Multimedia Tools and Applications (2020) 79:7413–7433
https://doi.org/10.1007/s11042-019-08535-8

Detecting action-relevant regions for action recognition using a three-stage saliency detection technique

Xiaofang Wang[1] · Chun Qi[2]

Received: 16 November 2018 / Revised: 5 October 2019 / Accepted: 26 November 2019 /
Published online: 23 December 2019

Abstract
Dense tracking has been proven successful in action recognition, but it may produce a large number of features in background, which are not so relevant to actions and may hurt recognition performance. To obtain the action-relevant features for action recognition, this paper proposes a three-stage saliency detection technique to recover action-relevant regions. In the first stage, low-rank matrix recovery optimization is employed to decompose the overall motion of each sub-video (temporally split video) into a low-rank part and a sparse part, and the latter is used to compute initial saliency to discriminate candidate foreground from definite background. In the second stage, using the dictionary formed by the patches in definite background, the sparse representation for each patch in candidate foreground is obtained based on motion and appearance information to compute the refined saliency, which ensures the action-relevant regions tend to be distinguished more clearly from background. In the third stage, the saliency is spatially updated based on the motion and appearance similarity so that the action-relevant regions can be better highlighted due to the increase of spatial saliency coherence. Finally, a binary saliency map is created by comparing the updated saliency with a given threshold to indicate action-relevant regions, which is fused into dense tracking to extract action-relevant trajectory features in a video for action recognition. Experimental results on four benchmark datasets demonstrate that the proposed method performs better than the conventional dense tracking and competitively with its improved versions.

Keywords Action recognition · Saliency detection · Action-relevant region · Dense tracking

✉ Xiaofang Wang
wxf2012@stu.xjtu.edu.cn

Chun Qi
qichun@mail.xjtu.edu.cn

[1] School of Electronic and Information Engineering (Department of Physics), Qilu University of Technology (Shandong Academy of Sciences), Jinan, China

[2] School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China

1 Introduction

The videos captured from realistic scenes usually contain various types of human actions, which range from simple gestures such as *running* and *clapping* to complex activities such as *ice dancing* and *basketball*. Recognizing what actions are performed in videos is an important task in computer vision, which can be applied to intelligent video surveillance, patient monitoring, human-machine interaction, etc. Although extensive researches have been devoted, it is still a challenge to extract discriminative features from realistic videos for action recognition.

The dense tracking technique, which extracts trajectory features by tracking densely sampled points across frames [44], has shown to be successful in action recognition. However, the traditional dense tracking does not discriminate between action-relevant regions (i.e., the regions corresponding to the body parts or objects that participate in the action) and background, and may produce dense features in both areas, especially in the presence of camera motion. In contrast, when human recognizes actions in a video, he instinctively focuses his eyes on action-relevant regions and only gives a glance at surroundings occasionally. Thus, the features most useful for action recognition are those from action-relevant regions. The background features may encode some contextual information for action recognition, but they are generally far less informative than those in action-relevant regions. Incorporating them into action recognition may degrade recognition performance.

An intuitive improved mechanism is to perform dense tracking only in action-relevant regions, where the central issue is how to differentiate these regions from background. Since the action-relevant regions are usually more salient than background, many methods [33, 39, 40, 53] rely on saliency detection technique to gain high saliency for action-relevant regions so that they can be highlighted and discriminated. However, action-relevant regions cannot be always equivalent to salient regions. For example, some large action-relevant regions (e.g., the big torso of runner) may be not so salient for the uniform motion in them, and some background regions may also obtain high saliency for the irrelevant object motion (e.g., leaf swinging) in them. In these cases, a common saliency detection method may lead to wrong action-relevant region detection. Thus, for recovering action-relevant regions, a saliency detection method should be designed to exploit the difference between these regions and background more precisely. Partly inspired by the multi-stage idea used in many saliency detection methods [8, 14, 21, 38], we deduce that, if some definite background regions with very low motion saliency are segmented at first, the action-relevant regions will be easier to discriminate based on the fact that they tend to be distinct from the definite background regions more clearly than other regions. In Fig. 1, we try reconstructing some video frames using the linear combination of definite background regions in terms of motion information and find that the reconstruction errors of action-relevant regions are much larger than that of other regions. Based on this observation, we propose a three-stage saliency detection method to detect action-relevant regions for action recognition. Considering saliency does not vary considerably in a small spatiotemporal area, we temporally split a video into sub-videos, and detect the saliency of each sub-video on patch level.

The framework is shown in Fig. 2. In a realistic video, background motion caused by camera movement is usually uniform and lies in a low-rank subspace, while action motion is often irregular and sparse. In the first stage, we employ low-rank matrix recovery (LRMR) [48] to decompose the overall motion of a sub-video into a low-rank part and a sparse part and use the latter to compute initial saliency. Thus, some definite background regions will gain very low saliency and can be segmented at first. Among the rest (candidate foreground) regions, the action-relevant ones tend to be distinct from the definite background regions

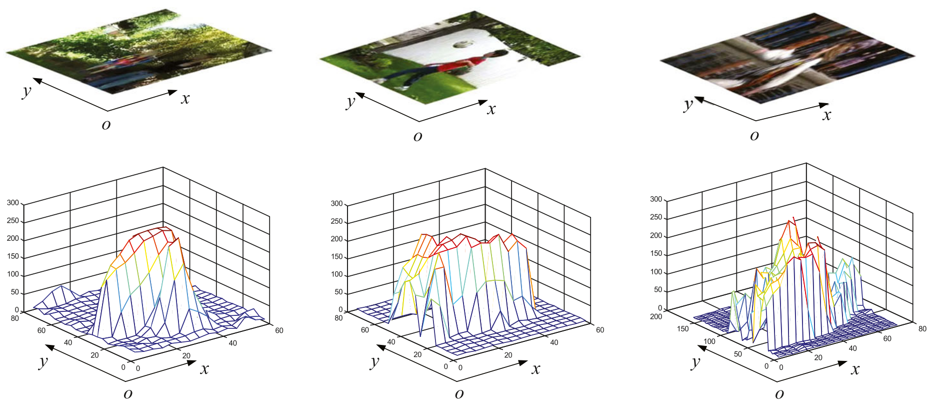

Fig. 1 Reconstruction error produced by reconstructing the video frames using the linear combination of definite background regions in terms of motion information

more clearly than the background ones. Thus, in the second stage, we learn sparse representation (SR) for candidate foreground regions using the dictionary formed by the definite background regions based on motion and appearance information, and use reconstruction error to compute refined saliency, so that the action-relevant regions have more chance to be highlighted. Considering saliency should be spatially coherent and some incorrect saliency values can be remedied by their neighboring saliency values, in the third stage, we spatially update the saliency based on the motion and appearance similarity to increase coherence. Comparing the saliency with a given threshold, a binary saliency map is created to indicate action-relevant regions. Finally, we fuse binary saliency maps into dense tracking to extract action-relevant trajectory features for action recognition.

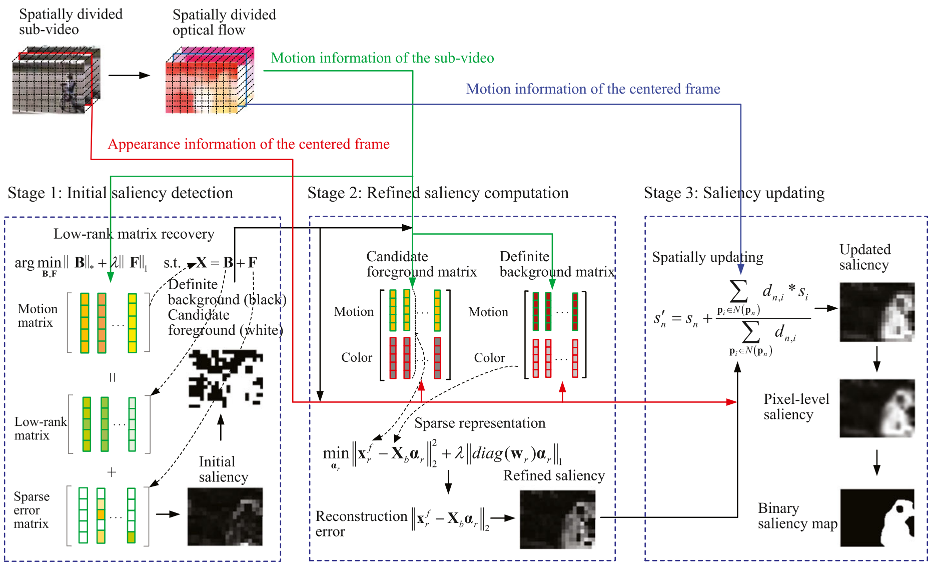
Spatially divided sub-video
Spatially divided optical flow
Motion information of the sub-video
Motion information of the centered frame
Appearance information of the centered frame
Stage 1: Initial saliency detection
Stage 2: Refined saliency computation
Stage 3: Saliency updating
Low-rank matrix recovery
$\arg\min_{\mathbf{B},\mathbf{F}} \|\mathbf{B}\|_* + \lambda\|\mathbf{F}\|_1$ s.t. $\mathbf{X}=\mathbf{B}+\mathbf{F}$
Definite background (black)
Candidate foreground (white)
Motion matrix
=
Low-rank matrix
+
Sparse error matrix
Initial saliency
Candidate foreground matrix
Definite background matrix
Motion
Color
Motion
Color
Sparse representation
$\min_{\boldsymbol{\alpha}_r} \|\mathbf{x}_r^f - \mathbf{X}_b\boldsymbol{\alpha}_r\|_2^2 + \lambda\|diag(\mathbf{w}_r)\boldsymbol{\alpha}_r\|_1$
Refined saliency
Reconstruction error $\|\mathbf{x}_r^f - \mathbf{X}_b\boldsymbol{\alpha}_r\|_2$
Spatially updating
$s_n' = s_n + \frac{\sum_{\mathbf{p}_i \in N(\mathbf{p}_n)} d_{n,i} * s_i}{\sum_{\mathbf{p}_i \in N(\mathbf{p}_n)} d_{n,i}}$
Updated saliency
Pixel-level saliency
Binary saliency map

