Supervised Nearest Neighbor Searching for Visual Tracking

Ji Zhang, Jie Sheng, Ankur Teredesai

Computer Science and Systems
Institute of Technology
University of Washington Tacoma

Abstract. Supervised learning algorithms have been widely applied in tracking-by-detection based methods for object tracking in recent years. Most of these approaches treat tracking as a classification problem and solve it by training a discriminative classifier and exhaustively evaluating every possible target position; problems thus exist for two reasons. First, since the classifier describes the common feature of samples in an implicit way, it is not clear how well the classifier can represent the feature of the desired object against others; second, the brute-force search within the output space is usually time consuming, and thus limits the competence for real-time application. In this project, we treat object tracking as a problem of information retrieval for streaming data. We propose to apply unsupervised learning by Locality Sensitive Hashing (LSH) and use Nearest Neighbor Searching (NNS) as the main engine for target detection. In addition, our method applies a Support Vector Machine (SVM) based supervised classifier cooperating with the unsupervised detector. Both the proposed tracker and several selected trackers are tested on some well accepted challenging videos; and the experimental results demonstrate that the proposed tracker outperforms the selected other trackers in terms of the effectiveness as well as the robustness.

Keywords: Tracking-by-detection, Unsupervised Learning, Nearest Neighbor Searching, Locality Sensitive Hashing, Support Vector Machine

1 Introduction

Object tracking in unconstrained scenarios is one of the fundamental problems of Computer Vision. The challenge in this topic lies mainly in the complexity of tracking environments such as severe illumination change, target deformation, background clutter, partial occlusion, to name just a few. Inspired by the tracking-by-detection framework [1], which has been proposed to treat object tracking as a detection problem and solve it by supervised learning and prediction, a major research axis has been focused on building online classifiers to describe the dynamic target with discriminative power against the background. Among this group, Garbner at al. [2] [3] and Stalder et al. [4] applied boosting algorithm for the classification task. Babenko [5] extended the idea by using

a more sophisticated sampling strategy to alleviate ambiguity among training data. Meanwhile, SVM has been deeply exploited; for example, the method proposed in [6] exploits an online-SVM algorithm called Larank [7] [8] and solves the curse of kernelization by maintaining a fixed-sized set of support vectors [9] [10]. Aiming at improved performance of that idea, several SVM-based trackers [11] [12] [13] has been proposed in three aspects respectively: increasing the sensitivity to new samples, strengthening the adaptive ability on target deformation, and fusing multiple features by binary code to build a better target model.

In spite of demonstrated success in previous research, classification-based trackers often suffer drifting problems where a tracker confidently tracks a position that covers the true target location in part and can hardly correct itself once it starts to make such mistake. The main contribution of this research project is that we propose a novel tracking algorithm based on unsupervised learning and similarity search. Using a new framework for incorporating unsupervised learning with supervised learning, our algorithm is carefully designed to prevent drifting problems and to improve the overall accuracy. Later we will show the effectiveness of our method by experimental results; here, we want to highlight the advantages of our algorithm over supervised approaches in the following two aspects.

First, treating tracking problem as a classification task is not straightforward and has inevitable defects. For most supervised algorithms [2] [5] [6] [14] [15], the basic idea is to train a function $h : X \to R$, where X is feature space and h returns a classification confidence score evaluating how likely a sample would be the target. The function h acts as a summary of target appearance so far from the beginning, and is updated online by continuously adding more training samples frame by frame. This is where severe drifting problems may occur because training samples at different time are equally treated in those algorithms, i.e., old samples have the same influence on h as new samples do, while the new ones in fact should have more influence in order to keep h up-to-date. Previous research has been done to alleviate this problem by weighting more on the new data [11]. However, underrating old data may lead to incomplete view of the target and the fragility to noise. The dilemma is due to the inherent flaw of supervised methods, that is, trying to train an implicit function describing common feature of all the training samples at the cost of each samples uniqueness. According to that implicit function, however, it is difficult to evaluate how well the trained function can represent the varying target and how confident it is to track.

An intuitive understanding of tracking by human vision may help us come up with better ideas. After confirming a target, what a human being does is to keep memorizing new appearance of the target while detecting it according to the memory. This mechanism can be well exported to computer vision system by applying unsupervised learning and nearest neighbor searching based on it. Specifically, the process of unsupervised learning corresponds to the human keeping memorizing target, and Nearest Neighbor Searching (NNS) acts like human eye detection based on the learned data. Our approach uses this mechanism

and applies Locality Sensitive Hashing (LSH) as the unsupervised structure for nearest neighbor searching. We claim that our novel tracking framework overcomes the main flaw of supervised methods. On the one hand, thanks to the large capacity provided by hash tables, all possible appearance of the target can be learned, and this ensures complete knowledge of all possible appearance of the target; on the other hand, the uniqueness of a sample is well preserved because each sample is individually stored without information loss, and NNS is able to output a clear path to find the target based on this complete set of samples rather than a vague summary of them.

Second, there is a lot of redundancy in the detection steps of supervised methods. Suppose $X \in R^d$ is the feature space where d is the feature dimension, and Y is the search space with all possible target positions, then the new target position is estimated by the following function

$$y = \underset{y \in Y}{\operatorname{argmax}}\, h(x_t^{p_{t-1} \circ y}) \tag{1}$$

where $p_{t-1} \in P$ is the target position on $t-1$ with $t = 1, 2, ..., T$ being the discrete time instants, $y_t \in Y$ is the transformation such that the new position is approximated by the composition defined as $p_t = p_{t-1} \circ y_t$, and $x_t^p \in X$ is the feature of a sample at position p at time t. It is usually in a brute force manner to find y_t and is usually time costly but inevitable due to the fact that supervised classification or regression merely predicts a value rather than a vector or a structure that explicitly represents the desired transformation. Therefore, every potential sample needs to be enumerated to find the optimal one. In contrast, NNS returns the specific sample that is the most similar to the query. That means, the output of our core engine is not a value but a datum in the same format with the query. If we record each datum's relative location to the target, then after NNS, we can consider the NN's relative position as the query's and easily get the target position by its inverse. Theoretically, the position estimation function in our approach becomes

$$y_t = g(x_t^{p_{t-1}}) \tag{2}$$

where g retrieves the nearest neighbor of $x_t^{p_{t-1}}$ and returns the associated attribute, without checking every possible transformation in the output space. Therefore, brute-force searching in the output space is avoided and time complexity is reduced from $O(n)$ to $O(1)$ where N is the amount of candidates in the output space Y.

