Real-time Visual Tracking Based on Improved Perceptual Hashing

Mengjuan Fei ¹, Zhaojie Ju ², Xiantong Zhen ³, Jing Li ^{4*}

¹ Institute of Cyber-Systems and control, Zhejiang University, HangZhou, 310027, China

² Intelligent Systems and Biomedical Robotics Group, School of Computing, University of Portsmouth, Portsmouth, PO1 3HE, U.K.

³ The University of Western Ontario, London, ON, Canada.

⁴ Laboratory of Ubiquitous Vision Perception and Intelligent Computing, School of Information Engineering, Nanchang University, Nanchang, 330031, China

* Corresponding Author Email: jingli@ncu.edu.cn Phone Number: +86 (0) 791 83969675

Abstract

Video object tracking represents a very important computer vision domain. In this paper, a perceptual hashing based template-matching method for object tracking is proposed to efficiently track objects in challenging video sequences. In the tracking process, we first apply three existing basic perceptual hashing techniques to visual tracking, namely average hash (aHash), perceptive hash (pHash) and difference hash (dHash). Compared with previous tracking methods such as mean-shift or compressive tracking (CT), perceptual hashing-based tracking outperforms in terms of efficiency and accuracy. In order to further improve the accuracy of object localization and the robustness of tracking, we propose Laplace-based Hash (LHash) and Laplace-based Difference Hash (LDHash). By qualitative and quantitative comparison with some representative tracking algorithms, experimental results show that our improved perceptual hashing-based tracking algorithms perform favorably against the state-of-the-art algorithms under various challenging environments in terms of time cost, accuracy and robustness. Since our improved perceptual hashing can be a compact and efficient representation of objects, it can be further applied to fusing with depth information for more robust RGB-D video tracking.

Keywords: Visual tracking; Perceptual hashing; AHash; PHash; DHash

1. Introduction

Visual tracking aims at generating the movement trajectories of an object by calculating the moving object's state information such as position, speed and its corresponding and matching relationships between frames. As an important topic in computer vision, not only does visual tracking provide reliable data for motion analysis and scene understanding, but it also underpins a wide range of applications, including military navigation guidance [1,2], autonomous driving [3,4], activity analysis [5,6,7], human-computer interfaces [8,9] and intelligent

supervision [10,11]. Recently, researchers have proposed numerous algorithms to this fundamental subject in the literature [12,13,14,15,16]. On the whole, most algorithms fall into three main categories which are kernel tracking, point tracking and silhouette tracking [17].

In the kernel tracking category, kernel refers to the object shape and appearance, while tracking is based on modeling the kernel's content by a rectangular template or a density-based appearance model with an associated histogram. Point tracking detects point-represented objects in consecutive frames and finds feature correspondences of the same objects among frames. Silhouette-based methods provide an accurate shape description for those complex shape objects such as head, hands and shoulders.

Ideally, we hope to address the object tracking problem in real-world scenarios rather than in a lab environment, while the real-world scenarios typically include appearance changes and diversity of object motion. That is, the key problem with visual tracking is how to represent the appearance of an object in a robust way and an effective appearance model that can deal with these visual changes is always of prime importance for the success of tracking. Point tracking and silhouette tracking aim at finding object regions in each frame based on an object model which can be generated in the form of a color histogram, object contours or object edges using previous frames, whereas they sometimes cannot achieve good tracking results. At this point, kernel tracking represents state-of-the-art tracking performance.

Numerous researches in kernel tracking have attempted to deal with all challenges simultaneously. Black and Jepson [18] learned an off-line subspace model to represent the object for tracking, where tracking a previously viewed object in an image sequence is deemed as the detection of object changes due to its motion or the camera movement. Kwon et al. [19] presented a robust tracking algorithm that can track an object whose appearance and motion change drastically. In their algorithm, for the efficient design of observation and motion models as well as trackers, they proposed a visual tracking decomposition scheme in which the observation model is decomposed into multiple basic observation models. These models are constructed by sparse principal component analysis (SPCA) of a set of feature templates. Karavasili and Nikou [20] proposed a visual tracking method using the differential Earth Mover's Distance (EMD) between Gaussian mixtures models (GMM) and Kalman filtering. They first model the appearance of the target by a GMM and then employ the DEMD for tracking. Moreover, in order to address occlusion, they estimate the location of the target as an observation of a time varying Kalman filter. Their results showed significant improvement in tracking performance in the presence of occlusion. Recently, target templates have been expressed based on sparse representation and used in

the l_1 tracker [21]. The l_1 tracker demonstrates promising robustness compared with many existing trackers, although its computational complexity is rather high. Inspired by the success of the l_1 tracker and motivated by the high-computational complexity problem, Li et al. [22] proposed real-time compressive sensing (CS) tracking by exploiting the signal recovery power of compressive sensing. They further extended the l_1 tracker by using the orthogonal matching pursuit algorithm to accelerate the CS tracking.

Recent tracking algorithms proposed in the literature perform well in terms of accuracy, such as compressive sensing for tracking [23], visual tracking based on learning compact binary codes [24,25], sparse representation [26,27], SVM classifier [28], and semi-supervised learning [29,30]. However, the above-mentioned algorithms demand high-computational costs owing to the complexity of their appearance models.

