A Robust Approach to Hough-based Action Detection

Kensho Hara

Abstract

There has been rapid growth in computers' understanding of the real world through images and videos. One of the most important aspects of real-world information is human actions. To support and communicate with humans, computers must understand their actions. In addition, a huge number of videos exist in the world; to manage so much data, automatic understanding of these videos by computers is required. Finally, understanding actions is especially important because the main subjects of most videos are humans.

Our goal is to develop an automated approach to understanding actions in videos. In this study, we focus on action detection in videos; action detection finds where, when, and what actions occur within videos. Target actions in this study, such as *kicking a ball* and *running*, are constructed by primitive elements such as *lifting a leg* and *swinging an arm*.

We focus on Hough-based action detection methods. Hough-based methods extract local spatiotemporal features from an entire video, then cast votes for action class labels, positions and scales. Here, an action position is usually defined as a spatiotemporal center position. Voting scores are calculated by accumulating the votes at each position based on all local features for each related action class. The local maxima of the accumulated voting scores indicate candidate detected actions. These methods manage the actions that have scores over a threshold as detected actions. Hough-based methods can detect actions robustly with partial observations; the votes based on observed local features are not affected by unobserved local features, because the voting process for each local feature is performed independently. When actions are spatially occluded by other objects and humans, only partial observations are available. Robustness is also useful to early action detection, which can use only early observations of actions.

Various factors, such as occlusion, human orientation variety, motion similarity, temporal variations, and action manners variety, make accurate action detection from videos difficult. To detect actions accurately, detection methods should be robust to

these factors. Hough-based methods are already robust to occlusions. In this study, we propose methods that increase robustness to four additional factors: *human orientation variety, motion similarity, temporal variations*, and *action manners variety*. This thesis is organized with the following chapters.

Chapter 1 presents background, problems, and purpose of this thesis. Contributions of this study are also provided in this chapter.

Chapter 2 reviews related work; we provide an overview of feature representation methods for action recognition and detection and describe related approaches to action detection.

Chapter 3 describes the basic Hough-based action detection algorithm and the implementation of our baseline method.

Chapter 4 presents a Hough-based method that uses multiview videos to provide additional robustness to variety in human orientations. The appearances of actions change relative to a person's orientation to cameras. Multiview videos are synchronous videos captured from multiple cameras. Capturing actions with multiview videos gives observations from various viewpoints. These observations include different relative orientations of human subjects to the cameras. Therefore, these observations reduce the differences in relative orientation between training and test data and contribute the robustness to human orientation variety. Our proposed method casts independent votes in each view. Here, we assume that human feet touch the ground plane when they start an action. We then integrate votes in global coordinates based on assumptions using homographic transformations. The proposed method uses multiview information effectively and detects actions robustly.

Chapter 5 describes a novel Hough-based action detection method to overcome the problem of motion similarity. Discriminating between similar local motions that exist in different action classes is difficult; in such cases, conventional Hough-based methods often cast votes for false action classes. The false votes do not occur randomly such that they depend on relevant action classes. We introduce vote distributions, which are distributions of the voting scores for each action class. We assume that the distribution of false votes includes important information necessary for improving action detection. These distributions are used to build a model that represents the characteristics of Hough voting, including false votes. This method estimates likelihood using this model and reduces the influence of false votes, which leads to robustness to motion similarities across action classes.

Chapter 6 presents a method for achieving robustness to temporal variations that can exist in the same action class. Conventional Hough-based methods perform

poorly for actions with temporal variations because such variations change the temporal relation between local feature positions and action positions. Some votes may be scattered in the temporal dimension because of such variations. We propose a method for concentrating scattered votes through time warping. The proposed method estimates the offsets between scattered and concentrated voting positions based on conventional Hough votes. The offsets warp the scattered votes to concentrate them, providing a method for robustness even in the presence of temporal variations.

Chapter 7 describes a method that focuses on the number of local features. Various factors, such as occlusions, human orientation variety, temporal variations, and action manners variety, change not only the feature descriptors of actions but also the number of local features. Conventional Hough-based methods perform poorly with variations in the number of local features extracted from actions. Changes in voting scores that depend on the number of local features produce difficulties in determining a voting score detection threshold. Our proposed method improves two parts of the Hough-based method. The first is the extraction of local features; the proposed method reduces the method's dependency on the number of local features based on spatial scales. It adjusts the number of local features for each spatial scale using a sampling method. The second part is detection thresholding. The proposed method determines appropriate thresholds for voting scores based on the number of local features by learning the relation between the number of local features and voting scores. These changes reduce the influence of the number of local features (i.e. improving robustness in different aspects from previous chapters).

Finally, Chapter 8 concludes this thesis and presents directions for future work.

Acknowledgement

I first would like to express my gratitude to my supervisor, Professor Kenji Mase. His insightful comments and suggestions guided me in developing this study. In addition, his great research vision always inspired me.

I also would like to thank my thesis committee, Professor Hiroshi Murase, Professor Yoshiharu Ishikawa, and Professor Katsuhiko Toyama for taking time to review my thesis and giving important comments. Their comments improved quality of this thesis significantly.

Dr. Takatsugu Hirayama was always eager to discuss this work, and these discussions were essential for writing this thesis. I also wish to express my gratitude to Dr. Yu Enokibori and Dr. Junya Morita. Their comments gave me fresh ideas because their areas of research are slightly different from mine.

I would like to offer my special thanks to all the members of the Mase laboratory. Mr. Fumiharu Tomiyasu and Ms. Xueting Wang. Daily discussion with them advanced my research and motivated me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Backgrounds

There has been rapid growth in computers' understanding of the real world via images and videos. Recently, computers have begun to outperform humans in object recognition in images [He et al., 2015]. Computers that use cameras for visual input could support human activities and communicate with humans naturally. In addition, a vast number of images and videos exist in the world. To manage this quantity of data, automatic understanding of these images and videos by computers is required.

One important type of real-world information in is human actions. To support and communicate with humans, computers must understand their actions. For example, understanding actions in surveillance cameras can lead to automatic surveillance systems in which computers contribute to safety by finding suspicious people, automatic monitoring of elderly people can help with care activities, and intelligent devices could be controlled by humans without touch. Robots collaborate with humans by understanding their actions. Understanding actions is also important for video descriptions, because the main subjects of most videos are humans. Descriptions increase the ease of searching for specific actions in videos and video summaries.

Actions have a hierarchical structure. The definition of this structure has been defined differently by many researchers [Nagel, 1988, Bobick, 1997, Moeslund et al., 2006, Aggarwal and Ryoo, 2011]. This thesis adopts the hierarchical structure of [Moeslund et al., 2006] as shown in Figure 1.1. This structure is constructed from three elements: *action primitives, actions*, and *activities*. *Action primitives* are the atomic components of human body movements. *Actions* are constructed

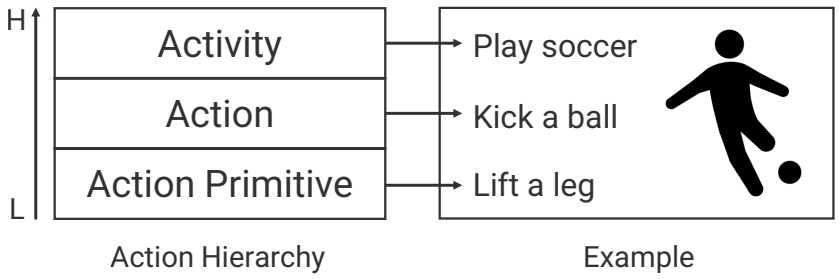


Figure 1.1: Action hierarchy.

Figure 1.2: Examples of action detection. In this figure, examples are shown only in spatial dimensions, whereas actual action detection is performed in spatiotemporal dimensions. Parts of the images in this figure are contained in the IXMAS [Weinland et al., 2006a], UCR Videoweb Activities [Denina et al., 2011], UT-Interaction [Ryoo and Aggarwal, 2010], and UCF-Sports [Soomro and Zamir, 2014] datasets.

by multiple *action primitives*. *Activities* represent general categories and have semantics, whereas *actions* represent specific movements. Figure 1.1 shows an example of an action hierarchy in soccer. An *action primitives* could be *lift a leg*, *swing an arm*, etc. *Actions* are sequences of primitives, such as *kick a ball* and *run*. *Activities* is the highest-level representation and, in this case, corresponds to *play soccer*. In this study, we focus on *actions* which are the key element in the action hierarchy.

Typical tasks that focus on *actions* are action recognition and detection. Action recognition methods classify actions throughout an entire video. These methods assume that the video is already somehow segmented so that it only contains one action. Action detection methods search for all instances of targeted actions in a scene. Besides classifying actions, these methods localize actions spatiotemporally. Most applications deal with videos that contain multiple action instances and need to detect each instance. Herein, we focus on action detection, of which Figure 1.2 shows some examples.

Most action recognition and detection methods represent actions with feature vectors. The feature vectors often describe both the appearance and motions of

actions. State-of-the-art methods represent actions using features that are hand-crafted [Wang et al., 2015a], with deep learning [Simonyan and Zisserman, 2014], or a hybrid of the two techniques [de Souza et al., 2016]. These methods perform machine learning using these feature vectors to recognize and detect actions.

1.2 General Problems in Action Detection

There are a variety of factors that make accurate action detection from videos difficult. In this section, we introduce important factors of them.

Human orientation variety: The appearance of actions changes depending on the orientation of the person performing them relative to the camera. Both the left and right images in Figure 1.3 (a) represent the action *Punch*, but the appearances of the action are different significantly. Such changes degrade action detection performance.

Motion similarity: If similar local motions exist across different action classes, discriminating between such motions is difficult. The left and right images in Figure 1.3 (b) are *Sit Down* and *Pick Up*, respectively. They belong to different action classes, but include similar stooping motions. Such similar motions cause false classification.

Temporal variations: Temporal variation exists within an action class. That is, the same action can be performed at a variety of speeds. Both the top and bottom image sequences in Figure 1.3 (c) are *Kick*. Comparing these actions, the temporal positions in which a foot is raised are the same, but they differ for lowered feet. Such variations change the temporal structure of actions and confuse detection methods.

Action manners variety: Actions can be performed in a variety of manners even if they are in the same class. Both the left and right images in Figure 1.3 (d) are *Punch*, but the left *Punch* is straight whereas the right *Punch* is performed from the side. Such variety leads to large intraclass variations.

Occlusion: If human regions are occluded by other humans or objects, we cannot observe complete action sequences. Figure 1.3 (e) shows an example of an occluded action. The arms and lower bodies in the image cannot be observed. In such situations, detection methods must work only using the partial observations.

Background clutter: The motions of humans and other moving objects in the background of images functions as noise. The image in Figure 1.3 (f) contains the motions of pedestrians that are not performing any target actions. Such motions

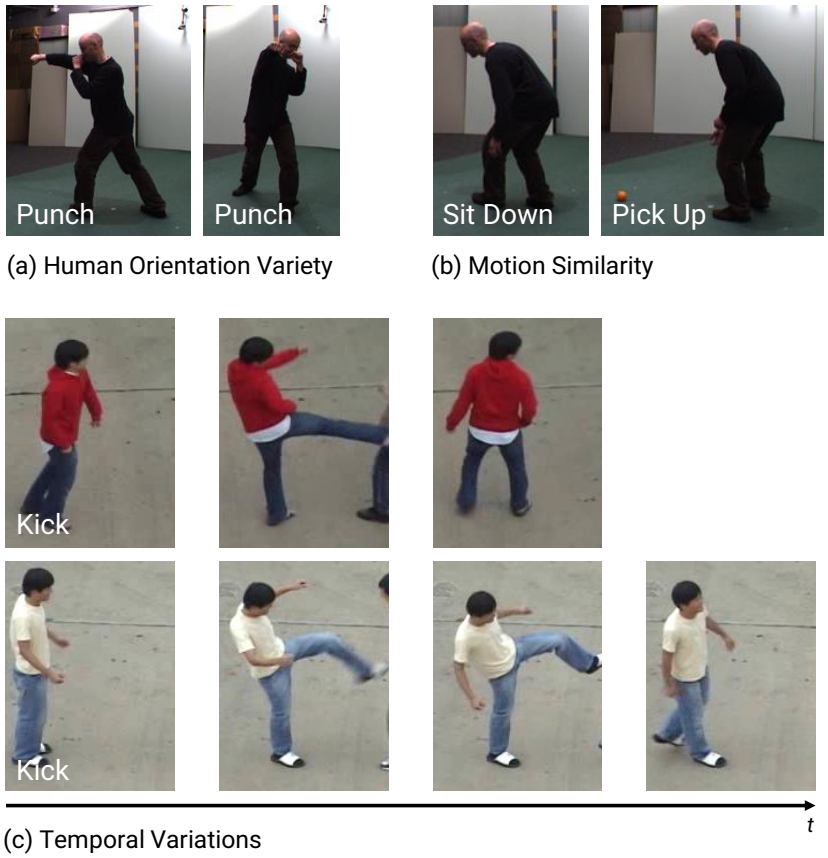


Figure 1.3: Various factors that make action detection difficult.

have a negative influence on accurate action detection.

Global camera motions: Movement of the camera also creates noise. Figure 1.3 (g) shows global motions resulting from camera movement. The green lines in the figure indicate the flow of motions; homogeneous flows exist throughout the and affect the apparent motion of target actions.

These variations are caused by humans and camera configurations. To detect actions accurately, detection methods should be robust to these factors.

1.3 Contributions

In this study, we focus on a Hough-transform-based approach [Mikolajczyk and Uemura, 2008, Yao et al., 2010, Yu et al., 2011a, Yu et al., 2012, Vijay Kumar and Patras, 2013], which is a typical approach for action detection. An overview of the Hough-based approach is shown in Figure 1.4. The Hough-based approach extracts local spatiotemporal features from an entire video, then casts votes for action classes, spatiotemporal positions, and scales based on extracted local features. The voting scores are calculated by accumulating the votes at each position based on all local features for each related action class. The local maxima of accumulated voting scores indicate actions (i.e. class labels, spatial bounding boxes, and time intervals).

An advantage of the Hough-based approach is its robustness to detecting actions when presented with partial observations. When actions are spatially occluded by other objects and humans, only partial observations are available. This robustness is also useful for early action detection. The votes based on observed local features are unaffected by unobserved local features, because the voting process of each local feature is performed independently. Another advantage of these methods is their computational efficiency. Because the voting process classifies actions and estimates positions simultaneously, the Hough-based approach does not require an exhaustive search of the spatiotemporal video space. We focus on Hough-based action detection because of these advantages.

In this study, we aim to improve the robustness of action detection to various impeding factors. Hough-based methods are already robust to occlusions. However, conventional Hough-based methods are not robust to the other factors we have discussed. We propose methods that provide robustness to four of the other factors: *human orientation variety, motion similarity, temporal variation*, and *action manners variety*. By combining these methods, we can achieve robust action detection in a wider variety of environments.

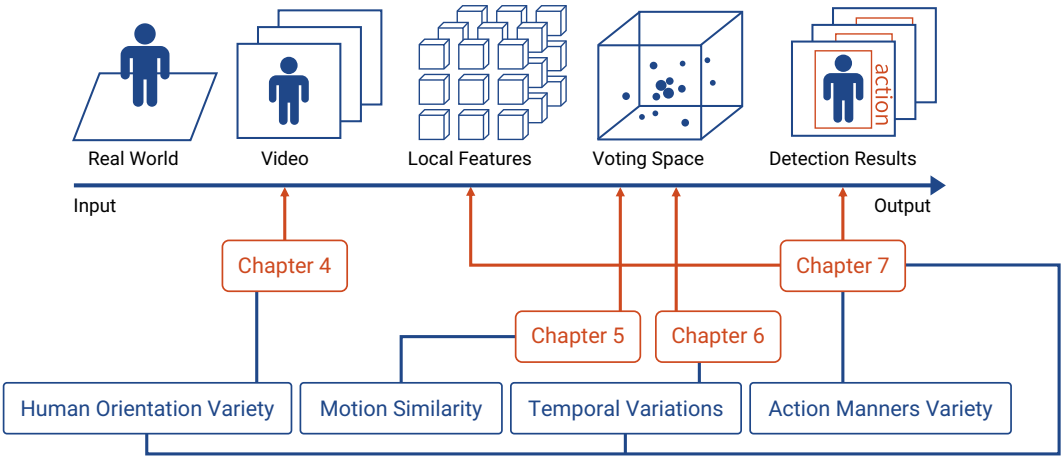


Figure 1.4: Flow of the Hough-based approach and the relation between the various factors and the structure of this thesis.

First, we propose a Hough-based method that uses multiview videos to increase robustness to *human orientation variety*. Multiview videos are synchronous videos captured from multiple cameras. Capturing actions with multiview video gives observations from several viewpoints that include different relative orientations of the subject to the cameras. Therefore, these observations reduce the differences in relative orientation between training and test data and contribute the robustness to human orientation variety. To use multiview videos in Hough-based action detection, we propose integrating Hough votes by homographic transformation. Our proposed method casts independent votes for each view. Here, we assume that human feet touch the ground plane when they start an action. We then integrate the votes in global coordinates using homographic transformations based on this assumption. The proposed method uses multiview information effectively and detects actions robustly.

Next, we propose a novel Hough-based method to overcome the problem of *motion similarity*. Conventional Hough-based methods often cast votes for false action classes when similar motions exist across classes. The false votes do not occur randomly, such that they depend on relevant action classes. We introduce vote distributions, which are distributions of the voting scores for each action class. We assume that the distribution of false votes includes important information necessary to improving action detection. These distributions are used to build a model that represents the characteristics of Hough voting and includes false votes. This method estimates likelihood and reduces the influence of false votes using the model. This reduction leads to robustness to motion similarity.

We also propose a method to achieve robustness to *temporal variation*. Conventional Hough-based methods perform poorly for actions with temporal variations because such variations change the temporal relation between the local feature positions and action positions. Some votes may be temporally scattered because of such variations. We propose a method for concentrating scattered votes through time warping. The proposed method estimates the offsets between the scattered and concentrated voting positions based on the conventional Hough votes. The offsets warp the scattered votes to concentrate them and provide robustness to temporal variations.