The remaining of this report is organized as follows. In Section 2 and Section 3 we will discuss the learning and detection steps of our algorithm, respectively. In section 4, the proposed cooperation strategy between LSH and SVM will be described in details. Section 5 presents the implementation procedures of our approach and two algorithms are provided; one for the learning step and the other for the detection step of the proposed approach. Experiments to examine the difference between our proposed tracker and 10 other selected trackers are

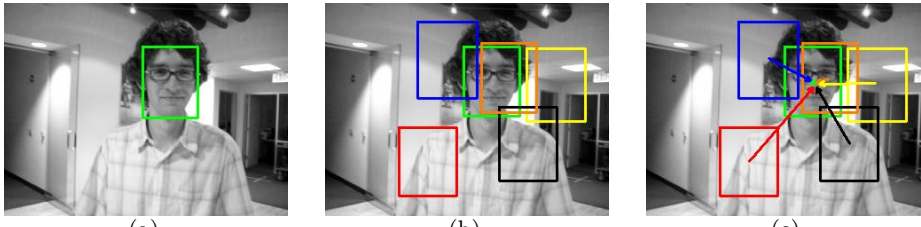
conducted in Section 6. Both the quantitative and qualitative performance evaluations will be discussed, according to experimental results. Effectiveness and robustness of our tracker will be demonstrated. Section 7 concludes the report.

2 Technical Details

2.1 Learning

Benefiting from the key-value indexing architecture of LSH tables, the value domain can be filled with a structure that directly represents the transformation. Specifically, as shown in Figure 1, in frame $f_t \in F$, we take a fixed number of samples within a preset radius centered at the target position $p = p_t \circ y$ by transformation $y \in Y$. We extract the sample feature x_t^p and generate a key-value pair $< x_t^p, y^{-1} >$ where y^{-1} is the inverse of y such that $p \circ y^{-1} = p_t \circ y \circ y^{-1} = p_t$, and insert it into the hash tables. By doing this, the tracker learns how each sample can be transformed back to the center, and this is the foundation of our detection step, to be described later.

Without loss of generality, we only consider targets 2D translation to simplify the problem, and therefore the transformation between two boxes can be represented as a vector $y \in R^n$ indicating 2D displacement from one to another. Extension to more complex transformation is straightforward and can be realized by simply substitute y with a more sophisticated structure that describes the enriched transformation model.



(a) (b) (c)

Fig. 1. Illustration of our learning strategy: (a) a random frame from a test video; (b) a certain number of boxes are generated around the current target box (in green) for sampling; (c) for each sample (in colors other than green), a transformation structure is associated with it. Since only translation is considered, the transformation structure here is a 2D vector indicating the offset from the sample to the target

2.2 Detection

Figure 2 illustrates the detection step with an example in two consecutive frames. As mentioned earlier, the position estimation function g in Equation (1)

searches for a querys nearest neighbor by its key and returns the corresponding value. Given a previous target position p_{t-1} at time t, the task of detection is to estimate a transformation y_t according to Equation (1) such that the new target position is achieved by the composition $p_t = p_{t-1} \circ y_t$. However, a simple calculation by Equation (1) is not sufficient to get accurate results, since nearest neighbor searching is an approximation method with random errors and they may accumulate along the tracking. We propose two ways to minimize the potential error. The first is to do the position estimation in a gradient-descent-like way. The second is to cooperate with supervised classification. We describe the former in this section, and details about the latter will be discussed in the next section.

We refer to the mechanism of gradient descending and propose to do NNS in an iterative manner. In frame f_t, we apply Equation (1) for at most M rounds rather than only once. On each round $m = 1, 2, ..., M$, we obtain a new estimated target position by $p_t^m = p_t^{m-1} \circ y^m$, where we define $p_t^0 = p_{t-1}$ as the target position in f_{t-1}, and $y^m = g(X_t^{p_t^{m-1}})$ means the transformation y^m in round M is obtained according to the output of round $m-1$. The iteration stops either when it reaches M rounds or when the transformation y^m is small enough. Since only 2D translation is considered in this project, we define that y^m is sufficiently small when $\|y^m\|_2$ decreases to less than a preset threshold δ. Ideally, this condition should be as strict as $\|y^m\|_2$ being exactly zero which corresponds to the exact expected target position. However, since our calculation is discrete and based on feature similarity rather than real gradient, convergence on $\|y^m\|_2$ is not guaranteed. Considering the high flexibility of tracking environment, it may be the most case where $\|y^m\|_2$ shakes around a small value around zero. That is why we set a small threshold δ to stop iteration softly. On the other hand, if the computation fails to converge into the threshold, it probably means the target just suffered a large change on appearance and a certain number of iteration would suffice for a good estimation. Eventually, we have the final transformation $y_t = y^m, 1 \leq m \leq M$.