Template matching as a kind of kernel tracking has been proved to be one of the most efficient and simplest approaches in object tracking. However, traditional template matching based on color histogram and complex image features is computationally expensive and of weak real-time performance.

In recent years, perceptual hashing [31,32,33] has been well known for its ability to map an image's content to a short digest, regardless of data formats and manipulations it suffers. It has been applied to a wide area of applications, for example, it is a promising solution for multimedia content authentication and usually used to measure the similarity of two images in image retrieval. In the last decade, many researchers have devoted themselves to developing perceptual hashing algorithms. Yang [32] used the mean of image blocks to obtain a perceptual hash code. Jie [34] proposed a block-DCT and PCA-based image perceptual hashing technique and explored the algorithm in tamper detection. In [35], Laradji and Lahouari presented a new perceptual image hashing approach that exploits the image color information using hypercomplex and QFT representations. Simulation results clearly indicated the superior retrieval performance of the proposed QFT-based perceptual hashing technique. Although perceptual hashing has been used in various applications, there is no research applying perceptual hashing to visual tracking.

In the light of above observations, we propose a simple and fast perceptual hashing algorithm which builds a robust image representation for template matching based object tracking. In this paper, we first apply three existing perceptual hashing methods, namely average hash (aHash), perceptive hash (pHash), and difference hash (dHash), to respectively build three appearance models for tracking. The proposed tracking algorithms can run in real time and outperform some state-of-the-art trackers on challenging video sequences. To deal with even

more complicated scenarios, we further propose two improved perceptual hashing methods (namely Laplace-based hash and Laplace-based difference hash) and apply them to object tracking. Consequently, these two improved algorithms perform favorably against the three existing perceptual hashing methods on most video sequences in terms of accuracy and robustness. Compared with existing template matching based on color, textures, HOG et al., our proposed tracking approach based on image perceptual hash can track objects fast and accurately.

The remainder of this paper is organized as follows. Section 2 describes three existing perceptual hashing algorithms and two proposed perceptual hashing algorithms. In Section 3, the proposed perceptual hashing based tracking algorithm is presented. Experimental results are shown in Section 4. We conclude the paper and point out directions for future work in Section 5.

2. Perceptual Hashing Algorithms

A key component of our proposed perceptual hashing-based tracking algorithms is to obtain the perceptual hash codes. A common framework of these algorithms includes the following four stages:

(1) Preprocessing. Each time, we take an image as input. In order to obtain the main features of the input image such as brightness, it is resized to $M \times M$ or $(M+1) \times M$ via resampling and bi-cubic interpolation to make the obtained hash codes resistant to images with various resolutions. Afterwards, the image is converted to a grayscale one;

(2) Feature extraction. Features are extracted from the preprocessed image based on its perceptual content, where the most common ones include low-frequency discrete cosine transform (DCT), discrete wavelet transform (DWT) coefficients, Fourier-Mellin domain, color histogram and feature points [36];

(3) Quantization. Since each column of perceptual features is an indicator that reflects information of the input image, quantization is directly based on the coefficients or spatial features. By uniform quantization, a continuous perceptual hashing vector is discretized into an intermediate perceptual hash vector containing $M \times M$ elements;

(4) Hash computation. To compute perceptual hash codes, we binarize each component of the obtained perceptual hash vector with a threshold value T .

In this section, we first describe three most commonly used perceptual hashing algorithms, i.e., average hash (AHash), perceptive hash (pHash) [37], and difference hash (dHash). After that, we introduce the newly proposed perceptual hashing algorithms in detail, namely the Laplace-based hash (LHash) and Laplace-based

difference hash (LDHash).

2.1 Average Hash (aHash)

An image is a two-dimensional signal with different frequency components, where low-frequency components describe a wide range of structural information and high-frequency components describe specific details such as edges and texture.

Average hash (aHash) is one of the simplest and fastest perceptual hashing techniques. It mainly uses low-frequency information of images by the following procedure:

- (1) Resize the input image to 8×8 by bi-cubic interpolation without keeping the aspect ratio;
- (2) Convert the resized color image into a grayscale one and calculate its mean value M ;
- (3) Normalize each pixel value $A(i)$ into a binary form by comparing it with M :

$$h(i) = \begin{cases} 0 & \text{if } A(i) \leq M \\ 1 & \text{otherwise} \end{cases} \quad (1 \leq i \leq 64); \quad (1)$$

- (4) Combine the binarization results into a 64-bit binary code.

2.2 Perceptive Hash (pHash)

Although aHash is simple and fast, it may be too rigid by using the mean value. For example, if we apply a gamma correction or color histogram equalization to the image, the mean value is influenced and the final hash code is changed. Perceptive hash (pHash) is a more robust algorithm which extends the average method by using a discrete cosine transform (DCT) [38] to obtain the most sensitive information of the human vision system [37].