Finally, different from the three methods above, we propose a method that focuses on the number of local features. Various factors, such as *human orientation variety, temporal variations*, and *action manners variety*, change not only the feature descriptors of actions but also the number of local features extracted from them. Conventional Hough-based methods perform poorly when variable numbers of local features are extracted from actions. Changes in voting scores caused by changes in the number of local features result in difficulties in determining a detection threshold using the voting scores. Our proposed method improves two parts of the Hough-based method: local feature extraction and detection thresholding. First, the proposed method reduces the dependency on the number of local features using spatial scales. It adjusts the number of local features for each spatial scale using a sampling method. Second, the proposed method determines appropriate thresholds for voting scores based on the number of local features by learning the relation between the number of local features and voting scores. These two changes reduce the influence of the number of local features, which is affected by various factors. Reducing the influence improves robustness to human orientation variety, temporal variations, and action manners variety in different aspects from the three methods above.

1.4 Structure of this Thesis

The structure of this thesis is as follows. Chapter 2 reviews related work. Chapter 3 describes the basic Hough-based action detection algorithm. Chapter 4 presents the method that uses multi-view videos to improve robustness to human orientation variety. In Chapter 5, we propose a Hough-based method to overcome the problem of motion similarity. Chapter 6 introduces the time-warping of Hough votes to make the method robust to temporal variations. Chapter 7 describes a method that

considers the number of local features to increase robustness to various factors. Chapter 8 concludes this thesis and presents future directions for this work. Figure 1.4 shows the relation between the thesis structure and the factors discussed.

Chapter 2

Related Work for Action Detection

In this chapter, we review various work in action detection and recognition. Table 2.1 summarizes the approaches taken in related studies. Our proposed methods use local feature representation and detect actions using a Hough-based approach. The characteristics of each approach are discussed in the following sections.

We also discuss some work in object recognition and detection, which have motivated many methods of action recognition and detection.

2.1 Feature Representation

First, we review feature representations of actions. Actions should be represented as feature vectors for recognition and detection. Various methods for representing actions have been proposed, and they can be divided into two approaches: local and global feature representations. In the following sections, we review both approaches. In Sections 2.1.1 and 2.1.2, we review both approaches, then discuss both approaches in Section 2.1.3.

2.1.1 Local Feature Representation

Local feature representation describes an action using a set of local features. The Hough-based approach, which are the focus of this work, use this representation.

Local feature representation consists of two parts: local feature detection and local feature description. First, the representation detects local features, which are small regions in videos. It then describes each local feature to represent the region.

		Feature Representation	
		Local	Global
Detection Approach	Hough	***Ours***, [Mikolajczyk and Uemura, 2008], [Gall et al., 2011], [Yu et al., 2011a], [Yu et al., 2012], [Vijay Kumar and Patras, 2012], [Vijay Kumar and Patras, 2013]	
	Sliding Window	[Yuan et al., 2011], [Yu et al., 2011b]	[Cao et al., 2010], [Derpanis et al., 2010], [Tian et al., 2013], [Ke et al., 2007], [Siva and Xiang, 2010], [Oneata et al., 2014b]
	Action Proposals		[Oneataetal.,2014a], [Jain et al., 2014], [Gemert et al., 2015], [Yu and Yuan, 2015], [Weinzaepfeletal.,2015], [Gkioxari and Malik, 2015]
	Action Recognition (for pre-segmented sequences)	[Laptev, 2005]	[Schüldt et al., 2004], [Dollar et al., 2005], [Wang et al., 2009], [Wang et al., 2015a], [Jain et al., 2013], [Ji et al., 2013], [Karpathy et al., 2014], [Simonyan and Zisserman, 2014], [Feichtenhofer et al., 2016], [Wang et al., 2015b]

Table 2.1: Related Work.

Local Feature Detection

First, we introduce local feature detection methods, divided into two approaches: sparse and dense detectors. Sparse detectors find characteristic points in videos to represent actions using only a few local features. Popular sparse detectors are the spacetime interest point (STIP) [Laptev, 2005] and cuboid [Dollar et al., 2005] detectors. The STIP detector is an extension of the Harris corner detector [Harris and Stephens, 1988] from a 2D spatial domain to a 3D spatiotemporal domain. STIPs detect corner points in 3D spatiotemporal space. Cuboid detectors find feature points using spatial Gaussian and temporal Gabor filters.

In contrast with sparse detectors, dense detectors contribute to better action representation. In some studies, simple dense sampling of local features in videos has achieved higher action recognition accuracy than sparse detectors, and denser sampling leads to better accuracy [Wang et al., 2009, Gall et al., 2011]. The most popular dense detector method is called dense trajectories (DTs) [Wang et al., 2013]. DTs not only sample local features densely, but also track them using dense optical flows. DTs define regions around the trajectories based on optical flow as those of local features. The trajectory-aligned regions achieve higher recognition accuracy than simple densely sampled regions. Furthermore, Wang et al. improved DTs by removing global camera motion, which enables descriptors to focus on essential human motions [Wang et al., 2015a].

A disadvantage of dense detectors is their high computational cost in the feature description step because the number of local features is large. Shi et al. tried to accelerate such detectors by random sampling of local features [Shi et al., 2013]. They showed that random sampling accelerates action recognition with slight sacrifices to accuracy.

In this study, we use STIPs as a local feature detection method because of their performance and availability. We also use DTs to discuss the results of both sparse and dense detectors in Chapter 7. The original implementations of both STIPs[1] and DTs[2] are available online.

Local Feature Description

Next, we review description methods for local features. Two types of description are often used: appearance and motion. To represent actions, knowing which parts of

[1] https://www.di.ens.fr/~laptev/download.html#stip
[2] https://lear.inrialpes.fr/people/wang/dense_trajectories

the human body move and in which directions is important. Appearance and motion represents these two aspects, respectively.

Various descriptors for appearance exist. Low-level descriptors use image intensity and its derivatives in the x and y dimensions [Schüldt et al., 2004, Dollar et al., 2005, Gall et al., 2011]. The computational cost of these descriptors is small, but the accuracy obtained from their use is relatively low compared with richer descriptors. Object recognition and detection often use gradient-based descriptors, such as histograms of oriented gradients (HOG) [Dalal and Triggs, 2005] and scale-invariant feature transform (SIFT) [Lowe, 2004]. Motivated by this method, Laptev et al. used HOG as an appearance descriptor for action recognition [Laptev et al., 2008].

Intensity derivatives in the temporal dimension and optical flow are mainly used as motion descriptors. Like appearance, the derivative in the temporal dimension is used as a low-level motion descriptor [Schüldt et al., 2004, Dollar et al., 2005, Gall et al., 2011]. Various optical-flow-based descriptors are used to represent motion: Gall et al. used the absolute values of optical flows in the x- and y-directions [Gall et al., 2011], Laptev et al. used histograms of optical flow (HOF) [Laptev et al., 2008], and Wang et al. used trajectory of optical flow and motion boundary histograms (MBH) that represent optical flow gradients [Wang et al., 2013]. Wang et al. showed that MBH are more robust to camera motion than HOF.

Some descriptors can represent appearance and motion simultaneously. These descriptors calculate 3D gradients in spacetime. Everingham et al. proposed HOG3D, which is a spatiotemporal extension of spatial HOG [Everingham et al., 2008]. Similarly, Scovanner et al. extended SIFT from a 2D spatial domain to a 3D spatiotemporal domain in a method called 3D SIFT [Scovanner et al., 2007].

In this study, we use the original implementation of STIPs and DTs, as described in the previous section. The implementations includes both local feature detection and description steps. The implementation of STIPs use HOG and HOF descriptors. The implementation of DTs use trajectory, HOG, HOF, and MBH descriptors. We also used the implementations for the local feature description.

2.1.2 Global Feature Representation

Global feature representation describes an action using a feature vector. Whereas the Hough-based approach uses the local feature representation (i.e. used in this study), other detection approaches, such as the sliding window and action proposals, often use this global feature representation. We present a discussion of the local and

global feature representation in Section 2.1.3.

In this section, we review the two popular types of methods for these descriptions: local feature encoding and convolutional neural network (CNN)-based features.

Local Feature Encoding

One of the most popular global feature representations is local feature encoding, which converts local feature representation to a global feature representation. A typical encoding method is the bag of visual words (BOVW), which was inspired by natural language processing and has been proposed for image recognition [Csurka et al., 2004]. A BOVW generates a codebook by clustering local features extracted from many images or videos. K-means and Gaussian mixture models are often used for clustering. Then, global representations are calculated by encoding local features into the nearest code words. Many studies have shown that the BOVW representation is also effective for action recognition [Schüldt et al., 2004, Wang et al., 2009, Wang et al., 2015a].

The BOVW representation has some extensions, such as the vector of locally aggregated descriptors (VLAD) [Jégou et al., 2012] and the Fisher vector [Sánchez et al., 2013], that consider the distribution of local features in the feature space to improve recognition. VLAD uses the mean information of the distribution of the local features assigned to each code word. Jain et al. demonstrated the effectiveness of VLAD in action recognition [Jain et al., 2013]. The Fisher vector uses covariance information as well as means; when using improved dense trajectories (HOG, HOF, and MBH descriptors), the Fisher vector has achieved state-of-the-art performance in action recognition [Wang et al., 2015a].

CNN-based Features

CNNs achieve state-of-the-art performance for various computer vision tasks. CNNs learn feature representations using deep convolution layers. Deep learned features by CNNs perform better than hand-crafted features in many tasks. Specifically, the success of CNN-based features for object recognition [He et al., 2015, He et al., 2016] and detection [Girshick et al., 2016, Liu et al., 2016] has motivated the use of CNN-based features for action recognition and detection.

Two approaches exist for handling temporal information in CNNs for action. One approach uses only video and fuses information from multiple frames. The other approach first calculates optical flows from the video, and then uses both the

video and optical flows as a two-stream input.

First, we review methods using the first approach. Ji et al. proposed a 3D spatiotemporal convolution to capture temporal information [Ji et al., 2013]. The convolution filters in the temporal dimension fuse information from multiple frames to capture temporal information. Karpathy et al. explore multiple approaches for fusing information from multiple frames in a CNN [Karpathy et al., 2014]. Their proposed slow fusion, which progressively connects convolutional layers from each frame, achieves better performance than early or late fusion methods.

The second approach uses optical flow as explicit motion information. Simonyan and Zisserman proposed a two-stream CNN that uses image and optical flow sequences as separate input [Simonyan and Zisserman, 2014]. They showed that fusion of the two streams results in higher recognition accuracy compared with separate networks. Feichtenhofer et al. explored fusion methods for two-stream networks, finding that a convolutional fusion method at the convolutional layers of the two streams achieves the best performance [Feichtenhofer et al., 2016].

CNN-based features have achieved state-of-the-art performance in recent studies. However, the performance difference for action recognition and detection between CNN-based and hand-crafted features is not significant compared with the same differences in object recognition and detection. Combining CNN-based and hand-crafted features performs better than either feature type alone [Wang et al., 2015b, Feichtenhofer et al., 2016].

2.1.3 Discussion

Local feature representation describes an action using a set of local features, whereas global feature representation describes an action using a feature vector. Here, we discuss the advantages of each representation.

A local feature representation contributes to the robustness of detection to partial observations. Observed local features are not affected by unobserved local features because local feature processes are performed independently. Therefore, local feature representations provide robustness to occlusions and enable early action detection. A disadvantage of local feature representations is their inferior discrimination ability. Even if actions from different classes differ when observing their full sequences, some local parts are often similar. Discriminating between such local parts represented by local features is difficult.

In contrast with local feature representation, global feature representation can

describe an entire action sequence. These descriptions include complex information such as the co-occurrence of multiple parts. Therefore, global feature representation is more robust to motion similarity.

In this study, we focus on local feature representation because of its robustness to occlusions. To achieve robustness to motion similarity, we propose the method described in Chapter 5.

2.2 Detection Approach

Here, we review typical approaches for action detection: Hough-based, sliding window, and action proposal approaches.

2.2.1 Hough-based Approach

The Hough-based approach extracts local spatiotemporal features from an entire video and then casts votes for action classes, positions, and scales. The local maxima of the voting scores indicate possible detected actions.

The implicit shape model (ISM) [Leibe et al., 2008], which is now a commonly used Hough-based method, was originally proposed for object detection. The ISM generates a codebook of local features using an unsupervised clustering algorithm and uses the codebook to cast votes for object positions based on the local features extracted from an image.

Some studies have applied the ISM framework to action recognition and detection [Mikolajczyk and Uemura, 2008, Gall et al., 2011, Yu et al., 2011a, Yu et al., 2012, Vijay Kumar and Patras, 2012, Vijay Kumar and Patras, 2013]. To best of our knowledge, the study in [Mikolajczyk and Uemura, 2008] is first one applying the framework to actions. They detected actions only spatially in a 2D spatial and 1D scale voting space. Gall et al. proposed a supervised codebook using random forests to improve discriminative power [Gall et al., 2011]. They applied the method to not detection but recognition of actions. Here, we manage their method as action detection method because their method can be naturally applied to action detection. Similarly, Kumar et al. also proposed supervised codebook learning methods [Vijay Kumar and Patras, 2012, Vijay Kumar and Patras, 2013]. Yu et al. attempted fast search in the voting space using a max sub-path search strategy [Yu et al., 2011a]. They also proposed the method that is suitable when the number of labeled training data is very small [Yu et al., 2012]. Their method

uses the random projection tree, which can be constructed in unsupervised way, as a codebook method. Although various methods are proposed, robustness to various factors described in Section 1.2 is not explored sufficiently.

The Hough transform was originally proposed in 1962. Many methods have since extended the Hough transform for various tasks. The history of the Hough transform is described in Section 3.1.1.

2.2.2 Sliding Window Approach

The sliding window approach classifies actions in each subvolume of a video sequence while changing the spatiotemporal position of that subvolume, searching all possible subvolumes in the space exhaustively.

Motivated by the success of sliding window approaches for object detection [Viola and Jones, 2001, Felzenszwalb et al., 2010], many action detection methods have adopted the sliding window approach [Cao et al., 2010, Derpanis et al., 2010, Ke et al., 2007, Siva and Xiang, 2010, Yuan et al., 2011, Yu et al., 2011b, Tian et al., 2013]. Many of these methods [Cao et al., 2010, Derpanis et al., 2010, Ke et al., 2007, Siva and Xiang, 2010, Tian et al., 2013] have used global feature representations, whereas some [Yuan et al., 2011, Yu et al., 2011b] use local feature representations.

Because the search spaces of action detection are larger than those of object detection, the computational cost of the sliding window approach is high, although some research has attempted to reduce these costs. Yuan et al. proposed a branch-and-bound search for action detection [Yuan et al., 2011]. Their search method accelerates the sliding window approach and uses local feature representation. Yu et al. proposed two-round coarse-to-fine searches [Yu et al., 2011b]. Their method downsamples videos and searches the video space coarsely. Then, their method searches actions at a finer scale. Oneata et al. also applied branch-and-bound search to action detection [Oneata et al., 2014b]. They proposed an approximated Fisher vector encoding and showed that sliding-window-based detection that uses this encoding can be accelerated using branch-and-bound search. These efficient methods accelerate action detection, though the types of feature vectors that are used are restricted.

2.2.3 Action Proposals Approach

The action proposal approach reduces the searched windows to accelerate the sliding window approach, because the computational cost of a straightforward sliding window is high. This approach finds general action regions and only classifies these regions, whereas the sliding window approach classifies all possible windows.

The proposal approach was originally proposed for object detection using CNN-based features, which are computationally expensive. In object detection, the object proposal approach finds general object regions first and only classifies these regions. Efficient region discovery allows detection methods to use high-cost features, such as CNN-based features. An R-CNN [Girshick et al., 2016] is a typical method of proposal-based detection. R-CNNs find general object regions using selective search [Uijlings et al., 2013] and then classify each region using a CNN.

Motivated by the success of the R-CNN, proposal-based approaches to action detection have been explored. To find general action regions, two approaches were proposed. One approach is based on bottom-up merging in 3D spatiotemporal, similar to a selective search. Oneata et al. proposed spatiotemporal object proposals based on video segmentation [Oneata et al., 2014a]. Their method calculates superpixels in each frame and merges them in spatiotemporal space. Each merged region is classified to detect actions. A similar method was proposed by Jain et al. [Jain et al., 2014]. A disadvantage of these methods is their high computational cost, due to video segmentation. Gemert et al. proposed a more efficient method that uses DTs instead of video segmentation [Gemert et al., 2015]. Their method assumes that the feature vector for classifying actions is composed of DTs. Their method finds the general action regions and classifies actions in the regions using the DTs shared in both steps. The shared features make action detection efficient.

Another approach links spatial candidate regions in each frame to finds the general regions of actions. Yu et al. detected humans in each frame and used them as spatial candidate regions [Yu and Yuan, 2015]. They then temporally linked the candidate regions as a set coverage problem. They detected actions by classifying the regions of the linked candidates. Weinzaepfel et al. used EdgeBoxes [Larry Zitnick, 2014], which find general object regions, to find spatial candidate regions and tracked these regions using a tracking-by-detection approach [Weinzaepfel et al., 2015]. They also detected actions by classifying the tracked regions. Gkioxari et al. used selective search to find the spatial candidate regions [Gkioxari and Malik, 2015]. Unlike the above methods, their method classified each spatial candidate before the linking. Their method linked the candidate regions using the Viterbi

algorithm based on the classified scores and overlap ratios of the spatial candidate regions.

2.2.4 Discussion

The Hough-based approach uses local feature representation and performs the voting process independently for each local feature. In contrast, the sliding window and action proposal approaches mainly use global feature representations and classify action regions that have been found in advance. In the following, we discuss the advantages of each approaches.

One advantage of the Hough-based approach is its computational efficiency. The other detection approaches need processing for each action region that the Hough-based approach does not, because of its simultaneous action classification and position estimation in the voting process. In addition, the advantages of local feature representations (i.e., robustness to occlusions) also contribute to this detection approach.

The sliding window and action proposal approaches are superior in discrimination ability to the Hough-based approach. Because these approaches decide on the action regions in advance, they can use global feature representations. The advantages of global feature representations contribute to these detection approaches.