2.3 Cooperation with supervised classification

Collaboration between tracker and validator has been successfully applied in the previous research [4] [16]. In [16], an independent fern-like structure is used to learn a binary classifier in a supervised manner, and the tracking results are verified by it in order to ensure high accuracy. In [4], an online supervised recognizer is trained as an adaptive prior for tracking. The proposed method in this project applies the similar idea, and we note that our supervised classifier can greatly improve the tracker, in the following two points of view.

Firstly, a potential issue of NNS is that the nearest neighbor may not be associated with the best transformation. This could be ascribed to large appearance variation of the target and the intrinsic approximation of NNS. Inspired by [16], we propose to use the classifier verification strategy to solve this problem. In the detection step, we retrieve K nearest neighbors instead of only one, and

Fig. 2. Illustration of our detection strategy. (a) is one frame of the video at time $t-1$ and the green box is the target position, while the orange box in (b) is one of the training samples learned at $t-1$. Then the target moves towards left and (c) is the frame at time t, but the target box still stays where it was in (a). To detect the new target position, a nearest neighbor searching is done on the sample in green box in (c), and the result indicates that the green sample in (c) is in similar situation of the orange sample in (b). Given that, the tracker retrieves the transformation associated with the orange box, and moves the green one towards left to the blue one, which is the new target position at time t as shown in (d)

sort them in descending order according to the similarity with the query. Then we make a temporal transformation based on each of these neighbors associated models. The image patch within the transformed box is sent to the classifier for a verification. If the feedback is positive, it means the transformation is good. We do such verification from the most similar neighbor to the least, and it stops once the classifier reports the first positive one. If there is no such neighbor then we consider the LSH detection to be plausible, and simply pick the first one regardless of its verification result. Plausible LSH detection doesnt necessarily mean tracking failure but only indicates conflict between the tracker and the verifier, and when it happens we always choose the trackers output. In fact, during our experiment such conflict occurs for very few times. Figure 3 shows the percentage of frames where positive nearest neighbors exist for each of our 15 test videos. As shown, successful cooperation between LSH tracker and SVM verifier occurs for the most time and the average percentage of verified detection is above 97%.

Secondly, in most supervised approaches, new data is sampled once per frame to update the classifiers. It is beneficial to do learning so often in supervised setting, since more data can provide richer information about the target and alleviate the ambiguity of the classifier. In contrast, our LSH based algorithm may suffer from frequent updating. The main reason is that LSH tables preserve each individual sample rather than integrate all of them. Therefore, with a higher updating frequency, consecutive samples would be closer on the timeline and they would become more similar. As a result, NNS is more likely to be confused when selecting the nearest neighbor, leading to inaccurate output. Ideally, learning should run faster when the target changes significantly and slower when it doesnt. That is to say, the frequency is controlled by the level of target variation.

Now we need a way to determine when the target is undergoing fast change. We notice that well-structured classifiers may also be efficient tools to detect

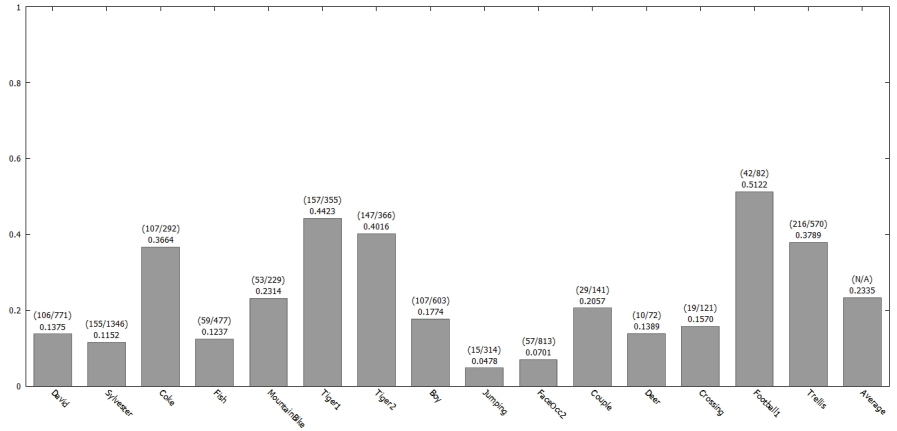


Fig. 3. Number of frames containing positive neighbors and their percentages

data novelty, and SVM is one of them [17] [18]. The support vectors can be regarded as a brief description of all previous samples, and increment of support vectors during learning is usually associated with large adjustment of the split hyper-plan. This is a sign of singular data among the training data set. In the tracking scenario, it can be interpreted as significant appearance change of the target. Inspired by this, we propose a simple novelty detection strategy: variation occurs only when there are new positive support vectors added. Since we update the SVM classifier in every frame, support vectors increase simultaneously with the targets change, but the frequency is much less than once per frame. Figure 4 shows the percentage of frames where new positive support vectors occur for all the 15 test videos. As we can see, the value varies from 4.78% to 51.22%, which is mainly due to the different levels of target variation in different videos. Meanwhile, the average percentage is only 23.35%, indicating that less than one fourth of frames have the necessity to update the LSH database. Both the variance in percentage and the low mean value demonstrate the effectiveness and efficiency of our dynamic strategy.

2.4 Implementation Details

The streaming form of image data requires the algorithm to be in an online manner. It is intuitive to implement LSH online, since we only need to periodically insert new sample and do queries based on current data. When the size of data reaches the maximum capacity of hash tables, we simply delete the oldest data to create space. For the online SVM, some research has been done on this topic [6] [7] [8] [10]. We refer to Struck [6] and take advantage of the accuracy and speed of Larank by using it only in 2-class setting. Similar with [6], we also apply a budget strategy to maintain a maximum number of support vectors in order to keep the computational cost acceptable.