Fig. 2 Framework of the three-stage action-relevant region detection method

The contributions of the proposed method are three-fold: 1) We propose to segment some definite background regions at first based on the initial saliency obtained from the sparse part of the motion produced by LRMR algorithm. 2) By obtaining the sparse representation for the candidate foreground regions based on the dictionary formed by the definite background regions and computing the refined saliency using reconstruction error, the action-relevant regions are more likely to be highlighted. 3) Saliency is updated in spatial domain to increase coherence, so that some incorrect saliency values can be rectified using their neighboring saliency values.

It is worth noting that, although the action-relevant region detection method is proposed to improve the dense tracking technique for action recognition, it is essentially a separate technique and can also be applied to many other existing action recognition frameworks such as the ones based on dense sampling [42], context modeling [5] and deep learning [4] to detect the actors and related objects or emphasize the action-relevant features.

The rest of this paper is organized as follows. Section 2 gives a review on the related works. Section 3 details the three-stage saliency detection method for recovering action-relevant region . Section 4 introduces saliency-based dense tracking for action recognition. Section 5 describes the experimental settings, datasets and results. Section 6 draws the conclusion.

2 Related works

2.1 Action recognition

In the past decades, extensive researches have been devoted to human action recognition and its related tasks, which involve recognizing actions in videos [7, 13, 26, 29, 33, 41, 44], recognizing activities from sensor data [18–20, 22], recognizing and detecting human actions in untrimmed videos [9, 51], etc. We just focus on the works that are most related to the proposed method.

In early researches, interest point tracking plays an important role in action recognition. To obtain long-term information for action recognition, Matikainen et al. [26] and Messing et al. [27] track a set of interest points in a video sequence through KLT tracker [23] to extract trajectories. Sun et al. [36] extract trajectories by tracking SIFT salient points over consecutive frames via pairwise SIFT matching. Sun et al. [37] and Bregonzio et al. [1] combine KLT tracker and SIFT matching to increase trajectory duration and density. However, recent researches show that it is more successful to extract features with a finer granularity by dense tracking. Wang et al. [44] extract dense trajectories by tracking densely sampled points across video frames via dense optical flow to describe actions and significantly improve the recognition performance.

Due to the harmful impact of background features, the improved techniques of dense tracking draw much attention in action recognition. To emphasize action-relevant features, some research works propose to improve dense tracking through saliency detection. Wang et al. [40] recover action-relevant regions in a video by detecting saliency through LRMR and perform dense tracking in these regions to extract action-relevant trajectories. Vig et al. [39] employ saliency-mapping algorithms to find informative regions and highlight the descriptors in these regions. Yi et al. [53] compute saliency maps from color, space and optical flow to obtain the trajectories containing only foreground motion. Because background trajectory features are mostly induced by camera motion, some research works aim to eliminate them by removing camera motion from videos. Wang et al. [41] prune background trajectories by estimating homography using RANSAC to cancel out camera motion from optical flow. Jain et al. [10] estimate dominant camera motion with a 2D affine motion model and separate it from optical flow. Wu et al. [49] use low-rank optimization to decompose dense trajectories into camera-induced component and object-induced component and use the latter to describe actions.

2.2 Saliency dection

Our action-relevant region detection method is partly inspired by some related works on image or video saliency detection, which adopt a somewhat multi-stage idea. Gao et al. [8] propose to use a first-pass RPCA to identify the likely regions of foreground, and then employ a second-pass block-sparse RPCA with fine-tuned parameter λ to accomplish a finer saliency detection. Tong et al. [38] first compute saliency using a bottom-up method to build background and foreground dictionaries, and then obtain saliency map using the LLC algorithm [43]. Li et al. [14] first extract boundary segment features to form background templates, then reconstruct the image by dense and sparse appearance models based on the templates and use reconstruction error to measure saliency.

Simlar to our method, there are also some research works [35, 52] employing the combination of LRMR and SR for saliency detection. Yan et al. [52] first transform image patches into feature space via SR using a learned dictionary, and then compute the saliency from the sparse part produced by performing LRMR over the transformed features. A similar pipeline is adopted in [35] for video saliency detection. The feature matrix of video cuboids is first obtained based on group SR and then decomposed into a low-rank part and a sparse part; the sparse part is used to compute saliency.

Although we also use a multi-stage idea and a combination of LRMR and SR, the pipeline of our method is totally different from above saliency detection methods. We first compute the initial saliency based on the sparse part of motion produced by LRMR to decide candidate foreground and definite background, then calculate the refined saliency by learning SR to reconstruct the candidate foreground regions based on the dictionary formed by the definite background, and finally improve the saliency by an updating operation. This pipeline is specially designed to compute saliency by exploiting the difference between action-relevant regions and background, which helps to obtain high saliency contrast and thus is better at action-relevant region detection.

3 Action-relevant region detection

Considering that action-relevant regions do not vary considerably in short time, we split the temporal duration of a video to obtain a number of short sub-videos and compute a saliency map for each sub-video to indicate the action-relevant regions of all frames in it. Besides, since saliency also does not vary significantly in a small spatial region, we spatially divide each sub-video into a number of small patches and detect the action-relevant regions in it on patch level.

Assuming a video with T frames is denoted by $\mathbf{V} = [\mathbf{I}_1, \mathbf{I}_2, \cdots, \mathbf{I}_T]$, where $\mathbf{I}_t$ is the frame at time t, by splitting the temporal duration evenly, we can divide the video into K non-overlapped sub-videos, i.e., $\mathbf{V} = [\mathbf{V}_1, \mathbf{V}_2, \cdots, \mathbf{V}_K]$, where $\mathbf{V}_k = \left[\mathbf{I}_{(k-1)w+1}, \mathbf{I}_{(k-1)w+2}, \cdots, \mathbf{I}_{kw}\right]$, and w is the temporal length of the sub-video. Then, each sub-video is further spatially divided into MN non-overlapped patches with equal spatial size $s \times s$. The divided $\mathbf{V}_k$ can be represented by a block matrix

$$\mathbf{V}_k = \begin{bmatrix} \mathbf{P}_1 & \mathbf{P}_2 & \cdots & \mathbf{P}_N \\ \mathbf{P}_{N+1} & \mathbf{P}_{N+2} & \cdots & \mathbf{P}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{(M-1)N+1} & \mathbf{P}_{(M-1)N+2} & \cdots & \mathbf{P}_{MN} \end{bmatrix} \quad (1)$$

where $\mathbf{P}_n$ is the nth patch and its size is $s \times s \times w$. In the following, we use $\mathbf{V}_k$ as an example to introduce how to recover the action-relevant regions in each sub-video through three-stage saliency detection.

3.1 Initial saliency detection

In this stage, we compute the initial saliency to differentiate definite background regions from candidate foreground, which can be accomplished based on the motion difference between action-relevant regions and background. If a video is captured with a moving camera, the background motion is spatially uniform and lies in a low-rank subspace, while the action motion is spatially irregular and can be seen as sparse error. For this reason, we use LRMR to separate the overall motion of a sub-video into a low-rank part and a sparse error part, and use the latter to measure the initial saliency, which is then used to classify the patches in the sub-video into a candidate foreground set and a definite background set.