In this study, we focus on the Hough-based approach, proposing methods that improve its robustness to factors other than occlusions.

Chapter 3

The Hough Transform and Action Detection

3.1 The Hough Transform

3.1.1 History of the Hough Transform

The Hough transform was initially proposed for line detection [Hough, 1962]. The basic idea of the Hough transform is a transformation from the original image space to a parameter space of the lines. First, the Hough transform detects edges, which are the elements of the lines, in an image. Then, the all lines that can pass through each edge are managed as the candidates of the lines in the image. The Hough transform represents a line using two parameters described later and casts votes for the parameters corresponding to each candidate line in the two-dimensional parameter space. Based on the edges that lie on a common line, the Hough transform casts votes for the parameters of the common line as well as the other parameters of the lines that pass through the edges. Therefore, the accumulated voting score for the parameters of the common line is high. The Hough transform detects lines by finding such high scores in the parameter space. We call the parameter space a voting space in this study.

The Hough transform originally represents a line as $y = ax + b$ using two parameters a and b [Rosenfeld, 1969]. Parameters a and b are the slope and the intercept of the line, respectively. A problem of this representation is that slope a of the vertical line is infinity. It is hard to implement the method that uses such unbounded parameter. Duda and Hart represents a line as $\rho = x \cos\theta + y \sin\theta$ using

two parameters ρ and θ [Duda and Hart, 1972]. Parameters ρ is the distance between the origin and the closest point on the line, and θ is the angle of the normal vector at the closest point, respectively. Because distance ρ is bounded by the size of the image and angle θ is also bounded, these parameters are bounded and suitable for implementation. Using ρ-θ representation, a point in a image space is transformed to a sinusoidal curve in the voting space.

The Hough transform has been extended for other shapes such as circles [Duda and Hart, 1972] and ellipses [Tsuji and Matsumoto, 1978]. For instance, a circle can be represented by a 2D center position and a radius. The Hough transform for circle detection casts votes in the 3D voting space (2D center position and 1D radius) [Duda and Hart, 1972]. Ballard et al. generalized the Hough transform for arbitrary shapes [Ballard, 1981]. The method, which is called the generalized Hough transform, assumes that the shape of the target object is known. Because the method does not restrict the target shapes, some of the shapes cannot be represented by mathematical formulations, unlike lines and circles. To cast votes for the parameters of the shapes, the method defines a reference position of the target and uses the offsets from the edge position to the reference position. Using a template image of the target, the method prepares an R-table that maps the gradient direction of an edge detected in the template image to the offset. Note that one gradient direction is mapped to multiple offsets in the R-table. To detect the targets in a image, the method detects edges in the image, calculates the gradients of edges, casts votes based on the R-table, and detects targets by finding local maxima in the voting space. Here, the voting space consists of four dimensions: a 2D reference position, scale and rotation of the target.

Computational cost of the original Hough transform is high because the method casts votes for all possible candidates. Kimme et al. reduced the computational cost for circle detection using directions of the gradients [Kimme et al., 1975]. The directions of the gradients shows the directions to centers of circles. The method should cast votes for only the circles that are located along the directions. The generalized Hough transform also use the directions of the gradients.

Leibe et al. introduced modern computer vision techniques into the generalized Hough transform in the ISM (also described in Section 2.2.1) [Leibe et al., 2008]. Instead of the edges and R-table, the ISM uses local features, such as SIFT [Lowe, 2004], and a codebook of the local features, respectively. The ISM generates the codebook using unsupervised clustering algorithm. The codebook maps descriptors of the local features and the offsets between feature positions to reference object positions. The ISM uses the codebook to casts votes for object positions based on

local features extracted from an image.

Recently, as well as object detection, the Hough-based approaches are applied to various computer vision tasks such as object tracking [Gall et al., 2011]; pose estimation of heads, [Fanelli et al., 2011], objects [Tejani et al., 2014], and humans [Girshick et al., 2011]; facial expression recognition [Fanelli et al., 2010]; and action recognition [Gall et al., 2011] and detection [Vijay Kumar and Patras, 2012]. These methods indicate high flexibility of the Hough transform. The Hough transform can manage various tasks by defining appropriate voting spaces.

Motivated recent success of CNNs, many approaches adopt CNN-based features. As described in Section and , the proposal approaches use CNN-based global feature representation. The Hough-based approach cannot use global feature representation. To use CNN-based features in the Hough-based approach, Riegler et al. combined CNN-based local features and the Hough-based approach for head pose estimation and facial feature localization [Riegler et al., 2014]. Their CNN model uses a local image patch as input and estimates a class label and offsets for head pose. Their method casts votes based on the estimation of the CNN model. They showed that using discriminative CNN-based features improve performance in the Hough-based approach.

3.1.2 Definition of the Hough Transform

As shown in the previous section, many methods based on the Hough transform have been proposed. The unique step of the Hough transform is the transformation from the input space to the voting space. In this study, we define Hough-based approaches as follows: Hough-based approaches transform the input space to the voting space through voting.

The Hough-based approach is different from a voting-based approach. For instance, consider the method that performs the majority voting process for the class labels of the target based on local features. The method casts votes based on only the feature vectors of the local features and do not transform the positions of the local features in the input space. Such method does not belong to the Hough-based approach.

3.2 Hough-based Action Detection

This section explains the basic Hough-based action detection algorithm. A Hough-based method detects actions by casting votes in a Hough voting space. To cast votes in the voting space, the Hough-based method generates a codebook of local features in a training step. The method then casts votes for action classes, positions, and scales by matching local features to codebook entries in a detection step. In this section, we first define the problem of action detection in Section 3.2.1. Then, we explain the training step in Section 3.2.2 and describe the detection step in Section 3.2.3. Finally, Section 3.2.4 describes the implementation of the Hough-based method used in this study.

3.2.1 Problem Definition of Action Detection

We first define the problem of action detection as a problem that outputs $\{\mathbf{D}_1, \ldots, \mathbf{D}_N\}$ from a video input, where $\mathbf{D} = [c, \mathbf{x}_{\mathrm{tl}}, \mathbf{x}_{\mathrm{br}}, t_{\mathrm{b}}, t_{\mathrm{e}}]$ is a detected action, c is a class label, $\mathbf{x}_{\mathrm{tl}}, \mathbf{x}_{\mathrm{br}} \in \mathbb{R}^2$ are the top-left and bottom-right of the bounding box that encloses the human performing the action, and $t_{\mathrm{b}}, t_{\mathrm{e}} \in \mathbb{R}^1$ are the beginning and end times of the action. Figure 3.1 shows an example of a detected action.

The Hough voting space for action detection consists of five-dimensions: action class label $c \in \mathbb{R}^1$, spatiotemporal position $\mathbf{x} \in \mathbb{R}^3$, and scale $s \in \mathbb{R}^1$. The spatiotemporal position is usually defined as a spatiotemporal center of an action (i.e., the spatial and temporal positions of actions are the centers of bounding boxes and time intervals, respectively). The scale is the height of the spatial bounding box of an action. Here, other scale parameters, such as aspect ratio and temporal duration, are fixed for each action class.

Two other definitions of action detection exist: temporal action detection [Gaidon et al., 2013] and action tube detection [Gkioxari and Malik, 2015]. Temporal action detection assumes that the video is already cropped spatially. It does not find a spatial bounding box (i.e., $\mathbf{x}_{tl}$ and $\mathbf{x}_{br}$) and detects actions in the temporal dimension. Action tube detection finds the spatial bounding box for each frame, whereas standard action detection only finds one bounding box for the whole action sequence.

3.2.2 Training

During training, the Hough-based method generates a codebook of local features using videos that include the following annotations: class labels, bounding boxes,

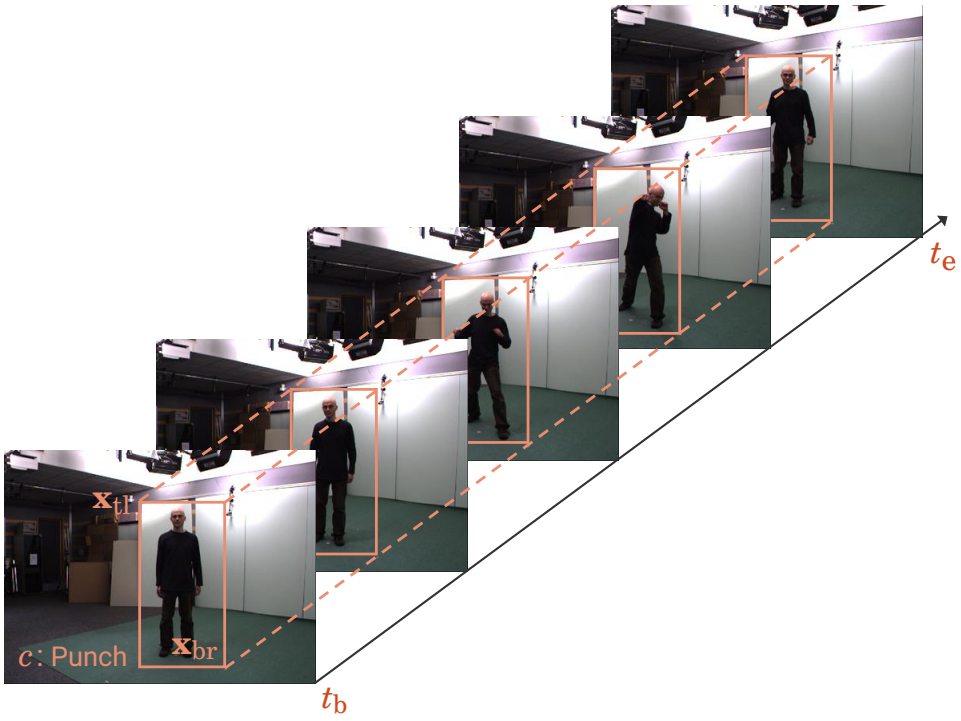


Figure 3.1: Problem definition of action detection.

and time intervals. First, the method crops videos using the bounding boxes and time intervals. The method extracts local spatiotemporal features from the cropped action sequences. Each local feature has an annotated class label and spatiotemporal offset from the local feature point to the annotated action position. Let $\mathbf{f} = (\mathbf{I}, \mathbf{a})$ be the local feature, where $\mathbf{I}$ is a visual feature vector and $\mathbf{a} = [c, \mathbf{d}, s]$ is an annotation for a local feature. Here, c is the class label, $\mathbf{d} \in \mathbb{R}^3$ is the spatiotemporal offset from the local feature point to the annotated action position, and s is the scale. In addition, the method extracts local features from negative samples that do not include any actions. The local features from the negative samples do not have offset $\mathbf{d}$.

The Hough-based method generates the codebook using training data; various codebook generation methods are used, including agglomerative clustering [Leibe et al., 2008], random forests [Gall et al., 2011], and random projection trees [Yu et al., 2012]. In this study, we use random forests. The details of codebook generation by random forests are described in Section 3.2.4. Each codebook entry stores a set of annotations consisting of class labels, offsets, and scales. The method casts votes using the set by matching local features to codebook entries.

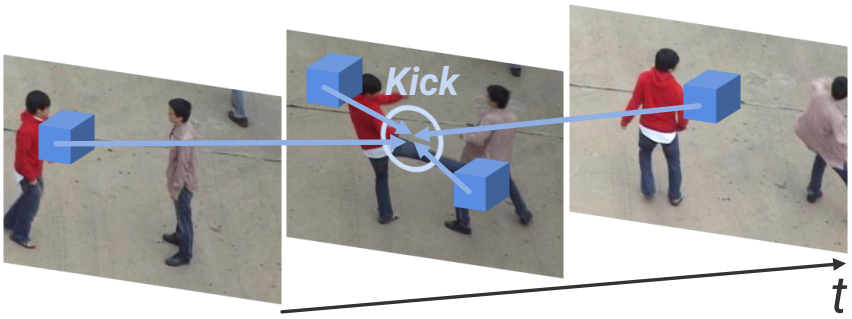


Figure 3.2: Hough-based action detection.

3.2.3 Detection

To detect actions, Hough-based methods cast votes for action classes, positions, and scales based on local features. Figure 3.2 shows an overview of Hough-based action detection.

First, we define the formulation of the probabilistic voting based on local feature $\mathbf{f}_\mathbf{y}$ extracted from position $\mathbf{y} \in \mathbb{R}^3$. The probability of an action of class c at position $\mathbf{x} \in \mathbb{R}^3$ of scale s based on local feature $\mathbf{f}_\mathbf{y}$ can be defined as

$$\begin{aligned} p(c, \mathbf{x}, s \mid \mathbf{f}_\mathbf{y}, \mathbf{y}) &= \sum_i p(c, \mathbf{x}, s \mid C_i, \mathbf{y}) p(C_i \mid \mathbf{f}_\mathbf{y}) \\ &= \sum_i p(\mathbf{x}, s \mid c, C_i, \mathbf{y}) p(c \mid C_i) p(C_i \mid \mathbf{f}_\mathbf{y}). \end{aligned} \tag{3.1}$$

The first term represents a position estimation for class c, the second term represents class estimation based on codebook entry C_i, and the third term represents the probability for matching between a local feature and a codebook entry. Codebook entry C_i stores set of annotations $\mathbf{A}_i = \{[c_i^1, \mathbf{d}_i^1, s_i^1], \ldots, [c_i^{N_i}, \mathbf{d}_i^{N_i}, s_i^{N_i}]\}$. The class estimation probability represented by the second term is calculated by the proportion of the class labels in $\mathbf{A}_i$. The position estimation probability represented by the first term can be defined as

$$p(\mathbf{x}, s \mid c, C_i, \mathbf{y}) = \frac{1}{|\mathbf{A}_i^c|} \sum_{[\mathbf{d}, s'] \in \mathbf{A}_i^c} G([\mathbf{x}, s], [\mathbf{y} + \mathbf{d}, s']), \tag{3.2}$$

where $\mathbf{A}_i^c$ is the set of annotations that has class label c in $\mathbf{A}_i$, G is a 4D Gaussian kernel, and $\mathbf{y} + \mathbf{d}$ refers to the voting position from local feature point $\mathbf{y}$ using offset $\mathbf{d}$. Using Equation (3.1), the voting score of an action of class c at position $\mathbf{x}$ of

scale s based on all local features extracted from a video can be defined as

$$v\left(c, \mathbf{x}, s \mid \mathbf{F}\right) = \sum_{\mathbf{f}_\mathbf{y} \in \mathbf{F}} p\left(c, \mathbf{x}, s \mid \mathbf{f}_\mathbf{y}, \mathbf{y}\right), \tag{3.3}$$

where $\mathbf{F}$ is the set of local features extracted from a video. The method finds local maxima of Equation (3.3) for each action class independently. Each local maximum has five-dimensions of information: class label, 2D-spatial and 1D-temporal position, and scale. The local maxima are candidate detected actions. Local maxima that have voting scores over a threshold are detected actions.

To generate a volume that centers around the action, the method calculates the spatial bounding box and time interval using scale s, the aspect ratio, and the duration. We use an average aspect ratio and duration in the training data.

3.2.4 Implementation

In this study, we implement the Hough-based method using Hough forests [Gall et al., 2011], which generate a codebook using random forests (i.e., ensembles of decision trees). This method optimizes parameters of the split functions of tree nodes during training.

This method trains the decision trees of random forests using a set of local features. Local features for each tree are randomly sampled from the entire set. During the training step for each tree, the method iterates the parameter optimization of the split function from the root to the leaf. To optimize node parameters, the method uses the set of local features classified into the node with the split function of the parent node. This iteration continues until each node satisfies termination criteria defined by the maximum depth or minimum number of local features in a node. Leaf node L stores $p(c \mid L)$, which is estimated by the label proportion of local features for each class c, and $\mathbf{D}_c^L$, which is a set of offsets for class c.

The split functions can be defined as

$$j_{i,q,r,\tau}(\mathbf{I}) = \begin{cases} 0 & \text{if } \mathbf{I}^i(q) < \mathbf{I}^i(r) + \tau \\ 1 & \text{otherwise,} \end{cases} \tag{3.4}$$

where i is a feature channel, q and r are the dimensions of $\mathbf{I}^i$, and τ is a threshold. Here, we use the multi-channeled feature vectors (i.e., $\mathbf{I} = [\mathbf{I}^1, \mathbf{I}^2, \cdots, \mathbf{I}^I]$, where I is the number of feature channels).

In the training step, i, q, r and τ are optimized. The method generates random parameter sets and selects the optimal one using either class uncertainty, which is defined as

$$U_1(\mathbf{A}) = -|\mathbf{A}| \sum_c p(c \mid \mathbf{A}) \ln p(c \mid \mathbf{A}), \tag{3.5}$$

where $\mathbf{A}$ is a set of local features and $|\cdot|$ denotes the number of elements in the set, or the uncertainty of the offsets, given as

$$U_2(\mathbf{A}) = \sum_c \left(\sum_{\mathbf{d} \in \mathbf{D}_c^{\mathbf{A}}} \left\| \mathbf{d} - \overline{\mathbf{d}_c^{\mathbf{A}}} \right\|^2 \right), \tag{3.6}$$

where $\mathbf{D}_c^{\mathbf{A}}$ is the offsets of the local features of class c in $\mathbf{A}$ and $\overline{\mathbf{d}_c^{\mathbf{A}}}$ is an average offset of $\mathbf{D}_c^{\mathbf{A}}$. Each node randomly chooses between these measures and selects the parameter set that minimizes uncertainty based on the split set of local features.

In the detection step, the method inputs local features into trained random forests. Here, random forests and leaf node L correspond to the codebook and codebook entry C, as described in Section 3.2.3. Each local feature is assigned to a leaf node (a codebook entry). Therefore, matching between the local features and the codebook is not probabilistic but deterministic. The method defines probability $p(L_i \mid \mathbf{f_y})$ as $1/K$ if $\mathbf{f_y}$ is assigned to leaf L_i and as 0 otherwise, where K is the number of trees. The probability of an action of class c at position $\mathbf{x} \in \mathbb{R}^3$ of scale s based on local feature $\mathbf{f_y}$ can be defined as

$$p(c, \mathbf{x}, s \mid \mathbf{f_y}, \mathbf{y}) = \frac{1}{K} \sum_i p(\mathbf{x}, s \mid c, L_i, \mathbf{y}) p(c \mid L_i). \tag{3.7}$$

The local maxima of Equation (3.7) indicate candidate actions.

