Visual Tracking via Locality Sensitive Histograms

Shengfeng He[‡] Qingxiong Yang[‡*] Rynson W.H. Lau[‡] Jiang Wang[‡] Ming-Hsuan Yang[§]
[‡]Department of Computer Science City University of Hong Kong
[§]Electrical Engineering and Computer Science University of California at Merced
shengfeng_he@yahoo.com, {qiyang, rynson.lau, jiangwang6}@cityu.edu.hk, mhyang@ucmerced.edu

Abstract

This paper presents a novel locality sensitive histogram algorithm for visual tracking. Unlike the conventional image histogram that counts the frequency of occurrences of each intensity value by adding ones to the corresponding bin, a locality sensitive histogram is computed at each pixel location and a floating-point value is added to the corresponding bin for each occurrence of an intensity value. The floating-point value declines exponentially with respect to the distance to the pixel location where the histogram is computed; thus every pixel is considered but those that are far away can be neglected due to the very small weights assigned. An efficient algorithm is proposed that enables the locality sensitive histograms to be computed in time linear in the image size and the number of bins. A robust tracking framework based on the locality sensitive histograms is proposed, which consists of two main components: a new feature for tracking that is robust to illumination changes and a novel multi-region tracking algorithm that runs in real-time even with hundreds of regions. Extensive experiments demonstrate that the proposed tracking framework outperforms the state-of-the-art methods in challenging scenarios, especially when the illumination changes dramatically.

1. Introduction

Visual tracking is one of the most active research areas in computer vision, with numerous applications including augmented reality, surveillance, and object identification. The chief issue for robust visual tracking is to handle the appearance change of the target object. Based on the appearance models used, tracking algorithms can be divided into generative tracking [4, 17, 19, 14, 3] and discriminative tracking [7, 2, 11, 8].

Generative tracking represents the target object in a particular feature space, and then searches for the best matching score within the image region. In general, it does not requires a large dataset for training. Discriminative tracking treats visual tracking as a binary classification problem to define the boundary between a target image patch and the background. It generally requires a large dataset in order to achieve good performances. While numerous algorithms of two categories have been proposed with success, it remains a challenging task to develop a tracking algorithm that is both accurate and efficient. In order to address challenging factors in visual tracking, numerous features and models have been used to represent target objects.

The appearance of an object changes drastically when illumination varies significantly. It has been shown that two images, taken at the same pose but under different illumination conditions, cannot be uniquely identified as being the same object or different ones [10]. To deal with this problem, numerous methods have been proposed based on illumination invariant features [5]. Early works on visual tracking represent objects with contours [9] with success when the brightness constancy assumption holds. The eigentracking approach [4] operates on the subspace constancy assumption to account for the appearance change caused by illumination variation based on a set of training images. Most recently, Harr-like features and online subspace models have been used in object tracking with demonstrated success in dealing with large lighting variation [13, 2, 3, 22]. Notwithstanding the demonstrated success, these methods often require time-consuming mixture models or optimization processes.

Due to their simplicity, intensity histograms are widely used to represent objects for recognition and tracking. However, spatial information of object appearance is missing in this holistic representation, which makes it sensitive to noise as well as occlusion in tracking applications. To address this problem, algorithms based on multi-region representations have been proposed. The fragment-based tracker [1] divides the target object into several regions and represents them with multiple local histograms. A vote map is used to combine the votes from all the regions in the target frame. However, the computation of multiple local histograms and the vote map can be time consuming even with the use of integral histograms [15]. As a trade-off between accuracy and speed, the fragment-based method [1] uses up to 40 regions to represent the target object and thus causes jitter effects. To account for large geometric appearance changes, recent multi-region trackers combine local and global representations by adapting the region locations

*Correspondence author.

to the geometric change, instead of using a fixed grid layout [1]. In [20], each target object is modeled by a small number of rectangular blocks, which positions within the tracking window are adaptively determined. In [13, 22], a fixed number of object parts are dynamically updated to account for appearance and shape changes. All these methods achieve better accuracy at the expense of speed.

In this paper, we propose a novel locality sensitive histogram algorithm, which takes into account contributions from every pixel in an image instead of from pixels only inside local neighborhoods as the local histogram algorithm. It operates in a way similar to conventional image histograms. However, instead of counting the frequency of occurrences of each intensity value by adding ones to the corresponding bin, a floating-point value is added to the corresponding bin for each occurrence of an intensity value. The floating-point value declines exponentially with respect to the distance to the pixel location where the locality sensitive histogram is computed. Thus, the proposed histogram is more suitable for applications such as visual tracking, which assigns lower weights to pixels further away from the target center (as these pixels more likely contain background information or occluding objects, and hence their contribution to the histogram is diminished).

The proposed histogram has an $O(NB)$ complexity, where N is the number of pixels and B is the number of bins. This facilitates a framework for real-time object tracking. This tracking framework effectively deals with drastic illumination changes by extracting dense illumination invariant features using the proposed locality sensitive histogram. It also handles significant pose and scale variation, occlusion and visual drifts, out of plane rotation and abrupt motion, and background clutters with the use of a novel multi-region tracking algorithm. Unlike existing multi-region trackers that need to represent a target object with a limited number of non-overlapping regions due to the use of rectangular local histograms, the proposed multi-region tracker efficiently describes a target object with a large number of overlapping regions using the locality sensitive histogram, which takes into account contributions from every pixel adaptively. This unique property facilitates robust multi-region tracking, and the efficiency of the locality sensitive histogram enables the proposed tracker to run in real time. The contributions of this paper are summarized as follows:

- a histogram that takes into account contributions from every image pixel adaptively and facilitates a real-time tracking framework;
- an efficient illumination invariant feature for tracking; and
- a multi-region tracking algorithm that outperforms state-of-the-art algorithms both in terms of accuracy and speed.