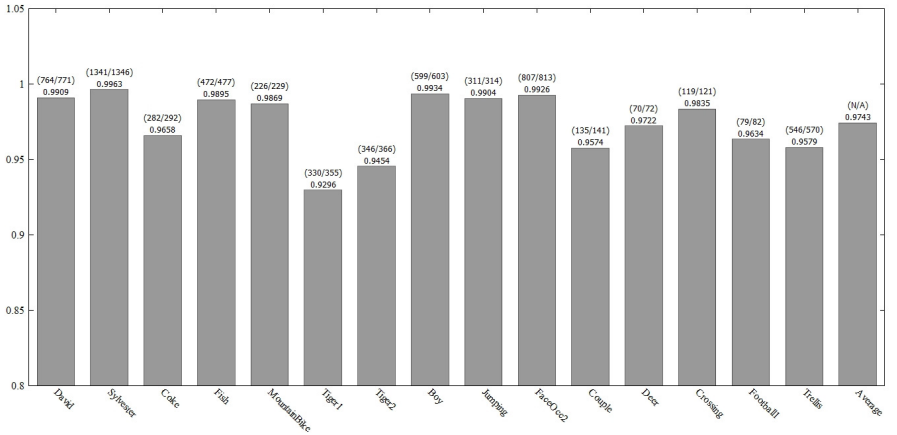


Fig. 4. Number of frames where new positive support vectors are added and their percentages

The concept of LSH was firstly introduced by Indyk and Motwani in [19]. The main property of LSH is that similar items have high probability to be collided being mapped into the same buckets. During the past decade many variants have been proposed and their success has been demonstrated [20] [21] [22] [23] [24]. In this research we use Multi-Probe Locality Sensitive Hashing (MPLSH) [22] as the implementation of LSH. MPLSH is based on p-stable distribution hashing [20], where each hash function is defined by:

$$h_{\mathbf{a},\mathbf{b}}(\mathbf{v}) = \lfloor \frac{\mathbf{a} \cdot \mathbf{v} + \mathbf{b}}{\mathbf{W}} \rfloor \tag{3}$$

where a is a normalized d-dimensional random vector generated by a Gaussian distribution, W is a real number and b is a random number chosen in [0, W]. The multi-probe strategy aims at generating a probing sequence of buckets that are most likely to contain the most similar items, in order to obtain as many nearest neighbors as possible within an acceptable computing expense.

The main workflow of the learning step is described in Algorithm 1. For the first frame we conduct learning for both the LSH database and the SVM classifier. For the rest frames, we update the SVM classifier and check if any new positive support vector is added. If so, or if not but it is at the period point for static update, we update the LSH database. The sampling strategy for the SVM classifier is similar to [15] with a little modification, where positive samples are taken at every possible position that is close to the target, while negative samples are randomly chosen within an area far away from the target.

Algorithm 2 summarizes the detection step. As it states, estimation of the new target position is done for M rounds iteratively. For each round, K nearest neighbors are retrieved and sorted in descending order. SVM is then applied to

Algorithm 1 Learning step of our tracking algorithm (from the second frame)

Input:

the new target position p_t in the t-th frame f_t;

Obtain a set $S_pos = (s_j, y_i)|i = 1, 2, ..., L$ by sampling at each position within a radius $R1$ centered at p_t, then obtain the negative set $S_neg = (s_j, y_j)|j = 1, 2, ...M$ by randomly selecting M samples out of a radius R_2 with $R_1 < R_2$;

Update the SVM classifier by S_pos and S_neg, check if there is new positive support vector added;

if there is new positive support vector added **or** it is at the update period point, **then**

    Obtain a set $S = (s_i, y_i)|i = 1, 2, ..., N$ containing all the N samples within a radius R_2 centered at p_t, with y_i being the associated transformation of s_i. Then insert each s_i into the LSH database;

end if

Algorithm 2 Detection step of our tracking algorithm

Input:

t-th frame f_t; target position in the $t-1$-th frame p_{t-1};

Output:

target position p_t in the t-th frame $f - t$;

$p^0 = p_{t-1}$;

for $i = 1$ to M **do**

    Do nearest neighbor searching on p^{i-1} and get the neighbor set $P = p_k|k = 1, 2, ..., k$ containing K nearest neighbors in descending order of similarity with p^{i-1};

    for $j = 1$ to K **do**

        Obtain the temporal target position $p_j^{'} = p_j \circ y_j$ where y_j is the transformation associated with p_j;

        Extract the image patch I_j at $p_j^{'}$ and do a verification $SVM(I_j)$;

        if $SVM(I_j) = true$ **then**

            break;

        end if

    end for

    if there is one patch I_j that is verified to be valid **then**

        $p^i \leftarrow p_j^{'}$;

        $y^i \leftarrow y_j$;

    else

        $p^i \leftarrow p_1^{'}$;

        $y^i \leftarrow y_1$;

    end if

end for

$p_t = p^i$;

verify each of them in order to pick the first valid one. The iteration stops either when it reaches M rounds or the most recent transformation y^i is smaller than a threshold δ. In this project we set δ as 5 in pixels.

3 Experiment

3.1 Experiment Setup

We implement our algorithm by C++ in Linux-32bit on a PC with an Intel Core i5 2.5GHz CPU and 4GB RAM. Similar with [6], we use Haar-like feature and generate a 192-dimension vector for each image patch. The Multi-Probe Locality Sensitive Hashing(MPLSH) is implemented by lshkit, a publicly available C++ library of the MPLSH algorithm, with the parameters where W is 1.0, the number of hash tables L is 4, the probing step level T is 5, and the nearest neighbors number K is 10. The online SVM algorithm, Larank, is implemented by our own code with reference to [6] but under the two-class setting, and we set the support vector budget as 200. A Gaussian kernel is applied and the slack value is set as 100. The learning radius for LSH database is 20. The positive sample radius R_1 for classifier update is set as 2 and the negative radius R_2 is 50. The static update frequency F is set as one per 4 frames for all the sequences.