The DCT is widely used in signal and image processing, especially for lossy data compression for reducing redundancy and correlation in general images since it has a strong "energy compaction" property [39,40]. It transforms images from the pixel domain to the frequency domain by separating an image into a collection of frequencies and scalars. In such a way, most information of the image tends to be concentrated into a few low-frequency components of the DCT and only the coefficient of some low-frequency components is not 0.

The implementation of pHash includes the following steps:

- (1) Resize the image to 8×8 and convert it to the grayscale level;
- (2) Perform a 8×8 DCT on the grayscale image and obtain a 8×8 DCT coefficient matrix, where the energy of the image will be gathered into a few low-frequency DCT coefficients;
- (3) As proposed in [41], 64 low-frequency DCT coefficients are selected for hash extraction after discarding

high frequency coefficients;

(4) Calculate the mean value of the DCT coefficients:

$$m = \frac{1}{64} \sum_{r=1}^8 \sum_{c=1}^8 d_{r,c}, \quad (5)$$

where $d_{r,c}$ indicates the DCT coefficients;

(5) Compute the hash code. The input image's pHash is obtained via thresholding these DCT coefficients with respect to their mean value. The 2D coefficient matrix can be flattened to form a 1D-array of 64 elements.

Let the DCT coefficients of the array be denoted as C_i , $i = 0, K, 63$, then the perceptual hash code can be obtained via binarization:

$$h_i = \begin{cases} 0 & C_i < m \\ 1 & C_i \geq m \end{cases}, \quad i = 0, K, 63. \quad (6)$$

where h_i is the bit of the perceptual hash at position i and m is the mean value of the DCT coefficients;

(7) Convert the 64 bits into a 64-bit integer. Finally, a perceptual hash code, also named the fingerprint of the image, is obtained.

2.3 Difference Hash (dHash)

Difference Hash (dHash) works on the difference between two adjacent pixels. As aHash, it is very simple to implement but is far more accurate in tracking relative gradient directions. The dHash algorithm is nearly identical to aHash and here is how it works:

(1) Resize the input image to 8×8 irrespective of all different image alterations. Like aHash and pHash, this step also removes high frequencies and details of the image;

(2) Convert the image to the grayscale level and calculate the difference between adjacent pixels as follows:

$$dpx_{r,c} = px_{r,c+1} - px_{r,c}, \quad (7)$$

where $px_{r,c}$ is the gray value of the resized image at r -th row and c -th column, and $dpx_{r,c}$ is the difference value. In this way, the pixels per row yield 8 differences between adjacent pixels and 8 differences of 8 rows become an 8×8 image. Then, all the rows are concatenated to form a 1D-array of 64 bits;

(3) Compute the hash code and assign bits. Each bit is set to 0 or 1 based on whether the right pixel is brighter than the left pixel.

2.4 Proposed Laplace-based Hash (LHash)

The Laplace transform is a kind of integral transform in applied mathematics and widely used in image processing such as image sharpening and edge detection. In visual tracking, some objects in video sequences move very fast and motion blur is inevitable. The essence of image blurring is performing a low-pass filtering operation on images, and therefore we can perform an inverse operation on images, such as difference operation, to highlight the image details. Since the Laplace transform is a kind of differential operator, applying it can enhance edge regions of the image and weaken slowly varying gray areas. So in this paper, in order to solve the problem of image blur and track objects more accurately, we propose to apply the Laplace transform to generate perceptual hash codes, which is named the Laplace-based Hash (LHash). Furthermore, based on the proposed LHash, we find that the Laplace transform is an image sharpening operator which is suitable for edge detection. Several existing perceptual image hashing algorithms use edge detectors for feature extraction in identification and authentication of multimedia content [42]. In their researches, simulation results show that edge detection-based perceptual hash codes have strong robustness and good discriminability. To this end, we apply LHash to object tracking.

2.4.1 Definition of Laplace Transform

The discrete Laplacian is defined as the sum of the second derivatives of the Laplace operator. For two dimensions, it is given by

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}, \quad (8)$$

where $f(x, y)$ is the input image and $\nabla^2 f(x, y)$ indicates the sum of the second-order partial derivatives on the x-axis and y-axis of each pixel in the image.

In the discrete form, the most common Laplace operator is given by

$$\nabla^2 f(x, y) = f(x+1, y) + f(x, y-1) + f(x, y+1) - 4f(x, y). \quad (9)$$

For two-dimensional image signals, the discrete Laplacian can be given as the convolution with the following kernel:

$$D_{xy}^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \quad (10)$$

2.4.2 Algorithm Implementation

Figure 1: Flowchart of the proposed LHash.

This specific process is described as follows:

- (1) Convert the input image to the grayscale level and resize it to 8×8 ;
- (2) Perform the Laplace transform on the resized image. This step can enhance the grayscale contrast in order to improve the robustness and accuracy of perceptual hash codes;
- (3) Calculate the mean value of the image obtained in Step (2) and put it as a threshold for binarization;
- (4) Construct the hash code to 64 bits.

2.5 Proposed Laplace-based Difference Hash (LDHash)

Since an edge is considered as an abrupt grey-level change and an edge detector describes each pixel's edge attributes, most existing perceptual image hash functions use edge detectors for feature extraction (e.g. [44]). In this case, the function of gradient in edge detection can be taken into account. To this end, we propose Laplace-based difference hash (LDHash) for object tracking by combining gradient and Laplacian together in edge detection.