2. Locality Sensitive Histograms

The conventional image histogram is a 1D array. Each of its values is usually an integer indicating the frequency of occurrence of a particular intensity value. Let $\mathbf{I}$ denote an image. The corresponding image histogram $\mathbf{H}$ is a B-dimensional vector defined as:

$$\mathbf{H}(b) = \sum_{q=1}^{W} Q(\mathbf{I}_q, b), \quad b = 1, \ldots, B, \tag{1}$$

where W is the number of pixels, B is the total number of bins, and $Q(\mathbf{I}_q, b)$ is zero except when intensity value $\mathbf{I}_q$ (at pixel location q) belongs to bin b. If computed using Eq. 1, the computational complexity of the histogram is linear in the number of bins at each pixel location: $O(B)$. However, the complexity can be reduced to $O(1)$ in practice because the addition operation in Eq. 1 can be ignored when $Q(\mathbf{I}_q, b) = 0$.

Local histograms, on the other hand, record statistics within a region of a pixel in an image. They are computed at each pixel location, and have been proved to be very useful for many tasks like the bilateral filtering [25, 16], median filtering [25, 12] and tracking [15, 1]. The computational complexity of the brute-force implementation of the local histograms is linear in the neighborhood size. Nevertheless, this dependency can be removed using integral histogram [15], which reduces the computational complexity to $O(B)$ at each pixel location. Let $\mathbf{H}^{\mathbf{I}}_p$ denote the integral histogram computed at pixel p. It can be computed based on the previous integral histogram computed at pixel $p-1$ in a way similar to the integral image [6, 23]:

$$\mathbf{H}^{\mathbf{I}}_p(b) = Q(\mathbf{I}_p, b) + \mathbf{H}^{\mathbf{I}}_{p-1}(b), \quad b = 1, \ldots, B. \tag{2}$$

For simplicity, let $\mathbf{I}$ denote a 1D image. $\mathbf{H}^{\mathbf{I}}_p$ contains the contributions of all the pixels to the left of pixel p, and the local histogram between pixel p and another pixel q on the left of p is computed as $\mathbf{H}^{\mathbf{I}}_p(b) - \mathbf{H}^{\mathbf{I}}_q(b)$ for $b = 1, \ldots, B$.

For the local histogram, pixels inside a local neighborhood has equal contribution. However, for applications such as object tracking, pixels further away from the target center should be weighted less as they more likely contain background information or occluding objects. Hence, their contribution to the histogram should be diminished. We propose a novel *locality sensitive histogram* algorithm to address this problem. Let $\mathbf{H}^{E}_p$ denote the locality sensitive histogram computed at pixel p. It can be written as:

$$\mathbf{H}^{E}_p(b) = \sum_{q=1}^{W} \alpha^{|p-q|} \cdot Q(\mathbf{I}_q, b), \quad b = 1, \ldots, B, \tag{3}$$

where $\alpha \in (0, 1)$ is a parameter controlling the decreasing weight as a pixel moves away from the target center. Same

as the local histogram, the computational complexity of its brute-force implementation presented in Eq. 3 is $O(WB)$ per pixel. However, similar to the integral histogram, the proposed locality sensitive histogram also has an efficient algorithm when the input image is 1D:

$$\mathbf{H}_p^E(b) = \mathbf{H}_p^{E,left}(b) + \mathbf{H}_p^{E,right}(b) - Q(\mathbf{I}_p, b), \quad (4)$$

where

$$\mathbf{H}_p^{E,left}(b) = Q(\mathbf{I}_p, b) + \alpha \cdot \mathbf{H}_{p-1}^{E,left}(b), \quad (5)$$

$$\mathbf{H}_p^{E,right}(b) = Q(\mathbf{I}_p, b) + \alpha \cdot \mathbf{H}_{p+1}^{E,right}(b). \quad (6)$$

Based on Eqs. 5 and 6, pixels on the right of pixel p do not contribute to $\mathbf{H}_p^{E,left}$ while pixels on the left of pixel p do not contribute to $\mathbf{H}_p^{E,right}$. The summation of $\mathbf{H}_p^{E,left}$ and $\mathbf{H}_p^{E,right}$, however, combines the contribution from all pixels and the weight of the contribution drops exponentially with respect to the distance to pixel p. Clearly, only B multiplication and B addition operations are required at each pixel location in order to compute $\mathbf{H}_p^{E,left}$ (or $\mathbf{H}_p^{E,right}$). Thus, the computational complexity of the locality sensitive histogram is reduced to $O(B)$ per pixel. A special constraint is that $Q(\mathbf{I}_p, \cdot)$ in Eqs. 5 and 6 is a B-dimensional vector containing zero values except for the bin corresponding to the intensity value $\mathbf{I}_p$ at pixel p. Hence, the addition operation in Eqs. 5 and 6 can be removed except for the bin corresponding to the intensity value $\mathbf{I}_p$.

