PAPER

Vote Distribution Model for Hough-Based Action Detection

Kensho HARA[†a)], ***Student Member***, **Takatsugu HIRAYAMA**[†,††b)], ***Member***, ***and*** **Kenji MASE**[†c)], ***Fellow***

SUMMARY Hough-based voting approaches have been widely used to solve many detection problems such as object and action detection. These approaches for action detection cast votes for action classes and positions based on the local spatio-temporal features of given videos. The voting process of each local feature is performed independently of the other local features. This independence enables the method to be robust to occlusions because votes based on visible local features are not influenced by occluded local features. However, such independence makes discrimination of similar motions between different classes difficult and causes the method to cast many false votes. We propose a novel Hough-based action detection method to overcome the problem of false votes. The false votes do not occur randomly such that they depend on relevant action classes. We introduce vote distributions, which represent the number of votes for each action class. We assume that the distribution of false votes include important information necessary to improving action detection. These distributions are used to build a model that represents the characteristics of Hough voting that include false votes. The method estimates the likelihood using the model and reduces the influence of false votes. In experiments, we confirmed that the proposed method reduces false positive detection and improves action detection accuracy when using the IXMAS dataset and the UT-Interaction dataset.

key words: *action detection, Hough transform, Hough forests*

1. Introduction

Automatically recognizing and detecting human action in videos are widely used in applications such as surveillance systems, video indexing, and human computer interaction. Action recognition methods classify an action throughout an entire video. A typical approach uses a holistic representation of a video scene, such as bag-of-words of local features [1]. Action detection methods search for all instances of action in a scene. Besides classifying actions, these methods localize actions both in space and time. Most applications deal with videos that contain multiple instances of action and detect each instance.

Most action detection methods are based on a sliding-window approach [2]–[10] or Hough transform [11]–[15]. Sliding-window approaches classify actions in each sub-volume of a video sequence while changing the spatio-temporal position of that sub-volume. By contrast, Hough-based approaches extract the local features of a video and then cast votes for action classes and spatio-temporal positions based on each local feature. The local maxima of voting scores indicate possible detected actions. Here, the voting scores are calculated by accumulating the votes at each position based on all local features. Compared with sliding window approaches, Hough-based approaches are robust to occlusions. Votes based on visible local features are not affected by occluded local features because the voting process of each local feature is performed independently from others. Our study focuses on Hough-based action detection.

The independence of the voting process causes a problem. If a similar local motion exists between different action classes, discriminating such motions in the voting process is difficult. It naturally votes for those relevant action classes. Therefore, Hough-based approaches are prone to cast many false votes for wrong action classes and, thus, detect many false positives.

In this study, we propose a novel method for overcoming the false-votes problem by examining the cause of false votes. Similar local features cause false votes. The false votes do not occur randomly such that they depend on relevant action classes. Hough-based approaches essentially cast votes not only for a certain class but also for other specific classes even when only an action is performed. These characteristics are different according to each class and do not necessarily correlate. We assume that the distribution of false votes include important information necessary to improving action detection. Our proposed method learns these characteristics of Hough voting. We introduce vote distributions, which represent the voting scores for each action class, as shown in Fig. 1. Our proposed method builds a model that represents those characteristics based on vote distributions. The method estimates the likelihood using the model and reduces the influence of false votes.

The main contribution of this paper is that our vote distribution model improves the performance of Hough-based action detection by reducing the influence of false votes caused by similar local motion between different action classes. Experimental results in this paper support the assertion.

The remainder of this paper is organized as follows. Section 2 reviews related studies and Sect. 3 describes conventional Hough-based action detection. We explain our proposed method in Sect. 4 and analyze our experimental results in Sect. 5. Section 6 concludes the study.

Manuscript received December 24, 2015.
Manuscript revised July 9, 2016.
Manuscript publicized August 19, 2016.

[†]The authors are with the Graduate School of Information Science, Nagoya University, Nagoya-shi, 464–8603 Japan.
[††]The author is with the Graduate Program for Real-World Data Circulation Leaders, Nagoya University, Nagoya-shi, 464–8603 Japan.
a) E-mail: kensho@cmc.ss.is.nagoya-u.ac.jp
b) E-mail: hirayama@is.nagoya-u.ac.jp
c) E-mail: mase@nagoya-u.jp

DOI: 10.1587/transinf.2015EDP7503

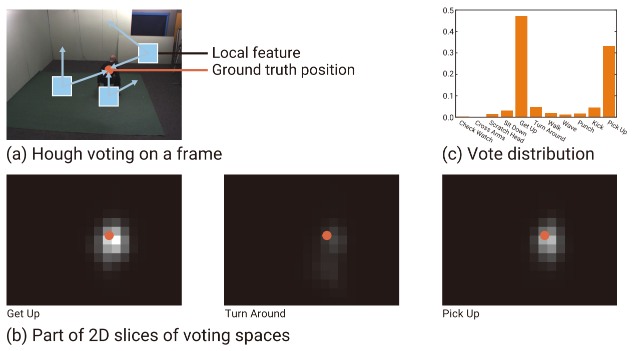

Fig. 1 Hough voting and vote distribution. (a) This frame is an example of *Get Up* class. A Hough-based approach casts votes based on local features of the frame. (b) The votes can be visualized as 2D slices of voting spaces. The pixel values of the slices mean the voting scores of the votes for each action class. (c) A vote distribution describes the voting scores for each action class at a given position. This vote distribution is calculated at the ground truth position indicated by the circles.

2. Related Work

In this section, we review studies related to this research. Related work includes object- and action detection methods. The object detection methods motivate many of the action detection methods.

As mentioned in previously, action detection methods are divided into sliding-window and Hough-based approaches. Motivated by the success of sliding-windows for object detection [16], many sliding-window-based action detection methods exist for domain adaptation [3], action descriptors [4], action representation [10], and discriminative representation in crowded scenes [2], [5]. Because the search spaces of action detection are larger than those of object detection, some research has attempted to reduce computational costs by using branch-and-bound [6] and coarse-to-fine searches.

In contrast to these sliding-window-based approaches, Hough-based approaches, which are the main focus of this research, avoid exhaustive searches in spaces and are robust to occlusions. Hough transform was initially proposed for line detection and then extended for general object detection [17]. Leibe et al. proposed the implicit shape model (ISM) that is now a commonly used Hough-based object detection method [18]. ISM generates a codebook of local features. It then uses the codebook to casts votes for object positions based on local features extracted from an image. Some studies have applied the ISM framework to action detection [11]–[15].

Hough-based approaches possess the problem of false votes. Many studies have been conducted that consider this problem for both object and action detection. One approach for overcoming the problem involves generating a discriminative codebook. Maji et al. [19] and Wohlhart et al. [20] proposed approaches to learn discriminative weights of the codebook using max-margin frameworks. Some studies have adopted a supervised manner to generate a codebook using random forests [21] and locality-constrained linear coding [15]. These studies have improved the training step that generates a codebook in order to achieve robustness to false votes. Our proposed method both constructs a vote distribution model in the training step and generates a codebook. In addition, the proposed method does not restrict the type of codebook generation method. We can combine the proposed method with the aforementioned methods.

Other studies have attempted to improve the voting process to solve the problem of false votes. Razavi et al. indicated that sparsity of local appearance is an effective measure for discriminating foreground and background features [22]. In the voting process, their method selects foreground features based on the sparsity measure and reduces the influence of background features. Their method is effective only for false votes generated by background features. Some studies have attempted to group local features to reduce the independence of Hough voting [23], [24]. They have improved the voting process to manage multiple dependent local features. In general, these studies have improved the voting process but only find the local maxima of votes at completion. Our proposed method calculates vote distributions including false votes from the voting space when the voting process is completed. In addition, the proposed method can be combined with these other methods that improve the voting process.

Woodford et al. optimized vote weights for each class by minimizing entropy in the voting space of each class separately when the voting process is completed [25]. They assumed that only one vote created by each local feature is correct. This assumption is flawed when background features exist that generate no correct votes. Their method would enlarge weights of false votes generated by background features in the optimization. Our proposed method is not affected by background features unless the features change vote distributions.

2 Similar to the vote distributions, Hoai et al. introduced relative class scores (RCS) for action recognition [26]. RCS is a vector representation of output scores of a multi-class action classifier. The main difference between the vector representing vote distributions and RCS is whether each method sorts the elements of vector. The scores would become independent from the action classes because of sorting so that RCS-based action recognition could not achieve satisfactory accuracy. We experimentally confirmed that the vote distributions are superior to RCS for Hough-based action detection.