Three evaluation metrics are applied here: VOC overlap ratio (VOR), center location error (CLE) and tracking success rate (SR), all calculated between algorithm output boxes B_o and ground truth boxes B_g. VOR is defined by $\frac{area(B_o \cap B_g)}{area(B_o \cup B_g)}$, CLE is defined as the average distance between the centers of B_o and B_g, and SR is defined as the ratio of frames whose VOR are larger than 50% among all frames.

Table 1. The VOC overlap ratio (VOR) comparison results of 10 trackers over 15 sequences with an average value for each tracker. Numbers are represented in percentage. The best and the second best in each row is shown in bold and underlined, respectively

	Ours	**MTT**	**CT**	**Frag**	**DFT**	**BSBT**	**L1APG**	**IVT**	**MIL**	**Struck**	**LSH**
Deer	**45.77**	21.30	2.73	11.73	45.58	13.29	39.61	13.71	37.85	44.86	4.93
Sylvester	73.63	71.17	60.79	60.35	25.43	62.14	35.68	54.60	8.46	**73.66**	62.17
Fish	**87.67**	64.51	66.92	49.21	65.86	60.60	86.18	81.54	61.92	82.19	85.70
Coke	**67.09**	59.28	26.91	3.69	15.01	27.04	21.66	10.56	3.25	56.55	16.30
Crossing	**71.18**	17.53	60.15	30.06	59.92	20.93	17.43	22.76	64.73	54.95	35.77
Couple	**68.28**	37.48	5.39	60.24	8.73	5.70	34.85	7.27	42.11	50.89	9.10
David	**55.49**	43.19	18.29	2.33	24.82	41.54	54.80	9.37	5.35	47.19	32.42
FaceOcc2	77.21	75.65	59.21	66.19	**77.94**	57.20	70.45	71.77	58.81	77.46	72.12
Football1	**74.49**	56.91	28.59	28.59	74.34	11.96	38.89	51.76	25.52	56.58	72.90
Boy	**78.02**	40.42	41.73	41.73	19.39	67.51	35.13	25.75	2.21	74.96	33.97
Jumping	64.31	6.96	31.52	**65.08**	5.96	26.25	42.22	20.25	28.31	60.84	12.11
MountainBike	77.14	71.08	12.95	10.59	**77.25**	63.20	67.98	39.79	12.60	70.80	73.51
Tiger1	**64.11**	32.44	13.40	34.95	58.17	27.37	42.48	12.96	4.95	63.90	12.41
Tiger2	55.00	46.58	41.79	7.66	34.39	21.50	58.94	17.86	12.52	**60.48**	12.97
Trellis	**71.91**	60.74	34.22	35.71	41.74	15.27	42.56	16.73	26.70	70.89	40.53
Average	**68.75**	47.02	33.64	33.87	42.30	34.77	45.92	30.45	26.35	63.08	38.46

3.2 Quantitative performance evaluation

In this section we evaluate the performance of our algorithm by comparison with other 10 trackers. Among them, [25], [15], [26], [27], [4], [28], [29] have Matlab implementation in [30] and we use it directly in our experiment. For [5], [6], and [31], we use the code posted on their corresponding websites. Fifteen publicly available sequences from [30] are selected for the comparison, and we use each sequence with its original length. The chosen sequences contain challenges such as illumination variation, fast motion, partial occlusion, in-plane and out-of-plane rotation, to name just a few. The ground truth data for the sequences are also collected from [30] except *David*, where we use the ground truth of all the frames from frame No.1 to frame No.771 given by [16] instead of from 300 to 771. The longer one contains larger illumination change and we regard it as a higher evaluation standard for algorithm performance. Table 1-3 show the results of performance comparison under the three criteria. Figure 3 provides a complete view of performance by plotting CLE frame-by-frame for all the 11 trackers on 15 sequences. As we can see, our algorithm has overall better performance over others, and even for the ones where our tracker does not get the best, its performance is still close to the best. For a concise display, we only show CLE plots here. VOR and SR plots can be found in our supplementary material.

Table 2. The center location error (CLE) comparison results of 10 trackers over 15 sequences with an average value for each tracker. Numbers are represented in pixels. The best and the second best in each row is shown in bold and underlined, respectively

	Ours	**MTT**	**CT**	**Frag**	**DFT**	**BSBT**	**L1APG**	**IVT**	**MIL**	**Struck**	**LSH**
Deer	**12.35**	26.80	114.12	82.35	12.48	43.25	17.84	40.24	14.99	12.87	55.84
Sylvester	**5.78**	6.51	15.48	14.75	53.96	12.01	28.02	37.58	82.73	6.00	13.00
Fish	**2.86**	13.26	13.51	25.41	12.71	31.75	3.69	3.70	15.08	4.74	3.79
Coke	**5.81**	8.53	21.52	66.97	31.12	18.01	22.98	43.94	59.91	7.77	33.30
Crossing	**2.26**	40.20	5.52	37.52	7.88	45.82	53.37	3.01	4.19	6.66	25.43
Couple	**4.59**	36.51	103.84	9.54	112.79	88.10	27.45	96.20	37.75	33.32	111.77
David	**6.12**	17.75	41.67	114.50	41.97	18.51	11.69	220.06	75.50	14.01	24.28
FaceOcc2	6.96	9.18	17.69	14.71	7.47	31.03	9.06	6.91	19.37	**6.79**	10.02
Football1	**3.35**	11.90	18.75	18.75	5.76	46.82	13.94	19.12	17.46	7.04	3.76
Boy	**1.34**	24.23	29.85	29.85	76.43	5.48	35.10	51.80	26.84	1.92	19.38
Jumping	5.53	53.03	16.92	**4.50**	63.35	27.36	23.33	36.73	16.89	6.18	40.39
MountainBike	**3.14**	5.09	113.18	120.07	3.62	7.23	5.32	43.25	112.80	5.30	3.82
Tiger1	**7.49**	33.44	48.15	28.24	11.81	30.51	22.84	119.08	56.28	7.66	57.42
Tiger2	9.19	13.09	14.39	71.30	24.93	31.12	8.23	37.08	42.81	**7.19**	52.25
Trellis	**7.35**	13.42	39.43	56.25	48.22	80.69	38.69	122.35	54.70	8.53	59.14
Average	**5.61**	20.86	40.93	46.31	34.30	34.51	21.44	58.74	42.49	9.07	34.24