In practice, most of the applications use normalized histograms. Thus, a normalization step, including a summation operation over all bins, for obtaining the normalization factor is required at each pixel location. A brute-force implementation of this summation operation is obviously $O(B)$ per pixel. Nevertheless, let n_p denote the normalization factor at pixel p, according to Eq. 3,

$$\begin{aligned} n_p &= \sum_{b=1}^{B} \mathbf{H}_p^E(b) = \sum_{q=1}^{W} \alpha^{|p-q|} \cdot \left(\sum_{b=1}^{B} Q(\mathbf{I}_q, b) \right) \\ &= \sum_{q=1}^{W} \alpha^{|p-q|}. \end{aligned} \quad (7)$$

Hence, the computation of the normalization factor n_p is independent of the number of bins B. In addition, as the histogram presented in Eq. 3 can be computed in $O(B)$ per pixel using Eqs. 4-6, n_p can also be computed in the same way:

$$n_p = n_p^{left} + n_p^{right} - 1, \quad (8)$$

$$n_p^{left} = 1 + \alpha \cdot n_{p-1}^{left}, \quad (9)$$

$$n_p^{right} = 1 + \alpha \cdot n_{p+1}^{right}. \quad (10)$$

In this case, the normalization factors can be computed in $O(1)$ per pixel. Furthermore, the normalization factors are independent of the image content and thus can be precomputed to further reduce the computational complexity.

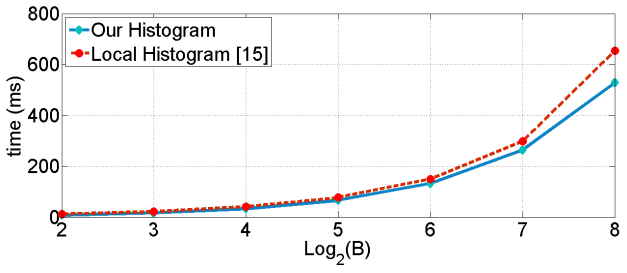


Figure 1: Speed comparison. The red dash curve shows the runtime of computing the local histograms from the integral histogram for a 1 MB image w.r.t. the logarithm of the number of bins B. The blue solid curve shows the runtime of computing the proposed locality sensitive histograms. Note that the speed of the two algorithms are very close and can be computed in real time when the number of bins is relatively low. For instance, when $B = 16$, the locality sensitive histograms can be computed at 30 FPS.

The algorithm presented in Eqs. 4-6 (or Eqs. 8-10) is developed for 1D images. However, its extension to multi-dimensional images is straightforward. We simply perform the proposed 1D algorithm separately and sequentially in each dimension. Figure 1 shows the speed of the proposed algorithm for computing the locality sensitive histogram for a 1 MB 2D image with respect to the logarithm of the number of bins B. Note that it is as efficient as the integral histogram.

3. Illumination Invariant Features

Images taken under different illumination conditions have drastic effects on object appearance. Under the assumption of affine illumination changes, we can synthesize images of the scene presented in Figure 2 (a) captured under different illumination conditions as shown in Figure 2 (b) and (c). As the corresponding pixels have different brightness values, the intensity (or color) of the image cannot be considered as invariants for matching.

In this section, we propose a new method for extracting dense illumination invariant features. It is basically an image transform to convert the original image into a new image, where the new pixel values do not change when the illumination changes. Let $\mathbf{I}_p$ and $\mathbf{I}'_p$ denote the intensity values of pixel p before and after an affine illumination change. We have:

$$\mathbf{I}'_p = \mathcal{A}_p(\mathbf{I}_p) = a_{1,p}\mathbf{I}_p + a_{2,p}, \quad (11)$$

where $a_{1,p}$ and $a_{2,p}$ are two parameters of the affine transform $\mathcal{A}_p$ at pixel p.

Let $\mathbf{H}_p^S$ denote the histogram computed from a window S_p centered at the pixel p, and b_p denote the bin corresponding to the intensity value $\mathbf{I}_p$. According to the definition of

(a) (b) (c) (d) (e) (f)

Figure 2: Illumination invariant features. (a)-(c) are input images with different illuminations. (d)-(f) are the corresponding outputs.

the histogram, the number of pixels in S_p whose intensity value resides in $[b_p - r_p, b_p + r_p]$ is:

$$\mathcal{I}_p = \sum_{b=b_p-r_p}^{b_p+r_p} \mathbf{H}_p^S(b), \tag{12}$$

where parameter r_p controls the interval of integration at pixel p. If r_p scales linear with the illumination so that

$$r'_p = a_{1,p} r_p, \tag{13}$$

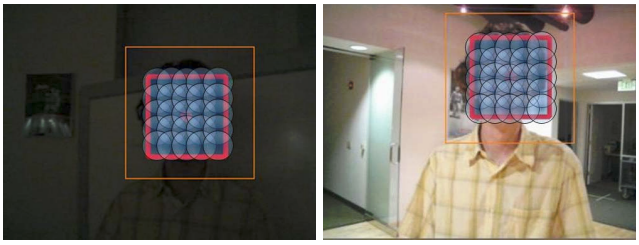
the integrated value $\mathcal{I}'_p$ obtained under a different illumination condition corresponds to the number of pixels with intensity value resides in $[a_p^1 b_p + a_{2,p} - a_{1,p} r_p, a_p^1 b_p + a_{2,p} + a_{1,p} r_p] = [a_{1,p}(b_p - r_p) + a_{2,p}, a_{1,p}(b_p + r_p) + a_{2,p}] = [\mathcal{A}_p(b_p - r_p), \mathcal{A}_p(b_p + r_p)]$. Ignoring the quantization error, $\mathcal{I}'_p$ is equal to $\mathcal{I}_p$. Thus, $\mathcal{I}_p$ is independent of affine illumination changes and can be used as a matching invariant under different illumination conditions as long as Eq. 13 holds. We let:

$$r_p = \kappa |\mathbf{I}_p - \bar{\mathbf{I}}_p|, \tag{14}$$

where $\kappa = 0.1$ is a constant, $\bar{\mathbf{I}}_p = \frac{1}{|S_p|} \sum_{q \in S_p} \mathbf{I}_p$ is the mean intensity value of window S_p and $|S_p|$ is the number of pixels in S_p. With an additional assumption that the affine illumination changes are locally smooth so that the affine transform is the same for all pixels inside window S_p, we have:

$$\begin{aligned} r'_p &= \kappa |\mathbf{I}'_p - \bar{\mathbf{I}}'_p| \\ &= \kappa |a_{1,p}\mathbf{I}_p + a_{2,p} - \frac{1}{|S_p|} \sum_{q \in S_p} (a_{1,p}\mathbf{I}_q + a_{2,p})| \\ &= a_{1,p} \kappa |\mathbf{I}_p - \bar{\mathbf{I}}_p| = a_{1,p} r_p. \end{aligned} \tag{15}$$

As a result, Eq. 13 holds when the interval r_p is obtained adaptively from Eq. 14.



(a) Template. (b) Current frame.

Figure 3: Multi-region tracking. The target template is defined manually in the first frame. In each subsequent frame, the red rectangle indicates the tracked object. The circles show the multiple regions. The orange rectangle indicates the search region for object tracking.

In practice, it is inaccurate to define an exact local window inside which the affine illumination transform remains unchanged. Hence, we replace histogram $\mathbf{H}_p^S$ in Eq. 12 with the proposed locality sensitive histogram $\mathbf{H}_p^E$, which adaptively takes into account the contribution from all image pixels. In addition, we use a "soft" interval to reduce the quantization error. Eq. 12 becomes:

$$\mathcal{I}_p = \sum_{b=1}^{B} \exp\left(-\frac{(b - b_p)^2}{2\max(\kappa, r_p)^2}\right) \cdot \mathbf{H}_p^E(b), \tag{16}$$

where $\bar{\mathbf{I}}_p = \sum_{b=1}^{B} \mathbf{H}_p^E(b) \cdot b$. In practice, $a_{2,p}$ is relatively small; thus r_p can be replaced by $\kappa \mathbf{I}_p$. The invariants computed from Figure 2 (a)-(c) are presented in Figure 2 (d)-(f), respectively. Unlike the intensity values, they remain the same even under dramatic illumination changes, and are used as the input to the tracking algorithm proposed in Section 4.

4. Multi-Region Tracking

The proposed multi-region tracking algorithm aims at capturing the spatial information of a target object, which is missing in single region tracking, to account for appearance change caused by factors such as illumination and occlusion. While it is important to use multiple regions to represent a target objects, it is not necessarily useful to use a large number of them as the incurred computational cost of local analysis and region-by-region matching is prohibitively high. We exploit the proposed locality sensitive histograms for multi-region tracking, since illumination invariant features and region matching scores can be computed efficiently. One application of our method is to adapt the fragment-based method [1] to demonstrate the effect of using a large number of regions for robust object tracking.

4.1. Tracking via Locality Sensitive Histograms

The proposed tracking algorithm represents a target object with multiple overlapping regions, each of which de-

scribes some local configuration. The spatial relationship of these regions remains fixed and is used for region-to-region matching between the template and the potential target object of the current frame. The spacing between regions depends on the size of the target object and the user defined number of regions. Representing the object appearance by hundreds of regions allows the proposed tracker to better handle occlusion and large appearance change. The locality sensitive histograms also allow us to process a large number of regions in real-time.

The regions are weighted based on their locations from each region center. This facilitates more robust matching results when the regions are partially occluded. The fragment-based tracking method approximates the kernel function with weights by computing histograms from three rectangular regions of different sizes. This approximation scheme significantly increases the computational cost. In contrast, a weighting kernel is inherent in the proposed algorithm.

Since object movements are typically non-ballistic, object tracking entails only searches within the nearby area of the current target location. Figure 3 shows an example of the proposed multi-region tracking algorithm, where the blue circles indicate the regions. For the sake of clarity, only a few regions are shown here. The orange rectangle indicates the search region of the current frame.

Based on the location of the target object center in the previous frame, we aim to locate the new center location within the search region. Similar to recent tracking-by-detection methods, exhaustive search within the search region is performed in this work, where every pixel is considered as a candidate target center. Each region at the candidate location is matched correspondingly against the template to produce a score, based on the Earth Mover's Distance (EMD) [18].