Before utilizing LRMR to detect the saliency of sub-video $\mathbf{V}_k$, we arrange the motion in it on patch level to form a motion matrix. To this end, we first obtain the motion vector of each patch by vectorizing the optical flow of all points in the patch in the order of spatiotemporal position. The motion vector of $\mathbf{P}_n$ is formed by $\mathbf{x}^n = [\, u_1^n \ v_1^n \ \cdots \ u_i^n \ v_i^n \ \cdots \ u_{s\times s\times w}^n \ v_{s\times s\times w}^n \,]$, where u_i^n and v_i^n are respectively the horizontal and vertical components of the optical flow at the ith point in $\mathbf{P}_n$. When the motion vectors of all patches in $\mathbf{V}_k$ are obtained, we stack them column-wise in the order of spatial position to obtain a motion matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}^1 & \mathbf{x}^2 & \cdots & \mathbf{x}^{MN} \end{bmatrix}$, where $\mathbf{X}$ is a 2D matrix and its size is $(2s^2w) \times (MN)$.

The overall motion of sub-video $\mathbf{V}_k$ is encoded in $\mathbf{X}$. We decompose $\mathbf{X}$ into a low-rank matrix $\mathbf{B}$ and a sparse error matrix $\mathbf{F}$ by solving the following relaxed LRMR problem [48],

$$\arg\min_{\mathbf{B},\mathbf{F}} \|\mathbf{B}\|_* + \lambda\|\mathbf{F}\|_1 \quad \text{s.t.} \quad \mathbf{X} = \mathbf{B} + \mathbf{F} \quad (2)$$

where $\|.\|_*$ is the nuclear norm of a matrix, λ trades off rank versus sparsity. According to [16, 48, 49], the setting of λ is highly dependent on the dimension of the matrix to be decomposed. We empirically set $\lambda = 1.1/\sqrt{max(2s^2w, MN)}$ based on the recommended settings in these works. The optimization problem in (2) is solved with the inexact Augmented Lagrange Multiplier (IALM) algorithm [16].

After decomposition, the uniform background motion in sub-video $\mathbf{V}_k$ is mostly deposited in the low-rank matrix $\mathbf{B}$, while the irregular action motion is mostly conveyed in the sparse error matrix $\mathbf{F}$. Each column in $\mathbf{F}$ encodes the action motion of a patch in the sub-video. We compute the l_∞ norms for all columns in $\mathbf{F}$ and obtain a vector $\hat{\mathbf{s}}_k = \begin{bmatrix} \hat{s}_1 & \hat{s}_2 & \cdots & \hat{s}_{MN} \end{bmatrix}$, where $\hat{s}_n$ is used to measure the initial saliency of patch $\mathbf{P}_n$ in sub-video $\mathbf{V}_k$.

In general, the patches in action-relevant regions contain irregular action motion and produce high saliency values in $\hat{\mathbf{s}}_k$, while the patches in background contain uniform background motion and produce small values in $\hat{\mathbf{s}}_k$. However, in some cases, some action motion in large action-relevant regions is also uniform and deposited in matrix **B**, and some background motion may contain subtle spatial changes and be included in matrix **F**. Thus, the initial saliency cannot always correctly distinguish all action-relevant regions from background. We just compare it with a relatively small threshold T_s to classify all patches into a candidate foreground set S_f and a definite background set S_b. More specifically,

$$\begin{array}{l} S_f = \left\{\mathbf{P}_i \left| \hat{s}_i \geqslant T_s,\ i = 1, 2, \cdots, MN \right.\right\} \\ S_b = \left\{\mathbf{P}_i \left| \hat{s}_i < T_s,\ i = 1, 2, \cdots, MN \right.\right\} \end{array} \tag{3}$$

The threshold T_s is small enough in (3) so that nearly all possible foreground patches are included in S_f and the patches retained in S_b nearly all come from the true background. If the saliency of all patches in the sub-video is smaller than T_s, it means that the sub-video contains no action-relevant regions, and the following steps are not processed.

3.2 Saliency refinement

As a small threshold is used in (3), some background patches with subtle motion changes are retained in the candidate foreground set S_f. In this stage, to recover the true action-relevant patches from S_f, we compute refined saliency for each patch in S_f so that the patches in action-relevant regions can be distinguished more clearly from those in background.

Generally, even though the motion in some large action-relevant regions is spatially uniform, it still tends to be distinct from the motion in definite background. On the other hand, even though the motion in some background regions contains subtle changes, it is still apt to resemble the motion in definite background. Thus, we learn a sparse representation for the motion vector of each patch in the candidate foreground set S_f based on the dictionary constructed by the motion vectors of all patches in the definite background set S_b, and use the reconstruction error to measure patch saliency. With this technique, the action-relevant patches in S_f are difficult to reconstruct and thus obtain high saliency, while the background patches in S_f are easy to reconstruct and thus obtain low saliency. As such, the saliency contrast between action-relevant regions and background is sharpened.

In some cases, the motion of some action-relevant patches happens to resemble that of background patches, but its appearance (e.g., color) may be distinct from that of background ones. To enable action-relevant patches to have more chances to gain big reconstruction error, we combine appearance with motion to refine the saliency of each patch in S_f. To this end, we form a color matrix for the centered frame of the sub-video on patch level. When sub-video $\mathbf{V}_k$ is spatially divided into patches, the centered frame $\mathbf{I}_k^c$ in it is also divided as

$$\mathbf{I}_k^c = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_N \\ \mathbf{p}_{N+1} & \mathbf{p}_{N+2} & \cdots & \mathbf{p}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{p}_{(M-1)N+1} & \mathbf{p}_{(M-1)N+2} & \cdots & \mathbf{p}_{MN} \end{bmatrix} \tag{4}$$

where $\mathbf{p}_n$ is the nth patch and its size is $s \times s$. We obtain the color vector of each patch by vectorizing the color values of all pixels in it. The color vector of $\mathbf{p}_n$ is formed by $\mathbf{a}^n = \left[\, L_1^n\ a_1^n\ b_1^n\ \cdots\ L_i^n\ a_i^n\ b_i^n\ \cdots\ L_{s\times s}^n\ a_{s\times s}^n\ b_{s\times s}^n \,\right]^T$, where L_i^n, a_i^n and b_i^n are the color values of the ith pixel in $\mathbf{p}_n$ in Lab space. By stacking the color vectors of all patches in the order of spatial position, we obtain the color matrix of $\mathbf{I}_k^c$, i.e., $\mathbf{A} = \left[\, \mathbf{a}^1\ \mathbf{a}^2\ \cdots\ \mathbf{a}^{MN} \,\right]$, where $\mathbf{A}$ is a 2D matrix and its size is $(3s^2) \times (MN)$.

The motion and appearance information of $\mathbf{V}_k$ can be combined by concatenating the motion and color matrices into a feature matrix $\mathbf{G}$

$$\mathbf{G} = \begin{bmatrix} \mathbf{X}' \\ \mathbf{A}' \end{bmatrix} \tag{5}$$

where $\mathbf{X}'$ and $\mathbf{A}'$ are respectively formed by the normalized columns in $\mathbf{X}$ and $\mathbf{A}$. Each column in $\mathbf{G}$ is the feature vector of a patch in $\mathbf{V}_k$, whose length is $(2s^2w + 3s^2)$. The feature vectors of all patches in S_f are stacked column by column in given order to form the candidate foreground feature matrix $\mathbf{X}_f = \begin{bmatrix} \mathbf{x}_1^f & \mathbf{x}_2^f & \cdots & \mathbf{x}_R^f \end{bmatrix}$, where $\mathbf{x}_r^f$ is the feature vector of the rth patch in S_f, and R is the number of patches in S_f. Similarly, the feature vectors of all patches in S_b form the definite background feature matrix $\mathbf{X}_b = \begin{bmatrix} \mathbf{x}_1^b & \mathbf{x}_2^b & \cdots & \mathbf{x}_S^b \end{bmatrix}$, where $\mathbf{x}_s^b$ is the feature vector of the sth patch in S_b, and S is the number of patches in S_b.

Then, the refined saliency for each patch in S_f can be computed by learning a sparse representation for the corresponding vector in $\mathbf{X}_f$ using $\mathbf{X}_b$ as dictionary. Take the rth patch in S_f for example, the sparse representation for $\mathbf{x}_r^f$ can be obtained by solving the following optimization problem

$$\min_{\alpha_r} \left\| \mathbf{x}_r^f - \mathbf{X}_b \alpha_r \right\|_2^2 + \lambda \|\alpha_r\|_1 \tag{6}$$

where λ is a regularization parameter that trades off reconstruction error versus sparsity, and α_r is the reconstruction coefficient vector.