Chapter 4

Vote Integration in Multi-view Videos for Robustness to Human Orientation Variety

4.1 Introduction

This chapter focuses on the problem of human orientation variety. Changes in action appearances are caused by the relative orientation of a human to the camera. When the difference in relative orientation between training and test data is large, accurate detection is difficult. To detect human actions, an approach that employs multiview videos can manage orientation variety. Multiview videos are synchronous videos captured from multiple cameras. Such observations, from various viewpoints, include a variety of orientations of the human relative to the cameras. Therefore, these observations reduce the difference in relative orientation between the training and test data and contribute to the robustness to orientation variety.

Most conventional methods for actions captured from multiview videos are not detection methods, but recognition methods [Yan et al., 2008, Naiel et al., 2011, Zhu et al., 2013]. As described in Section 1.1, action recognition methods can be applied only to segmented video. To detect actions in multiview videos that contain multiple action instances, action detection methods for multiview videos are required.

In this chapter, we propose a Hough-based action detection method for multiview videos. The method first casts votes independently in each view, then integrates them in global coordinates. To integrate votes, we use a homographic transformation, similar to [Sternig et al., 2011]. We assume that human feet touch the ground plane

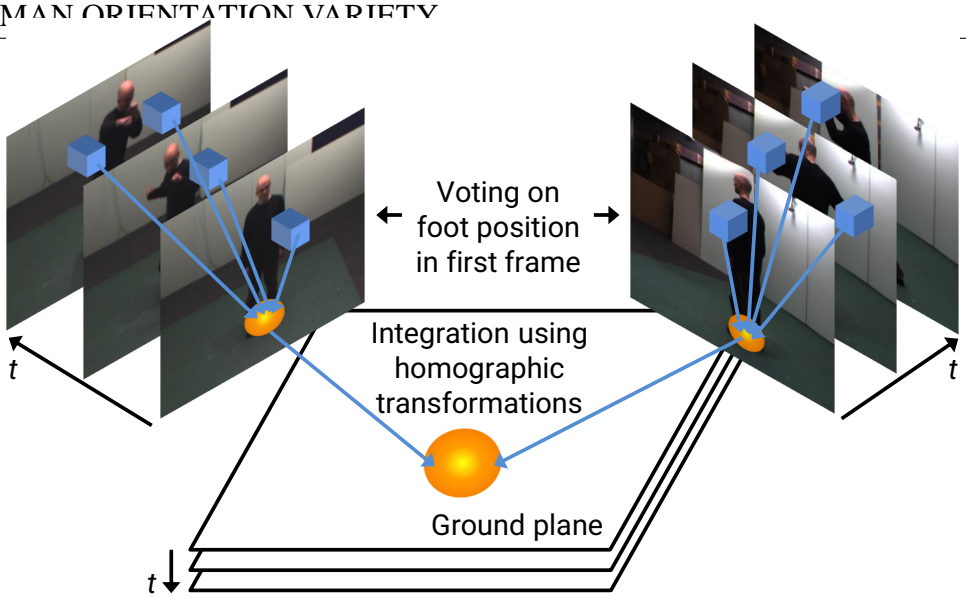


Figure 4.1: Main idea of our proposed approach. Votes are cast on human foot positions in the first frame and integrated using homographic transformation on the ground plane in global coordinates.

when they start an action. Based on this assumption, the proposed method integrates votes on the ground plane in global coordinates. Figure 4.1 shows its main idea.

The remainder of this chapter is organized as follows. Section 4.2 reviews related studies. We explain our proposed method in Section 4.3 and analyze our experimental results in Section 4.4. Section 4.5 summarizes this chapter.

4.2 Related Work

Other action recognition methods using multiview videos exist. Naiel et al. proposed majority-voting of classifiers trained in each view [Naiel et al., 2011]. Yan et al. proposed weighted voting of outputs in each view based on 3D reconstructed models [Yan et al., 2008]. Zhu et al. also adopted weighted voting based on the prediction uncertainty of a classifier [Zhu et al., 2013]. In contrast, we propose a multiview action detection method using multiview integration of Hough votes.

Many researchers have developed view-invariant features that robustly recognize human actions. Here, *view* is similar to relative orientation. Zheng and Jiang proposed a framework that learns dictionaries such that pairs of videos taken in two views have similar sparse representations [Zheng and Jiang, 2013]. Junejo et

al. claimed that distances between extracted low-level features for all pairs of time frames are stable under a variety of human orientations [Junejo et al., 2011]. These view-invariant features enable allow action descriptions that are robust description of actions to the orientation variety. However, the orientation variety changes affect not only the description, but also the spatial structure of local features. It is difficult to apply the view-invariant features to the Hough-based methods.

Sternig et al. proposed a Hough voting framework for multiview human tracking [Sternig et al., 2011]. First, they independently cast votes for the human foot position in each view. Then, they projected the votes onto the ground plane in global coordinates using homographic transformation to integrate the votes from each view. We apply this integration method to multiview action detection.

4.3 Hough-based Action Detection for MultiView Videos

This section explains the proposed Hough-based method for multiview videos. Our proposed method integrates votes from each view with a homographic transformation that maps a position in one plane to a position in another plane. The proposed method transforms a position in an image plane of view v (X_v, Y_v) into a position in the ground plane in global coordinates $(X_\mathrm{G}, Y_\mathrm{G})$. This transformation can be defined as

$$\tilde{\mathbf{x}_{i_\mathrm{v},t}} = \mathbf{H}_{i_\mathrm{v}} \tilde{\mathbf{X}_t}, \tag{4.1}$$

where $\mathbf{H}_{i_\mathrm{v}}$ is the homography matrix of view i_v, $\mathbf{x}_{i_\mathrm{v},t} \in \mathbb{R}^2$ is a position of the image plane in view i_v at frame t, $\mathbf{X}_t \in \mathbb{R}^2$ is a position on the ground plane in global coordinates at frame t, and the tilde denotes homogeneous coordinates. Homogeneous coordinates are usually used for projective geometry. For example, $\mathbf{u} = [x, y, \cdots]$ becomes $\tilde{\mathbf{u}} = [x, y, \cdots, 1]$ in homogeneous coordinates. $\tilde{\mathbf{u}}' = [x, y, \cdots, m]$ corresponds to $\mathbf{u}' = [x/m, y/m, \cdots]$ in orthogonal coordinates.

The homography matrix can be calculated using four point correspondences, which is easy compared with full-camera calibration. However, positions transformed by homography must be located on a common plane in the global coordinate system. Votes by the conventional method do not satisfy this condition. To solve this problem, our proposed method defines the action position as the foot position in the first frame of an action. In the following, we explain the reason for this definition in detail. We then describe Hough-based action detection for multiview videos using

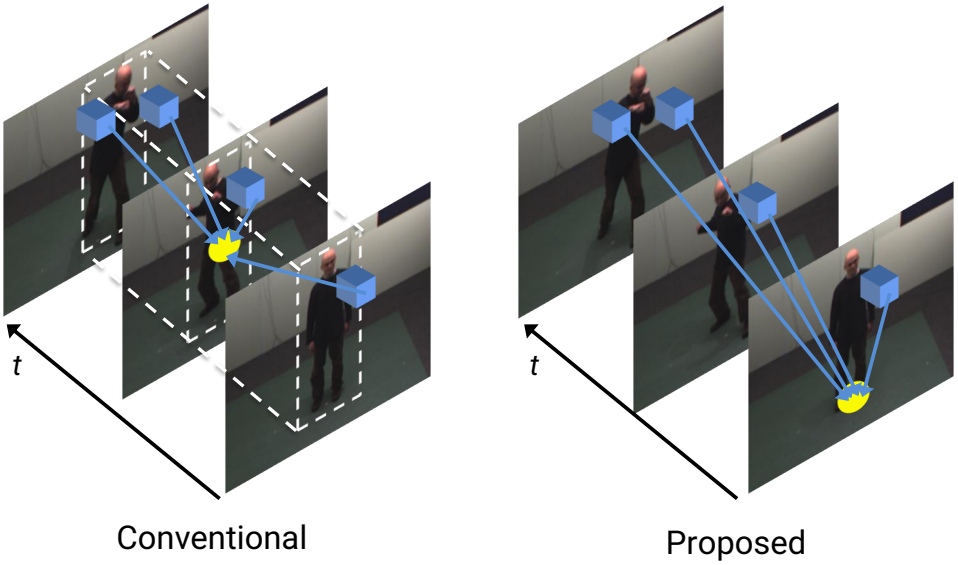


Figure 4.2: The definition of the action position in conventional methods and our proposed methods. Conventional methods define the action position as the spatiotemporal center, whereas the proposed method defines it as the foot position in the first frame of the action.

homographic transformation.

4.3.1 Definition of the Action Position

Most conventional Hough-based methods define an action position as the spatiotemporal center in the video of an action, as shown in Figure 4.2 (left). This definition spatially concentrates votes in the center of the human body. The centers differ depending on various factors, such as pose and the height of the human, and are not located on a common plane in global coordinates. Therefore, this method cannot integrate the votes by homographic transformations.

In contrast, we define the action position as the human's foot position in the first frame of an action, as shown in Figure 4.2 (right). A person's feet inevitably touch the ground plane when he or she begins an action. This means that the position of the feet is on the ground plane. The Hough-based method casts votes on the ground plane using the proposed definition and can integrate votes by homographic transformations.

4.3.2 Vote Integration by Homographic Transformation

Our proposed method first casts votes in X_{i_v}, Y_{i_v}, T-space for each view i_v independently using Equation (3.7). It then integrates the votes in X_G, Y_G, T-space. Here, X_G, Y_G is the ground plane in global coordinates. Using homographic transformations, the integration can be defined as

$$v(c, \mathbf{X} \mid \mathbf{F}_{\text{all}}) = \sum_{i_v} v\left(c, \mathbf{x}_{i_v} \mid \mathbf{F}_{i_v}\right), \tag{4.2}$$

where $\mathbf{F}_{\text{all}}$ is the set of local features extracted from a multi-view video; $\mathbf{F}_{i_v}$ is the set of local features extracted from a video of view i_v; $\mathbf{X} \in \mathbb{R}^3$ is a position in X_G, Y_G, T-space; and $\mathbf{x}_{i_v} \in \mathbb{R}^3$ is a position in X_{i_v}, Y_{i_v}, T-space. $v\left(c, \mathbf{x}_{i_v} \mid \mathbf{F}_{i_v}\right)$ corresponds to Equation (3.1), and the relation between $\mathbf{X}$ and $\mathbf{x}_{i_v}$ is represented by Equation (4.1). Here, we ignore scale s in the voting process because scales of each view are different. Backprojection of contributing votes [Leibe et al., 2008] can estimate the scales of the actions.

Figure 4.1 shows the voting and integration using homographic transformations. We integrate the votes for each spatiotemporal position and view and obtain scores in X_G, Y_G, T-space. The integration denotes that the voting scores in each view are added for each position. The proposed method finds local maxima of Equation (4.2) for each action class independently. The local maxima are candidates actions in X_G, Y_G, T-space. The method defines local maxima with voting scores over a threshold as detected actions.

Our proposed method sums the voting scores of each view represented as Equation (4.2). We expect that if actions captured from some views are significantly different from the actions in training data, actions captured from other views are similar to the actions in training data. The votes in similar views work well and detect actions accurately.

4.4 Experiments

We explain the experiments performed in this section. We compared our proposed method with one baseline and three other methods [Yan et al., 2008, Naiel et al., 2011, Zhu et al., 2013]. The difference between the baseline method and the proposed method is the input videos. The baseline method only uses single-view videos, whereas the proposed method uses multiview videos. The other methods

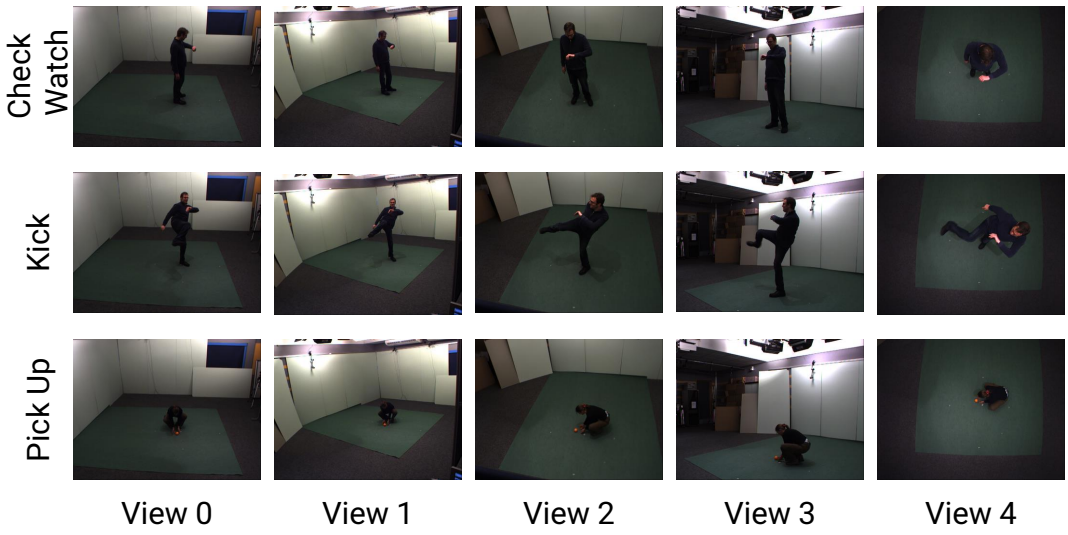


Figure 4.3: Examples from the IXMAS dataset.

are multiview action recognition methods. To compare with these methods, we used videos that are temporally segmented to contain only one action in these experiments.

4.4.1 Dataset

We used two different dataset: the IXMAS dataset [Weinland et al., 2006a] and the UCR Videoweb Activities dataset [Denina et al., 2011]. The IXMAS dataset is captured in an indoor experimentally environment. The dataset 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 ten actors and recorded by five cameras. The resolution and frame rate of the videos were 390×291 and 23 fps, respectively. The orientations of the actions differed from one actor to another. We calculated the homography matrix using the camera calibration data provided by the dataset. The ground truth, which is set at the human foot position in the first frame, is manually annotated for each video of each view. Figure 4.3 shows the examples of the IXMAS dataset.

The UCR Videoweb Activities dataset was captured in a complex outdoor environment. The dataset comprises over 2.5 hours of videos captured by 4–8 cameras. We chose 100 scenes for this experiment, each of which was captured by 2–3 cameras and contains one action. These scenes include six action classes: hug, shake hands, shove, stand up, pick up object, and throw object. The resolution and frame rate of the videos were 640×480 or 704×480 pixels and 30 fps, respectively. We

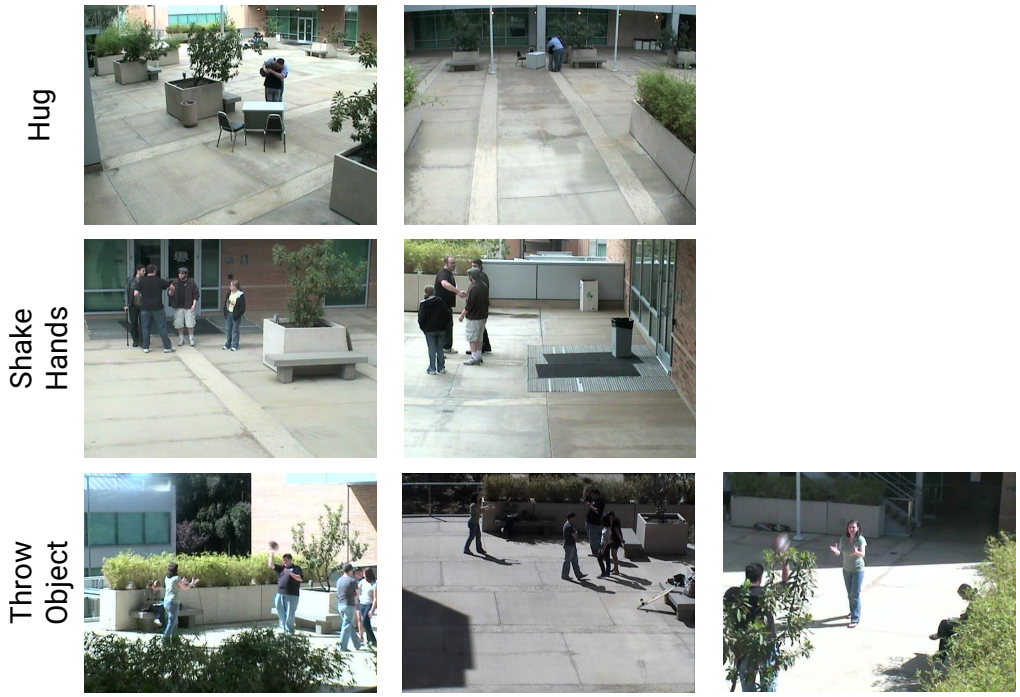


Figure 4.4: Examples from the UCR Videoweb Activities dataset.

calculated the homography matrix using manually corresponding points between each camera. The ground truth was also manually annotated for each video of each view. The ground truth positions of hug and shake hands are the center between the foot positions of the two people. Figure 4.4 shows examples from the UCR Videoweb Activities dataset.

4.4.2 Evaluation method

We evaluated the methods using recognition accuracy and localization error. We calculated the recognition accuracy by comparing the recognized label with the correct label. We calculated the localization error from the distance between detected position $\mathbf{x}^l_{i_\mathrm{v}}$ and ground truth $\mathbf{x}^g_{i_\mathrm{v}}$ in $X_{i_\mathrm{v}}, Y_{i_\mathrm{v}}, T$-space. Position $\mathbf{x}^l_{i_\mathrm{v}}$ was projected from detected position $\mathbf{X}^l$ in $X_\mathrm{G}, Y_\mathrm{G}, T$-space using (4.1). We evaluated the localization error by independently calculating the spatial and temporal distances, because the former are pixel units and the latter are frame units. The spatial distance is calculated between the spatial information of $\mathbf{x}^l_{i_\mathrm{v},t^l}$ and that of ground truth $\mathbf{x}^g_{i_\mathrm{v},t^g}$. The temporal distance is calculated between t^l and temporal ground truth t^g. Note that the localization error is evaluated only for actions that are correctly recognized.