Although the conventional EMD is computational expensive, the EMD between two normalized 1D histograms can be computed in time linear in the number of bins [21]. It is equal to the ℓ_1-distance between their cumulative histograms [26]. The proposed locality sensitive histogram is normalized as presented in Section 2. The distance between histograms can be computed as:

$$d(S_1, S_2) = \sum_{b=1}^{B} |C_1(b) - C_2(b)|, \tag{17}$$

where S_1 and S_2 are the corresponding regions of the template (centered at $p1$) and the candidate (centered at $p2$) in the current frame. C_1 and C_2 are the cumulative histograms of $\mathbf{H}^E_{p1}$ and $\mathbf{H}^E_{p2}$. They are defined as: $C(b) = \sum_1^b \mathbf{H}^E_p(b)$.

Once the matching scores of all regions within a candidate region are computed, a vote map is obtained. Similar to the fragment-based tracking method, we use a least-median-squares estimator to accumulate all the votes. This scheme has been shown robust to occlusion, since it assumes that at least one quarter of the target is visible, and the occluded regions are considered as outlier measurements. The final tracking result is the candidate object with the lowest joint score (as the vote map measures the dissimilarity between regions).

4.2. Online Template Update

Visual tracking with a fixed template is not effective over a long period of time as the appearance may have changed significantly. It is also likely to cause jitter and drift as observed in the fragment-based tracking method [1]. To address these issues, we update the region histograms of the template gradually. Taking advantage of using multiple regions, updating a fraction of them in each frame allows the template to adapt to the appearance change and alleviate the tracking drift problem. Once the new target location is determined, the local histograms are updated as follows:

$$H^E_{p1}(\cdot) = H^E_{p2}(\cdot) \quad \text{if } F_1 \cdot M < d(S_1, S_2) < F_2 \cdot M, \tag{18}$$

where M is the median distance of all the regions at the new position. F_1 and F_2 are the forgetting factors, which define the appearance model update rate. In general, the regions with high dissimilarity represent occlusion or the background. The regions with low dissimilarity are perfectly matched. Thus we suggest to update only the regions with medium dissimilarity. In our implementation, we set the forgetting factors F_1 and F_2 as 0.96 and 1.04, respectively. For a target object with 1,000 regions, about 30 of them are updated in each frame.

5. Experiments

This section evaluates the effectiveness and efficiency of the proposed tracking method. We have implemented the proposed method in C using single core and tested it on a PC with an Intel i5 3.3GHz CPU and 8GB RAM. We have evaluated it using 20 video sequences, which contain challenging factors, including drastic illumination changes, pose and scale variation, heavy occlusion, background clutter, and fast motion. We have compared the proposed tracker with 12 state-of-the-art trackers (the implementations provided by the authors were used for fair comparisons). Three of them are multi-region based methods, including the fragment-based tracking method (**Frag**) [1], the articulating block and histogram tracker (**BHT**) [20] and the local-global tracker (**LGT**) [22]. The others are the recent tracking methods, including the real time L1 tracker (**L1T**) [3], the super-pixel tracker (**SPT**) [24], the real-time compressive tracker (**CT**) [27], the multiple instance learning tracker (**MIL**) [2], the structured output tracker (**Struck**) [8], the visual tracking decomposition method (**VTD**) [14], the **TLD** tracker [11], the distribution

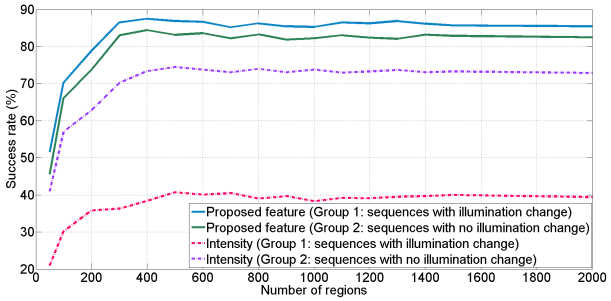


Figure 4: Success rate (%) of the proposed tracker with respect to different numbers of regions.

field tracker (**DFT**) [19] and the multi-task sparse learning tracker (**MTT**) [28]. For the trackers that involve random feature selection, they are evaluated five times and the average scores are considered as the final scores.

5.1. Quantitative Evaluation

Two evaluation criteria are used in our experiments: *center location error* and *tracking success rate*, both computed against manually labeled ground truth. Let $\frac{area(B_T \cap B_G)}{area(B_T \cup B_G)}$ denote the overlap ratio, where B_T and B_G are the bounding boxes of the tracker and of the ground-truth, respectively. When the overlap ratio is larger than 0.5, the tracking result of the current frame is considered as a success.

Figure 4 shows the tracking performance of our method with respect to different numbers of regions. The curves show the average success rates of 20 video sequences. We observe that the tracking performance of our method reaches its peak when the number of regions reaches 400, thus we use 400 regions in the following experiments. To compare the effectiveness of the proposed feature with the popular feature using intensity, we divide the 20 sequences into two groups. Group 1 contains 6 sequences with illumination change and group 2 includes all remaining 14 sequences with other challenging factors. We then test the tracker on the two groups of video sequences to evaluate the performance of using our feature and using intensity. Figure 4 shows that the proposed feature outperforms intensity not only on sequences with illumination change, but also on sequences without illumination change. This indicates that our proposed feature is more effective than intensity.