For a one-dimensional array of image pixels, a one-dimensional gradient image can be obtained by calculating the difference between adjacent pixels. The gradient image $f_d(x, y)$ represents the relative changes in brightness and intensity of the input image $f(x, y)$ and identifies the relative gradient directions of the input image. Since Laplacian locates the positions where the zero-crossings of $f_d''(x, y)$ occur and only occur at the edge points of $f_d(x, y)$, we perform the Laplace transform on $f_d(x, y)$ and obtain the edge

image $f_{dLap}(x, y)$.

The implementation process of the algorithm is described in Figure 2.

Figure 2: Flow diagram of the proposed LDHash.

3. Object Tracking based on Perceptual Hashing

In this section, we apply three basic perceptual hashing methods (i.e., aHash, pHash, dHash) and our proposed algorithms (i.e., LHash and LDHash) to object tracking. In this paper, object tracking is achieved by minimizing the similarity between the tracked object region of the current frame and the search window in the next frame. Since we use Hamming distance to measure the similarity between a pair of image perceptual hash codes, we briefly introduce the Hamming distance before describing the experiments.

3.1 Hamming Distance

In information theory, the Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In our methods, the Hamming distance is used to measure the similarity between a pair of image perceptual hash codes h_1 and h_2 , which is defined as

$$d(h_1, h_2) = \sum_{i=1}^L [h_1(i) - h_2(i)]^2, \quad (11)$$

where $h_1(i)$ and $h_2(i)$ are the i^{th} elements of h_1 and h_2 , respectively. The smaller the Hamming distance, the more similar the two images of the input perceptual hash codes. A Hamming distance of 0 indicates that two images are very similar. If the Hamming distance is smaller than a pre-defined threshold T_d , the two images are

deemed to be visually identical. Otherwise, they are different. In this paper, our goal is to find the minimum Hamming distance between the tracked object region of the current frame and the search window in the next frame.

3.2 Tracking

For each video sequence, we select the tracking target in the first frame and aim to find the most similar location to the target in remaining frames. After saving the tracked target, we scan the whole incoming image frame and compute perceptual hashing for each scan window to measure the similarity between it and the target via the Hamming distance. The scan window with the smallest Hamming distance will be most similar to the target, and it is also the place at which the target locates in an image frame. The implementation process is described as follows:

(1) Capture the first frame of a video sequence and use mouse to select the target regions $\mathcal{R}_O$ with a rectangle searching box of size $A \times B$, the size of searching box depends on the size of object;

(2) Capture the next frame of the video sequence and select the image area around the target in the previous frame as scanned regions $\mathcal{R}_s$ to speed up the process;

(3) Search the defined scanned regions $\mathcal{R}_s$ with the rectangle scan window $\mathcal{R}_w$ of size $A \times B$ and separately calculate the perceptual hash codes H_0 of the target region and H_w of search windows. After computing the Hamming distance $d(H_0, H_w)$ between H_0 and each H_w , all obtained Hamming distances in this scanned region are compared and the position $r_w(x, y)$ of scan window with the smallest Hamming distance is selected;

(4) Update the object rectangle region position $r_0(x, y)$ as $r_w(x, y)$ and mark the object in order to adapt to variations of the targets;

(5) End the cycle if the video sequence is finished, otherwise go to step (2) to capture the next frame and track objects.

We define the length of video sequences as L and the scanning step number as n , the whole algorithm is presented in Table 1. The algorithm is appropriate for three basic perceptual hashing tracking methods and our proposed LHash and LDHash tracking algorithms. When performing different tracking methods, we only need to change the step of obtaining hash codes.

Table 1: Perceptual Hashing based Tracking Algorithm.

```

Initialize:  $L, r_0, r_s, r_w, n, i = 1, j = 1$ 
1: while:  $i < L$  do
2:     while  $j < n$  do
3:          $d(H_0, H_w) = \text{Hamming}(H_0, H_w)$  ;
4:          $j = j + 1$ ;
5:     end
6:      $r_w(x, y) = \arg \min d(H_0, H_w)$ ;
7:      $r_0(x, y) = r_w(x, y)$ ;
8:      $H_0 = H_w$ ;
9: end while