A preliminary version of this study appeared in [27]. Voting trends described in [27] are based on the same concept as vote distributions. This study presents a detailed discussion of the concept of vote distributions. We also discuss conducted experiments and our evaluation of robustness to occlusions.

3. Hough-Based Action Detection

This section explains the conventional Hough-based action detection based on Hough forests [12]. Two steps are used to detect actions: training and detection. In Sect. 3.1, we explain the training step and in Sect. 3.2 we describe the detection step.

Before describing the training step, we must indicate the configurations of the Hough transform in this study. We define a spatio-temporal center position of an action as a ground-truth position based on the previous study [12], [14]. We use the 4D voting space consisting of 2D-spatial, 1D-temporal, and 1D-scale spaces. The scale space represents the height of the spatial bounding box of a given action. Other scale parameters such as the spatial aspect ratio and temporal duration are managed as fixed.

3.1 Training

During the training step, a Hough-based method generates a codebook of local features. In this section, we explain the codebook based on Hough forests [21]. However, please note that our proposed method can employ any codebook generation method, such as agglomerative clustering [18] and random projection trees [14].

The Hough forests-based method generates the codebook using random forests [28] that are ensemble classifiers composed of many decision trees. The codebook represents the relationship among local features, action classes, and positions.

Local features are extracted from action videos as training data. Each tree of the random forests is constructed from a set of local features $\mathcal{F} = \{\mathbf{f}_i = (\mathbf{I}_i, c_i, \mathbf{d}_i, \sigma_i, \tau_i, h_i)\}$, where i is an index of a local feature; $\mathbf{I}_i$ is a visual feature vector; c_i is the class label of an action; $\mathbf{d}_i \in \mathbb{R}^3$ is a spatio-temporal displacement vector from a feature position to an action position; σ_i and τ_i are spatial and temporal scales of a local feature, respectively; and h_i is the height of the spatial bounding box of a given action. $\mathbf{I}_i$ can be multi-channeled to accommodate multiple features (i.e., $\mathbf{I}_i = (\mathbf{I}_i^1, \mathbf{I}_i^2, \cdots, \mathbf{I}_i^F)$, where F is the number of feature channels). We use space-time interest points [29], [30] for local features in the experiments. Their details are described in Sect. 5.

Each nonleaf node of a tree employs the following binary test:

$$b_{f,q,r,o}(\mathbf{I}) = \begin{cases} 0 & \text{if } \mathbf{I}^f(q) < \mathbf{I}^f(r) + o \\ 1 & \text{otherwise,} \end{cases} \tag{1}$$

where f is a feature channel, q and r are the dimensions of $\mathbf{I}^f$, and o is an offset.

During training step, random forests iterate classification of the local features into two nodes. This iteration is continued until each node satisfies the termination criteria defined by the maximum depth or minimum number of local features in a node. Each leaf node stores information from the assigned features: $\{(c_i, \mathbf{d}_i, \sigma_i, \tau_i, h_i)\}$. Leaf node L can calculate the class probability $p(c \mid L)$ by the proportion of features for each class c.

Each node generates a set of binary tests with random values f, q, r, and o. We use two measures to select the most suitable binary test. The first measure is class uncertainty:

$$U_1(\mathcal{A}) = -|\mathcal{A}| \sum_c p(c \mid \mathcal{A}) \ln p(c \mid \mathcal{A}), \tag{2}$$

where $\mathcal{A}$ is a subset of local features, $|\cdot|$ denotes the number of elements in the set, and $p(c \mid \mathcal{A})$ is a class probability calculated by the proportion of features in $\mathcal{A}$ for each class c. The second measure is the uncertainty of the displacement vectors:

$$U_2(\mathcal{A}) = \sum_c \left(\sum_{\mathbf{d} \in \mathbf{D}_c^{\mathcal{A}}} \left\| \mathbf{d} - \overline{\mathbf{d}_c^{\mathcal{A}}} \right\|^2 \right), \tag{3}$$

where $\mathbf{D}_c^{\mathcal{A}}$ is the displacement vectors of the features of class c in $\mathcal{A}$ and $\overline{\mathbf{d}_c^{\mathcal{A}}}$ is an average vector of $\mathbf{D}_c^{\mathcal{A}}$.

Each node randomly chooses between these measures and selects binary test b^*, which minimizes uncertainty in the generated set. This minimization is represented by the following equation:

$$\arg\min_{b^*} \left(U\left(\{\mathbf{f} \mid b^* = 0\}\right) + U\left(\{\mathbf{f} \mid b^* = 1\}\right) \right), \tag{4}$$

where $\{\mathbf{f} \mid b^* = 0\}$ is the set of the local features that output 0 by the function b^*.

3.2 Detection

The method casts votes for action classes, positions, and scales to detect actions. These votes are cast in the 4D voting spaces. Consider that the local feature $\mathbf{f}_\mathbf{y}$ extracted from position $\mathbf{y} = [\mathbf{y}^\mathrm{s}, y^\mathrm{t}] \in \mathbb{R}^3$ results in leaf node $L_\mathbf{y}^k$ of tree k. Here, $\mathbf{y}^\mathrm{s} \in \mathbb{R}^2$ and $y^\mathrm{t} \in \mathbb{R}^1$ are the spatial and temporal positions of $\mathbf{y}$, respectively. A vote based on training local feature $\mathbf{f}_i$ that is assigned to the leaf node can be represented by the following equation:

$$\begin{aligned} \mathbf{v}(\mathbf{y}, i) &= \left[\mathbf{v}^\mathrm{s}(\mathbf{y}, i), v^\mathrm{t}(\mathbf{y}, i), v^\mathrm{h}(\mathbf{y}, i) \right], \\ \mathbf{v}^\mathrm{s}(\mathbf{y}, i) &= \mathbf{y}^\mathrm{s} + \frac{\sigma_\mathbf{y}}{\sigma_i} \mathbf{d}_i^\mathrm{s}, \\ v^\mathrm{t}(\mathbf{y}, i) &= y^\mathrm{t} + \frac{\tau_\mathbf{y}}{\tau_i} d_i^\mathrm{t}, \\ v^\mathrm{h}(\mathbf{y}, i) &= \frac{\sigma_\mathbf{y}}{\sigma_i} h_i, \end{aligned} \tag{5}$$

where $\mathbf{v}^\mathrm{s}$, v^t, and v^h are votes for the spatial, temporal, and scale positions; $\sigma_\mathbf{y}$ and $\tau_\mathbf{y}$ are the spatial and temporal scales of feature $\mathbf{f}_\mathbf{y}$; and $\mathbf{d}_i^\mathrm{s}$ and d_i^t are the spatial and temporal displacement vectors of $\mathbf{d}_i$, respectively. The votes for class c based on $L_\mathbf{y}^k$ can be defined as:

$$\mathcal{V}_c^{L_\mathbf{y}^k} = \left\{ \mathbf{v}(\mathbf{y}, i) \,\middle|\, i \in \mathcal{I}_c^{L_\mathbf{y}^k} \right\}, \tag{6}$$

where $\mathcal{I}_c^{L_\mathbf{y}^k}$ denotes the indices of the training data of class c assigned to leaf node $L_\mathbf{y}^k$.

Using the votes, the voting score of an action of class c at position $\mathbf{x} \in \mathbb{R}^4$ can be defined as:

$$V\left(c, \mathbf{x} \,\middle|\, \mathcal{V}^{L_{\mathbf{y}}^k}\right) = \left(\frac{1}{\left|\mathcal{V}_c^{L_{\mathbf{y}}^k}\right|} \sum_{\mathbf{v} \in \mathcal{V}_c^{L_{\mathbf{y}}^k}} G(\mathbf{x}, \mathbf{v})\right) p\left(c \,\middle|\, L_{\mathbf{y}}^k\right), \tag{7}$$

$$G(\mathbf{x}, \mathbf{v}) = \exp\left(\frac{-\|\mathbf{x}^{\mathrm{s}} - \mathbf{v}^{\mathrm{s}}\|^2}{w^{\mathrm{s}}}\right) \times \exp\left(\frac{-\|x^{\mathrm{t}} - v^{\mathrm{t}}\|^2}{w^{\mathrm{t}}}\right) \times \exp\left(\frac{-\|x^{\mathrm{h}} - v^{\mathrm{h}}\|^2}{w^{\mathrm{h}}}\right) \tag{8}$$

where $\mathcal{V}^{L_{\mathbf{y}}^k}$ is a set of the votes for all classes based on $L_{\mathbf{y}}^k$ and $p\left(c \,\middle|\, L_{\mathbf{y}}^k\right)$ is a class probability calculated by the proportion of local features assigned to $L_{\mathbf{y}}^k$ for each class c. Here, G is the 4D Gaussian kernel function with three kernel bandwidths for space w^{s}, time w^{t}, and scale w^{h}. Using the following equation, the voting score based on all votes can be defined as

$$V(c, \mathbf{x} \,|\, \mathcal{V}) = \sum_{\mathbf{y} \in \mathbf{Y}} \left(\frac{1}{K} \sum_{k=1}^{K} V\left(c, \mathbf{x} \,\middle|\, \mathcal{V}^{L_{\mathbf{y}}^k}\right)\right), \tag{9}$$

where $\mathcal{V}$ is the votes based on all local features and trees, $\mathbf{Y}$ is the set of positions of the local features, and K is the number of trees. The method finds local maxima of Eq. (9) for each action class independently. The local maxima specify the action positions of the corresponding action class in the 4D voting space that consists of 2D-spatial, 1D-temporal, and 1D-scale spaces.