Considering background patches in S_f are more correlated with the patches that are spatially close to them in definite background set S_b, we assign a weight to each atom in $\mathbf{X}_b$ to further enable those patches to be reconstructed easily. Specifically, the weight for the ith atom $\mathbf{x}_i^b$ is computed by

$$w_i = \exp\left(\frac{dist\,(c_r, c_i)}{\sigma_w} \right) \tag{7}$$

where c_r and c_i are the centers of the patches corresponding to the reconstructed $\mathbf{x}_r^f$ and atom $\mathbf{x}_i^b$ respectively; $dist\,(c_r, c_i)$ is the normalized Euclidean distance between them; σ_w is a parameter adjusting the decay speed of weight, which is set to 0.05. The weights of all atoms form a weight vector $\mathbf{w}_r = \begin{bmatrix} w_1 & w_2 & \cdots & w_S \end{bmatrix}$. Introducing $\mathbf{w}_r$ in (6), we obtain the weighted sparse representation optimization problem [3]

$$\min_{\alpha_r} \left\| \mathbf{x}_r^f - \mathbf{X}_b \alpha_r \right\|_2^2 + \lambda \|diag(\mathbf{w}_r)\alpha_r\|_1 \tag{8}$$

We solve (8) using the optimization toolbox of SPArse Modeling Software (SPAMS) [24], and empirically set $\lambda = 0.15$. When the sparse representation is obtained, we compute the reconstruction error to measure the saliency of the rth patch in S_f

$$s_r = \left\| \mathbf{x}_r^f - \mathbf{X}_b \alpha_r \right\|_2 \tag{9}$$

Since (7) produces smaller weight for the shorter distance, the background patches in S_b that are spatially close to the reconstructed patch are emphasized in (8), which helps to obtain lower reconstruction error for the background patches in S_f.

Similarly, we calculate the saliency values of all patches in S_f. Combining them with those of the patches in S_b, which are obtained with LRMR in the first stage, we construct a patch-level saliency matrix $\mathbf{S}_k$ according to the patch position in the divided sub-video $\mathbf{V}_k$

$$\mathbf{S}_k = \begin{bmatrix} s_1 & s_2 & \cdots & s_N \\ s_{N+1} & s_{N+2} & \cdots & s_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{(M-1)N+1} & s_{(M-1)N+2} & \cdots & s_{MN} \end{bmatrix} \tag{10}$$

where the element s_n is the saliency of patch $\mathbf{P}_n$ in the divided $\mathbf{V}_k$. If all the elements in $\mathbf{S}_k$ are smaller than a small threshold, no patch in $\mathbf{V}_k$ contains action motion, and $\mathbf{S}_k$ is set to zero. Otherwise, the values in it are rescaled to [0, 255].

3.3 Saliency updating

Through saliency refinement, the action-relevant patches have more chances to gain high saliency contrast against background, but there are still some outliers that cannot be correctly highlighted. In this stage, we further improve the saliency detection results through a spatial updating operation.

When the saliency is computed patch by patch in the first two stages, the spatial coherence of saliency between neighboring patches is ignored. In fact, the saliency values of the adjacent patches located in the same body part or background should be spatially coherent. If the spatial saliency coherence is enhanced, some wrongly suppressed action-relevant patches may be highlighted due to the high saliency of their adjacent action-relevant patches, and on the other hand, some wrongly highlighted background patches may be suppressed due to the low saliency of their adjacent background patches. Based on this observation, we update the saliency value of each patch by adding a weighted sum of the saliency values of its spatially adjacent patches. The weight is the probability that two patches belong to the same body part or background, which is measured by the motion and appearance similarity between the two patches in the centered frame of the sub-video.

To update the saliency value of $\mathbf{P}_n$, we first compute the motion and appearance similarity between $\mathbf{p}_n$ and each 8-neighborhood patch $\mathbf{p}_i$ in the centered frame of $\mathbf{V}_k$

$$d_{n,i} = \exp\left(-\frac{dist\left(H_n, H_i\right)}{\sigma_d}\right) \tag{11}$$

where H_n and H_i are the feature vectors formed by concatenating the HOG and HOF features of patch $\mathbf{p}_n$ and $\mathbf{p}_i$ respectively, $dist\left(H_n, H_i\right)$ is the Euclidean distance between H_n and H_i, and σ_d is a parameter adjusting the decay speed of similarity, which is empirically set to 0.5. Based on the patch similarity, the saliency value of $\mathbf{P}_n$ is updated by

$$s_n' = s_n + \frac{\sum\limits_{\mathbf{p}_i \in N(\mathbf{p}_n)} d_{n,i} * s_i}{\sum\limits_{\mathbf{p}_i \in N(\mathbf{p}_n)} d_{n,i}} \tag{12}$$

where $N\left(\mathbf{p}_n\right)$ is the neighborhood of $\mathbf{p}_n$ in the centered frame. When all the saliency values in $\mathbf{S}_k$ are updated, we resize it to the original spatial size of the video through nearest-neighbor interpolation to generate a pixel-level saliency matrix $\tilde{\mathbf{S}}_k$. By thresholding each element in $\tilde{\mathbf{S}}_k$ with a given threshold T_s', we obtain the binary saliency map $\mathbf{M}_k$ for sub-video $\mathbf{V}_k$

$$\mathbf{M}_k(x, y) = \begin{cases} 1 & \tilde{\mathbf{S}}_k(x, y) \geqslant T_s' \\ 0 & otherwise \end{cases} \tag{13}$$

where $\tilde{\mathbf{S}}_k(x, y)$ is the saliency value of the pixel (x, y) in any frame of the sub-video and $\mathbf{M}_k(x, y)$ is the action-relevant region indicator of (x, y). If $\mathbf{M}_k(x, y) = 1$, the pixel belongs to action-relevant regions, otherwise it belongs to background.

4 Saliency-based dense tracking and action recognition

To evaluate recognition performance, we fuse binary saliency maps into dense tracking technique [44] to extract trajectories in action-relevant regions. Feature points are densely sampled on a grid spaced by 5 pixels and tracked from one frame to the next using filtered dense optical flow. When the points are tracked in the next frame, we see them as candidate succeeding trajectory points and employ the binary saliency map to determine whether they are in action-relevant regions or not. Only those located in action-relevant regions are regarded as valid trajectory points. To ensure dense coverage of trajectories, if no tracked point is found in a 5×5 neighborhood in action-relevant regions, a new point is sampled and added to the tracking process. After the tracking process is finished, all points on the same track form a trajectory. We remove static trajectories and those with sudden large displacements, which are mostly irrelevant to the interested action.

Four types of trajectory descriptors (Shape, HOG, HOF and MBH) are computed for all trajectories. To reduce the influence of camera motion, we compute the motion descriptors (Shape, HOF and MBH) from the rectified optical flow, in which camera motion is removed through the technique used in the improved dense trajectory method [41]. Each type of descriptors is encoded independently using Fisher vector [31, 32] with the same settings as [41] to obtain the final video-level representation. We use the SVM classifier and multiple kernel learning to predict action class, where four linear kernels are used, each corresponding to one type of video-level representation.

5 Experiments

5.1 Experimental settings

We carry out experiments on four benchmark datasets to evaluate the proposed method, namely, YouTube, Hollywood2, HMDB51 and UCF101. When detecting action-relevant regions, we resize video frames with a factor of 1/4 for all datasets to reduce computational costs. The sub-video length is set to $w = 5$ frames and the spatial size of patches is set to $s = 4$ pixels, which are shown to be optimal by the experiments in [40]. We set the initial saliency threshold $T_s = 10$, the final saliency threshold $T_s' = 100$, which are proved to be most appropriate by the experiments.

5.2 Action datesets

Hollywood2 dataset [25] is composed of the videos extracted from 69 movies, which fall into 12 actions classes: *answering phone*, *driving car*, *eating*, *fighting person*, *getting out of* *car*, *hugging person*, *hand shaking*, *kissing*, *running*, *sitting down*, *sitting up* and *standing up*. The action videos in it are split into two sets in experiments, i.e., 823 training samples and 884 test samples.

YouTube dataset [17] consists of 1168 videos, which are divided into 11 action categories: *Basketball shooting*, *biking*, *diving*, *golf swinging*, *horseback riding*, *soccer juggling*, *swinging*, *tennis swinging*, *trampoline jumping*, *volleyball spiking* and *walking (with a dog)*. The videos in each category are grouped into 25 groups. The LOO (Leave-One-Out) scheme on group basis is employed in experiments, using one group of each category for testing and the others for training in each run. Recognition performance is measured by averaging the recognition accuracy over 25 runs.