Table 1 and Table 2 show the tracking performance and the speed (in frame rate) of our method with the 12 other methods. We note that the **TLD** tracker does not report tracking result (or bounding box) when the drift problem occurs and the target object is re-detected. Thus we only report the center location errors for the sequences that the **TLD** method does not lose track of target objects. The proposed tracker performs favorably against the state-of-the-art algorithms as it achieve the best or the second best performance in most of sequences using both evaluation criteria. In addition, our tracker is very efficient which runs at 25.6 frames per second (FPS). Figure 5 shows some tracking results of different trackers. For presentation clarity, we only show the results by the trackers with the top average success rate, i.e., **CT** [27], **MIL** [2], **TLD** [11], **Struck** [8], **DFT** [19] and **MTT** [28].

5.2. Qualitative Evaluation

The 20 sequences that we use are placed at our project website[1], and in the supplementary material for reference. We qualitatively evaluate the tracking results of these 20 sequences in four different ways as follows.

Illumination, pose and scale variation. We evaluate sequences with different kind of illumination variations. In the *Man*, *Car* and *Crowds* sequences, the object appearances change drastically due to cast shadows and ambient lights. Only the **L1T**, **Struck** and the proposed trackers are able to adapt the illumination variation well. The *David Indoor* and *Trellis* sequences contain gradual illumination changes and pose variation. We note that in most of the reported results using the both sequences, only a subset of frames are used (i.e., not from the very beginning of the *David* sequence when the face is in complete dark, or until the very end of the sequence of the *Trellis* sequence where there are both sudden pose and lighting variations). We use the full sequences for better assessment of all tracking algorithms. In these two sequences, only the proposed algorithm is able to track the targets successfully in most of the frames. This can be attributed to the use of the proposed features which are insensitive to illumination variation. Likewise, most of the other trackers do not perform well in the *Shaking* sequence since the object appearance changes drastically due to the stage light and sudden pose change. In addition, the proposed tracker performs well in the *Basketball* and *Bolt* sequences where the target objects undergo large pose variation.

Occlusion and drift. The target objects are partially occluded in the *Bird*, *Occluded face 2* and *Woman* sequences. For the *Woman* sequence, the target object enclose the whole body instead of only upper body used in the fragment-based tracking method [1]. Most tracking methods do not perform well when the objects are heavily occluded. By exploiting a large number of regions, the relative spatial information between regions is maintained and thus occlusion is handled well by the proposed algorithm. Tracking drift usually occurs when the target object is heavily occluded. Few trackers recover from tracking drift since these methods focus on learning the appearance change. The *Box* sequence is challenging as the target is heavily occluded in several frames. Only the proposed algorithm and

[1] http://www.cs.cityu.edu.hk/~shengfehe2/visual-tracking-via-locality-sensitive-histograms.html

Sequence	Ours	LIT [3]	SPT [24]	CT [27]	Frag [1]	MIL [2]	Struck [8]	VTD [14]	TLD [11]	BHT [20]	LGT [22]	DFT [19]	MTT [28]
Basketball	**9.3**	14.9	20.0	62.9	25.2	102	165	**8.6**	– –	153	15.2	239	287
Biker	**8.9**	40.4	52.9	26.0	79.8	31.0	**12.7**	69.4	– –	30.1	85.5	46	76.1
Bird	**8.3**	57.8	14.1	17.1	26.6	17.2	23.9	61.9	– –	14.7	15.6	**10.9**	73.2
Board	**10.9**	174	52.6	34.7	25.2	35.0	34.6	**17.8**	– –	42.4	36.6	145	50.3
Bolt	**6.4**	186	72.0	9.0	73.2	**8.1**	8.4	17.7	– –	149	48.6	50.9	9.0
Box	**13.1**	77.5	169	107	65.8	109	**10.6**	114.1	– –	111	68.9	120	54.8
Car	**3.9**	36.8	**4.5**	38.3	40.2	37.9	6.4	35.3	9.4	156	60.9	39.0	34.1
Coupon	**5.4**	69.7	6.1	17.1	35.7	18.6	5.7	65.3	7.9	45.7	21.9	5.5	**4.5**
Crowds	5.9	15.3	433	351	435	465	**5.0**	380	– –	344	232	**4.5**	226
David indoor	**10.6**	28.5	29.8	25.8	103	44.1	30.5	64.6	**10.4**	122	13.4	28.6	10.7
Dragon baby	**20.0**	72.8	49.7	**30.8**	51.1	48.6	59.1	42	– –	83.1	87	148	78.9
Man	**2.1**	2.8	23.1	9.4	48.6	37.7	**2.4**	20.7	4.4	73.0	24.1	40.0	2.6
Motor rolling	**13.1**	187	**76.2**	198	110	178	84	148	– –	137	178	170	180
Occluded face 2	**4.0**	20.2	37.8	**13.2**	15.5	15.3	16.4	15.9	11.5	44.5	30.9	23.8	16.8
Shaking	13.8	42.0	144	33.4	195	**12.9**	43.2	24.0	– –	237.9	35.7	**8.7**	52.2
Surfer	**7.4**	122	78.2	78.6	150	128	14.6	93.3	**5.4**	93.3	188.1	143	93.1
Sylvester	**6.1**	23.4	47.8	9.1	14.6	12.6	6.7	9.7	13.7	11.1	23.9	39	**5.1**
Tiger2	**8.5**	46.3	85.7	11.9	45.7	**7.7**	9.8	32.9	– –	66.8	27.9	33.6	25.5
Trellis	**8.3**	**15.5**	18.8	67.3	100	65.8	22.9	32.4	– –	75.4	16.1	51.3	56.9
Woman	**6.4**	136	8.9	130	7.3	144	**5.5**	136	– –	71.9	19.3	8.6	154
Average	**8.6**	68.4	71.2	62.5	82.3	76.5	**28.4**	69.5	– –	103	61.5	67.8	74.5

Table 1: The average center location errors (in pixels) of the 20 sequences. The best and the the second best performing methods are shown in red color and blue color, respectively. The total number of frames is 10,918. The entry '– –' for **TLD** indicates that the value is not available as the algorithm loses track of the target object.