Each vote is calculated based on one local feature as described in this section. One local feature describes only a local region in the given video. Discriminating the local features is difficult if a similar local motion exists between different action classes. Therefore, the Hough-based method is prone to cast many false votes for wrong action classes. In the next section, we show the manner in which our proposed method reduces the influence of false votes.

4. Hough-Based Action Detection with Vote Distribution Model

We introduce the vote distribution model to the conventional Hough-based method in order to improve robustness to false votes. Our proposed method reduces the influence of false votes by learning the characteristics of Hough voting. Figure 3 shows the flow of our proposed method. We explain vote distributions in Sect. 4.1. In Sect. 4.2, we show the training step of the vote distribution model. Finally, we describe the detection step of our proposed method in Sect. 4.3.

4.1 Vote Distribution

We represent the characteristics of Hough voting using the voting scores for all action classes. Similar local features generate false votes but the false votes do not occur randomly such that they depend on relevant action classes. The conventional Hough-based approaches basically cast votes for not only a certain class but also other specific classes, as shown in Fig. 2. We define the normalized voting scores for all classes at a position as a vote distribution. The vote distribution represents characteristics of the voting. Specifically, the vote distribution at position $\mathbf{x} \in \mathbb{R}^4$ can be represented by the following equation:

$$\mathbf{V}(\mathbf{x}) = \left[\frac{V\left(c^1, \mathbf{x} \,\middle|\, \mathcal{V}\right)}{Z}, \ldots, \frac{V\left(c^N, \mathbf{x} \,\middle|\, \mathcal{V}\right)}{Z}\right],$$
$$Z = \sum_{j=1}^{N} V\left(c^j, \mathbf{x} \,\middle|\, \mathcal{V}\right), \tag{10}$$

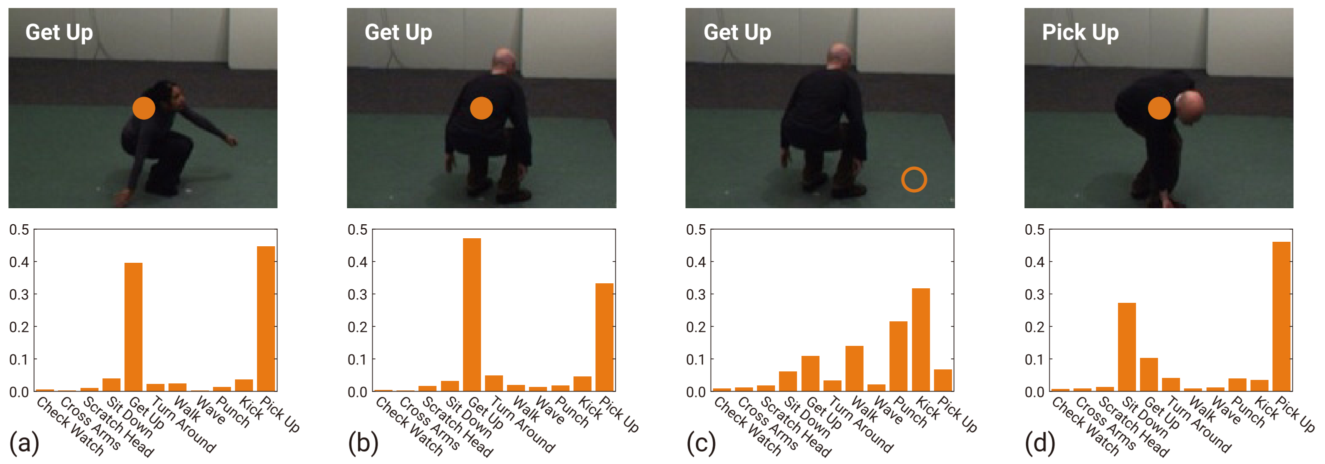

Fig. 2 Examples of vote distributions calculated at circle positions. The filled and open circles indicate the ground truth and other positions, respectively. (a), (b), and (c) are examples of the *Get Up* class, and (d) is an example of the *Pick Up* class. The vote distributions of only (a) and (b) are similar. These examples reveal that the calculated vote distributions at the ground truth positions are similar if they are of the same class.

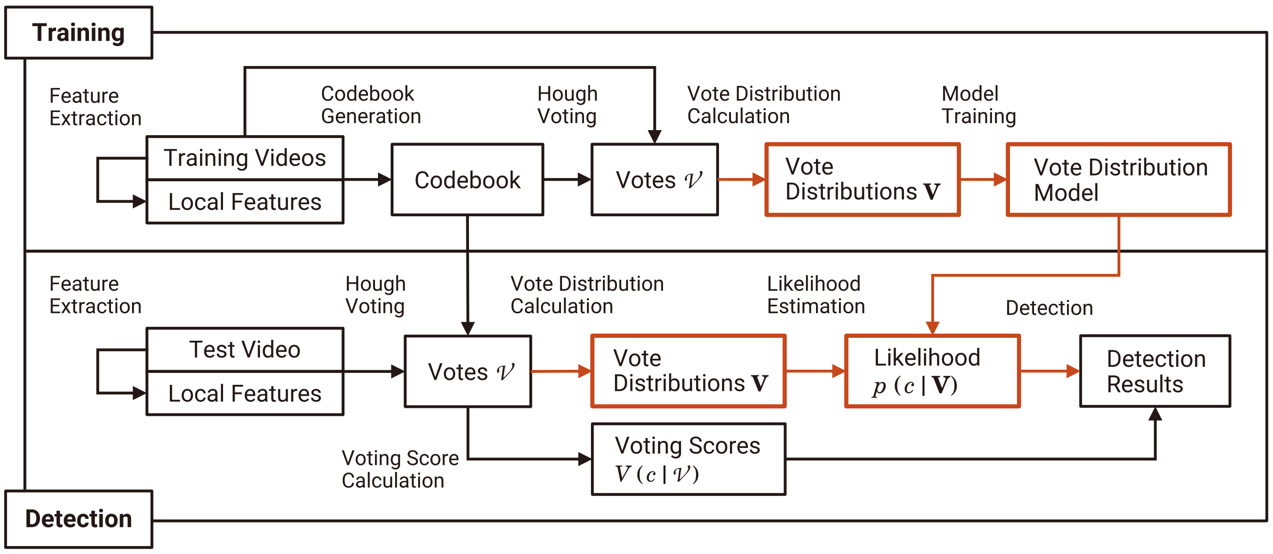

Fig. 3 Flow of our proposed method. The orange boxes are our proposed elements.

where N is the number of action classes and Z is the normalization constant.

Figure 2 shows the characteristics of Hough voting represented by the vote distribution. A vote distribution at a ground truth position of an action has a high score on the correct class. The distribution also has a high score on classes having similar local motions. In this figure, the distribution of the *Get Up* class shows high scores on *Get Up* and *Pick Up* classes, whereas the distribution of *Pick Up* class shows high scores on *Sit down* and *Pick Up* classes. The characteristics are different depending on each class and position. They do not necessarily correlate. The proposed method builds a vote distribution model by learning these characteristics.

4.2 Training

To reduce the influence of false votes, our proposed method builds a vote distribution model that represents the vote distributions of each class. Consider false votes for wrong action classes generated by similar motions. If votes for the correct action class and other specific wrong action classes exist in the training step, we can estimate that the likelihood for the wrong action classes is low based on the trained characteristics. This estimation enables us to reduce the influence of false votes.