HMDB51 dataset [12] contains 6,766 video clips extracted from commercial movies as well as YouTube. In our experiments the original videos are used, which are divided into 51 action categories, each including a minimum of 101 clips. We use the original experimental setup [12], where three train-test splits are used. For each split, 70 videos are used for training and 30 videos for testing in each class. Recognition performance is measured by the average recognition accuracy over three splits.

UCF101 dataset [34] consists of 13,320 video clips collected from YouTube, which are divided into 101 categories. The videos in each category are grouped into 25 groups. In our experiments, the protocol with three train-test splits [34] is followed and recognition performance is reported as the average accuracy over these splits.

5.3 Saliency and action-relevant region detection results

5.3.1 Saliency detection results of different stages

We visualize the saliency detection results of different stages for some action videos from four datasets in Fig. 3. The videos are all contaminated with various types of camera motion, such as shaking, travelling and zooming. As indicated in Fig. 3b, the initial saliency obtained in the first stage can not highlight all the action-relevant regions. The candidate foreground set and the definite background set are respectively displayed in white and black in Fig. 3c. Obviously, the former set contains not only nearly all the action-relevant patches, but also some background patches, and the patches in the latter set nearly all come from background. As indicated in Fig. 3d, after the saliency is refined in the second stage, the saliency contrast is greatly sharpened and the action-relevant regions are significantly highlighted. As shown in in Fig. 3e, after spatially updated in the third stage, the saliency in action-relevant regions is even more prominent and continuous, which well distinguishes the action-relevant regions from background.

5.3.2 Final saliency visualization and comparison

We compare the final saliency detection results with that of two recent video saliency detection methods [21, 45], which are not specially designed for recovering action-relevant regions and have shown excellent performance in common video saliency detection. To this end, we randomly select some videos from four action datasets, which are shown in Fig. 4a. All of them contain various degrees of background motion except the last two. The saliency is computed using the three methods under their default parameter settings and the binary

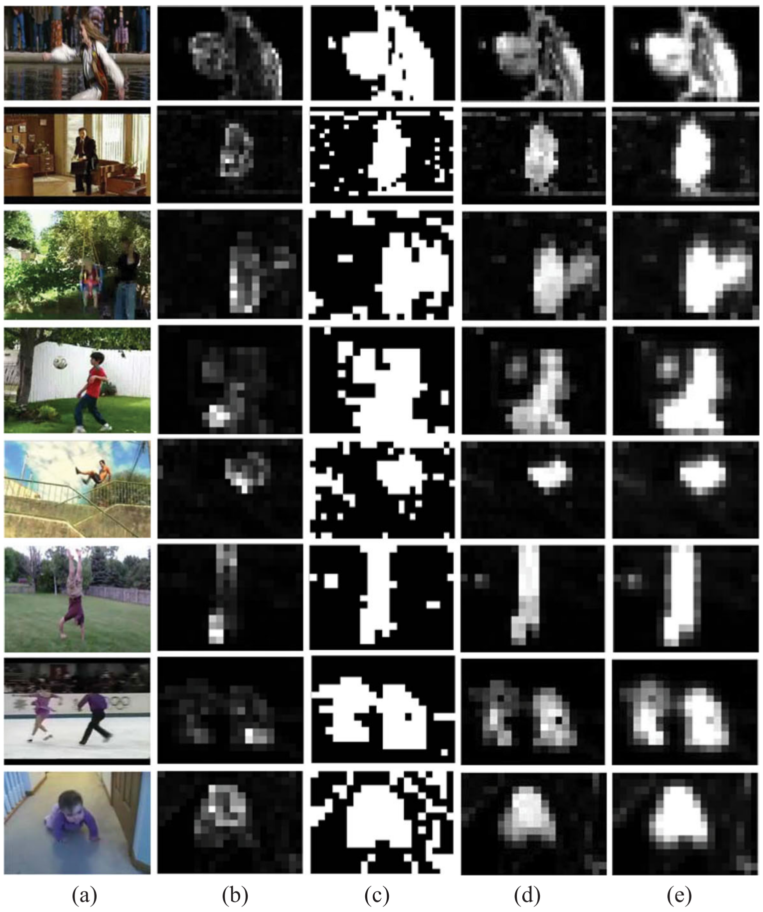

Fig. 3 Saliency detection results of different stages: **a** sample frame, **b** initial saliency of the first stage, **c** candidate foreground set (white) and definite background set (black), **d** refined saliency of the second stage, and **e** updated saliency of the third stage

saliency maps are all obtained with threshold $T_s' = 100$. From Fig. 4b and c, we can see, whether for the videos with dynamic or static background, the final saliency obtained by our method well highlights action-relevant regions, and the binary saliency maps offer good coverage over the action-relevant regions in most cases. In contrast, as shown in Fig. 4d and f, the saliency contrast obtained by [45] and [21] is not as sharp as ours, and in the binary saliency maps yielded by [45] and [21] shown in Fig. 4e and g, background regions are frequently wrongly segmented as action-relevant regions, and conversely action-relevant regions are also often mistaken for background. Obviously, our method is more suitable for action-relevant region detection than common video saliency detection methods.

5.3.3 Evaluation of appearance information in saliency refinement

In the second stage of saliency detection, we introduce static appearance information (color) and combine it with motion information to refine saliency, so that the patches in

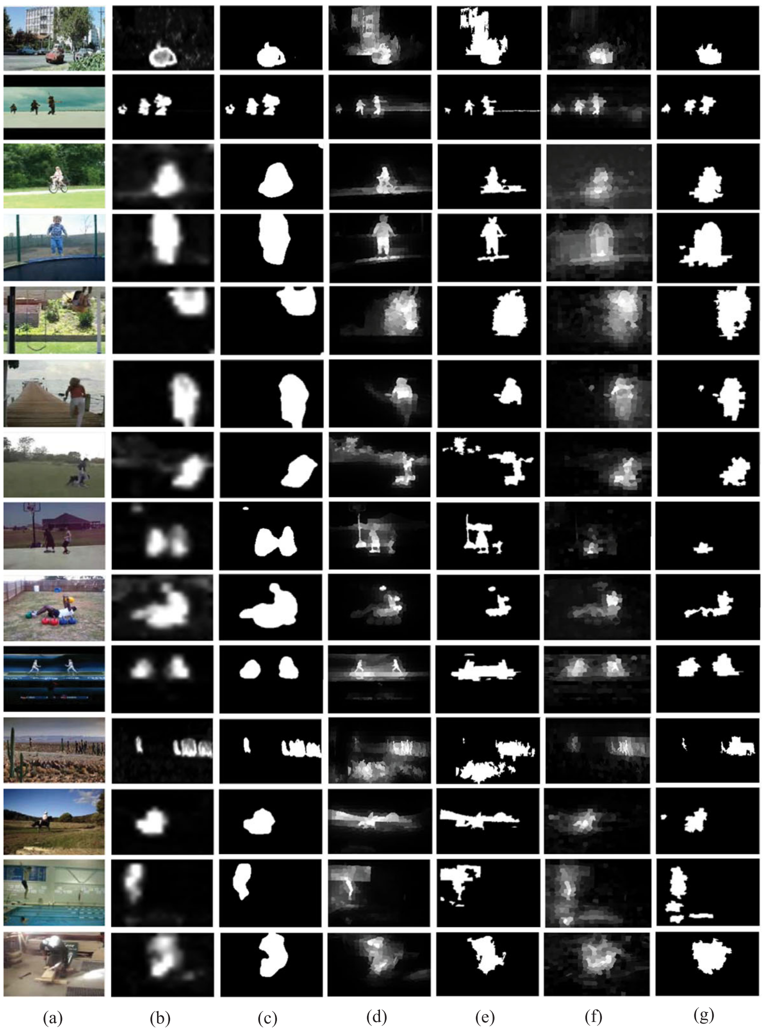

Fig. 4 Comparison of the saliency detection results of different methods: **a** sample frame, **b** saliency of our method, **c** binary saliency map of our method, **d** saliency of [45], **e** binary saliency map of [45], **f** saliency of [21], and **g** binary saliency map of [21]

action-relevant regions have more chances to obtain high saliency. To evaluate the influence of appearance information on saliency detection, we try removing it from feature matrix **G** and using only motion information for saliency refinement. The saliency detection results on some videos are visualized and compared with those obtained under the default settings in Fig. 5. As we can observe, when both appearance and motion information is employed, higher saliency is gained in the inner parts of action-relevant regions and the binary saliency maps cover the action-relevant regions more completely, which demonstrates that incorporating appearance information is beneficial to saliency refinement.