Sequence	Ours	LIT [3]	SPT [24]	CT [27]	Frag [1]	MIL [2]	Struck [8]	VTD [14]	TLD [11]	BHT [20]	LGT [22]	DFT [19]	MTT [28]
Basketball	**87**	75	84	32	78	27	2	**96**	1	20	44	3	3
Biker	**69**	23	44	34	14	40	**49**	45	38	46	7	46	44
Bird	**98**	44	74	53	48	58	48	13	12	71	5	**91**	13
Board	**93**	3	47	73	**82**	76	71	81	16	38	5	23	63
Bolt	**81**	18	8	66	15	73	**76**	26	3	6	2	8	68
Box	**84**	4	8	33	42	18	**90**	34	60	8	9	37	25
Car	**92**	43	**73**	43	40	38	59	44	58	10	11	43	49
Coupon	**100**	24	98	58	67	77	**100**	38	98	58	12	**100**	**100**
Crowds	76	59	7	9	2	4	**82**	8	16	4	3	**85**	9
David indoor	**93**	41	64	46	35	24	67	32	90	7	24	45	**92**
Dragon baby	**67**	16	28	30	35	38	**43**	33	15	28	4	23	24
Man	**100**	98	41	60	21	21	**100**	31	98	18	8	22	**100**
Motor rolling	**83**	5	28	11	24	9	11	6	14	**30**	1	10	5
Occluded face 2	**100**	60	22	**100**	80	94	79	77	76	43	8	49	82
Shaking	73	12	3	**81**	8	45	9	77	1	2	23	**95**	2
Surfer	**75**	1	3	3	2	2	67	2	**86**	2	1	3	3
Sylvester	**89**	48	34	74	66	70	87	72	76	78	8	54	**96**
Tiger2	**66**	10	3	65	5	**77**	65	17	26	5	2	21	27
Trellis	**91**	67	**72**	35	18	34	70	54	31	18	2	45	34
Woman	**83**	8	80	6	71	6	**87**	5	30	34	4	80	8
Average	**85.0**	33.0	41.1	45.6	37.7	41.6	**63.1**	39.6	42.3	26.3	9.2	44.2	42.4
Average FPS	**25.6**	10.2	0.2	**34.8**	3.5	11.3	13.5	0.5	9.5	3.3	2.8	5.8	1.0

Table 2: The success rates (%) and the average frames per second (FPS) of the 20 sequences. The best and the second best performing methods are shown in red color and blue color, respectively. The total number of frames is 10,918.

the **Struck** method are able to relocate the target after heavy occlusion. As only some regions are updated at any time instance by the proposed method, the tracking drift problem can be better handled where heavy occlusion occurs.

Out of plane rotation and abrupt motion. The target objects in the *Biker* and *Surfer*[2] sequences undergo large out of plane rotation with abrupt movement. Most the trackers except the proposed algorithm and the **Struck** method do not perform well in these sequences. The *Dragon baby* sequence is downloaded from Youtube where the baby moves abruptly in action scenes. The proposed algorithm is able to track the baby well despite all the abrupt movement and out of plane rotation. The motorbiker performs acrobatic movement with 360 degree rotation in the *Motor rolling* sequence. While the proposed algorithm tracks the target object throughout the sequence, the other methods do not perform well.

Background clutters. In the *Tiger2* and *Board* sequences, the target objects undergo fast movement in cluttered backgrounds. The **MIL** tracker and the proposed algorithm perform well whereas the others fail to locate the target objects.

6. Conclusion

In this paper, we propose a novel locality sensitive histogram method and a simple yet effective tracking framework. Experimental results show that the proposed multi-

[2]Since we do not consider object scale here, only part of the sequences are used. The scale approach in [1] can also be used in our method.