To build the model that estimates the likelihood based on vote distributions, we use a vote distribution as a feature vector. A multi-class classifier trained by using these vectors can work as a vote distribution model and output the likelihood $p(c \mid \mathbf{V})$ using vote distribution $\mathbf{V}$.

Before extracting the vectors, the proposed method prepares the codebook described in Sect. 3.1. The method then executes the conventional voting process described in Sect. 3.2 on videos from the training data. We define vote distributions at the ground truth and surrounding position as positive data, and vote distributions far from the ground-truth position as negative data. The proposed method builds the vote distribution model using the positive and negative data. We note that the training videos for the codebook should be different from those for the model. If they were the same, some votes are cast to the ground truth position perfectly. Positive data generated using such votes causes overfitting of the vote distribution model.

In this study, we use random forests as the classifier. Therefore, a vote distribution model is a nonparametric model based on random forests. The model calculates the likelihood $p(c \mid \mathbf{V})$ based on vote distribution $\mathbf{V}(\mathbf{x})$. Here, the random forests use vote distributions instead of local features. The random forests adopt only the class uncertainty measure of Eq. (2) because the vote distributions do not have displacement vectors.

4.3 Detection

In the detection step, our proposed method initially calculates votes $\mathcal{V}$ using the conventional voting process. We can then calculate the vote distributions using Eq. (10) and estimate the likelihood based on the distribution. The proposed method detects actions using both the voting scores and the likelihood. The proposed method multiply the scores by the likelihood. Low likelihood for wrong action classes estimated by the model modulates the voting scores accumulated by false votes. The multiplied voting score of an action of class c at position $\mathbf{x}$ can be defined as:

$$V(c, \mathbf{x} \mid \mathcal{V}, \mathbf{V}(\mathbf{x})) = V(c, \mathbf{x} \mid \mathcal{V})\, p(c \mid \mathbf{V}(\mathbf{x})), \qquad (11)$$

The proposed method finds local maxima of Eq. (11) for each action class independently. The local maxima specify the action positions of the corresponding action class in the 4D voting space.

Figure 4 shows an example of reducing the effect of false votes when using the proposed method. The first row

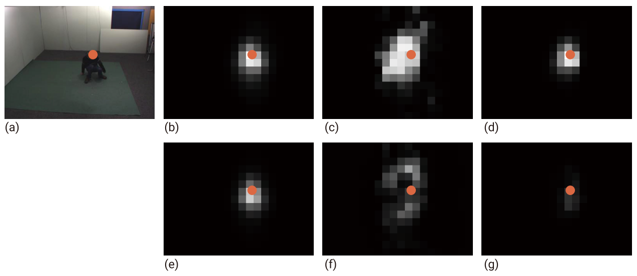

Fig. 4 Reducing the influence of false votes. The first row shows a video frame, the voting scores, the likelihood based on the vote distribution, and the product of the voting scores and likelihood of the *Get Up* class. The second row lists the voting scores, likelihood, and product of the *Pick Up* class. (a) is a video frame from *Get Up*. The circles indicate the ground truth positions of the action in the video. The second column visualizes 2D slices of voting spaces. The third column shows the estimated likelihood based on the vote distribution. The fourth column shows the product of the second and third columns. (g) shows that our proposed method reduces the influence of false votes in (e) whereas the correct votes in (b) remain in (d).

of the figure shows a video frame, the voting scores, the likelihood based on the vote distribution, and the product of the voting scores and likelihood of the *Get Up* class. The second row of the figure provides the voting scores, likelihood, and product of the *Pick Up* class. The second, third, and fourth columns correspond to $V(c, \mathbf{x} \mid \mathcal{V})$ in Eq. (9), $p(c \mid \mathbf{V}(\mathbf{x}))$, and $V(c, \mathbf{x} \mid \mathcal{V}, \mathbf{V}(\mathbf{x}))$ in Eq. (11), respectively. The pixel values of (b) denote the effect of correct votes for the *Get Up* class. The effect remains in (d) by using the high values estimated by the proposed method in (c). By contrast, the pixel values of (e) refer to the influence of false votes for the *Pick Up* class while the *Get Up* class is performed. The influence is reduced in (g) by using the low values estimated by the proposed method in (f). This kind of reduction enhances the robustness of the proposed method to false votes.

High values exist at the nonground truth positions of the correct class and the positions of the wrong class, as shown in Fig. 4 (c) and (f), respectively. These values represent to noise caused by estimation based on vote distributions. This kind of noise often occurs at positions having extremely low voting scores, such as at values far from the ground truth positions shown in (b) and (e). Because the vote distributions are the normalized voting scores as shown in Eq. (10), a slight difference in extremely low voting scores causes considerable variation of vote distributions by normalization. Therefore, estimation at such positions is incorrect. However, such noise does not influence detection performance considerably because such noise is multiplied by extremely low voting scores.

Our proposed method assumes that the conventional voting process works well. However, even if the voting process performs well, false votes caused by similar motions lead to false detections. The proposed method reduces the influence of such false votes to improve detection performance.

5. Experiments

We evaluated our proposed method using two public action datasets: INRIA Xmas motion acquisition sequences (IXMAS) dataset [31] and UT-Interaction dataset [32].

In experiments, we compared the proposed method with three baseline methods and a related method proposed by Hoai et al. [26]. The first method is the conventional Hough-based action detection described in Sect. 3. The second method adds nonmaximum suppression over action classes to the first method. The voting score of the second method can be defined as

$$V^{\max}(c, \mathbf{x} \mid \mathcal{V}) = \begin{cases} V(c, \mathbf{x} \mid \mathcal{V}) & \text{if } c = c^{\max}_{\mathbf{x}} \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where $c^{\max}_{\mathbf{x}} = \operatorname{argmax}_c V(c, \mathbf{x} \mid \mathcal{V})$. We adopted the second method because the voting score of the correct class is likely greater than that of other classes at its ground truth position as shown in Fig. 2. If the voting score of the detection class is not larger than that of other classes at its detection position, the detection is likely a false positive. The second method removes such detections. The third method uses the vote distribution model. During the detection step, the third method only uses the likelihood based on a vote distribution whereas the proposed method uses both the voting scores and likelihood. The related method uses the RCS similar to vote distributions. RCS vectors are designed in a one-vs-all manner. A first element of a RCS vector is the voting score of the target action class. All other elements of the vector are the voting scores sorted in the descending order for other action classes. Therefore, RCS vectors identify only target class whereas our vote distributions identify all classes by original order of the classes. The method estimates the likelihood based on RCS vectors.

We used space-time interest points [29] and histograms of oriented (spatial) gradient/histograms of optical flow (HOG/HOF) descriptors [30] for the local features of both the proposed and the baseline methods. The parameter settings were as follows: spatial scales $\sigma^2 = \{2, 4, 8, 16, 32, 42\}$ and temporal scales $\tau^2 = \{2, 4\}$. The dimensions of the HOG and HOF descriptors were 72 and 90, respectively.

The Hough-based methods must find the local maxima after calculating the voting scores, such in Eqs. (9), (11), and (12). In the experiments, we used quick shift [33] to identify the initial coarse local maxima. We then refined the local maxima using mean shift [34].

The local maxima indicated the spatio-temporal positions and heights of the spatial bounding boxes of actions. To generate spatio-temporal volumes from the local maxima, we require aspect ratios and temporal durations. We used average values for the aspect ratios and temporal durations from ground truth labels of training data.

We evaluated the methods using f-score. A detection is correct when the detection class label is correct and the overlap ratio between the detection volume and ground truth volume is greater than 0.5. We adopted the intersection-over-union criterion for the overlap ratio.

5.1 Datasets

The IXMAS dataset [31] includes 11 actions: *Check Watch*, *Cross Arms*, *Scratch Head*, *Sit Down*, *Get Up*, *Turn Around*, *Walk*, *Wave*, *Punch*, *Kick*, and *Pick Up*. Each action was performed three times by 10 actors and recorded by five cameras. The action orientations differ from one actor to another. The dataset provides ground truth labels that include a time interval and human silhouettes for each action execution. We defined the spatio-temporal center of the time interval and the bounding boxes generated using the silhouettes as the ground truth of the action positions. Figure 5 shows examples from the dataset. The resolution and frame rate of the videos are 390×291 pixels and 23 fps, respectively.