However, As shown in the first two rows in Fig. 5, there are still some regions in large action-relevant areas cannot be recovered by incorporating appearance information. One

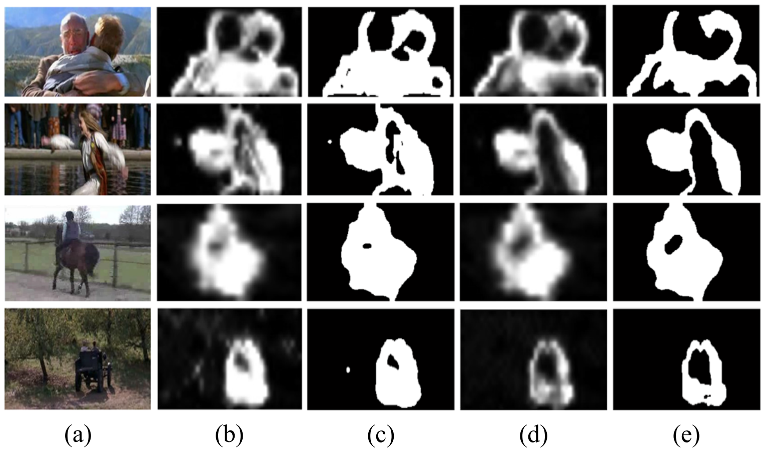

Fig. 5 Comparison of the saliency detection results with or without static appearance information: **a** sample frame, **b** saliency of default settings, **c** binary saliency map of default settings, **d** saliency without appearance information, and **e** binary saliency map without appearance information

possible reason is that these regions are wrongly classified into definite background in the first stage due to very uniform motion in them, and thus cannot be highlighted in the second stage. The other possible reason is that these regions happen to be similar to background in both appearance and motion. Both cases cannot be handled by the proposed method.

5.3.4 Evaluation of computational complexity

In this section, the computational complexity for each stage of the saliency detection method is evaluated on the 150 videos of *BaseballPitch* class in UCF101 dataset, whose average size is $320 \times 240 \times 100$. The experiment is conducted in MATLAB on a personal computer with 4.2 GHz CPU/16G RAM. and the time consumption per frame for each stage is shown in Table 1. Obviously, the first stage is most time consuming. Although it has been shown in [16] that the IALM can be empirically more than five times faster than the other algorithms, unfortunately, the computational costs are still non-negligible. In contrast, the second stage is much more computationally efficient, because the dictionary for SR is directly formed by stacking the feature vectors of the patches in the definite background set, which bypasses the time-consuming dictionary learning. The updating operation in the third stage just involves a weighted summation of the saliency values in 8-neighborhood, and can also be implemented very efficiently. Overall, the total processing time for action-relevant region detection is 1122.11ms per frame, thus it still cannot work in a real-time manner on a typical personal computer.

Table 1 Time consumption per frame for different stages of saliency detection

Stages	The first stage	The second stage	The third stage	Three stages
Time consumption (ms)	1113.5	4.55	4.06	1122.11

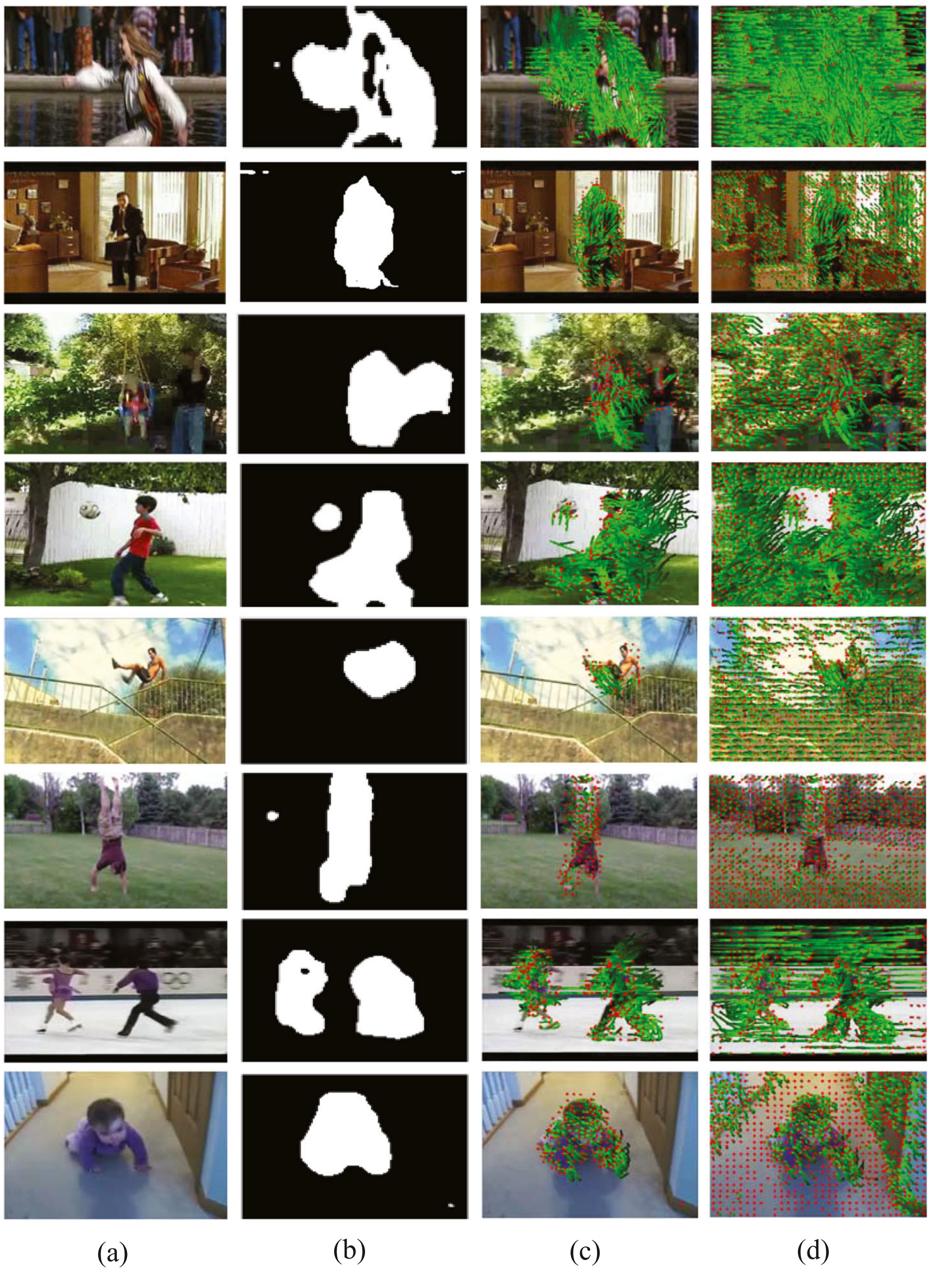

Fig. 6 Trajectory visualization and comparison: **a** sample frame, **b** binary saliency map, **c** trajectories produced by saliency-based dense tracking, and **d** trajectories produced by traditional dense tracking

Table 2 Recognition performance of different stages of saliency detection

Methods	Hollywood2 (%)	YouTube (%)	HMDB51 (%)	UCF101 (%)
Initial-saliency-DT	63.2	90.3	56.8	84.9
Refined-saliency-DT	64.5	91.0	57.7	85.6
Saliency-DT	64.8	91.4	58.9	86.1

5.4 Results of saliency-based dense tracking

The trajectories extracted by saliency-based dense tracking for the sample videos from four datasets are visualized in Fig. 6c. As can be seen, by fusing the binary saliency maps into dense tracking, the trajectories in action-relevant regions are well extracted, which are dense and continuous, and offer a good coverage. At the same time, the trajectories in background are well suppressed. For comparison, we also extract the dense trajectories using traditional dense tracking. As visualized in Fig. 6d, besides those in action-relevant regions, there are also a large number of trajectories produced in background.

5.5 Action recognition performance

5.5.1 Overall recognition performance

The default overall recognition results obtained based on the trajectories extracted from the action-relevant regions detected by the proposed method are shown in the last row of Table 2 indicated by ‘Saliency-DT’, which are 64.8% on Hollywood2, 91.4% on YouTube, 58.9% on HMDB51 and 86.1% on UCF101. To give a quantitative evaluation of each stage of saliency detection, we also compute the recognition results based on the action-relevant regions obtained from the initial saliency produced by the first stage and the refined saliency produced by the second stage, which are respectively indicated by ‘Initial-saliency-DT’ and ‘Refined-saliency-DT’ in Table 2. Comparing the results in three rows, we observe that the performance of refined saliency is always superior to that of initial saliency, and non-negligible improvement is also witnessed when the saliency is spatially updated.