```

4. Experimental Results

4.1 Experimental Environment

In this section, we conduct tracking experiments with three existing perceptual hashing algorithms and our newly proposed perceptual hashing methods. Also, we perform a comparison with mean-shift [45] which is one of the most common tracking methods. All experiments were conducted on a personal computer with an Intel (R) Pentium (R) 2 GHz processor with 2 GB RAM under Microsoft Visual Studio 2010 combined with Open Source Computer Vision (OpenCV) [46]. In the experiments, eight publicly available video sequences are used for evaluating the tracking performance, which are “Animal”, “Singer1”, “Soccer”, “Shaking”, “Faceocc2”, “David”, “Sylv”, “Lemming”. These video sequences were captured in different scenarios and contain diverse challenges such as occlusion, object pose variation, background distraction, lighting changes and out-of-plane rotation. The “Animal”, “Singer1”, “Soccer”, “Shaking” sequences are provided in [43]; the “Faceocc2”, “David”, “Sylv” sequences are from [47] and the “Lemming” sequence is from [48]. It is worth noting that we conducted experiments fully on three benchmark databases [43,47,48], whereas the eight test sequences we selected for evaluation are the most challenging sequences from existing works [43,47]. Using a sliding window-search scheme, our proposed object tracking methods perform object localization based on perceptual hashing. Like the template matching method for tracking, we perform object tracking by computing the similarity between object regions of the previous frame and the searching window in the current frame. In the experiments, we first select a

tracking object area with the mouse. Then, the object image area is converted into an image hash code and the hamming distance between consequence frames is calculated. The searching box size is determined by tracked objects. And we resize it to 8×8 in the perceptual image hash algorithm.

For quantitative performance comparison, we introduce two popular tracking accuracy measurements termed as tracking success rate (TSR) and center location error (CLE) to evaluate the proposed tracking algorithms, compressive tracking (CT) and the mean-shift tracker. Given the tracking window ROI_t and the ground truth bounding box ROI_g , TSR is defined as

$$TSR = \frac{area(ROI_t \cap ROI_g)}{area(ROI_g)} \in [0, 1], \quad (12)$$

and CLE denotes the Euclidean distance between two centroids, given by

$$\rho = \|c_g - c_t\|^2. \quad (13)$$

4.2 Qualitative Comparison of All Tracking Algorithms

In this section, we present a qualitative comparison of five perceptual hashing-based tracking methods, the mean-shift algorithm [45] and compressive tracking (CT) [23] over some representative frames of eight video sequences. Based on the qualitative tracking results, we demonstrate the effectiveness and low-computational cost of perceptual hashing-based tracking algorithms, especially our proposed LHash and LDHash.

Figure 3: Qualitative tracking results of the five perceptual hashing-based tracking algorithms and the mean-shift tracker over some representative frames of eight video sequences. From up to down, these eight video sequences are “David”, “Sylv”, “Faceocc2”, “Lemming”, “Singer1”, “Soccer”, “Animal”, “Shaking”, respectively.

The “Singer1” video sequence entails the challenges of illumination variation, complex background and target size changes. Owing to the view and focal length changes of the camera, the point of view and size of the

tracked singer changes, and illumination variation occurs drastically. From the experimental results, we can see that at the 10th frame, the tracked singer begins to be lost by the mean-shift tracker; as illumination severely changes at the 98th frame, the singer is absolutely lost by the mean-shift tracker. Throughout the entire video sequence, our proposed perceptual hashing-based trackers present outstanding performance compared with the mean-shift tracker, although there is significant illumination change, complex background in the scene and varying size of the target. Furthermore, by comparing the tracking results of five perceptual hashing methods, we can find that our proposed LHash and LDHash can track objects more accurately than the other three existing perceptual hashing algorithms. For example, at the 160th frame and the 264th frame, the tracking windows of three existing perceptual hash codes appear to shift slightly relative to the target location, but the tracking windows of LHash and LDHash almost locate at the center of the singer.

Some video sequences encounter challenges of illumination variation, complex background and target size changes. The “Faceocc2” video sequence contains heavy occlusions and pose variations. At the 271th frame, the mean-shift tracking window is away from the target face due to occlusion by a book. However, the perceptual hashing-based trackers are still able to keep track of the target face. At the 720th frame, the target face is completely occluded by a book and a hat, but these perceptual hashing-based tracking methods can still track the face accurately. Also, for the “Soccer” sequence, the target player is heavily occluded by others many times and the player undergoes drastic motion blur and illumination changes. In these video sequences, our proposed perceptual hashing tracking methods show absolute advantages. At the 30th frame, the object undergoes drastic motion blur. In this case, compressive tracking begins to deviate from the object. At the 70th frame, both the mean-shift tracker and compressive tracking lose the object, but our proposed trackers can still track the object well.

The “Animal” sequence exhibits background distraction and target motion blur caused by abrupt motion of the object, where either the mean-shift tracker or compressive tracking fails to track the target after the 24th frame. On the contrary, both the newly proposed LHash and LDHash-based trackers can successfully track the target throughout the whole video sequence. Especially, LDHash locates the animal most accurately.

In the “Lemming” video sequence, there is severe image blur in the target region caused by ultra-high moving speed of the target, which is sometimes combined with occlusion. From the experimental result at the 232th frame, we can find that the mean-shift, aHash and pHash trackers lose the target because of serious blur, but LHash and LDHash can still track the target accurately for their sharpening functions with respect to blurred images. In particular, at the 404th frame, LDHash is able to track the object completely when all the other

trackers lose the object.

For the “David” sequence, both the illumination and pose of the target change gradually. All the five perceptual hashing methods track the object better than the mean-shift tracker. At the 161th frame, due to drastical change of object pose, the mean-shift tracking window appears big drift, while there is only little drift with aHash, pHash, dHash and LHash.