The UT-Interaction dataset [32] contains videos of the continuous execution of six action classes: *Shake Hands*, *Hug*, *Kick*, *Point*, *Punch*, and *Push*. The dataset contains 20 sequences, including 162 action executions. The dataset provides ground truth labels that include a time interval and a bounding box for each action execution. We adopted the same definition for the ground truth of the action position as for the IXMAS dataset. Figure 6 shows examples from the dataset. The resolution and frame rate of the videos are 720×480 pixels and 30 fps, respectively. In contrast to the IXMAS dataset, the UT-Interaction dataset includes simultaneous occurrence of multiple actions.

5.2 Results

5.2.1 IXMAS Dataset

We employed leave-one-actor-out cross validation that uses the data of one actor as the test data and the remainder as the training data. In the training step, the proposed and third baseline methods built a vote distribution model. To avoid overfitting described in Sect. 4.2, we divided the training data using the leave-one-actor-out strategy in each validation iteration. We generated a codebook using the larger data and generated the training data for the vote distribution model using the remainder. This generation was repeated while changing the division.

Table 1 lists the f-score averaged over all cameras and Fig. 8 shows example output from the proposed method. *Standard*, *Max*, and *VD Only* refer to the first, second, and the third baseline methods, respectively. *Hoai* is the related method using RCS [26]. *Avg* is the f-score averaged over all 11 classes. The proposed method achieves the highest results compared to the comparative methods. These results indicate that using vote distributions improves action detection performance. The performance of *Max* is superior to that of *Standard* but inferior to that of the proposed method. *Max* only considers whether the voting score for a class is the maximum over all action classes. By contrast, our proposed method uses the distribution of the voting scores over action classes. Therefore, the class that has a maximum score is insufficient, and the distribution is essential for improved action detection performance.

VD Only performs the worst, whereas our proposed method performs the best. *VD Only* uses the vote distribution model in the same manner as the proposed method. The difference between the two methods is based on whether the

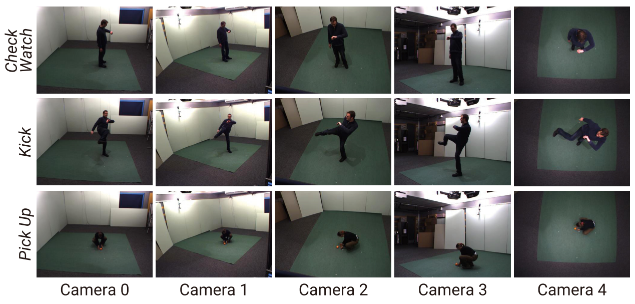

Fig. 5 Examples from the IXMAS dataset.

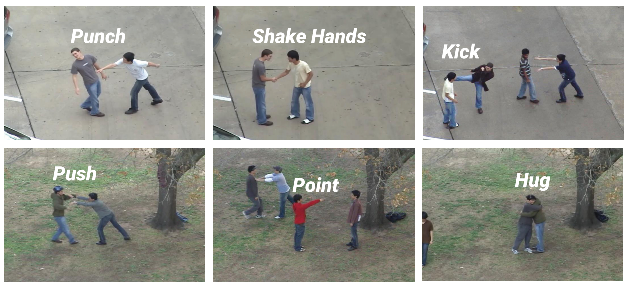

Fig. 6 Examples from the UT-Interaction dataset.

Table 1 Average f-score for all cameras on the IXMAS dataset

Method	Check Watch	Cross Arms	Scratch Head	Sit Down	Get Up	Turn Around	Walk	Wave	Punch	Kick	Pick Up	Avg
Standard	0.765	0.762	0.562	0.741	0.700	0.922	0.931	0.568	0.573	0.854	0.656	0.730
Max	0.784	0.792	0.622	0.820	0.745	0.959	0.931	**0.614**	**0.637**	0.857	0.702	0.769
VD Only	0.418	0.284	0.162	0.475	0.502	0.639	0.529	0.245	0.156	0.452	0.345	0.383
Hoai	0.780	0.795	0.607	0.819	0.747	0.945	0.956	0.589	0.577	0.844	0.696	0.759
Proposed	**0.818**	**0.810**	**0.656**	**0.863**	**0.782**	**0.963**	**0.967**	0.588	0.619	**0.862**	**0.747**	**0.789**

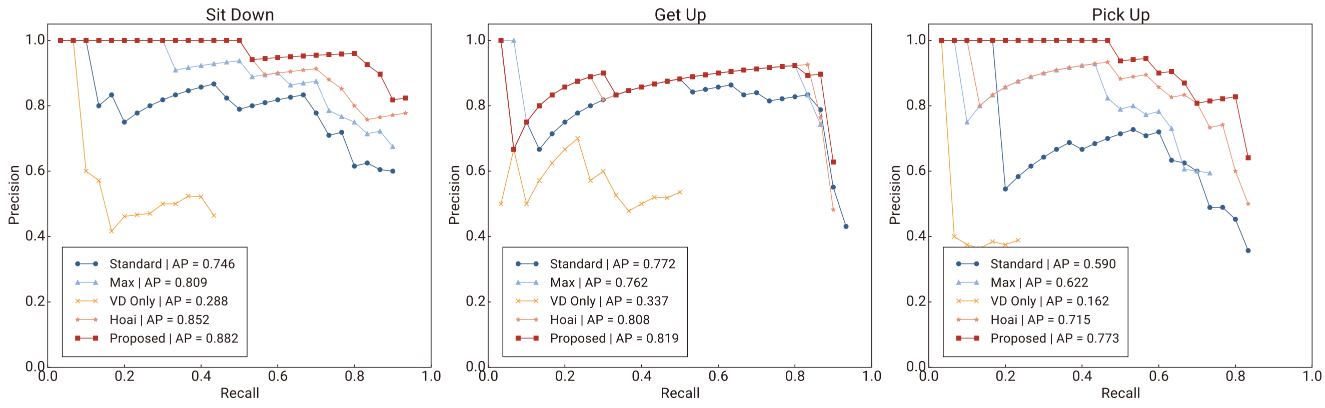

Fig. 7 Precision-recall curves of Camera 0 in the IXMAS dataset. AP means the average precision.

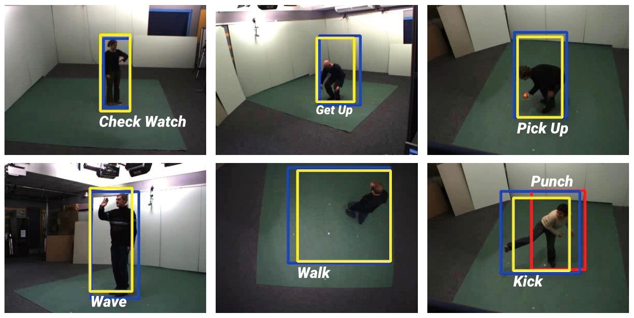

Fig. 8 Output examples of our proposed method on the IXMAS dataset. Yellow, red, and blue rectangles indicate true positives, false positives, and ground truths, respectively.

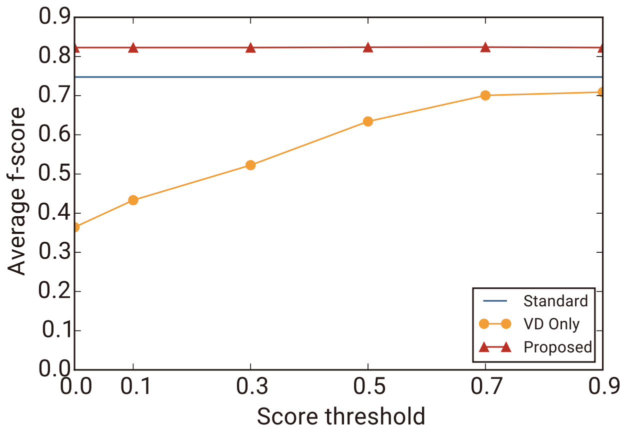

Fig. 9 F-score averaged over all action classes as a function of a score threshold on the IXMAS dataset.

method uses the voting scores as well as the model. The worst performance would be based on the incorrect estimation described in Sect. 4.3. Normalization of the vote distribution would generate an incorrect estimation at the positions with extremely low voting scores. To avoid the influence of the noise, we can use threshold for the voting scores. If the sum of the voting scores at a position over all action classes is lower than the threshold, the method does not estimate the likelihood based on the vote distribution and outputs zero probabilities for all action classes. Note that this threshold is different from the threshold for detection, and only decides whether the estimation is performed or not. Figure 9 shows the results of the methods using thresholding. The performance of *VD Only* improves as the score threshold increases. This result indicates that thresholding reduces the influence of the incorrect estimation. By contrast, the performance of our proposed method does not vary based on thresholding. The proposed method multiplies the voting scores by the likelihood. The proposed method reduces the influence of the incorrect estimation using the low voting scores. Therefore, multiplication is essential to improving action detection performance.