The default recognition performance is computed based on the trajectories in detected action-relevant regions with the motion descriptors obtained from rectified optical flow. To validate the superiority of the proposed method, we also compute the results of the counterpart with motion descriptors in original optical flow field, and the results of dense trajectories extracted by conventional dense tracking, which are respectively indicated with ‘Saliency-DT-original’ and ‘DT’ in Table 3. By comparison, when original optical flow is used in motion descriptors, the trajectories in detected action-relevant regions also show

Table 3 Comparison of recognition performance of dense trajectories (DT), trajectories in action-relevant regions with motion descriptors in original optical flow field (Saliency-DT-original) and default settings (Saliency-DT)

Methods	Hollywood2 (%)	YouTube (%)	HMDB51 (%)	UCF101 (%)
DT	60.4	86.8	52.4	81.2
Saliency-DT-original	62.6	89.7	56.1	84.6
Saliency-DT	64.8	91.4	58.9	86.1

obvious superiority over the dense trajectories. To be specific, the performance improvements are respectively 2.2% on Hollywood2, 2.9% on YouTube, 3.7% on HMDB51 and 3.4% on UCF101. Comparing 'Saliency-DT-original' with 'Saliency-DT', we find the recognition performance is further improved when rectified optical flow is used in motion descriptors.

5.5.2 Influence of saliency thresholds on recognition performance

Two saliency thresholds are involved in the action-relevant region detection method, i.e., the initial saliency threshold T_s and the final saliency threshold T_s'. We evaluate the impact of these two thresholds on recognition performance on Hollywood2 and YouTube datasets. When evaluating one threshold, we compute the recognition results by varying it with a given step and fixing the other to the default. As shown in Fig. 7a, the recognition performance fluctuates occasionally with T_s, and reaches the highest at $T_s = 12$ on Hollywood2 and $T_s = 10$ on YouTube. As shown in Fig. 7b, the recognition performance improves with T_s' at the beginning but deteriorates when T_s' exceeds a value on both datasets. The highest performance is reached at $T_s' = 100$ on Hollywood2 and $T_s' = 110$ on YouTube.

From Fig. 7a, we observe that the initial saliency threshold T_s should be small, but too small value is also not preferable. This is because a reasonable T_s, one on hand, should be small enough to guarantee all true action-relevant patches are enclosed in candidate foreground set, but on other hand, should not be too small to ensure the definite background set contains enough patches to form a dictionary. From Fig. 7b, it can also be seen that the final saliency threshold T_s' should be set appropriately, because too big T_s' tends to cause action-relevant regions to be wrongly classified as background ones, and conversely too small T_s' will increases the risk of wrongly classifying background regions as action-relevant ones.

5.5.3 Comparison with the state of the art

The best recognition results of our method are compared with those of the state-of-the-art trajectory-based methods in Table 4. On Hollywood2, although [11] yields the best result using trajectory cluster as global point to alleviate camera motion, our method outperforms all other methods, among which, [44] is the traditional dense tracking method, and [6, 10, 39–41] improve the dense tracking by suppressing background trajectory features using

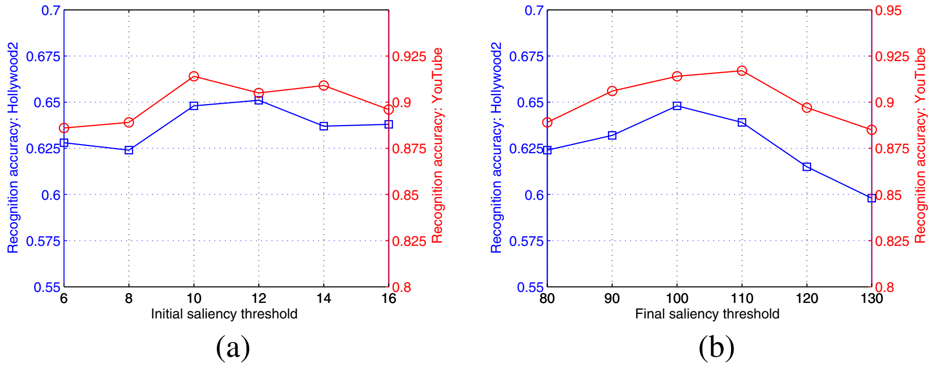

Fig. 7 Influence of saliency thresholds on recognition performance: **a** initial saliency threshold, **b** final saliency threshold

Table 4 Comparison with the state-of-the-art trajectory-based methods

Hollywood2		YouTube		HMDB51		UCF101	
Methods	Results(%)	Methods	Results(%)	Methods	Results(%)	Methods	Results(%)
Wang et al. [44]	59.9	Wang et al. [44]	85.4	Wang et al. [44]	48.3	Wang et al. [46]	85.9
Wang et al. [41]	64.3	Peng et al. [30]	87.6	Wang et al. [41]	57.2	Murthy et al. [28]	85.8
Wang et al. [40]	63.68	Wang et al. [40]	90.55	Murthy et al. [28]	58.8	Wang et al. [40]	85.69
Vig et al. [39]	61.9	Cho et al. [6]	86.1	Wu et al. [50]	56.36	Wu et al. [50]	84.16
Jain et al. [10]	62.5	Wu et al. [50]	90.82	Jiang et al. [11]	57.3	Jiang et al. [11]	87.2
Cho et al. [6]	60.5	Yao et al. [54]	85.9	Cai et al. [2]	55.9	Cai et al. [2]	83.5
Jiang et al. [11]	65.4	Weng et al. [47]	89.6	Weng et al. [47]	58.2		
		Yi et al. [53]	88.45	Li et al. [15]	56.7		
Ours	64.8	Ours	91.4	Ours	58.9	Ours	86.1

saliency detection or other techniques. On YouTube, our method performs the best among all the methods compared, which include the dense tracking method [44], its improved versions [6, 30, 40, 47, 53], and the methods employing improved VLAD [50] or Multi-view SC [54] to encode dense trajectory features. On HMDB51, our method performs competitively with the ordered trajectories [28], and is also superior to all other trajectory-based methods. On UCF101, our method gains the highest performance among all methods except [11].

6 Conclusion

To improve the conventional dense tracking techniques that do not discriminate between action-relevant regions and background when extracting features for action recognition, this paper proposes a three-stage saliency detection technique to recover action-relevant regions so that only action-related features are extracted to describe actions. Initial saliency is computed in the first stage based on LRMR to determine candidate foreground and definite background. Saliency is refined using SR in the second stage so that the true action-relevant regions are highlighted with sharper contrast against background. Saliency is spatially updated in the third stage to rectify some wrong saliency values through the enhancement of spatial saliency coherence. Then, binary saliency map is created to indicate action-relevant regions and incorporated into dense tracking to extract action-relevant trajectory features to describe actions. Experimental results show that the proposed method can detect action-relevant regions more accurately than common video saliency detection methods, and the recognition performance is superior to that of conventional dense tracking and competitive with that of existing improved versions.

Acknowledgements This work is supported in part by the National Natural Science Foundation of China (Grant No. 61572395) and the Project of Shandong Province Higher Educational Science and Technology Program (Grant No. J18KA345).