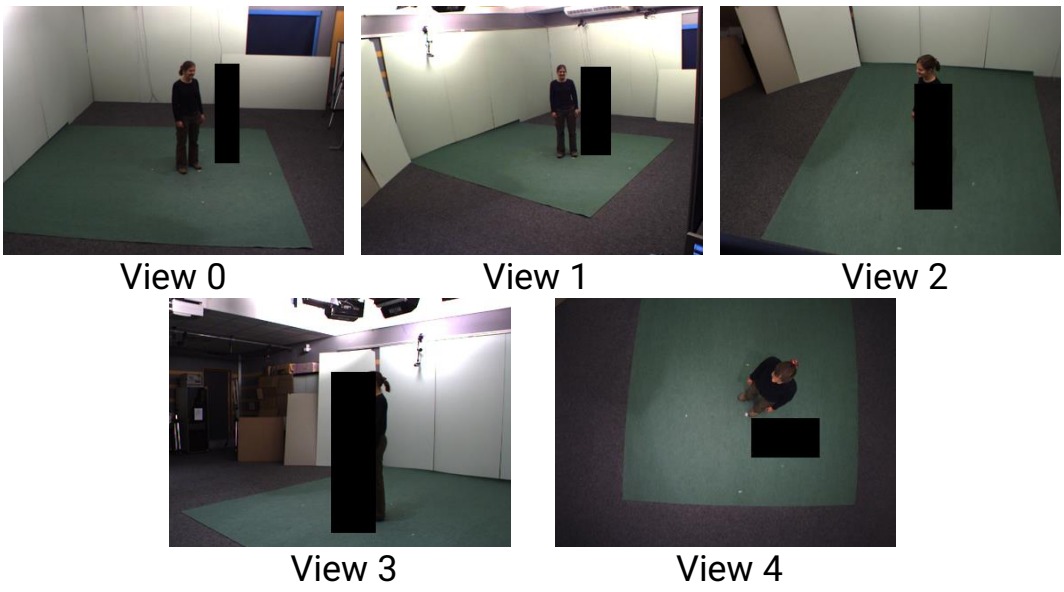


Figure 4.5: Occlusion setting in *S2*. The black rectangles are the artificially occluded regions.

4.4.3 Results

IXMAS Dataset

The experiments using the IXMAS dataset consisted of the following three settings:

Setting 1 (S1) Using all views during the training.

Setting 2 (S2) Using all views during the training and detecting occluded actions during the testing.

Setting 3 (S3) Using one view during the training.

Our proposed method used all views for detecting actions in all settings. *S1* shows the results for a large set of training data. *S2* shows the robustness to occlusions. (Figure 4.5 shows the occlusion in this setting.) *S3* shows the results when training data only include a subset of views. Our proposed method detects actions using all views in *S2*. A classifier is trained using one view, and our proposed method votes in each view using the same classifier. We compared the proposed method with the baseline method in this setting.

We employed leave-one-actor-out cross-validation that uses the data of one actor as the test data and the remainder as training data for all settings. We trimmed

Table 4.1: Recognition accuracy (%) for the IXMAS dataset in *Setting 1*.

Method	Position	All	View0	View1	View2	View3	View4
Proposed		89.1	-	-	-	-	-
Hough		-	83.3	82.1	80.9	86.7	70.0
Yan et al.	✓	78.0	72.0	53.0	68.0	63.0	-
Naiel et al.	✓	84.6	80.4	79.8	80.1	77.1	-
Zhu et al.	✓	88.0	71.5	78.7	73.9	-	-

the videos so that each contains one action. Both our proposed and the baseline methods detected the position of the largest voting scores in X_G, Y_G, T-space as the action position.

Setting 1

The methods used all views for training in *S1*. Figure 4.6 depicts a confusion matrix of the recognition accuracy averaged over all views using the proposed method in *S1*. The proposed method achieves high accuracy for most action classes, but fails in recognizing *Pick Up*, possibly because of the similar stooping motions that are part of *Sit Down* and *Pick Up*. It is difficult to accurately recognize actions that include similar motions to other actions.

Table 4.1 presents comparisons with the conventional methods. The *All* column shows the recognition results using multiview integration, and the other columns show the recognition results in each view. The *Position* column shows whether the method needs prior knowledge of the actor's position. The *Hough* rows shows the results of the baseline method that uses only single-view videos. Compared with *Hough*, the proposed method achieves higher accuracy. We confirmed that the proposed integration improves recognition accuracy. Compared with the other methods, our proposed method achieves the highest accuracy, indicating that it is also effective for action recognition tasks.

Table 4.2 shows the evaluation results of localization in *S1*. We evaluated the localization error in X_{i_v}, Y_{i_v}, T-space. The *Average* row shows the average result of all views. The errors of both the proposed and baseline methods are small and do not differ significantly. The baseline method localizes actions in each view accurately. These results indicate that accurate localization in each view enables vote integration in the proposed method.

Setting 2

We added artificial occlusions in *S2*. Figure 4.7 shows a confusion matrix of the recognition accuracy averaged over all views in *S2*. The average recognition accuracy

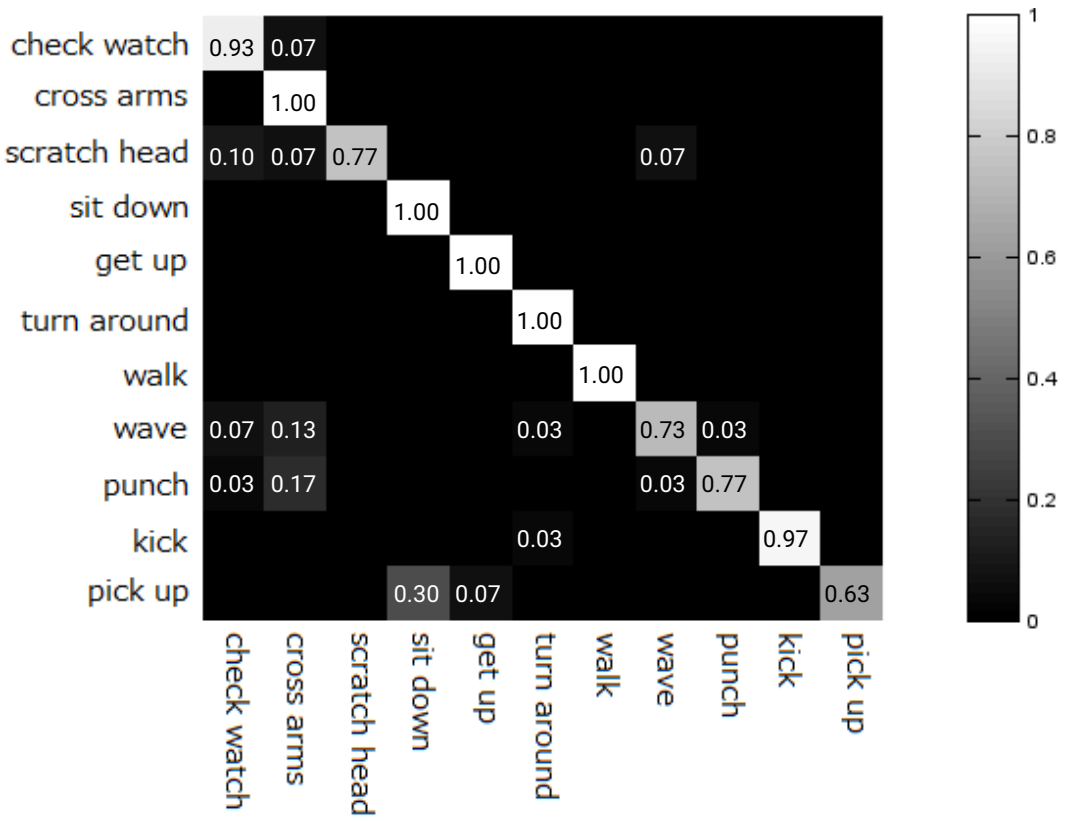


Figure 4.6: Confusion matrix of our proposed method in *S1*.

Table 4.2: Location estimation error for the IXMAS dataset in *Setting 1*.

Method		Spatial Distance (pixel)		Temporal Distance (frame)	
		Mean	Std. Dev	Mean	Std. Dev
Proposed	View0	10.7	8.8	4.4	6.4
	View1	9.1	8.0	4.4	6.4
	View2	12.5	9.3	4.4	6.4
	View3	11.8	7.8	4.4	6.4
	View4	12.8	10.9	4.4	6.4
	Average	11.4	9.1	4.4	6.4
Hough	View0	9.6	7.0	4.6	5.9
	View1	7.3	4.5	4.1	4.7
	View2	12.4	10.5	5.1	7.4
	View3	11.6	8.3	5.4	8.1
	View4	11.8	10.4	5.3	9.2
	Average	10.5	8.1	4.9	7.1

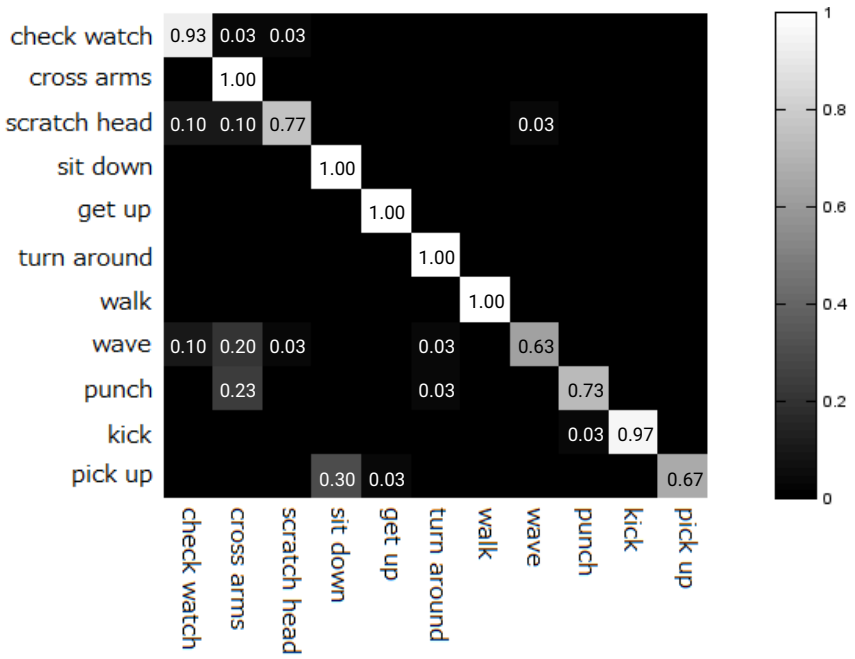


Figure 4.7: Confusion matrix of our proposed method for the IXMAS dataset in *Setting 2*.

for all action classes was 88.2%. Table 4.3 shows the results of the localization error in *S2*. The artificial occlusion covers a person in Views 2 and 3, as shown in Figure 4.5. The result of Figure 4.7 is competitive with the result in Figure 4.6. The result of Table 4.3 is also competitive with the result in Table 4.2, indicating that the proposed method is robust to large occlusions.

Setting 3

The methods used only one view for training in *S3*. Figure 4.8 depicts a confusion matrix of the recognition accuracy averaged over all views in *S3*. Table 4.4 shows the recognition accuracy results in *S3*. The rows except for *Average*, and the columns except for *Proposed*, in Table 4.4 indicate the view used for the training and detection, respectively. The proposed method uses all views for detection. The proposed method achieves high accuracy in all cases. Accuracy was high for the baseline method in cases that were trained using one view and recognized actions using the same view. However, the accuracy was very low for the baseline method when trained using one view and recognizing actions in another view. These results indicate that the multiview integration in the proposed method makes it robust to changes in viewpoint.

Compared with *S1*, the accuracy of the proposed method was low, which means

Table 4.3: Location estimation error for the IXMAS dataset in *Setting 2*.

Method		Spatial Distance (pixel)		Temporal Distance (frame)	
		Mean	Std. Dev	Mean	Std. Dev
Proposed	View0	9.3	7.9	3.9	4.5
	View1	8.1	6.7	3.9	4.5
	View2	13.1	8.8	3.9	4.5
	View3	12.9	8.2	3.9	4.5
	View4	11.1	8.6	3.9	4.5
	Average	11.1	8.6	3.9	4.5

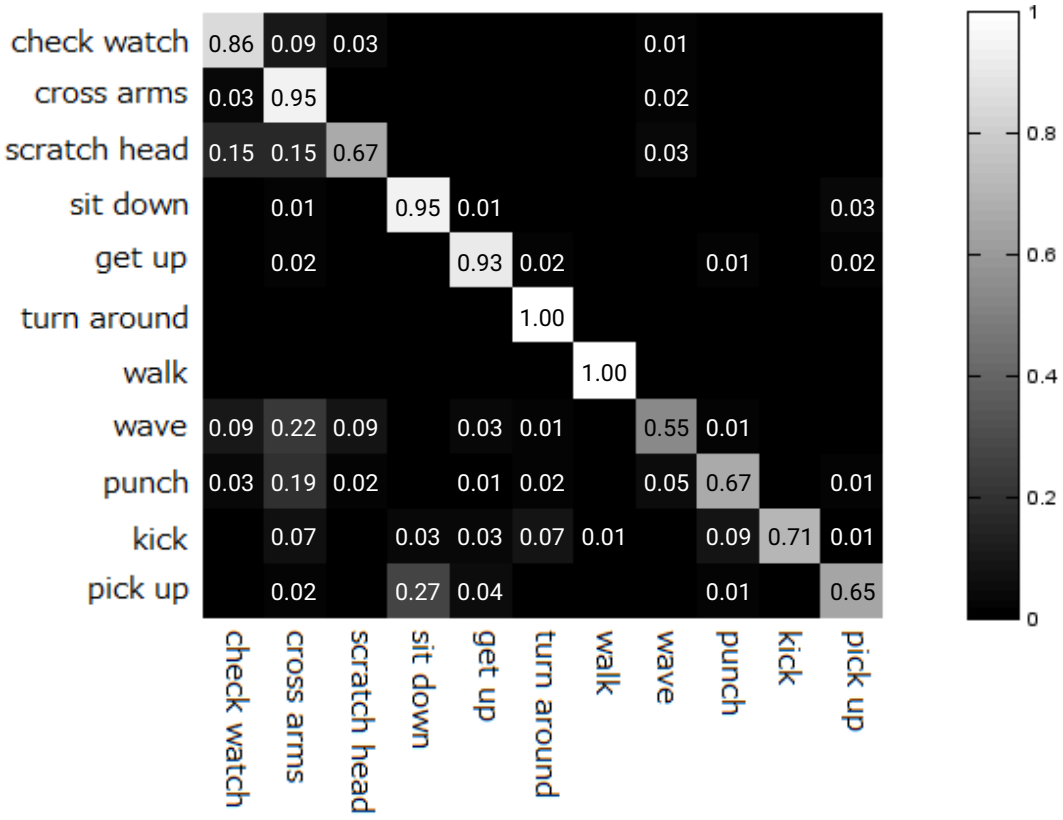


Figure 4.8: Confusion matrix of our proposed method in *Setting 3*.

Table 4.4: Recognition accuracy (%) for the IXMAS dataset in *Setting 3*.

Training \ Test	Proposed	Hough				
		View0	View1	View2	View3	View4
View0	81.8	84.2	78.5	53.9	54.8	27.0
View1	83.0	77.0	83.9	59.1	58.2	26.1
View2	79.7	48.8	45.2	80.0	61.2	46.7
View3	88.8	55.2	57.9	69.4	88.5	27.3
View4	72.1	16.4	12.7	33.3	17.3	73.3
Average	81.1	82.0				

Table 4.5: Means of the spatial error of location estimation (pixels) for the IXMAS dataset in *Setting 3*.

Training \ Test	Proposed	Hough				
		View0	View1	View2	View3	View4
View0	11.9	8.4	15.6	21.2	27.8	61.0
View1	9.9	19.1	7.1	19.7	29.7	62.7
View2	14.3	20.3	15.8	11.2	29.6	43.4
View3	13.0	25.5	24.0	28.2	11.7	95.8
View4	13.6	139.7	141.2	78.9	154.5	11.6
Average	14.7	10.5				

that training data captured from various viewpoints improves accuracy even for the multiview videos.

Tables 4.5 and 4.6 show the results of the spatial and temporal localization error in *S3*, respectively. The *Average* row shows the distances averaged over all views for training. The results in these tables shows similar tendencies to the accuracies in Table 4.4.

UCR Videoweb Activities Dataset

We divided the 100 scenes into five sets for each action class and employed 5-fold cross-validation in this experiment. In the training, we used 106 scenes captured from one camera in addition to the 100 test scenes. We removed scenes that have occlusions from the training data. The homography matrix in the dataset represents the relation between each camera. The proposed method integrates votes not in $X_{\mathrm{G}}, Y_{\mathrm{G}}, T$-space, but in one of the views. Each video contains one action. Both our proposed and the baseline method detected the position of the largest voting scores

Table 4.6: Means of the temporal error of location estimation (frames) for the IXMAS dataset in *Setting 3*.

Training \ Test	Proposed	Hough				
		View0	View1	View2	View3	View4
View0	4.3	4.5	5.4	8.5	9.3	10.4
View1	4.2	5.3	4.3	7.2	8.0	8.8
View2	4.8	8.0	5.9	5.0	7.1	7.9
View3	4.8	10.8	8.8	5.8	5.1	7.5
View4	5.1	26.4	32.2	18.1	27.5	4.8
Average	5.4	5.4				

Table 4.7: Results for the UCR Videoweb Activities dataset.

Method	Recognition Accuracy (%)	Spatial Distance (pixel)		Temporal Distance (frame)	
		Mean	Std. Dev	Mean	Std. Dev
Proposed	95.3	16.6	29.4	1.9	1.8
Hough	87.0	19.8	56.8	5.1	17.8

as in Section 4.4.3.