Our proposed method also achieves a higher f-score, outperforming *Hoai*. This result indicates that the vote distribution representation is superior to the RCS for Hough-based action detection. Figure 11 shows examples of vote distributions and RCS. The vote distributions for *Get Up* and *Pick Up* have different characteristics (see (a) and (b)). By contrast, RCSs for the two classes are similar because of sorting (see (c) and (d)). Therefore, class-conscious vote distributions are more effective than RCS.

Table 2 lists the f-scores of each camera averaged over all 11 classes. The proposed method outperforms the comparative methods except with respect to Camera 4. The results of all methods from Camera 4 are low. Camera 4 captured a person from an overhead location, but the motion captured from this location does not possess sufficient visual features for some classes that contain upward and downward motions, such as *Sit Down* and *Get Up*. Moreover, the aspect ratios in Camera 4 varied significantly depending on the direction in which the person is facing. We fixed the aspect ratio of the detection volume at the average value over all ground truth labels in each camera of the dataset. This fixed value cannot adapt to variation. The upward and downward motions and the fixed value cause low f-scores.

Figure 7 shows the precision-recall curves of the methods. The IXMAS dataset has 11 classes and 5 cameras. The number of precision-recall curves for each class and each camera is too large for visualization of the results. We selected three curves: *Sit Down*, *Get Up*, and *Pick Up* classes of Camera 0. Compared with the comparative methods, our proposed method achieves high precision for each recall value. These results indicate that the proposed method is effective for reducing false detections.

Figure 10 shows the confusion matrices for *Standard* and *Proposed*. For example, the value of row *Check Watch* and column *Cross Arms* refers to the ratio of the number of detections in which *Check Watch* was detected while *Cross Arms* is performed. *Standard* reveals some false detections between certain classes. For instance, the method detected *Sit Down* while *Pick Up* is performed. This false detection might be caused by similar stooping motions that are part of *Sit Down* and *Pick Up*. In our experiments, the proposed method reduced such false detections and improved detection accuracy.

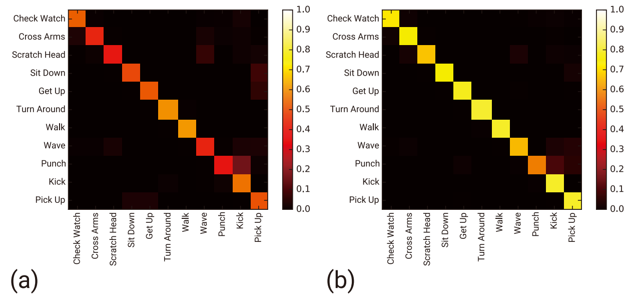

Fig. 10 Confusion matrix of: (a) *Standard* and (b) *Proposed* on the IXMAS dataset.

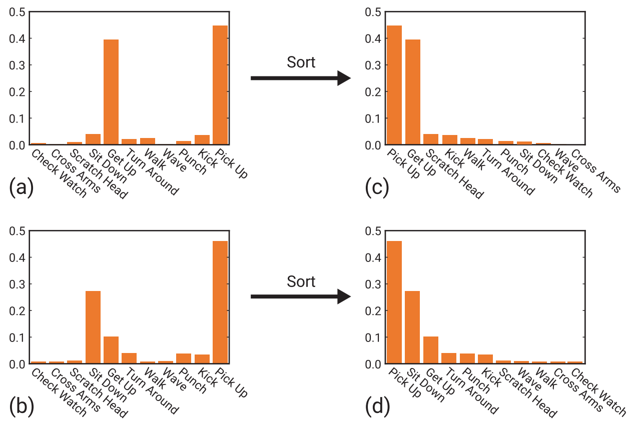

Fig. 11 Examples of vote distributions and RCS on the IXMAS dataset. (a) and (b) are proposed vote distributions of *Get Up* and *Pick Up*, respectively. (c) and (d) are RCS of *Get Up* and *Pick Up*, respectively.

Table 2 Average f-score over all classes of each camera on the IXMAS dataset

Method	Cam0	Cam1	Cam2	Cam3	Cam4
Standard	0.748	0.752	0.732	0.806	0.615
Max	0.784	0.782	0.765	0.852	**0.664**
VD Only	0.418	0.284	0.162	0.475	0.502
Hoai	0.789	0.805	0.715	0.854	0.634
Proposed	**0.823**	**0.824**	**0.783**	**0.872**	0.642

5.2.2 UT-Interaction Dataset

We employed 10-fold cross-validation that uses the data of two sequences as test data and the remainder as training data. The dataset contains motions that are not labeled. We used such motions as those in the *Others* class for the training. For detection, we did not cast votes for any class if the local features were classified into the *Others* class. For the training of a vote distribution model, we divided the training data as in the previous experiment. The 10-fold strategy was used for the division instead of the leave-one-actor-out strategy.

Table 3 lists the f-scores and Fig. 12 shows examples of output when using the proposed method. *Avg* is the f-score averaged over all six classes. *Proposed* achieves the highest f-score compared to those from the comparative methods. Figure 13 shows the precision-recall curves of the methods. Our proposed method achieves high precision for most recall values compared with the comparative methods, as well as the IXMAS dataset. These results show that the proposed method is also effective when multiple actions occur simultaneously.

Figure 14 shows the confusion matrices for *Standard*

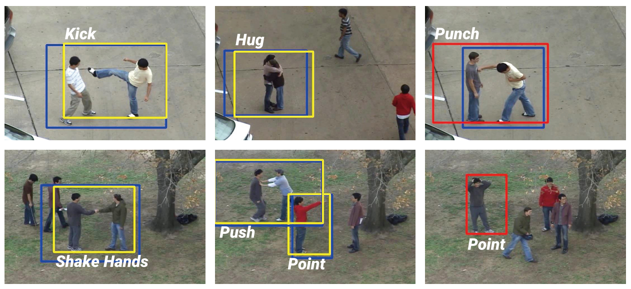

Fig. 12 Output examples of our proposed method on the UT-Interaction dataset. Yellow, red and blue rectangles indicate true positives, false positives, and ground truths, respectively.

Table 3 F-score on the UT-Interaction dataset

Method	Shake Hands	Hug	Kick	Point	Punch	Push	Avg
Standard	0.750	0.894	0.698	0.645	0.327	0.710	0.670
Max	0.808	0.889	0.667	0.656	0.258	0.714	0.665
VD Only	0.576	0.680	0.294	0.471	0.128	0.491	0.440
Hoai	0.807	**0.909**	0.698	0.636	**0.429**	0.733	0.702
Proposed	**0.873**	0.870	**0.700**	**0.657**	0.313	**0.833**	**0.708**