3.3 Qualitative performance evaluation

In this section we discuss several key components of our algorithm and illustrate their contribution to the overall performance of our tracker. We treat our algorithm with different components as different algorithms and we do the similar experiments as the ones conducted in the last section. Similarly, we use the

Table 3. The success rate (SR) comparison results of 10 trackers over 15 sequences with an average value for each tracker. Numbers are represented in pixels. The best and the second best in each row is shown in bold and underlined, respectively

	Ours	MTT	CT	Frag	DFT	BSBT	L1APG	IVT	MIL	Struck	LSH
Deer	26.76	7.04	1.41	7.04	22.54	11.27	**33.80**	9.86	16.90	21.13	2.82
Sylvester	**96.65**	84.98	84.76	70.71	28.55	72.27	33.80	67.88	0.37	93.98	76.28
Fish	**100.00**	81.09	71.22	41.81	60.29	65.34	**100.00**	**100.00**	83.19	**100.00**	**100.00**
Coke	**89.00**	85.57	13.06	3.44	9.62	24.74	22.68	13.40	1.37	57.73	15.12
Crossing	**100.00**	22.50	76.67	40.83	66.67	23.33	22.50	23.33	86.67	70.83	41.67
Couple	**76.43**	32.86	5.00	70.00	10.00	7.14	32.86	8.57	57.14	61.43	10.00
David	76.62	53.90	7.79	0.13	13.38	36.88	**84.42**	15.97	0.13	64.16	12.34
FaceOcc2	**100.00**	96.06	71.31	75.25	100.00	65.02	96.31	92.36	77.83	99.75	98.03
Football1	**100.00**	59.46	25.68	25.68	87.84	10.81	33.78	63.51	9.46	54.05	97.30
Boy	**99.34**	48.01	53.16	53.16	23.59	87.21	43.85	30.90	0.33	98.50	32.89
Jumping	**98.72**	6.39	16.29	96.81	8.63	31.31	64.22	25.56	1.28	93.61	12.46
MountainBike	**100.00**	**100.00**	16.67	11.84	**100.00**	89.04	89.91	52.63	16.23	90.35	98.25
Tiger1	**87.29**	37.57	2.54	40.11	72.88	25.14	43.50	15.54	2.82	86.72	10.45
Tiger2	63.84	51.23	33.70	9.59	39.18	13.15	75.34	20.82	11.51	**77.81**	14.25
Trellis	**95.61**	67.66	37.08	42.18	51.14	11.25	51.14	17.93	31.99	93.67	43.76
Average	**87.35**	55.62	34.42	39.24	46.29	38.26	55.21	37.22	26.48	77.58	44.37

three same criteria to evaluate the overall performance on all the 15 sequences. For each of the following two components, we tune one at a time and control the other by setting it as the default value. The defaults values are $F = 4$ and $K = 20$.

Table 4. Comparison results of our algorithm with different frequency values on 15 sequences based on VOC overlap rate (VOR), center location error (CLE), and success rate (SR). We only display the best result in red under each criterion for clarity and conciseness

	VOR					CLE					SR				
Sequences	F=1	F=2	F=4	F=8	F=inf	F=1	F=2	F=4	F=8	F=inf	F=1	F=2	F=4	F=8	F=inf
Deer	**47.03**	46.31	45.77	46.11	45.47	**11.98**	12.25	12.35	12.24	12.42	**33.80**	23.94	26.76	28.17	28.17
Sylvester	66.95	58.33	**73.63**	62.13	64.95	9.78	33.21	**5.78**	15.02	11.99	82.16	80.45	**96.65**	72.19	80.00
Fish	49.32	85.77	**87.67**	86.40	86.40	27.06	3.76	**2.86**	3.54	3.54	59.45	**100.00**	**100.00**	**100.00**	**100.00**
Coke	50.91	56.16	**67.09**	66.17	66.17	9.25	8.05	5.81	**5.60**	**5.60**	52.58	72.16	**89.00**	83.51	83.51
Crossing	36.64	28.32	71.18	**71.34**	19.30	25.78	54.94	**2.26**	2.61	64.34	48.33	38.33	**100.00**	**100.00**	24.17
Couple	54.75	65.73	**68.28**	27.92	47.88	17.83	5.55	**4.59**	70.95	42.61	64.29	75.71	**76.43**	32.86	64.29
David	54.34	43.74	**55.49**	53.39	53.39	6.61	16.74	**6.12**	8.71	8.71	75.32	52.21	**76.62**	74.42	74.42
Faceocc2	69.79	**78.44**	77.21	77.09	75.85	12.88	**6.27**	6.96	6.56	7.10	86.70	**100.00**	**100.00**	**100.00**	**100.00**
Football1	43.24	41.66	**74.49**	66.30	50.91	26.42	49.78	**3.35**	5.01	10.64	51.35	51.35	**100.00**	74.32	54.05
Boy	70.53	67.06	**78.02**	69.74	26.76	2.46	3.35	**1.34**	2.53	20.87	97.67	88.04	**99.34**	95.51	31.73
Jumping	62.51	**64.79**	64.31	19.76	63.68	5.86	**5.35**	5.53	54.23	5.77	97.44	98.40	98.72	29.71	**99.36**
MountainBike	33.74	62.82	**77.14**	55.43	20.35	63.72	6.73	**3.14**	48.80	105.07	40.35	91.23	**100.00**	69.74	26.32
Tiger1	23.34	28.12	**64.11**	54.29	9.57	58.52	62.94	**7.49**	10.84	55.84	19.77	36.16	**87.29**	50.28	5.08
Tiger2	21.48	46.18	**55.93**	19.65	18.02	36.75	18.61	**9.60**	60.29	79.96	24.38	48.22	**63.84**	24.93	22.19
Trellis	47.42	28.76	**71.91**	57.66	21.22	29.60	82.90	**7.35**	18.38	74.28	50.79	36.20	**95.61**	69.42	25.48