Figure 4.9 depicts a confusion matrix of the recognition accuracy of the proposed method. Table 4.7 shows the recognition accuracy and localization error of the proposed and baseline methods. Compared with the baseline method, the proposed method achieves high recognition accuracy and low localization error. These results indicate that vote integration in the proposed method is also effective in complex scenes.

4.5 Summary

We proposed an action detection method for multiview video sequences to provide robustness to orientation variety. Our method is based on a Hough voting framework and homographic transformations. We first vote for the action classes and human foot positions in each view and integrate these votes using homographic transformations. This integration enables us to robustly detect actions.

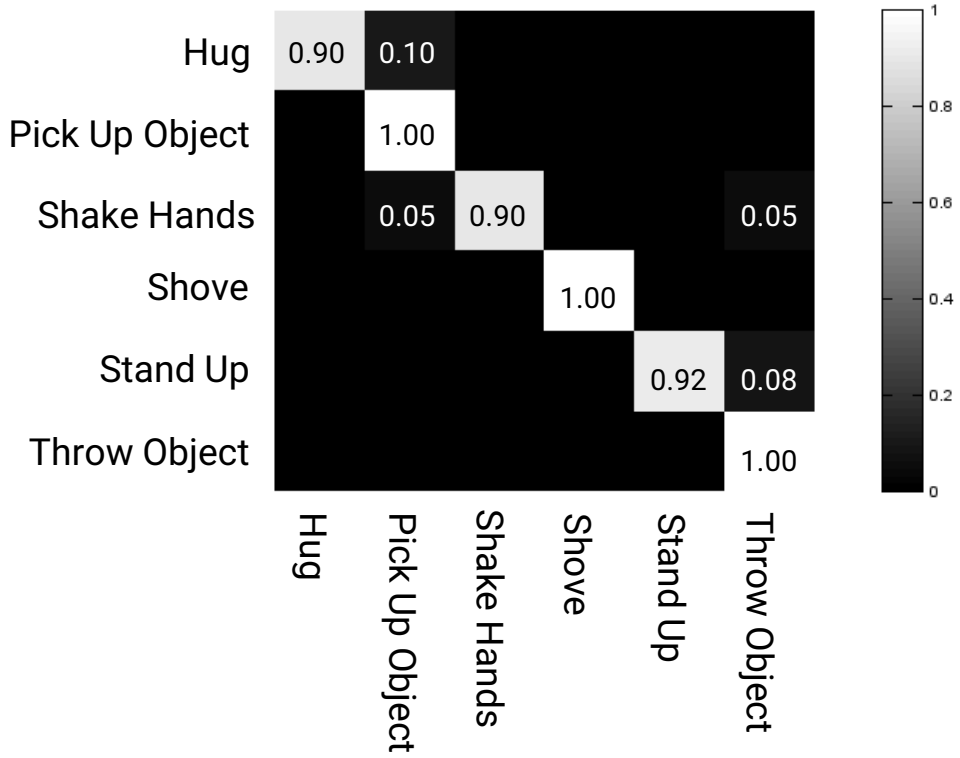


Figure 4.9: Confusion matrix of our method for the UCR Videoweb Activities dataset.

Chapter 5

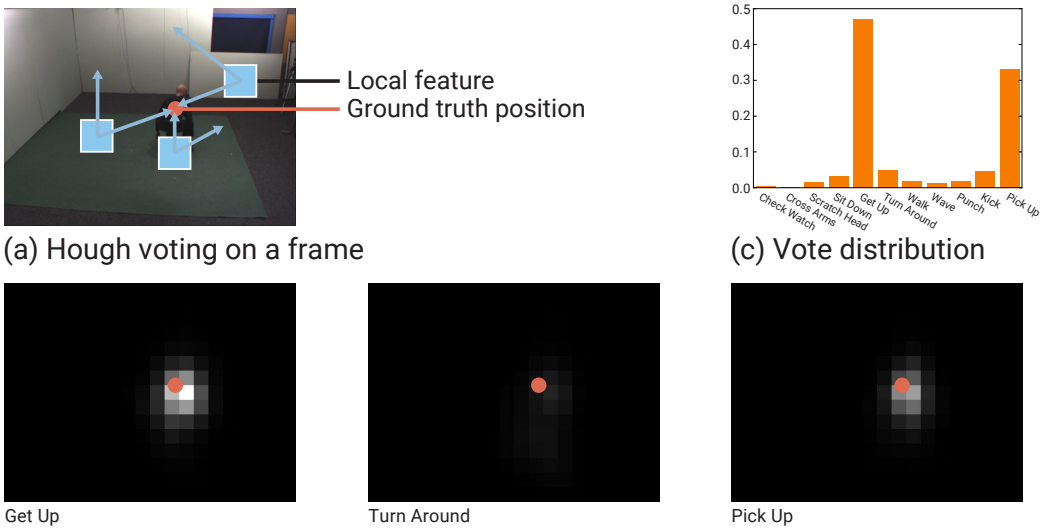
Vote Distribution Model for Robustness to Motion Similarity

5.1 Introduction

This chapter focuses on the problem of motion similarity. If similar local motions exist across different action classes, discriminating between such motions is difficult. Because the Hough voting process of each local feature is performed independently, the Hough-based method naturally votes for all relevant action classes. Therefore, the Hough-based method is prone to casting votes for the wrong action classes and, thus, detecting many false positives.

We propose a novel method for overcoming the false-vote problem by examining the cause of false votes. Similar local features can cause false votes. These false votes do not occur randomly; thus, 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 one action is performed. These characteristics are different based on each class and do not necessarily correlate. We assume that the distribution of false votes includes important information necessary to improving action detection. Our proposed method learns these characteristics. We then introduce vote distributions, which represent the voting scores for each action class, as shown in Figure 5.1. Our proposed method builds a model that represents characteristics based on vote distributions. The method estimates likelihood using the model and reduces the influence of false votes.

The main contribution of this chapter is that our vote distribution model improves the performance of Hough-based action detection by reducing the influence of false



(a) Hough voting on a frame

(c) Vote distribution

(b) Part of 2D slices of voting spaces

Figure 5.1: Hough voting and vote distribution. (a) This frame is an example of the Get Up class. A Hough-based approach casts votes based on local features of in the frame. (b) The votes can be visualized as 2D slices of voting spaces. The pixel values of the slices are the voting scores 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.

votes caused by similar local motions across action classes. The experimental results shown later in this chapter support this assertion.

The remainder of this chapter is organized as follows. Section 5.2 reviews related studies. We explain our proposed method in Section 5.3 and analyze our experimental results in Section 5.4. Section 5.5 summarizes the study in this chapter.

5.2 Related Work

Hough-based approaches must handle false votes. Many studies have been conducted that consider this problem for both object and action detection. One approach for overcoming false votes involves generating a discriminative codebook. Maji et al. and Wohlhart et al. proposed approaches to learning the discriminative weights of the codebook using max-margin frameworks [Maji and Malik, 2009, Wohlhart et al., 2012]. Some studies have adopted a supervised method for generating a codebook using random forests [Gall et al., 2011] and locality-constrained linear

coding [Vijay Kumar and Patras, 2013]. These studies have improved the training step, which generates a codebook, 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 therefore combine the proposed method with the methods in this section.

Other studies have attempted to solve the problem of false votes by improving the voting process. Razavi et al. indicated that sparsity of local appearance is an effective measure for discriminating foreground and background features [Razavi et al., 2012]. In the voting process, their method selects foreground features based on sparsity measures and reduces the influence of background features. This method is effective, but only for false votes generated by background features. Some studies have attempted to group local features to reduce the independence of Hough voting [Yarlagadda et al., 2010, Srikantha and Gall, 2014]. 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 voting is completed. The proposed method can also be combined with these other methods to further 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 [Woodford et al., 2014]. 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 the weights of false votes generated by background features during optimization. Our proposed method is not affected by background features, unless they change the vote distributions.

Hoai and Zisserman introduced relative class score (RCS), which is similar to vote distributions, for action recognition [Hoai and Zisserman, 2015]. RCS is a vector representation of output scores of a multiclass action classifier. The main difference between the vector representing vote distributions and RCS is whether each method sorts the vector elements. The scores would become independent of the action classes because of sorting, so that RCS-based action recognition could not achieve satisfactory accuracy. We experimentally confirmed that vote distributions are superior to RCS for Hough-based action detection.

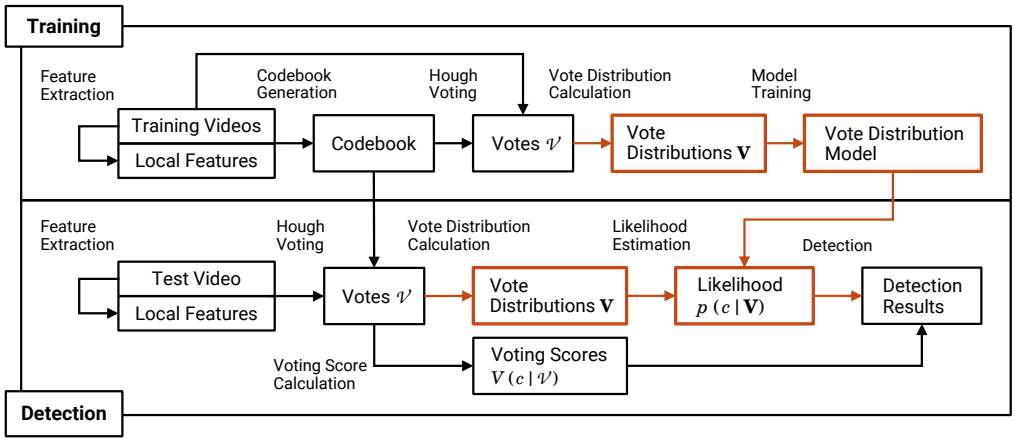


Figure 5.2: Flow of our proposed method. The orange boxes are our proposed elements.

5.3 Hough-based Action Detection with Vote Distribution Model

We introduce a vote distribution model to the conventional Hough-based method to improve its robustness to false votes. Our proposed method reduces the influence of false votes by learning the characteristics of Hough voting. Figure 5.2 shows the flow of our proposed method. We further explain vote distributions in Section 5.3.1. In Section 5.3.2, we demonstrate the training step of the vote distribution model. Finally, we describe the detection step of our proposed method in Section 5.3.3.

5.3.1 Vote Distribution

We represent the characteristics of Hough voting using the voting scores for all action classes. Similar local features generate false votes, although these false votes do not occur randomly. Rather, they depend on relevant action classes. Conventional Hough-based approaches basically cast votes for not only a certain class but also other specific classes, as shown in Figure 5.3. We define the normalized voting scores for all classes at a position as a vote distribution. The vote distribution represents voting characteristics. Specifically, the vote distribution at position $\mathbf{x} \in \mathbb{R}^3$ of scale

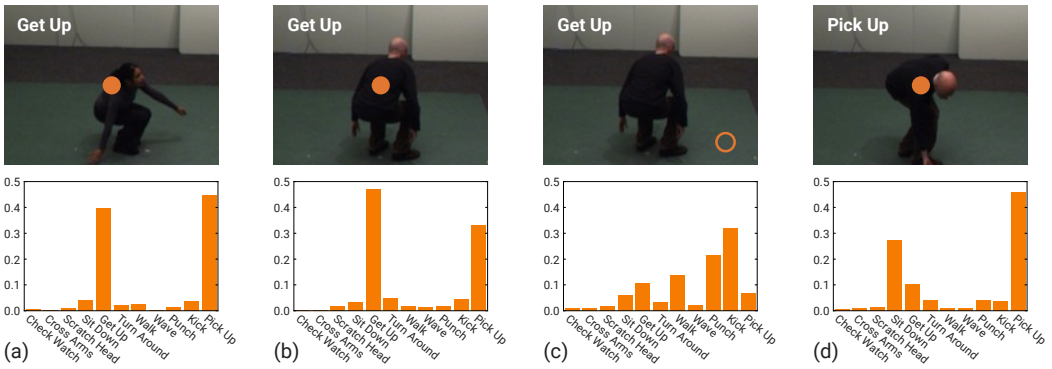


Figure 5.3: Examples of vote distributions calculated at circle positions. The filled and open circles indicate the ground truth and other positions, respectively. (a), (b), and (c) are examples of the *Get Up* class, and (d) is an example of the *Pick Up* class. The vote distributions of (a) and (b) are similar. These examples reveal that the calculated vote distributions at the ground truth positions are similar if they are of the same class.

s can be represented by the following equation:

$$\mathbf{V}(\mathbf{x}, s) = \left[\frac{v\left(c^1, \mathbf{x}, s \mid \mathbf{F}\right)}{Z}, \ldots, \frac{v\left(c^N, \mathbf{x}, s \mid \mathbf{F}\right)}{Z} \right],$$
$$Z = \sum_{j=1}^{N} v\left(c^j, \mathbf{x}, s \mid \mathbf{F}\right), \tag{5.1}$$

where v is the voting score represented by Equation (3.7), N is the number of action classes, and Z is the normalization constant.

Figure 5.3 shows the characteristics of Hough voting represented by a vote distribution. The vote distribution at the ground truth position of an action has a high score on the correct class. The distribution also has a high score on classes with similar local motions. In this figure, the distribution of the *Get Up* class shows high scores on the *Get Up* and *Pick Up* classes, whereas the distribution of the *Pick Up* class shows high scores on the *Sit down* and *Pick Up* classes. The characteristics differ depending on each class and position and do not necessarily correlate. The proposed method builds a vote distribution model by learning these characteristics.

5.3.2 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 a model that estimates likelihood for each class based on vote distributions, we use a feature vector. A multiclass classifier trained using these vectors can work as a vote distribution model and output likelihood $p(c \mid \mathbf{V})$ using vote distribution $\mathbf{V}$.

Before extracting the vectors, the proposed method prepares the codebook described in Section 3.2.4. The method then executes the conventional voting process described in Section 3.2.4 on videos from the training data. We define vote distributions at the ground truth and surrounding positions 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 would be cast at 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 likelihood $p(c \mid \mathbf{V})$ based on vote distribution $\mathbf{V}(\mathbf{x}, s)$. Here, the random forests use vote distributions instead of local features. and adopt only the class uncertainty measure of Equation (3.5) because the vote distributions do not have spatio-temporal offsets.

5.3.3 Detection

In the detection step, our proposed method initially calculates votes using the conventional voting process. We can then calculate the vote distributions using Equation (5.1) and estimate likelihoods based on the distribution. The proposed method detects actions using both the voting scores and likelihood, multiplying the scores by the likelihood. Low likelihoods 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 $\mathbf{x}$ of scale s can be defined as:

$$v\left(c, \mathbf{x}, s \mid \mathbf{F}, \mathbf{V}(\mathbf{x}, s)\right) = v\left(c, \mathbf{x}, s \mid \mathbf{F}\right) p\left(c \mid \mathbf{V}(\mathbf{x}, s)\right), \tag{5.2}$$

The proposed method finds local maxima of Equation (5.2) for each action class independently. The local maxima are candidate actions. The method uses the local maxima that have voting scores greater than a threshold as detected actions.

Figure 5.4 shows an example of reducing the effect of false votes when using the proposed method. The first row of the figure shows a video frame, the voting scores, the likelihood based on vote distribution, and the product of the voting scores and likelihood for the *Get Up* class. The second row of the figure provides the same for the *Pick Up* class. The second, third, and fourth columns correspond to $v\left(c, \mathbf{x}, s \mid \mathbf{F}\right)$ in Equation (3.7), $p(c \mid \mathbf{V}(\mathbf{x}, s))$, and $v\left(c, \mathbf{x}, s \mid \mathbf{F}, \mathbf{V}(\mathbf{x}, s)\right)$ in Equation (5.2), 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). In contrast, the pixel values of (e) refer to the influence of false votes for the *Pick Up* class when the *Get Up* action is performed. This influence is reduced in (g) using the low values estimated by the proposed method in (f). This reduction enhances the robustness of the proposed method to false votes.

High values exist at non-ground-truth positions of the correct class and positions of the wrong class, as shown in Figure 5.4 (c) and (f), respectively. These values represent 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 Equation (5.1), a slight difference in extremely low voting scores causes considerable variation in vote distributions during normalization. Therefore, estimation at such positions is incorrect. However, such noise does not considerably influence detection performance 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.

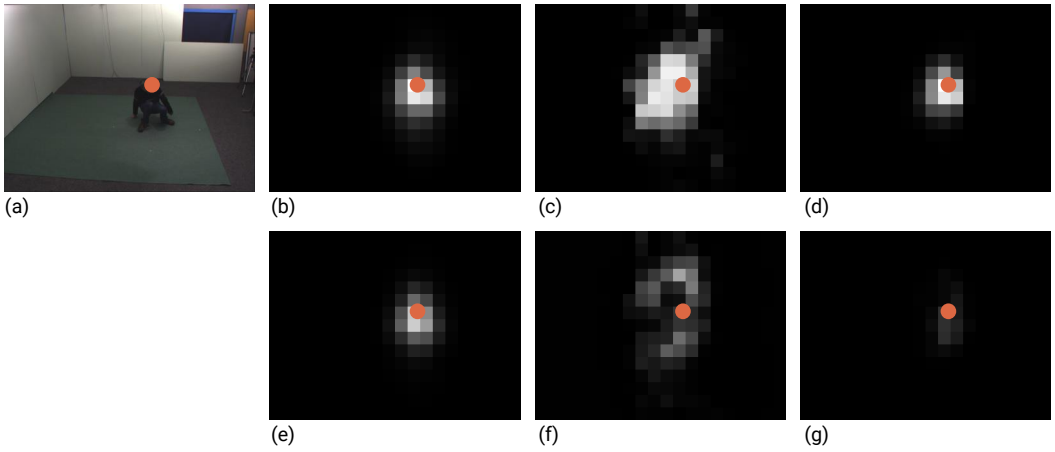


Figure 5.4: Reducing the influence of false votes. The first row shows a video frame, the voting scores, the likelihood based on the vote distribution, and the product of the voting scores and likelihood for the *Get Up* class. The second row lists the same information for the *Pick Up* class. (a) is 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).