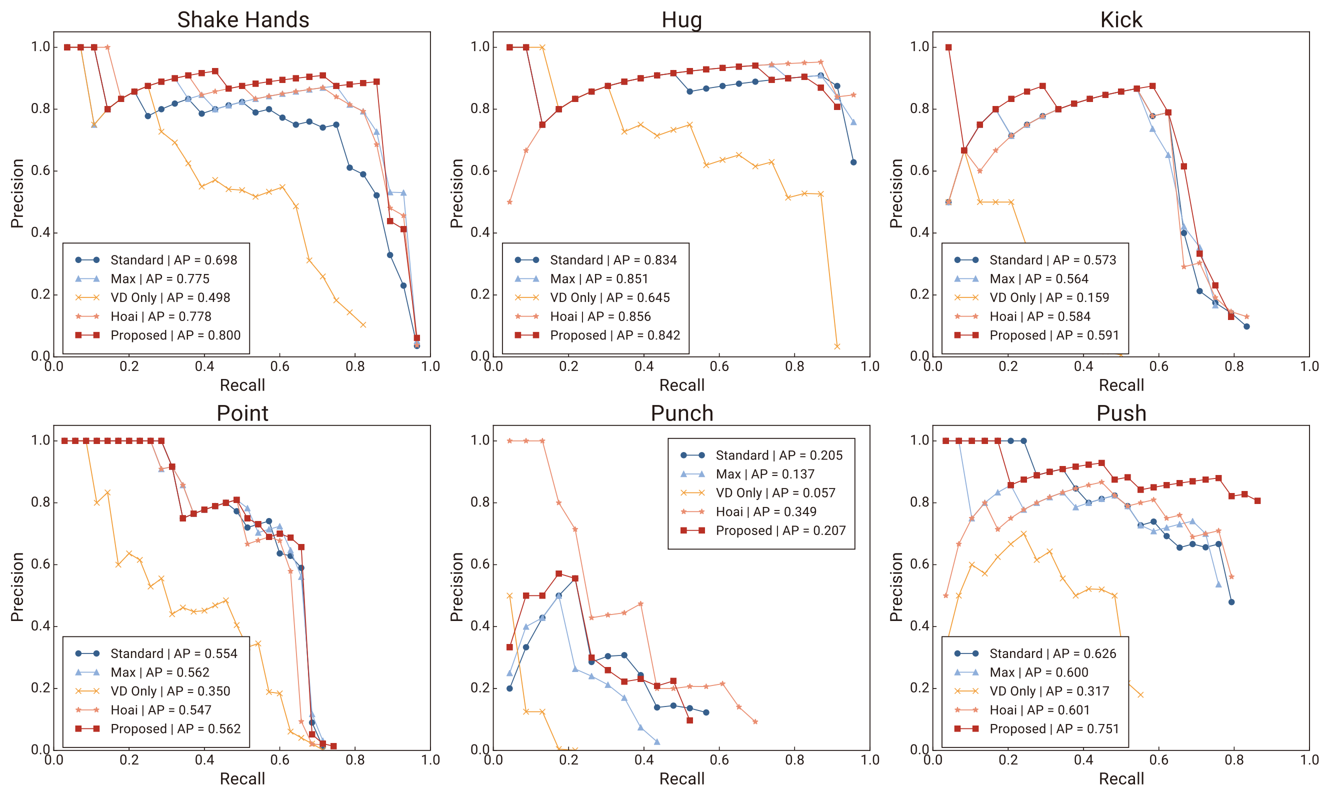

Fig. 13 Precision-recall curves in the UT-Interaction dataset. AP means the average precision.

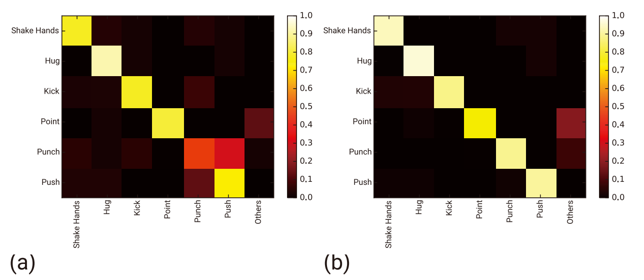

Fig. 14 Confusion matrix of: (a) *Standard* and (b) *Proposed* in the UT-Interaction dataset.

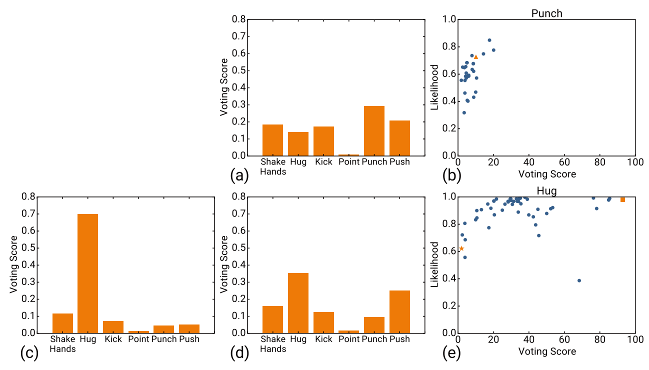

Fig. 15 Voting scores and likelihood based on vote distribution. Each data point in (b) and (e) represents a local maximum around ground truth positions for each class. (a) is the vote distribution of local maximum described by the orange triangle in (b). (c) and (d) are the vote distributions of local maxima described by the orange square and star in (e), respectively.

and *Proposed*. In our experiments, *Standard* detected *Push* when *Punch* is performed frequently. This false detection might be caused by similar arm motions that are part of *Push* and *Punch*. Similarly with respect to the IXMAS dataset, the proposed method reduced the numbers of false detection for actions, including similar motions in the UT-Interaction dataset.

The f-score of the proposed method for *Punch* is lower than that of *Standard*. This result would be caused by the situation only when correct votes are less in number. The vote distribution including such correct votes flattens out as shown in Fig. 15 (a). When the distribution flattens out, the difference in the frequency among action classes is small. The likelihood for correct action class based on such vote distribution tends to be low. Figure 15 (b) and (e) show the relation between original voting score and likelihood based on a vote distribution. Each data point in these figures represents a local maximum around ground truth positions for the corresponding class. Figure 15 (b) shows that the original voting scores for *Punch* are basically low. According to the low score, the likelihood for *Punch* is low. The proposed method degraded the voting score of the local maximum by multiplying the likelihood. The threshold for voting scores should be decreased to detect the degraded local maxima as true positives. The lower threshold would cause additional false positives so that it hurt the f-score for *Punch*.

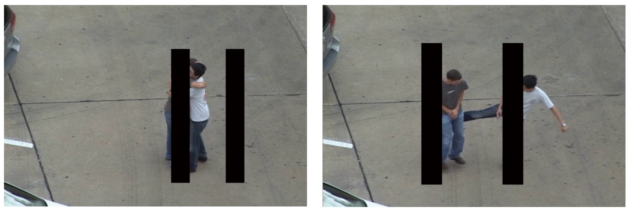

Fig. 16 Examples from the UT-Interaction dataset with artificial occlusions. The number of occluded regions in the examples is two.

Similarly, *Standard* outperformed our proposed method for *Hug*. Vote distributions of the local maxima that have high voting scores (i.e. the number of correct votes is large) are not flat but peaky as shown in Fig. 15 (c). The likelihood for *Hug* based on such distribution tends to be high. However, some local maxima have flat vote distributions as shown in Fig. 15 (d). Figure 15 (e) shows that such local maximum has low voting score and the likelihood for *Hug* based on such distribution tends to be low. Therefore, similar to *Punch*, the proposed method hurt the f-score for *Hug* because the multiplied voting score became lower.

Our proposed method performed better than *Hoai* except for *Hug* and *Punch*. As mentioned above, the proposed method did not improve the f-score for the two classes. This result also relates to above discussion.

The proposed method might degrade robustness to occlusions in the Hough-based method. If actions are occluded, the vote distributions are varied and this variation might cause poor estimation. Here, we evaluated robustness to occlusions using the UT-Interaction dataset with artificial occlusions and we generated artificial occlusions for each action. Figure 16 shows examples of the occlusions. We divided equally the width of the bounding box of each action into 10 regions. We then randomly chose regions as occluded. We used original local features in the training step. In the detection step, we removed the local features in the occluded regions. Other settings were the same as in the previous experiment that used the UT-Interaction dataset.

Figure 17 shows the f-score using the dataset with the artificial occlusions. The results when the number of occluded regions is zero are the same as those in Table 3. The f-score of both *Standard* and *Proposed* methods did not decrease significantly as the number of occluded regions increased. Therefore, the proposed method remains robust to occlusions and improves detection accuracy.

5.3 Limitations

We assume that our proposed method is applied to fixed-action categories. If the categories change, we have to rebuild not only a vote distribution model but also random forests for Hough voting. In addition, the number of categories might influence the performance of a vote distribution model. The number of dimensions of a vote distribution is equal to the number of categories. In this case, high-dimensional distributions have more information than low-

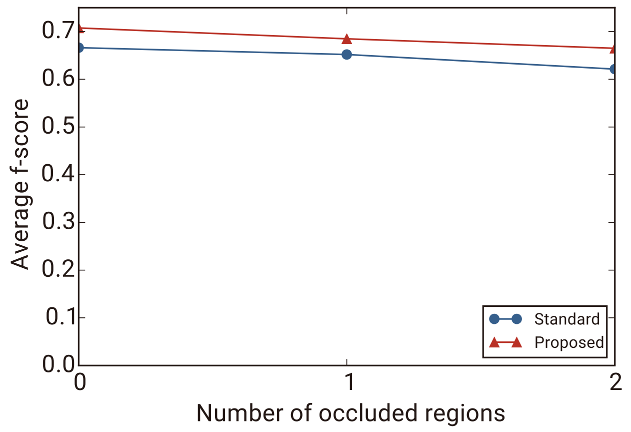

Fig. 17 F-score averaged over all action classes in the UT-Interaction dataset with artificial occlusions.

dimensional distributions. Therefore, when the number of categories increases, the performance of a vote distribution model might improve.

