by metric learning with weighted histogram representations," *Neurocomputing*, vol. 153, pp. 77–88, 2015.
[4] X. Li, C. Shen, A. Dick, and Z. Zhang, "Online metric-weighted linear representations for robust visual tracking," *PAMI*, p. 1, 2015.
[5] L. Si, R. Jin, S. C. Hoi, and M. R. Lyu, "Collaborative image retrieval via regularized metric learning," *Multimedia Systems*, vol. 12, no. 1, pp. 34–44, 2006.
[6] E. P. Xing, M. I. Jordan, S. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *NIPS*, 2002.
[7] W. Liu, C. Mu, R. Ji, S. Ma, J. R. Smith, and S.-F. Chang, "Low-rank similarity metric learning in high dimensions," in *AAAI*, 2015.
[8] B. Kulis, "Metric learning: A survey," *Foundations and Trends in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2012.
[9] A. Bellet, A. Habrard, and M. Sebban, "Metric learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 9, no. 1, pp. 1–151, 2015.
[10] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng, "Online and batch learning of pseudo-metrics," in *ICML*, 2004.
[11] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *ICML*, 2007.
[12] P. Jain, B. Kulis, I. S. Dhillon, and K. Grauman, "Online metric learning and fast similarity search," in *NIPS*, 2009.
[13] R. Jin, S. Wang, and Y. Zhou, "Regularized distance metric learning: Theory and algorithm," in *NIPS*, 2009.
[14] E. Liberty, "Simple and deterministic matrix sketching," in *SIGKDD*, 2013.
[15] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *TJMLR*, vol. 7, pp. 551–585, 2006.
[16] G. Chechik, U. Shalit, V. Sharma, and S. Bengio, "An online algorithm for large scale image similarity learning," in *NIPS*, 2009.
[17] G. H. Golub and C. F. Van Loan, *Matrix computations*, 2012, vol. 3.
[18] S. Hare, A. Saffari, and P. H. Torr, "Struck: Structured output tracking with kernels," in *ICCV*, 2011, pp. 263–270.
[19] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting." in *BMVC*, vol. 1, no. 5, 2006, p. 6.
[20] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," in *ECCV*, 2008, pp. 234–247.
[21] S. Stalder, H. Grabner, and L. Van Gool, "Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition," in *ICCV*, 2009, pp. 1409–1416.
[22] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *PAMI*, vol. 33, no. 8, pp. 1619–1632, 2011.
[23] K. Zhang, L. Zhang, M.-H. Yang, and D. Zhang, "Fast tracking via spatio-temporal context learning," *arXiv*, 2013.
[24] K. Zhang, L. Zhang, and M. H. Yang, "Fast compressive tracking," *PAMI*, vol. 36, no. 10, pp. 2002–2015, 2014.
[25] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: A benchmark," in *CVPR*, 2013, pp. 2411–2418.