References

1. Bregonzio M, Li J, Gong S, Xiang T (2010) Discriminative topics modelling for action feature selection and recognition. In: Proceedings of British machine vision conference, pp 1–11
2. Cai Z, Wang L, Peng X, Qiao Y (2014) Multi-view super vector for action recognition. In: 2014 IEEE Conference on computer vision and pattern recognition (CVPR), pp 596–603
3. Candés EJ, Wakin MB, Boyd SP (2008) Enhancing sparsity by reweighted l1 minimization. J Fourier Anal Appl 14(5–6):877–905
4. Carreira J, Zisserman A (2017) Quo vadis, action recognition? A new model and the kinetics dataset. volume 2017-January, pp 4724–4733
5. Caruccio L, Polese G, Tortora G, Iannone D (2019) EDCAR: a knowledge representation framework to enhance automatic video surveillance. Expert Syst Appl 131:190–207
6. Cho J, Lee M, Chang HJ, So H (2014) Robust action recognition using local motion and group sparsity. Pattern Recogn 47(5):1813–1825
7. Dollar P, Rabaud V, Cottrell G, Belongie S (2005) Behavior recognition via sparse spatio-temporal features. In: IEEE International workshop on visual surveillance and performance evaluation of tracking and surveillance, pp 65–72
8. Gao Z, Cheong LF, Wang YX (2014) Block-sparse rpca for salient motion detection. IEEE Trans Pattern Anal Mach Intell 36(10):1975–1987
9. Jain H, Harit G (2018) Unsupervised temporal segmentation of human action using community detection. In: 25th IEEE International conference on image processing (ICIP), pp 1892–1896

10. Jain M, Jegou H, Bouthemy P (2013) Better exploiting motion for better action recognition. In: Proceedings of IEEE conference on computer vision and pattern recognition (CVPR), pp 2555–2562
11. Jiang YG, Dai Q, Liu W, Xue X, Ngo CW (2015) Human action recognition in unconstrained videos by explicit motion modeling. IEEE Trans Image Process 24(11):3781–3795
12. Kuehne H, Jhuang H, Garrote E, Poggio T, Serre T (2011) HMDB: a large video database for human motion recognition. In: Proceedings of IEEE international conference on computer vision (ICCV), pp 2556–2563
13. Laptev I (2005) On space-time interest points. Int J Comput Vis 64(2):107–123
14. Li X, Lu H, Zhang L, Ruan X, Yang MH (2013) Saliency detection via dense and sparse reconstruction. In: Proceedings of IEEE international conference on computer vision (ICCV), pp 2976–2983
15. Li Q, Cheng H, Zhou Y, Huo G (2016) Human action recognition using improved salient dense trajectories. Comput Intell Neurosci 2016(5):1–11
16. Lin Z, Chen M, Ma Y (2009) The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. Eprint Arxiv, 9
17. Liu J, Luo J, Shah M (2009) Recognizing realistic actions from videos in the wild. In: Proceedings of IEEE conference on computer vision and pattern recognition (CVPR), pp 1996–2003
18. Liu Y, Nie L, Han L, Zhang L, Rosenblum DS (2015) Action2Activity: recognizing complex activities from sensor data. In: Proceedings of the 24th international conference on artificial intelligence, pp 1617–1623
19. Liu L, Cheng L, Liu Y, Jia Y, Rosenblum DS (2016) Recognizing complex activities by a probabilistic interval-based model. In: Proceedings of 30th AAAI conference on artificial intelligence, pp 1266–1272
20. Liu Y, Nie L, Liu L, Rosenblum DS (2016) From action to activity: sensor-based activity recognition. Neurocomputing 181:108–115
21. Liu Z, Li J, Ye L, Sun G, Shen L (2017) Saliency detection for unconstrained videos using superpixel-level graph and spatiotemporal propagation. IEEE Trans Circ Syst Vid Technol 27(12):2527–2542
22. Lu Y, Wei Y, Liu L, Zhong J, Sun L, Liu Y (2017) Towards unsupervised physical activity recognition using smartphone accelerometers. Multimed Tools Appl 76(8):10701–10719
23. Lucas BD, Kanade T (1981) An iterative image registration technique with an application to stereo vision (darpa). Nutr Cycl Agroecosyst 83(1):13–26
24. Mairal J, Mairal J (2012) SPAMS: a sparse modeling software, v2.3. http://spams-devel.gforge.inria.fr
25. Marszalek M, Laptev I, Schmid C (2009) Actions in context. In: Proceedings of IEEE conference on computer vision and pattern recognition (CVPR), pp 2929–2936
26. Matikainen P, Hebert M, Sukthankar R (2009) Trajectons: action recognition through the motion analysis of tracked features. In: Proceedings of IEEE international conference on computer vision workshops, pp 514–521
27. Messing R, Pal C, Kautz H (2009) Activity recognition using the velocity histories of tracked keypoints. In: Proceedings of IEEE International conference on computer vision (ICCV), pp 104–111
28. Murthy OVR, Goecke R (2015) Ordered trajectories for human action recognition with large number of classes. Image Vis Comput, 22–34
29. Nigam S, Khare A (2016) Integration of moment invariants and uniform local binary patterns for human activity recognition in video sequences. Multimed Tools Appl 75(24):17303–17332
30. Peng X, Qiao Y, Peng Q (2014) Motion boundary based sampling and 3d co-occurrence descriptors for action recognition. Image Vis Comput 32(9):616–628
31. Perronnin F, Sánchez J, Mensink T (2010) Improving the fisher kernel for large-scale image classification. In: Proceedings of European conference on computer vision (ECCV), pp 143–156
32. Sánchez J, Perronnin F, Mensink T, Verbeek J (2013) Image classification with the fisher vector: theory and practice. Int J Comput Vis 105(3):222–245
33. Somasundaram G, Cherian A, Morellas V, Papanikolopoulos N (2014) Action recognition using global spatio-temporal features derived from sparse representations. Comput Vis Image Underst 123(0):1–13
34. Soomro K, Zamir AR, Shah M (2012) UCF101: a dataset of 101 human actions classes from videos in the wild CRCV-TR-12-01
35. Souly N, Shah M (2016) Visual saliency detection using group lasso regularization in videos of natural scenes. Int J Comput Vis 117(1):93–110
36. Sun J, Wu X, Yan S, Cheong LF (2009) Hierarchical spatio-temporal context modeling for action recognition. In: Proceedings of IEEE computer society conference on computer vision and pattern recognition (CVPR), pp 2004–2011
37. Sun J, Mu Y, Yan S, Cheong LF (2010) Activity recognition using dense long-duration trajectories. In: Proceedings of IEEE international conference on multimedia and expo (ICME), pp 322–327
38. Tong N, Lu H, Zhang Y, Ruan X (2015) Salient object detection via global and local cues. Pattern Recogn 48(10):3258–3267

39. Vig E, Dorr M, Cox D (2012) Space-variant descriptor sampling for action recognition based on saliency and eye movements. In: Proceedings of European conference on computer vision (ECCV), vol 7578, pp 84–97
40. Wang X, Qi C (2016) Saliency-based dense trajectories for action recognition using low-rank matrix decomposition. J Vis Commun Image Represent, 41
41. Wang H, Schmid C (2013) Action recognition with improved trajectories. In: Proceedings of IEEE international conference on computer vision (ICCV), pp 3551–3558
42. Wang H, Ullah MM, Kläser A, Laptev I, Schmid C (2009) Evaluation of local spatio-temporal features for action recognition. In: Proceedings of British machine vision conference (BMVC)
43. Wang J, Yang J, Yu K, Lv F, Huang T, Gong Y (2010) Locality-constrained linear coding for image classification. In: Proceedings of IEEE conference on computer vision and pattern recognition (CVPR), pp 3360–3367
44. Wang H, Kläser A, Schmid C, Liu C-L (2013) Dense trajectories and motion boundary descriptors for action recognition. Int J Comput Vis 103(1):60–79
45. Wang W, Shen J, Yang R, Porikli F (2018) Saliency-aware video object segmentation. IEEE Trans Pattern Anal Mach Intell 40(1):20–33
46. Wang H, Schmid C LEAR-INRIA submission for the thumos workshop. In: http://crcv.ucf.edu/ICCV13-Action-Workshop/
47. Weng Z, Guan Y (2018) Action recognition using length-variable edge trajectory and spatio-temporal motion skeleton descriptor. EURASIP J Image Video Process 2018(1):8
48. Wright J, Ganesh A, Rao S, Ma Y (2009) Robust principal component analysis: exact recovery of corrupted low-rank matrices via convex optimization
49. Wu S, Oreifej O, Shah M (2011) Action recognition in videos acquired by a moving camera using motion decomposition of lagrangian particle trajectories. In: Proceedings of IEEE conference on computer vision and pattern recognition (CVPR), pp 1419–1426
50. Wu J, Zhang Y, Lin W (2014) Towards good practices for action video encoding. In: 2014 IEEE Conference on computer vision and pattern recognition (CVPR), pp 2577–2584
51. Wu Y, Yin J, Wang L, Liu H, Dang Q, Li Z, Yin Y (2018) Temporal action detection based on action temporal semantic continuity. IEEE Access 6:31677–31684
52. Yan J, Zhu M, Liu H, Liu Y (2010) Visual saliency detection via sparsity pursuit. IEEE Signal Process Lett 17(8):739–742
53. Yang Y, Pan H, Xiaokang D (2018) Human action recognition with salient trajectories and multiple kernel learning. Multimed Tools Appl 77(14):17709–17730
54. Yao T, Wang Z, Xie Z, Gao J, Feng DD (2017) Learning universal multiview dictionary for human action recognition. Pattern Recogn 64(C):236–244