4.3 Quantitative Comparison of All Tracking Algorithms

We measure the TSR, CLE and time cost of seven tracking methods on eight video sequences. Table 2 and Table 3 present the average TSRs and CLEs of five perceptual hashing trackers, compressive tracking, and the mean-shift tracker on each test video sequence. As seen from Figure 4, Table 2 and Table 3, we can easily find that all the perceptual hashing-based algorithms achieve better tracking performance than the commonly used mean-shift tracker. Besides, our perceptual hashing tracking outperforms the compressive tracking (CT) in most video sequences. In particular, our newly proposed perceptual hashing trackers (LHash and LDHash) almost achieve the best or the second best results in most sequences. As shown in Table 2, values in red fonts are the largest TSR indicating the optimal accuracy. Among the eight video sequences, LDHash presents the best tracking results on five video sequences (i.e., “Animal”, “Faceocc2”, “Lemming”, “Shaking”, “Singer1”), while LHash shows the best tracking performance on video sequences “Sylv” and dHash achieves the best result on video sequences “Soccer”. By contrast, CT only achieves the best result in the “David” sequence, but its TSR or CLE is very close to those of LHash and LDHash.

Table 2: Average Tracking Success Rate (TSR). Boldface indicates the optimal accuracy.

	aHash	dHash	pHash	LHash	LDHash	CT	mean-shift
David	0.65411	0.69708	0.70154	0.72285	0.712	0.7247	0.34853
Animal	0.74141	0.73374	0.733	0.72781	0.75153	0.11	0.10673
Faceocc2	0.73091	0.7624	0.67166	0.74283	0.80197	0.73615	0.34607
Lemming	0.32219	0.51565	0.20421	0.38287	0.59153	0.46246	0.57449
Shaking	0.33062	0.35613	0.40831	0.4985	0.52859	0.35038	0.28934
Singer1	0.59747	0.76088	0.60969	0.70104	0.76437	0.4703	0.20903
Soccer	0.61087	0.75444	0.61774	0.44589	0.73969	0.2036	0.30566
Sylv	0.467	0.38494	0.55637	0.72543	0.2968	0.71574	0.25747

Table 3: Average Center Location Error (CLE) (in pixels). Boldface represents the best performance.

	aHash	dHash	pHash	LHash	LDHash	CT	mean-shift
David	32.026	21.851	21.296	21.615	20.48	18.36	54.18
Animal	24.65	22.008	21.092	22.639	21.02	110.0821	162.88
Faceocc2	24.588	20.645	27.992	26.746	30.08	23.602	64.66
Lemming	161.54	128.14	192.87	138.57	113.54	122.44	163.31
Shaking	83.62	30.409	88.842	29.907	29.865	41.454	195.91
Singer1	13.468	18.474	18.46	14.071	13.182	25.114	62.588
Soccer	64.888	22.024	71.39	61.639	33.048	97.587	63.67
Sylv	32.242	49.559	20.204	11.478	55.642	13.63	57.59

Table 4: Average Time Cost (ms).

	aHash	dHash	pHash	LHash	LDHash	CT	mean-shift
David	99.4201	317.7176	1153.0012	219.7014	234.3012	786.704	15.1174
Animal	109.4768	435.8494	1437.9029	274.0576	685.1473	760.9203	50.8487
Faceocc2	135.9848	474.8533	1588.0166	358.463	723.0661	805.0147	15.9237
Lemming	64.5516	96.7111	429.021	185.1943	282.7665	774.703	68.1336
Shaking	37.0382	274.0915	199.3496	207.0067	147.573	728.503	38.6215
Singer1	86.3456	273.3337	1115.085	169.045	221.0157	594.995	46.0655
Soccer	106.0376	321.024	1331.2933	197.9395	402.1675	770.3618	63.8318
Sylv	39.9014	123.374	418.8416	88.069	106.376	791.1981	16.2994

4.4 Computational Cost Comparison of All Tracking Algorithms.

Real-time visual tracking systems require high accuracy and low-computational cost. What we try to do is to ensure the effectiveness of tracking and reduce the computational cost at the same time. In this paper, we use the average time cost of tracking in each frame of video sequences to measure the computational cost. Table 4 shows the average time cost of seven algorithms on each test sequence. As we can see, the average time cost of aHash, dHash, LHash and LDHash are all lower than that of CT on all test sequences. That is, our proposed perceptual hashing-based tracking algorithms not only outperform the others in terms of accuracy, but also with comparable

computational cost. On the contrary, for those really challenging videos, even though the mean-shift tracker requires low time cost, its tracking accuracy is lower than our proposed approaches.

The newly proposed perceptual hashing-based algorithms can further be applied to RGB-D video data. A single-object tracker could be used to represent a target object with a perceptual hashing code and single-valued depth information. First, the tracking object area is selected with the mouse. Then, the target image area is represented by an image hash code and single-valued depth information. Distance between the object area and the scan window can be computed to locate the tracking window with the minimum distance. According to the experimental results of [49], the approach naturally fusing perceptual hashing codes and depth data can yield much improved results in tracking compared with other color-based approaches. Perceptual hashing codes make the tracking algorithm run in real time; furthermore, the proposed method will track the object robustly even when a background hash code is similar to the tracking object by fusing depth data.