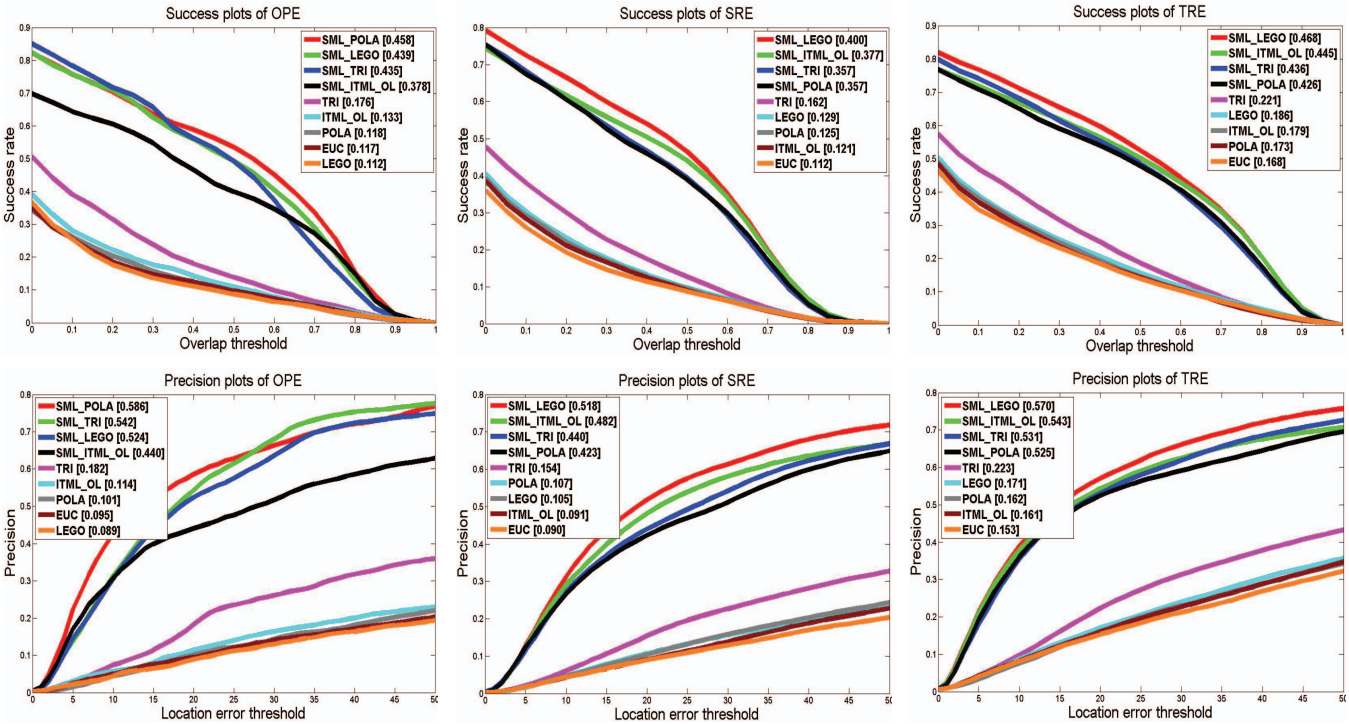

Fig. 5: Performance (OPE, SRE and TRE) of object tracking with and without sketching.

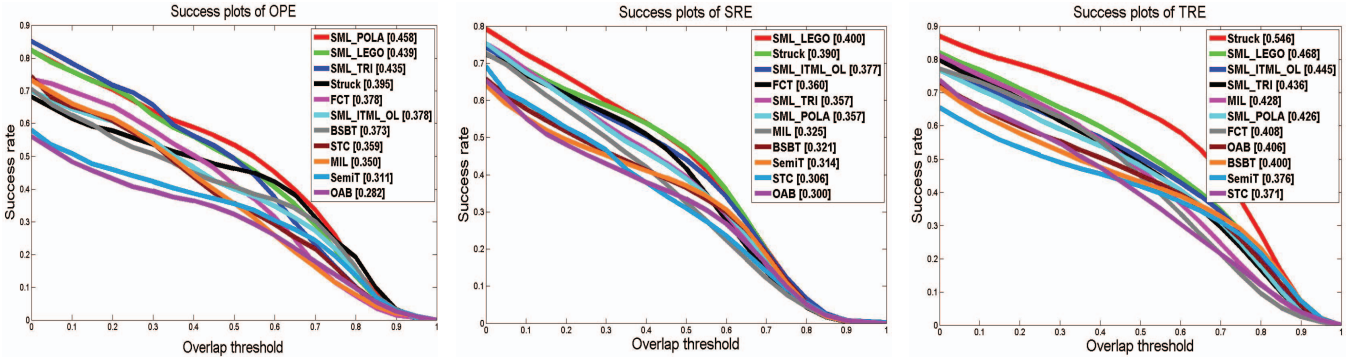

Fig. 6: Success plots of OPE, SRE and TRE of eleven trackers.

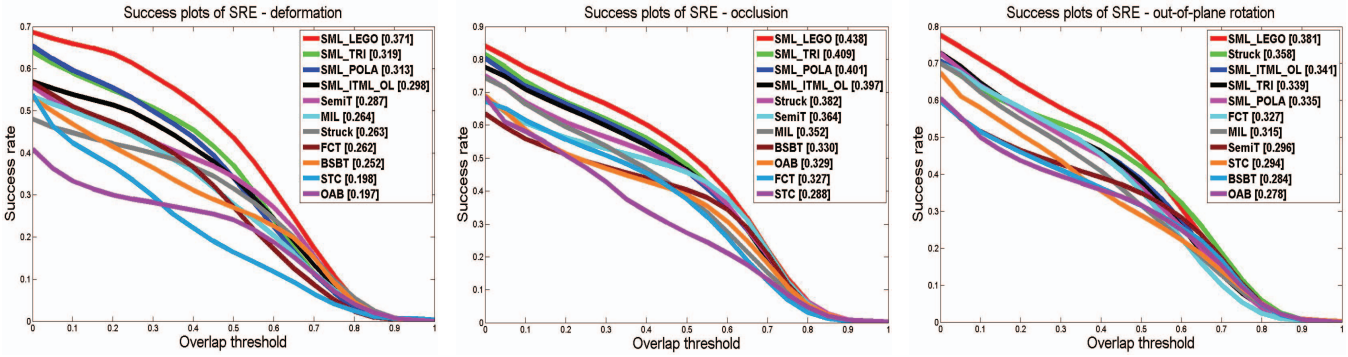

Fig. 7: Success plots of three attributes: deformation, occlusion and out-of-plane rotation.