We also assume that input videos into our proposed method are captured by stationary cameras. Applying the proposed method to videos captured by moving cameras would be difficult because the movements of cameras affect the spatial points of votes. Calculating global camera motions might remove this limitation. Global camera motions would adjust the spatial points of votes for each frame.

6. Conclusion

We proposed a novel Hough-based action detection method to enhance the robustness of detecting false votes. Our proposed method employed vote distributions, which represent the voting scores for each action class. The proposed method learns the characteristics of the Hough voting based on vote distributions in order to reduce the effect of false votes. The main contribution of this paper is that our vote distribution model improves the performance of the Hough-based action detection by reducing the influence of false votes caused by similar local motion between different action classes. In experiments, we confirmed that the proposed method reduces the number of false positive detections and improves action detection accuracy compared with the conventional methods. The proposed method achieved 0.789 and 0.708 f-scores on the IXMAS and UT-Interaction datasets, respectively. These results indicate that the proposed method works well in a controlled environment that contains multiple actions captured by a stationary camera.

A future study will improve the vote distribution model. Vote distributions vary temporally during actions because actions can be regarded as sequences of sub-actions. The current model does not manage such temporal variances of vote distributions. We believe that a new model that can represent such temporal variances will be more effective for action detection.

Acknowledgements

This work is supported by Grants-in-Aid for Scientific Research (26280074) and JSPS Fellows (16J09732).

References

[1] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," International Journal of Computer Vision, vol.103, no.1, pp.60–79, 2013.
[2] Y. Ke, R. Sukthankar, and M. Hebert, "Event detection in crowded videos," Proc. International Conference on Computer Vision, pp.1–8, 2007.
[3] L. Cao, Z. Liu, and T.S. Huang, "Cross-dataset action detection," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.1998–2005, 2010.
[4] K.G. Derpanis, M. Sizintsev, K. Cannons, and R.P. Wildes, "Efficient action spotting based on a spacetime oriented structure representation," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.1990–1997, 2010.
[5] P. Siva and T. Xiang, "Action detection in crowd," Proc. British Machine Vision Conference, pp.9.1–9.11, 2010.
[6] J. Yuan, Z. Liu, and Y. Wu, "Discriminative video pattern search for efficient action detection," IEEE Trans. Pattern Anal. Mach. Intell., vol.33, no.9, pp.1728–1743, 2011.
[7] G. Yu, J. Yuan, and Z. Liu, "Unsupervised random forest indexing for fast action search," Proc. IEEE Conference on Computer Vision and Pattern Recognitio, pp.865–872, 2011.
[8] A. Gaidon, Z. Harchaoui, and C. Schmid, "Temporal localization of actions with actoms," IEEE Trans. Pattern Anal. Mach. Intell., vol.35, no.11, pp.2782–2795, 2013.
[9] Z. Jiang, Z. Lin, and L.S. Davis, "A unified tree-based framework for joint action localization, recognition and segmentation," Computer Vision and Image Understanding, vol.117, no.10, pp.1345–1355, 2013.
[10] Y. Tian, R. Sukthankar, and M. Shah, "Spatiotemporal deformable part models for action detection," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.2642–2649, 2013.
[11] K. Mikolajczyk and H. Uemura, "Action recognition with motion-appearance vocabulary forest," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.1–8, 2008.
[12] A. Yao, J. Gall, and L.V. Gool, "A Hough transform-based voting framework for action recognition," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.2061–2068, 2010.
[13] G. Yu, J. Yuan, and Z. Liu, "Real-time human action search using random forest based Hough voting," Proc. ACM International Conference on Multimedia, pp.1149–1152, 2011.
[14] G. Yu, J. Yuan, and Z. Liu, "Propagative Hough voting for human activity recognition," Proc. European Conference on Computer Vision, pp.693–706, 2012.
[15] B. Vijay Kumar and I. Patras, "Supervised dictionary learning for action localization," Proc. International Conference on Automatic Face and Gesture Recognition, pp.1–8, 2013.
[16] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," IEEE Trans. Pattern Anal. Mach. Intell., vol.32, no.9, pp.1627–1645, 2010.
[17] D.H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," Pattern Recognition, vol.13, no.2, pp.111–122, 1981.
[18] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," International Journal of Computer Vision, vol.77, no.1-3, pp.259–289, 2008.
[19] S. Maji and J. Malik, "Object detection using a max-margin Hough transform," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.1038–1045, 2009.
[20] P. Wohlhart, S. Schulter, M. Köstinger, P. Roth, and H. Bischof, "Discriminative Hough forests for object detection," Proc. British Machine Vision Conference, pp.40.1–40.11, 2012.
[21] J. Gall, A. Yao, N. Razavi, L.V. Gool, and V. Lempitsky, "Hough forests for object detection, tracking, and action recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol.33, no.11, pp.2188–2202, 2011.
[22] N. Razavi, N.S. Alvar, J. Gall, and L.V. Gool, "Sparsity potentials for detecting objects with the hough transform," Proc. British Machine Vision Conference, pp.11.1–11.10, 2012.
[23] P. Yarlagadda, A. Monroy, and B. Ommer, "Voting by grouping dependent parts," Proc. European Conference on Computer Vision, pp.197–210, 2010.
[24] A. Srikantha and J. Gall, "Hough-based object detection with grouped features," Proc. International Conference on Image Processing, pp.1653–1657, 2014.
[25] O.J. Woodford, M.-T. Pham, A. Maki, F. Perbet, and B. Stenger, "Demisting the Hough transform for 3d shape recognition and registration," International Journal of Computer Vision, vol.106, no.3, pp.332–341, 2014.
[26] M. Hoai and A. Zisserman, Improving Human Action Recognition Using Score Distribution and Ranking, pp.3–20, Springer International Publishing, Cham, 2015.
[27] K. Hara, T. Hirayama, and K. Mase, "Trend-sensitive hough forests for action detection," Proc. International Conference on Image Processing, pp.1475–1479, 2014.
[28] L. Breiman, "Random forests," Machine Learning, vol.45, no.1, pp.5–32, 2001.
[29] I. Laptev, "On space-time interest points," Int. J. Comput. Vis., vol.64, no.2, pp.107–123, 2005.
[30] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp.1–8, 2008.
[31] D. Weinland, R. Ronfard, and E. Boyer, "Free viewpoint action recognition using motion history volumes," Computer Vision and Image Understanding, vol.104, no.2-3, pp.249–257, 2006.
[32] M.S. Ryoo and J.K. Aggarwal, "UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA)." http://cvrc.ece.utexas.edu/ SDHA2010/Human_Interaction.html, 2010.
[33] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," Proc. European Conference on Computer Vision, pp.705–718, 2008.
[34] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," Pattern Analysis and Machine Intelligence, vol.24, no.5, pp.603–619, 2002.

Kensho Hara received the B.E. degree in Information Engineering and the M.S. degree in Information Science from Nagoya Univeristy, in 2012 and 2014, respectively. He is currently a Ph.D. student at the Graduate School of Information Science, Nagoya University. His research interests include human action recognition and detection. He is a member of IEICE and IEEE.

Takatsugu Hirayama received the M.E. and D.E. degrees in Engineering Science from Osaka University in 2002 and 2005, respectively. From 2005 to 2011, he was a research assistant professor at the Graduate School of Informatics, Kyoto University. He is currently a designated associate professor at the Graduate School of Information Science, Nagoya University. His research interests include computer vision, human vision, human communication, and human-computer interaction. He has received the best paper award from IEICE ISS in 2014. He is a member of IEICE, the Information Processing Society of Japan (IPSJ), the Human Interface Society of Japan, ACM, and IEEE.

Kenji Mase received the B.S. degree in Electrical Engineering and the M.S. and Ph.D. degrees in Information Engineering from Nagoya University in 1979, 1981 and 1992, respectively. He became a professor of Nagoya University in August 2002. He is now with the Graduate School of Information Science, Nagoya University. He joined the Nippon Telegraph and Telephone Corporation (NTT) in 1981 and had been with the NTT Human Interface Laboratories. He was a visiting researcher at the Media Laboratory, MIT in 1988-1989. He has been with ATR (Advanced Telecommunications Research Institute) in 1995-2002. His research interests include gesture recognition, computer graphics, artificial intelligence and their applications for computer-aided communications. He is a member of the Information Processing Society of Japan (IPSJ), Japan Society of Artificial Intelligence (JSAI), Virtual Reality Society of Japan, Human Interface Society of Japan and ACM, a senior member of IEEE Computer Society, and a fellow of IEICE.