5. Conclusions and Future Work

In this paper, we propose new visual tracking methods based on perceptual hashing. Applying three existing perceptual hashing (aHash, dHash, pHash) as effective image representation, we obtain three robust trackers for visual tracking on challenging video sequences. Compared with the most commonly used mean-shift algorithm and the state-of-the-art compressive tracking algorithm, our methods achieve both higher accuracy and low-computational cost. To further improve the accuracy of object localization and the robustness of tracking, we propose two new perceptual hashing methods for tracking, which are Laplace-based hash (LHash) and Laplace-based difference hash (LDHash). Compared with three basic perceptual hashing algorithms on eight video sequences, experimental results indicate that the newly proposed methods are able to achieve more accurate and robust tracking, at the same time they demand low-computational cost. Therefore, we believe that LHash and LDHash should have good potential in more complex scenarios, such as face tracking, robot localization and navigation, infrared thermal imaging tracking applications, especially for RGB-D video tracking. Compared with some common videos, RGB-D videos provide not only high-quality image visual contents, but also depth information of a scene. Since our improved perceptual hashing can be a compact and efficient representation of visual content of objects, it can be further applied to fusing with depth information. Extracting the hash codes of color images and depth information simultaneously, a good complementary of these two kinds of features can be performed for more robust RGB-D video tracking.

For future topics, we intend to combine more information with our algorithms, such as color feature, angle

feature, and depth information. Also, to further improve the efficiency of visual tracking, we may apply some machine learning algorithms to the perceptual hashing-based tracking, such as SVM, PCA, etc.

Acknowledgement

This work is supported by National Natural Science Foundation of China (61463032, 61563035, and 81501560) and Scientific Research Foundation for Returned Scholars, Ministry of Education of China.

References

- [1] Chen J (2010) UAV-guided Navigation for Ground Robot Tele-operation in a Military Reconnaissance Environment, *Ergonomics*, 53: 940-950.
- [2] Cesetti A, Frontoni E, Mancini A, et al. (2010) A vision-based guidance system for UAV navigation and safe landing using natural landmarks. Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009. Springer Netherlands, pp 233-257.
- [3] Jia Z, Balasuriya A, Challa S (2008) Autonomous vehicles navigation with visual target tracking: Technical approaches. *Algorithms*, 1(2): 153-182.
- [4] Wang P K, Torrione P A, Collins L M, et al. (2012) Rapid position estimation and tracking for autonomous driving. *SPIE Defense, Security, and Sensing. International Society for Optics and Photonics*, pp 83871I-83871I.
- [5] Bulling A, Ward J, Gellersen H, et al. (2011) Eye movement analysis for activity recognition using electrooculography, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(4): 741-753.
- [6] Yu M, Liu L, Shao L (2015) Structure-Preserving Binary Representations for RGB-D Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, doi: 10.1109/TPAMI.2015.2491925.
- [7] Altinok A, El-Saban M, Peck A J, et al. (2006) Activity analysis in microtubule videos by mixture of hidden Markov models. *2006 IEEE Conference on Computer Vision and Pattern Recognition*, 2: 1662-1669.
- [8] Zhang P, Li N (2005) The intellectual development of Human-Computer Interaction research: A critical assessment of the MIS literature (1990-2002). *Journal of the Association for information Systems*, 6(11): 227-292.
- [9] Bulling A, Gellersen H (2010) Toward mobile eye-based human-computer interaction. *Pervasive Computing*, 9(4): 8-12.
- [10] Micalizio R, Scala E, Torasso P (2011) Intelligent supervision for robust plan execution. *AI* IA 2011: Artificial Intelligence Around Man and Beyond. Springer Berlin Heidelberg*, pp 151-163.
- [11] Perng M H, Chang H H (1993) Intelligent supervision of servo control. *Control Theory and Applications, IEE Proceedings D. IET*, 140(6): 405-412.
- [12] Zhang B C, Li Z G, Perina, Alessandro (2016) Adaptive local movement modeling for object tracking, *IEEE TCSVT*.
- [13] Yang H, Shao L, Zheng F, et al. (2011) Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18): 3823-3831.
- [14] Wang L, Liu T, Wang G, Chan K L and Yang Q (2015) Video Tracking Using Learned Hierarchical

Features. *IEEE Transactions on Image Processing*, 24(4): 1424-1435.