5.4 Experiments

We evaluated our proposed method using two public action datasets: the IXMAS [Weinland et al., 2006b] and UT-Interaction [Ryoo and Aggarwal, 2010] datasets.

We compared the proposed method with three baseline methods and a related method proposed by Hoai et al. [Hoai and Zisserman, 2015]. The first method is the conventional Hough-based action detection described in Section 3.2. The second method adds nonmaximum suppression over action classes to the first method. The voting score of the second method can be defined as

$$v^{\max}(c, \mathbf{x}, s \mid \mathbf{F}) = \begin{cases} v(c, \mathbf{x}, s \mid \mathbf{F}) & \text{if } c = c_{\mathbf{x}}^{\max} \\ 0 & \text{otherwise,} \end{cases} \tag{5.3}$$

where $c_{\mathbf{x}}^{\max} = \operatorname{argmax}_c V(c, \mathbf{x} \mid \mathcal{V})$. 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 Figure 5.3. 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. RCS vectors are designed in a one-vs-all manner. The first element of an 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 the other action classes. Therefore, RCS vectors identify only the target class, whereas our vote distributions identify all classes by the original order of the classes. The method estimates class likelihood based on RCS vectors.

We used STIP [Laptev, 2005] and HOG/HOF descriptors [Laptev et al., 2008] for the local features of both the proposed and 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 as in Equations (3.7), (5.2), and (5.3). In these experiments, we used quick shift [Vedaldi and Soatto, 2008] to identify the initial coarse local maxima. We then refined the local maxima using mean shift [Comaniciu and Meer, 2002].

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

5.4.1 Datasets

The IXMAS dataset is described in Section 4.4.1. The dataset provides ground truth labels that include a time interval and human silhouettes for each action execution. We defined the spatiotemporal center of the time interval and the bounding boxes generated using the silhouettes as the ground truth of the action positions.

The UT-Interaction dataset [Ryoo and Aggarwal, 2010] 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

Figure 5.5: Examples from the UT-Interaction dataset.

the action position as for the IXMAS dataset. Figure 5.5 shows examples from this dataset. The resolution and frame rate of the videos are 720×480 pixels and 30 fps, respectively. In contrast with the IXMAS dataset, the UT-Interaction dataset includes the simultaneous occurrence of multiple actions.

5.4.2 Results

IXMAS Dataset

We employed leave-one-actor-out cross-validation that uses the data of one actor as test data and the remainder as training data. In the training step, the proposed and third baseline methods built a vote distribution model. To avoid overfitting, as described in Section 5.3.2, we divided the training data using the leave-one-actor-out strategy in each validation iteration. We generated a codebook using the larger dataset and generated training data for the vote distribution model using the remainder. This generation was repeated while changing the division.

Table 5.1 lists the f-score averaged over all cameras and Figure 5.6 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 [Hoai and Zisserman, 2015]. *Avg* is the f-score averaged over all 11 classes. The proposed method achieves the highest results of all methods, indicating that using vote distributions improves action detection. The performance of *Max* is superior to that of *Standard*, but inferior to that of the proposed method. *Max*

Table 5.1: Average f-score for all cameras in 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**

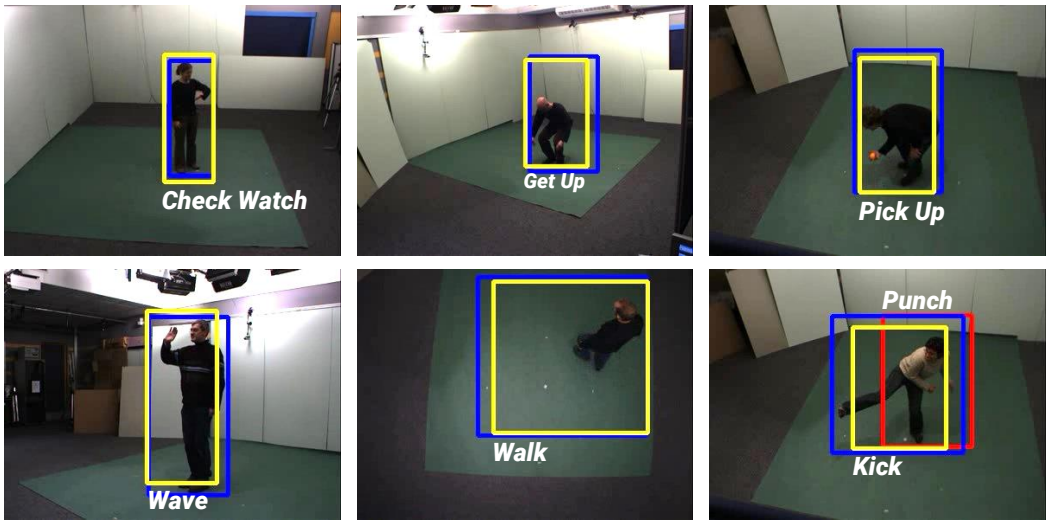


Figure 5.6: Output examples of our proposed method on the IXMAS dataset. Yellow, red, and blue rectangles indicate true positives, false positives, and ground truths, respectively.

only considers whether the voting score for a class is the maximum over all action classes. In contrast, our proposed method uses the distribution of 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, but our proposed method uses the voting scores as well as the model. This poor performance is based on the incorrect estimation described in Section 5.3.3. Normalizing the vote distribution would generate an incorrect estimation at positions with extremely low voting scores. To avoid the influence of noise, we can use a 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 probabilities of zero for all action classes.

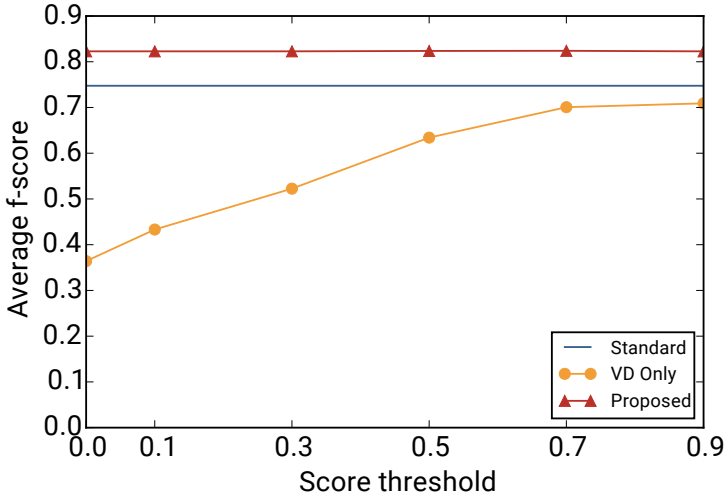


Figure 5.7: F-scores averaged over all action classes as a function of score threshold on the IXMAS dataset.

Note that this threshold is different from the threshold for detection, and only decides whether the estimation is performed or not. Figure 5.7 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 incorrect estimation. In contrast, the performance of our proposed method does not vary with changes in thresholding. The proposed method multiplies the voting scores by the likelihood and thus reduces the influence of 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 than *Hoai*, which indicates that the vote distribution representation is superior to RCS for Hough-based action detection. Figure 5.8 shows examples of vote distributions and RCS. The vote distributions for *Get Up* and *Pick Up* have different characteristics (see (a) and (b)). In 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 5.2 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 the from overhead; the motion captured from this location does not possess sufficient visual features for classes that contain upward and downward motions, such as *Sit Down* and *Get Up*. Moreover, the aspect ratios in Camera 4 varied significantly

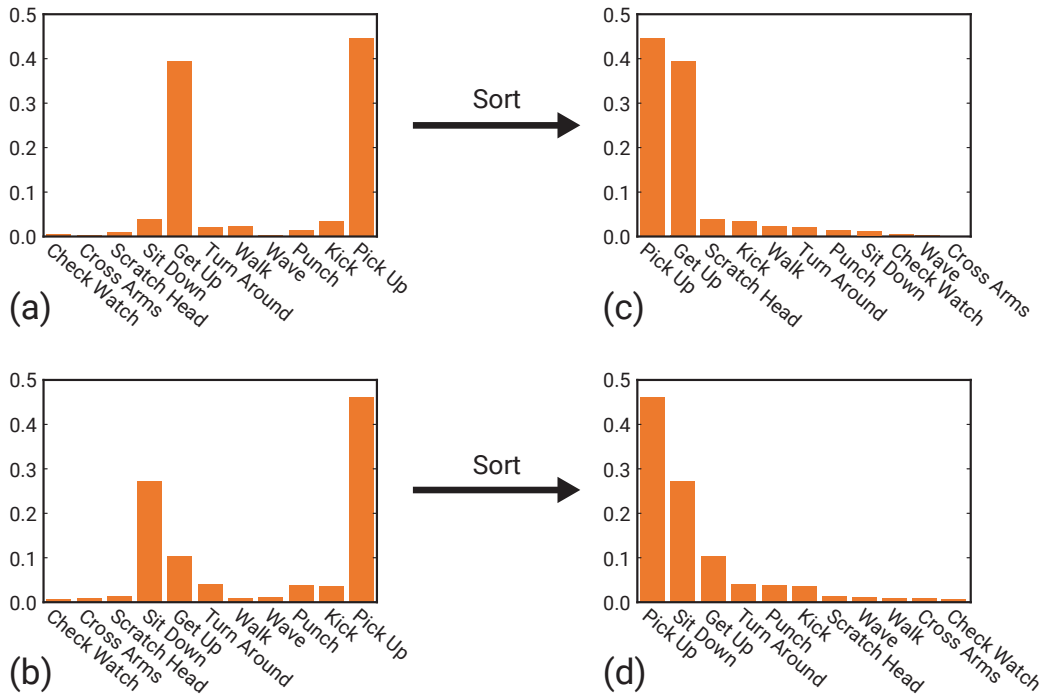


Figure 5.8: Examples of vote distributions and RCS on the IXMAS dataset. (a) and (b) are proposed vote distributions for *Get Up* and *Pick Up*, respectively. (c) and (d) are the RCS of *Get Up* and *Pick Up*, respectively.

Table 5.2: 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

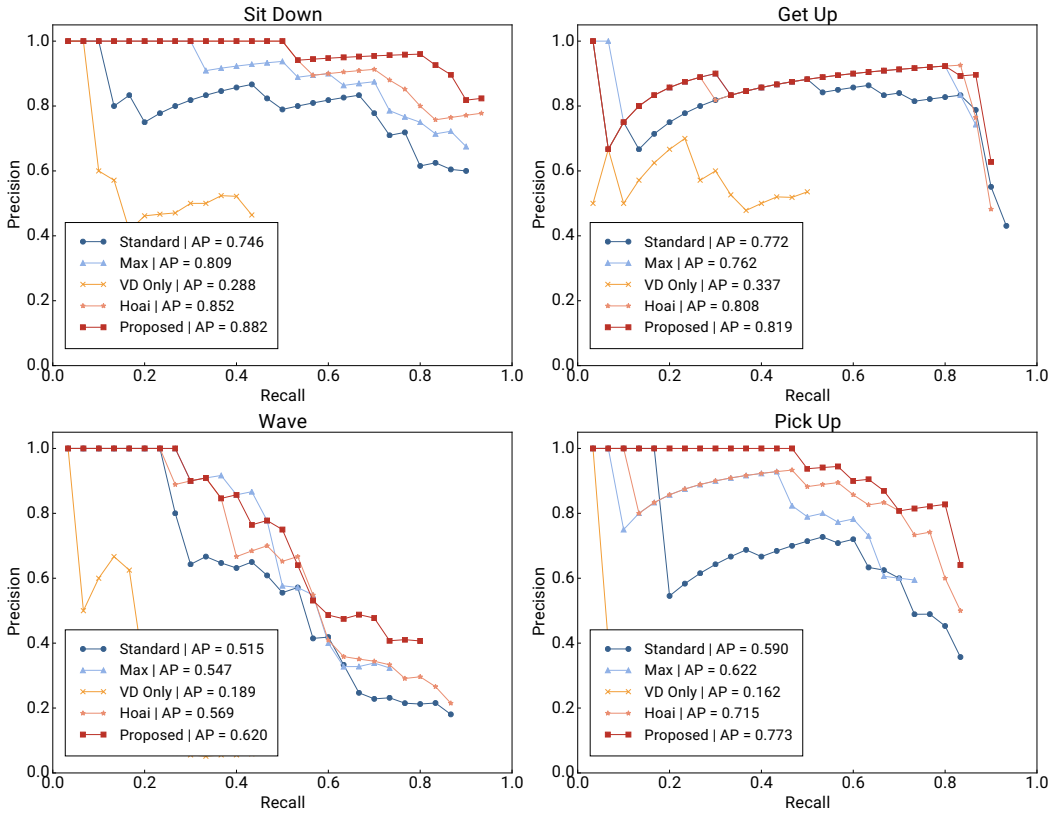


Figure 5.9: Precision-recall curves of Camera 0 in the IXMAS dataset. AP is the average precision.

depending on the direction in which the person faced. 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.

Figure 5.9 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 therefore selected three curves: the *Sit Down*, *Get Up*, *Punch*, and *Pick Up* classes of Camera 0. Compared with the other methods, our proposed method achieves high precision for each recall value. These results indicate that the proposed method is effective in reducing false detections.

Figure 5.10 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 detections in which *Check Watch* was detected while *Cross Arms* was performed. *Standard* reveals some false detections between certain classes. For instance, the

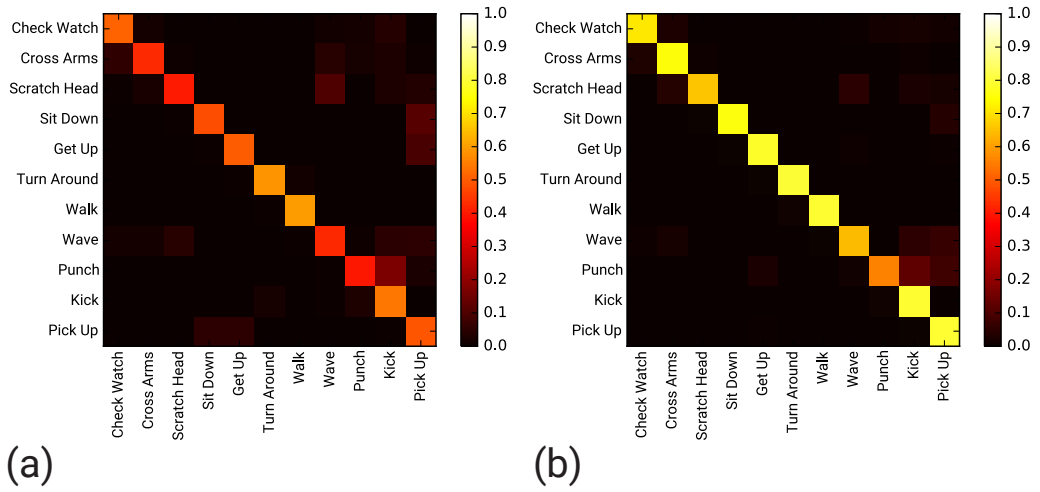


Figure 5.10: Confusion matrix of: (a) *Standard* and (b) *Proposed* in the IXMAS dataset.

method detected *Sit Down* when *Pick Up* was performed. This false detection might be caused by the 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.

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 placed such motions in the *Others* class for training. For detection, we did not cast votes for any class if the local features were classified into the *Others* class. For training the vote distribution models, we divided the training data as in the previous experiment, except the 10-fold strategy was used for division instead of the leave-one-actor-out strategy.

Table 5.3 lists the f-scores and Figure 5.11 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 other methods. Figure 5.12 shows the precision-recall curves of the methods. Our proposed method achieves high precision for most recall values, as for the IXMAS dataset. These results show that the proposed method is also effective when multiple actions occur simultaneously.

Table 5.3: 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**

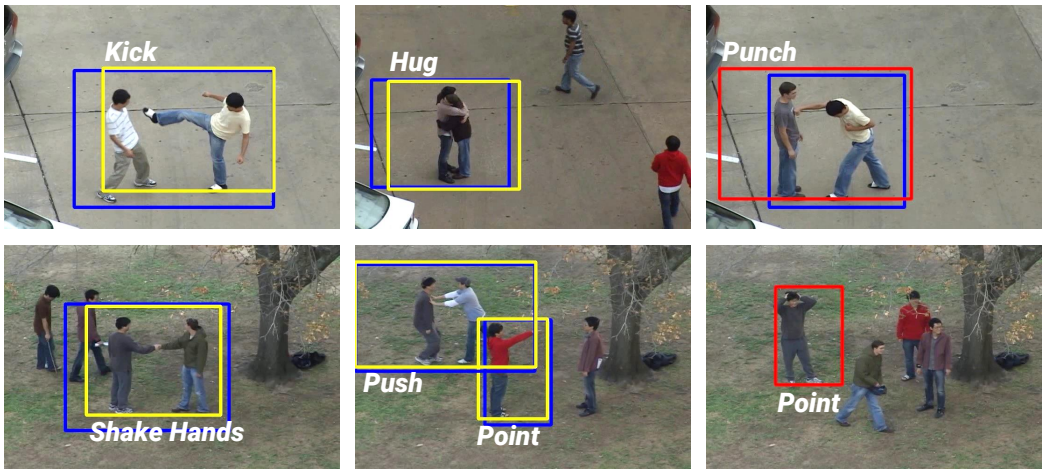


Figure 5.11: Output examples of our proposed method on the UT-Interaction dataset. Yellow, red and blue rectangles indicate true positives, false positives, and ground truths, respectively.