Static LSH update frequency *F* The static LSH update frequency determines the regular learning pace and therefore ensures the sufficiency of information. We set this parameter as 1, 2, 4, 8, infinite, respectively, meaning that the LSH database is updated every 1, 2, 4, 8 frames in addition to the dynamic strategy and infinite means there is no static frequency and updating only depends on the dynamic control. Table 4 shows the comparative performance under the

three metrics on all the 15 sequences. As we can observe, for each of the three criteria, our algorithm performs best in terms of overall accuracy when F is 4, which is a proper value to keep the learning pace reasonably slow and satisfy the purpose of preventing data redundancy.

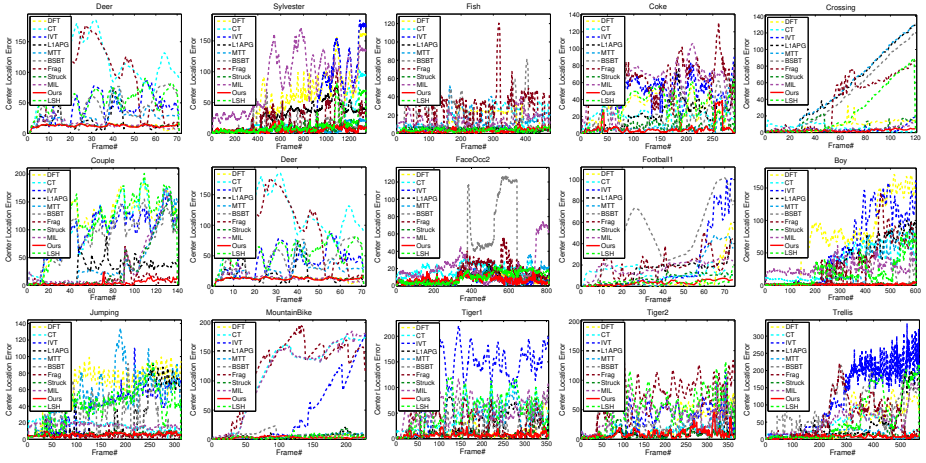


Fig. 5. The frame-by-frame center location error (CLE) plots for all the 15 sequences. Our method is drawn in red.

Number of retrieved nearest neighbors K Intuitively, the larger this parameter is, the more positive neighbors there will be, and correspondingly the more confident the results will be. However, a neighbor set with larger size also contains more weak neighbors, which may increase the risk of false verification by the classifier. Here, we aim to evaluate the effect of this factor by setting K as 5, 10, 15, and 20 and apply the same experiments on each of them. Table 5 shows the comparative performance under the three metrics on the 15 sequences. As shown, 10 is the best value for K in terms of overall performance. It is worth noticing that for most of the sequences the results of $K = 10$ and $K = 20$ are the same. This is a desirable property because it indicates that most positive nearest neighbor is among the top 10 in the neighbors list. Thus, it is not necessary to retrieve large amount of neighbors to achieve promising performance.

4 Conclusion

In this project we propose a novel tracking framework based on cooperation of unsupervised nearest neighbor searching and supervised classification. We regard tracking as a problem of information retrieval on streaming data, and use an SVM based classifier to improve its accuracy and control the LSH learning frequency. From a learning view of point, this strategy is an extension of the well-studied classification-based tracking, which assigns each sample

Table 5. Comparison results of our algorithm with different numbers of nearest neighbors on 15 sequences based on VOC overlap rate (VOR), center location error (CLE), and success rate (SR). We only display the best result in red under each criterion for clarity and conciseness

	VOR				CLE				SR			
Squences	K=1	K=5	K=10	K=20	K=1	K=5	K=10	K=20	K=1	K=5	K=10	K=20
Deer	41.95	45.77	45.77	45.77	20.25	12.35	12.35	12.35	29.58	26.76	26.76	26.76
Sylvester	72.62	73.17	73.63	73.63	6.22	5.97	5.78	5.78	95.76	95.76	96.65	96.65
Fish	84.34	87.63	87.67	87.67	4.58	2.85	2.86	2.86	100.00	100.00	100.00	100.00
Coke	70.01	63.83	67.09	17.74	4.69	6.44	5.81	52.27	87.29	82.47	89.00	23.37
Crossing	69.00	71.18	71.18	71.18	3.66	2.26	2.26	2.26	95.00	100.00	100.00	100.00
Couple	56.30	68.28	68.28	68.28	38.49	4.59	4.59	4.59	72.14	76.43	76.43	76.43
David	53.35	53.56	55.49	55.49	8.38	8.23	6.12	6.12	71.82	71.69	76.62	76.62
Faceocc2	77.78	77.21	77.21	77.10	6.71	6.97	6.96	7.01	100.00	100.00	100.00	100.00
Football1	30.12	55.28	74.49	74.49	83.71	10.54	3.35	3.35	32.43	62.16	100.00	100.00
Boy	76.96	76.96	78.02	78.02	1.56	1.56	1.34	1.34	99.00	99.00	99.34	99.34
Jumping	19.79	64.31	64.31	64.31	56.70	5.53	5.53	5.53	30.03	98.72	98.72	98.72
MountainBike	53.57	77.14	77.14	77.14	48.82	3.14	3.14	3.14	66.23	100.00	100.00	100.00
Tiger1	8.04	64.01	64.11	24.30	52.65	7.88	7.49	56.60	8.47	84.75	87.29	30.79
Tiger2	60.43	56.41	55.93	30.03	7.85	9.17	9.60	61.02	73.97	64.38	63.84	36.99
Trellis	64.78	65.09	71.91	46.21	12.31	11.58	7.35	57.03	73.99	85.59	95.61	54.13