- [15] Newcombe R, Fox D, Seitz S (2015) DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time. *IEEE Computer Vision and Pattern Recognition*, pp. 343-352.
- [16] Zhang B, Perina A, Li Z, Murino V, Liu J, Ji R (2016) Bounding Multiple Gaussians Uncertainty with Application to Object Tracking. *International Journal of Computer Vision*, pp. 1-16.
- [17] Yilmaz A, Javed O, Shah M (2006) Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4): 13.
- [18] Black M J, Jepson A D (1998) Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1): 63-84.
- [19] Kwon J, Lee K M (2010) Visual tracking decomposition. *IEEE Conference on Computer Vision and Pattern Recognition*, pp 1269-1276.
- [20] Karavasilis V, Nikou C, Likas A (2011) Visual tracking using the Earth Mover's Distance between Gaussian mixtures and Kalman filtering. *Image and Vision Computing*, 29(5): 295-305.
- [21] Mei X, Ling H (2011) Robust visual tracking and vehicle classification via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11): 2259-2272.
- [22] Li H, Shen C, Shi Q (2011) Real-time visual tracking using compressive sensing. *IEEE Conference on Computer Vision and Pattern Recognition*, pp 1305-1312.
- [23] Zhang K, Zhang L, Yang M H (2012) Real-time compressive tracking. *Computer Vision–ECCV 2012*. Springer Berlin Heidelberg, pp 864-877.
- [24] Li X, Shen C, Dick A, et al. (2013) Learning compact binary codes for visual tracking, *IEEE Conference on Computer Vision and Pattern Recognition*, pp 2419-2426.
- [25] Shao L, Liu L, Li X (2014) Feature learning for image classification via multiobjective genetic programming. *IEEE Transactions on Neural Networks and Learning Systems*, 25(7): 1359-1371.
- [26] Zhang S, Yao H, Zhou H, et al. (2013) Robust visual tracking based on online learning sparse representation. *Neurocomputing*, 100: 31-40.
- [27] Zhu F, Shao L (2014) Weakly-supervised cross-domain dictionary learning for visual recognition. *International Journal of Computer Vision*, 109(1-2): 42-59.
- [28] Avidan S (2004) Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8): 1064-1072.
- [29] Kalal Z, Matas J, Mikolajczyk K (2010) Pn learning: Bootstrapping binary classifiers by structural constraints. *IEEE Conference on Computer Vision and Pattern Recognition*, pp 49-56.
- [30] Liu L, Shao L (2013) Learning discriminative representations from RGB-D video data. *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, pp 1493-1500.
- [31] Chen N, Xiao H D, Wan W (2011) Audio hash function based on non-negative matrix factorisation of mel-frequency cepstral coefficients. *Information Security, IET*, 5(1): 19-25.
- [32] Yang B, Gu F, Niu X (2006) Block mean value based image perceptual hashing. *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp 167-172.
- [33] Liu L, Yu M, Shao L (2015) Multiview alignment hashing for efficient image search. *IEEE Transactions on Image Processing*, 24(3): 956-966.
- [34] Jie Z (2013) A novel block-DCT and PCA based image perceptual hashing algorithm. *arXiv preprint*

arXiv:1306.4079.

- [35] Laradji I H, Ghouti L, Khiari E H (2013) Perceptual hashing of color images using hypercomplex representations. IEEE International Conference on Image Processing, pp 4402-4406.
- [36] Li J, Allinson N M (2008) A comprehensive review of current local features for computer vision. Neurocomputing, 71(10): 1771-1787.
- [37] Monga V, Evans B L (2006) Perceptual image hashing via feature points: performance evaluation and tradeoffs. IEEE Transactions on Image Processing, 15(11): 3452-3465.
- [38] Watson A B (1994) Image compression using the discrete cosine transform. Mathematica journal, 4(1): 81.
- [39] Wen-Hsiung C, Smith C, Fralick S (1977) A Fast Computational Algorithm for the Discrete Cosine Transform. IEEE Transactions on Communications, 25(9): 1004-1009.
- [40] Narasimha M J, Peterson A M (1978) On the computation of the discrete cosine transform. IEEE Transactions on Communications, 26(6): 934-936.
- [41] Coşkun B, Sankur B (2004) Robust video hash extraction. Proceedings of the IEEE Conference on Signal Processing and Communications Applications, pp 292-295.
- [42] Weng L, Preneel B (2009) Shape-based features for image hashing. 2009. IEEE International Conference on Multimedia and Expo, pp 1074-1077.
- [43] Kwon J, Lee K M (2010) Visual tracking decomposition. (CVPR), IEEE Conference on Computer Vision and Pattern Recognition, pp 1269-1276.
- [44] Bhattacharjee S, Kutter M (1998) Compression tolerant image authentication. Proceedings of International Conference on Image Processing, 1: 435-439.
- [45] Comaniciu D, Ramesh V, Meer P (2000) Real-time tracking of non-rigid objects using mean shift. IEEE Conference on Computer Vision and Pattern Recognition, 2: 142-149.
- [46] Bradski G, Kaehler A (2008) Learning OpenCV: Computer vision with the OpenCV library. " O'Reilly Media, Inc."
- [47] Babenko B, Yang M H, Belongie S (2011) Robust object tracking with online multiple instance learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(8): 1619-1632.
- [48] Santner J, Leistner C, Saffari A, et al. (2010) Prost: Parallel robust online simple tracking. 2010 IEEE Conference on Computer Vision and Pattern Recognition, pp 723-730.
- [49] Yoon Y, Yun W, Yoon H, et al. (2014) Real-Time Visual Target Tracking in RGB-D Data for Person-Following Robots. Pattern Recognition (ICPR), 2014 22nd International Conference on. IEEE, pp. 2227-2232.