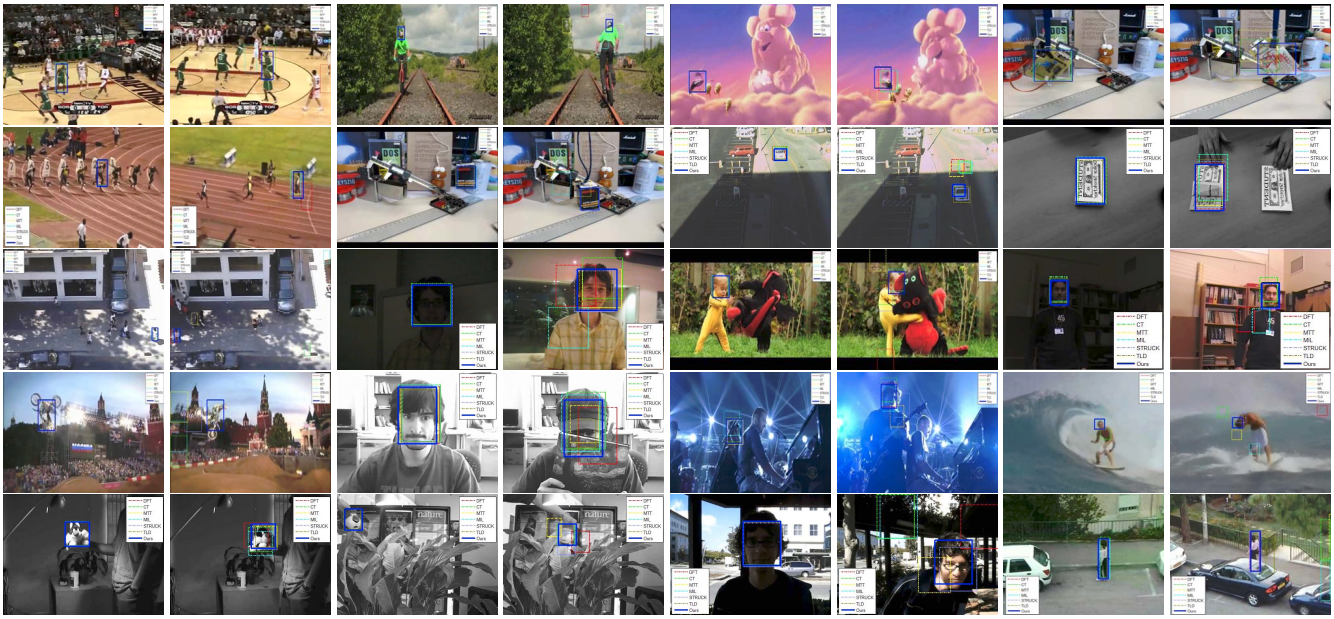


Figure 5: Screenshots of the visual tracking results. The figures are placed in the same order as Table 2. Red, green, yellow, azure, purple, olive and blue rectangles correspond to results from **DFT**, **CT**, **MTT**, **MIL**, **Struck**, **TLD** and **ours**.

region tracking algorithm performs favorably against numerous state-of-the-art algorithms. The proposed locality sensitive histograms can also be applied to other vision applications including efficient and effective bilateral filtering, which will be our future work.

Acknowledgements: This work was partially supported by two GRF grants from the Research Grants Council of Hong Kong (CityU 122212 and CityU 116010). M.-H Yang is supported in part by the NSF CAREER Grant #1149783 and NSF IIS Grant #1152576.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, pages 798 – 805, 2006.

[2] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *PAMI*, 33(8):1619 –1632, aug. 2011.

[3] C. Bao, Y. Wu, H. Ling, and H. Ji. Real time robust l1 tracker using accelerated proximal gradient approach. In *CVPR*, pages 1830 –1837, 2012.

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

[5] H. F. Chen, P. N. Belhumeur, and D. W. Jacobs. In search of illumination invariants. In *CVPR*, pages 1254–1261, 2000.

[6] F. Crow. Summed-area tables for texture mapping. In *SIGGRAPH*, pages 207–212, 1984.

[7] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, pages 234–247, 2008.

[8] S. Hare, A. Saffari, and P. Torr. Struck: Structured output tracking with kernels. In *ICCV*, pages 263 –270, 2011.

[9] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *IJCV*, 29:5–28, 1998.

[10] D. Jacobs, P. Belhumeur, and R. Basri. Comparing images under variable illumination. In *CVPR*, pages 610–617, 1998.

[11] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *PAMI*, 34:1409–1422, 2012.

[12] M. Kass and J. Solomon. Smoothed local histogram filters. *ACM TOG*, 29:100:1–100:10, 2010.

[13] J. Kwon and K. M. Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *CVPR*, pages 1208–1215, 2009.

[14] J. Kwon and K. M. Lee. Visual tracking decomposition. In *CVPR*, pages 1269–1276, 2010.

[15] F. Porikli. Integral histogram: a fast way to extract histograms in cartesian spaces. In *CVPR*, pages 829–836, 2005.

[16] F. Porikli. Constant time o(1) bilateral filtering. In *CVPR*, 2008.

[17] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1-3):125–141, 2008.

[18] Y. Rubner, C. Tomasi, and L. Guibas. The earth mover's distance as a metric for image retrieval. *IJCV*, 40(2):99–121, 2000.

[19] L. Sevilla-Lara and E. Learned-Miller. Distribution fields for tracking. In *CVPR*, pages 1910–1917, 2012.

[20] S. M. N. Shahed, J. Ho, and M.-H. Yang. Visual tracking with histograms and articulating blocks. In *CVPR*, 2008.

[21] S. Shirdhonkar and D. Jacobs. Approximate earth mover's distance in linear time. In *CVPR*, 2008.

[22] L. Čehovin, M. Kristan, and A. Leonardis. An adaptive coupled-layer visual model for robust visual tracking. In *ICCV*, 2011.

[23] P. Viola and M. Jones. Robust real-time object detection. *IJCV*, 57(2):137–154, 2004.

[24] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *ICCV*, pages 1323–1330, 2011.

[25] B. Weiss. Fast median and bilateral filtering. *ACM TOG*, 25(3):519–526, 2006.

[26] M. Werman, S. Peleg, and A. Rosenfeld. A distance metric for multidimensional histograms. *CVGIP*, 32(3):328 – 336, 1985.

[27] K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *ECCV*, 2012.

[28] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via multi-task sparse learning. In *CVPR*, pages 2042 –2049, 2012.