with a confidence value while our method directly assigns the transformation structure. Benefiting from that, our method avoids the brute-force search within the output space and successfully alleviates the intrinsic obscurity of supervised approaches in describing targets. We show experimentally that our tracker has superior performance over state-of-art trackers.

We do believe that the proposed tracking framework contributes to a brand new field where more unsupervised strategies could be introduced into tracking. Future studies can be done based on this framework, such as using other unsupervised structures for nearest neighbor searching and applying more sophisticated novelty detection algorithms for controlling updating frequency. Besides, transformation model can also be enriched in order to include more variation types such as rotation and scale changes.

References

1. Avidan, S.: Support vector tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(8) (2004) 1064–1072
2. Grabner, H., Grabner, M., Bischof, H.: Real-time tracking via online boosting. (2006) 47–56
3. Grabner, H., Leistner, C., Bischof., H.: Semi-supervised on-line boosting for robust tracking. (2006) 234–247
4. Stalder, S., Grabner, H., Gool., L.V.: Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. (2009) 1409–1416
5. Babenko, B., Yang, M.H., Belongie., S.: Visual tracking with online multiple instance learning. (2009) 983–990
6. Hare, S., Saffari, A., Torr., P.H.: Struck: Structured output tracking with kernels. (2011) 263–270

7. Bordes, A., Bottou, L., Gallinari, P., Weston., J.: Solving multiclass support vector machines with larank. (2007) 89–96
8. Bordes, A., Usunier, N., Bottou., L.: Sequence labelling svms trained in one pass. (2008) 146–161
9. Crammer, K., Kandola, J.S., Singer., Y.: Online classification on a budget. Volume 2. (2003) 5
10. Wang, Z., Crammer, K., Vucetic., S.: Multi-class pegasos on a budget. Volume 2. (2010) 1143–1150
11. Yao, R., Shi, Q., Shen, C., Zhang, Y., van den Hengel., A.: Robust tracking with weighted online structured learning. (2012) 158–172
12. Yao, R., Shi, Q., Shen, C., Zhang, Y., van den Hengel., A.: Part-based visual tracking with online latent structural learning. (2013) 2363–2370
13. Li, X., Shen, C., Dick, A., van den Hengel., A.: Learning compact binary codes for visual tracking. (2013) 2419–2426
14. Saffari, A., Leistner, C., Santner, J., Godec, M., Bischof., H.: Learning compact binary codes for visual tracking. (2009) 1393–1400
15. Zhang, K., Zhang, L., Yang., M.H.: Real-time compressive tracking. (2012) 864–877
16. Kalal, Z., Mikolajczyk, K., Matas., J.: Tracking-learning-detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(7) (2012) 864–877
17. Schlkopf, B., Williamson, R.C., Smola, A.J., Shawe-Taylor, J., Platt., J.C.: Support vector method for novelty detection. Volume 12. (1999) 582–588
18. Gmez-Verdejo, V., Arenas-Garca, J., Lazaro-Gredilla, M., Navia-Vazquez., A.: Adaptive one-class support vector machine. IEEE Transactions on Signal Processing **59**(6) (2011) 2975–2981
19. Indyk, P., Motwani., R.: Approximate nearest neighbors: towards removing the curse of dimensionality. (1998) 604–613
20. Datar, M., Immorlica, N., Indyk, P., Mirrokni., V.S.: Locality-sensitive hashing scheme based on p-stable distributions. (2004) 253–262
21. Weiss, Y., Torralba, A., , Fergus., R.: Spectral hashing. Volume 9. (2008) 6
22. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li., K.: Multi-probe lsh: efficient indexing for high-dimensional similarity search. (2007) 950–961
23. Kulis, B., Grauman., K.: Kernelized locality-sensitive hashing for scalable image search. (2009) 2130–2137
24. Heo, J.P., Lee, Y., He, J., Chang, S.F., Yoon., S.E.: Spherical hashing. (2012) 2957–2964
25. Zhang, T., Ghanem, B., Liu, S., Ahuja., N.: Robust visual tracking via multi-task sparse learning. (2012) 2042–2049
26. Adam, A., Rivlin, E., Shimshoni., I.: Robust fragments-based tracking using the integral histogram. (2006) 798–805
27. Sevilla-Lara, L., Learned-Miller., E.: Distribution fields for tracking. (2012) 1910–1917
28. Bao, C., Wu, Y., Ling, H., Ji., H.: Real time robust l1 tracker using accelerated proximal gradient approach. (2012) 1830–1837
29. Ross, D.A., Lim, J., Lin, R.S., Yang., M.H.: Incremental learning for robust visual tracking. International Journal of Computer Vision **77**(1–3) (2012) 125–141
30. Wu, Y., Lim, J., Yang., M.H.: Online object tracking: A benchmark. (2013) 2411–2418
31. He, S., Yang, Q., Lau, R.W., Wang, J., Yang., M.H.: Visual tracking via locality sensitive histograms. (2013) 2427–2434