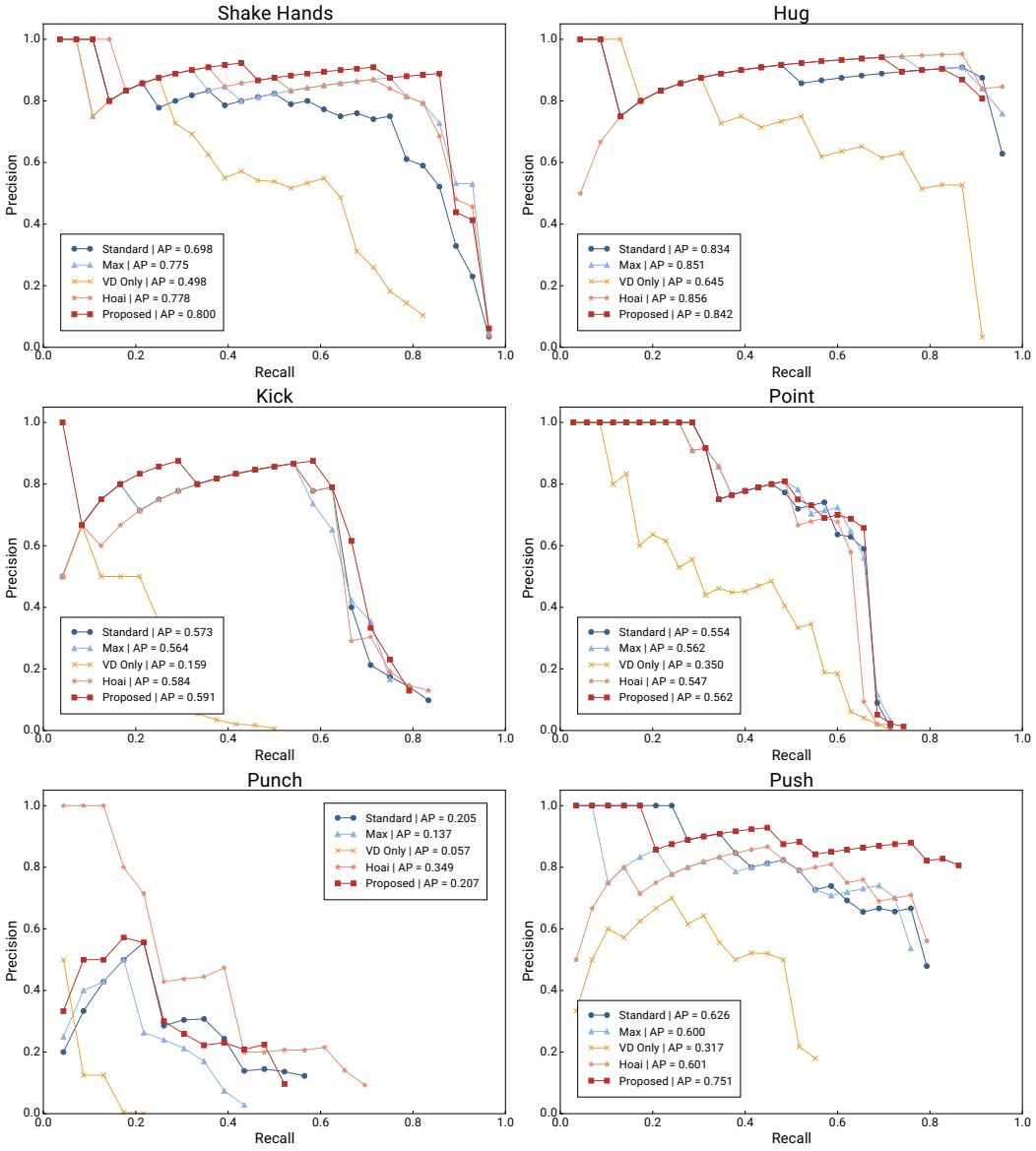


Figure 5.12: Precision-recall curves in the UT-Interaction dataset. AP is the average precision.

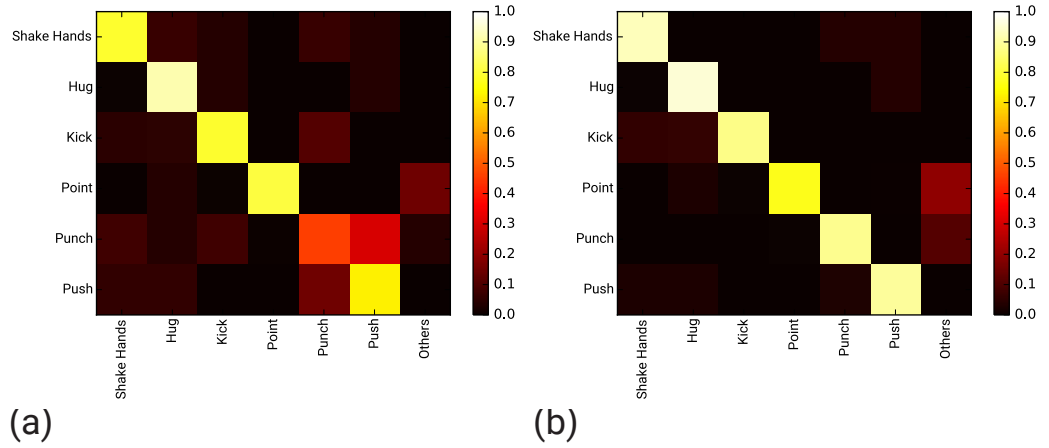


Figure 5.13: Confusion matrix of: (a) *Standard* and (b) *Proposed* for the UT-Interaction dataset.

Figure 5.13 shows the confusion matrices for *Standard* and *Proposed*. In our experiments, *Standard* frequently detected *Push* when *Punch* was performed, likely due to the similar arm motions that are part of *Push* and *Punch*. Similarly, with respect to the IXMAS dataset, the proposed method reduced the number of false action detections, including similar motions in the UT-Interaction dataset.

The f-score of the proposed method for *Punch* is lower than that for *Standard*. This result would only occur when there are fewer correct votes. The vote distribution including such correct votes flattens out, as shown in Figure 5.14 (a). When the distribution flattens out, the difference in frequency among action classes is small. The likelihood of the correct action class based on such vote distribution tends to be low. Figure 5.14 (b) and (e) show the relation between the original voting score and likelihood based on a vote distribution. Each data point in these figures represents a local maximum around the ground truth positions for the corresponding class. Figure 5.14 (b) shows that the original voting scores for *Punch* are low. Per the low score, the likelihood for *Punch* is low. The proposed method degraded the voting score of the local maximum by multiplying by the likelihood. The threshold for voting scores should be decreased to detect degraded local maxima as true positives. The lower threshold would cause additional false positives, hurting the f-score for *Punch*.

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 Figure 5.14(c). The likelihood

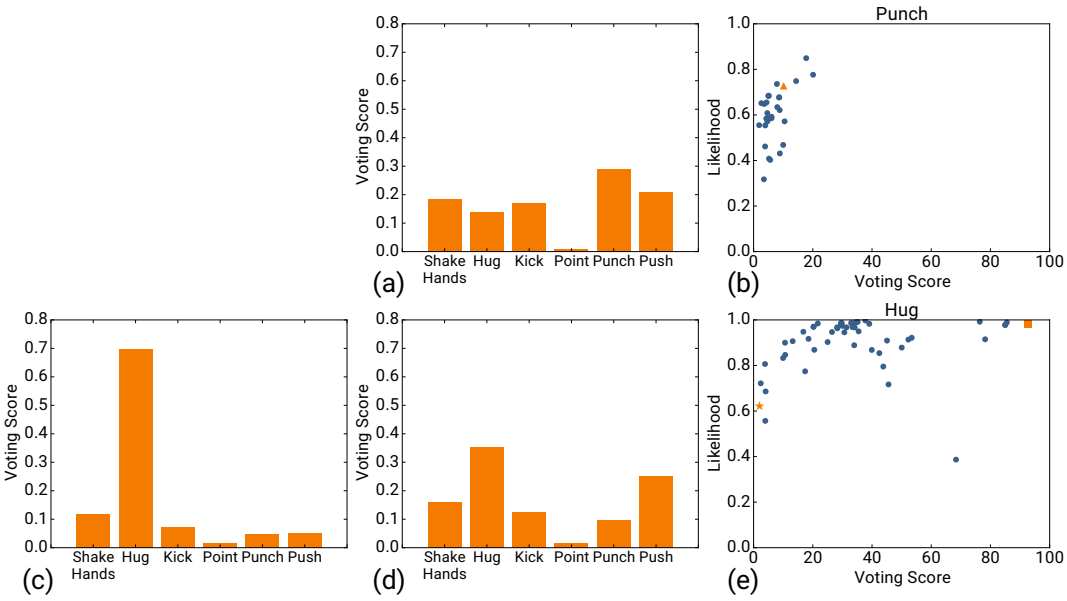


Figure 5.14: Voting scores and likelihood based on vote distribution. Each data point in (b) and (e) represents a local maximum around the ground truth positions for each class. (a) is the vote distribution of the local maximum described by the orange triangle in (b). (c) and (d) are the vote distributions of the local maxima described by the orange square and star in (e), respectively.

for Hug based on such distribution tends to be high. However, some local maxima have flat vote distributions, as shown in Figure 5.14 (d). Figure 5.14 (e) shows that such local maxima have low voting scores and the likelihood for *Hug* based on such distributions tends to be low. Therefore, like *Punch*, the proposed method hurt the f-score for *Hug* because the multiplied voting score decreased.

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 the 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, which we generated for each action. Figure 5.15 shows examples of the occlusions. We divided the width of the bounding box of each action equally into 10 regions. We then randomly chose regions to occlude. We used the 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 using the UT-Interaction dataset.

Figure 5.15: Examples from the UT-Interaction dataset with artificial occlusions. There are two occluded regions in these examples.

Figure 5.16 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 5.3. The f-score of both the *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.

5.5 Summary

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

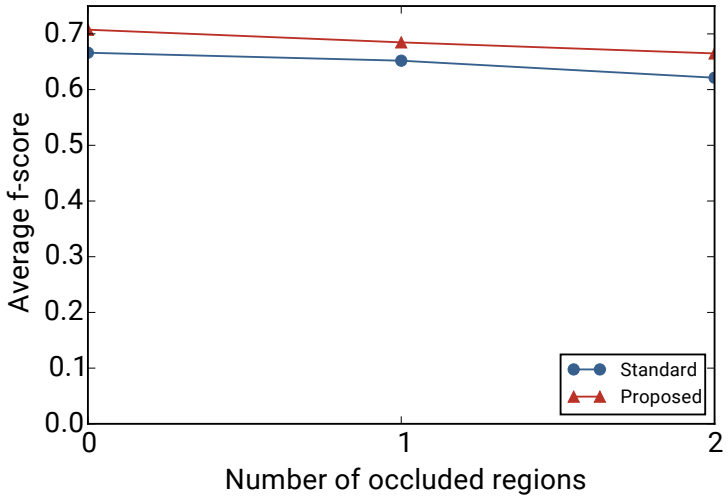


Figure 5.16: F-score averaged over all action classes in the UT-Interaction dataset with artificial occlusions.

Chapter 6

Time-warped Voting for Robustness to Temporal Variations

6.1 Introduction

This chapter focuses on the problem of temporal variations in action detection. Conventional Hough-based action detection methods perform poorly for actions with temporal variation. Most Hough-based methods cast votes in $X, Y, T-$space based on spatial and temporal relations between feature positions and action center positions. If any such variations exist, the temporal relation of the test data is different from that of the training data. Some votes do not concentrate on action center positions because of this difference as shown in Figure 6.1. The scattered votes lead to poor detection performance.

There are some methods for handling temporal variations, such as dynamic time warping (DTW) [Sakoe and Chiba, 1978] and hidden Markov models (HMM) [Bobick and Wilson, 1997]. Zhou and Torre aligned action sequences based on DTW [Zhou and Torre, 2012]. Lv and Nevatia recognize actions using their proposed model, which is based on HMM [Lv and Nevatia, 2007]. These methods can be applied to time series data. The Hough-based methods transform feature positions into voting positions. The transformation discards the temporal structure of the actions. Methods like DTW cannot easily be applied to Hough-based methods.

In this chapter, we propose a novel method for overcoming the temporal variation problem. We introduce time warping of votes using temporal offsets. We assume that even if scattered votes decrease the voting scores at the correct action center positions, the scores at the positions still form local maxima. Our proposed method

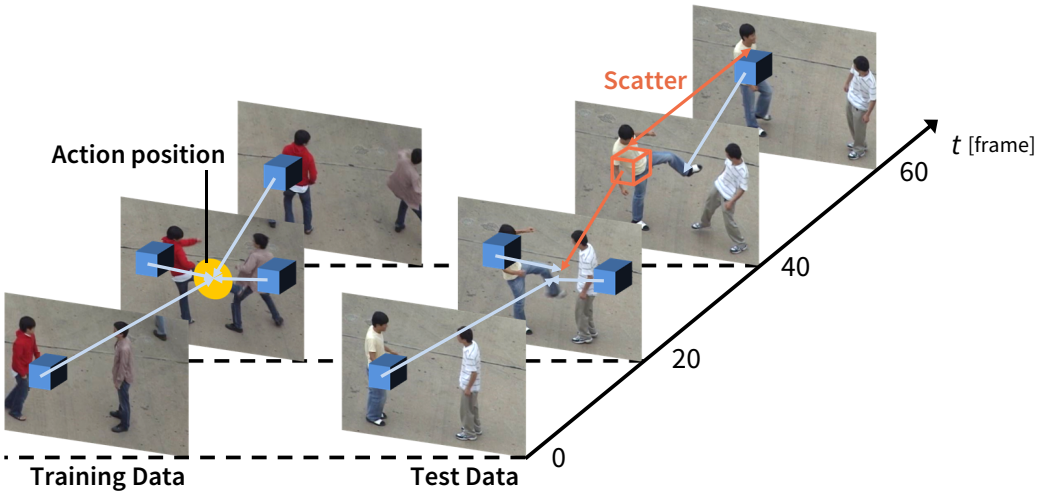


Figure 6.1: Hough voting and temporal variations of actions. Both the left and right video sequences contain *Kick* actions, though the duration of the actions differs. Some votes in the right sequence by a Hough-based method, using the left sequence as training data, are temporally scattered.

calculates offsets by finding the local maxima of the votes from voting positions. These offsets warp the scattered votes to concentrate them, making the method robust even in the presence of temporal variations.

6.2 Time-warped Voting

We propose a novel Hough-based method to overcome temporal variations in actions. Temporal variations in actions scatter votes in the temporal dimension. As shown in Section 3.2, Hough-based methods cast votes based on the spatiotemporal offsets between local feature positions and the action positions of the training data. When temporal variations occur between the training and test data, the offsets in the temporal dimension are different even if they are of similar features. These differences prevent the votes from concentrating on the correct action center positions and lower the resulting voting scores at these positions. In addition, the differences are propagated to the future temporal relation of the actions. Therefore, temporal variations affect the performance of the method significantly.

Our proposed method concentrates scattered votes by introducing another temporal offset to each local feature. The offsets are parameters to warp votes from their scattered positions to the correct action center positions. The reason why the

proposed method does not introduce the offset to each vote will be described later. Hough voting equations, with time warped voting, correspond to Equations (3.1), (3.2), and (3.7), and are represented by the following equation:

$$p(c, \mathbf{x}, s \mid \mathbf{f}_\mathbf{y}, \mathbf{y}, \boldsymbol{\theta}_\mathbf{y}) = \sum_i p(\mathbf{x}, s \mid c, C_i, \mathbf{y}, \boldsymbol{\theta}_\mathbf{y}) p(c \mid C_i) p(C_i \mid \mathbf{f}_\mathbf{y}), \tag{6.1}$$

$$p(\mathbf{x}, s \mid c, C_i, \mathbf{y}, \boldsymbol{\theta}_\mathbf{y}) = \frac{1}{|\mathbf{A}_i^c|} \sum_{[\mathbf{d}, s'] \in \mathbf{A}_i^c} G([\mathbf{x}, s], [(\mathbf{y} + \mathbf{d}) + \boldsymbol{\theta}_\mathbf{y}, s']), \tag{6.2}$$

$$v\,(c, \mathbf{x}, s) \mid \mathbf{F}, \boldsymbol{\Theta}) = \sum_{\substack{\mathbf{f}_\mathbf{y} \in \mathbf{F} \\ \boldsymbol{\theta}_\mathbf{y} \in \boldsymbol{\Theta}}} p\left(c, \mathbf{x}, s \mid \mathbf{f}_\mathbf{y}, \mathbf{y}, \boldsymbol{\theta}_\mathbf{y}\right), \tag{6.3}$$

The calculated offset of feature $\mathbf{f}_\mathbf{y}$ is $\boldsymbol{\theta}_\mathbf{y} = [\mathbf{0}, \theta_\mathbf{y}^\mathrm{t}] \in \mathbb{R}^3$, and $\boldsymbol{\Theta}$ is the set of the offsets. The spatial dimensions of the offsets are zero vectors because the offsets only affect the temporal dimension.

To calculate optimal offsets, we need the correct action positions. Obviously, the correct positions are unknown, so we need a measure for the correct positions. Some votes are scattered by temporal variations but are concentrated in the correct positions. Therefore, we assume that the voting scores obtained by the conventional method at the correct positions are forming local maxima. The proposed method adopts the voting scores as the measures and calculates the offsets from the given voting positions to the positions of the local maxima of the voting scores.

The proposed method first calculates the offset of each vote. Figure 6.2 shows the offset calculation. The proposed method finds the closest local maximum position in the temporal dimension around the original voting position using mean shift. As described above, we assume that the correct action center positions form local maxima. Let $v^\mathrm{t} \in \mathbb{R}^1$ and $m^\mathrm{t} \in \mathbb{R}^1$ be temporal positions of a vote and a local maximum, respectively. Offset θ is calculated as follows: $\theta = m^\mathrm{t} - v^\mathrm{t}$. As shown in Equation (3.1), the Hough-based method casts multiple votes based on one local feature. Multiple offsets are also calculated based on one local feature.

The proposed method selects one offset for each local feature. We assume, similar to [Woodford et al., 2014], that only one vote generated by each local feature is correct. We consider that using all the offsets of each vote to the closest local maximum position would lead to false concentrations. Therefore, we propose to select the most reliable offset. Let $\mathbf{v}_{\mathbf{y},i} \in \mathbb{R}^3$ and $\boldsymbol{\theta}_{\mathbf{y},i} \in \mathbb{R}^3$ be the voting position and the offset of the i-th vote of feature $\mathbf{f}_\mathbf{y}$, respectively. The proposed method selects the best offset (i.e., the highest score) based on the voting score at the local maximum

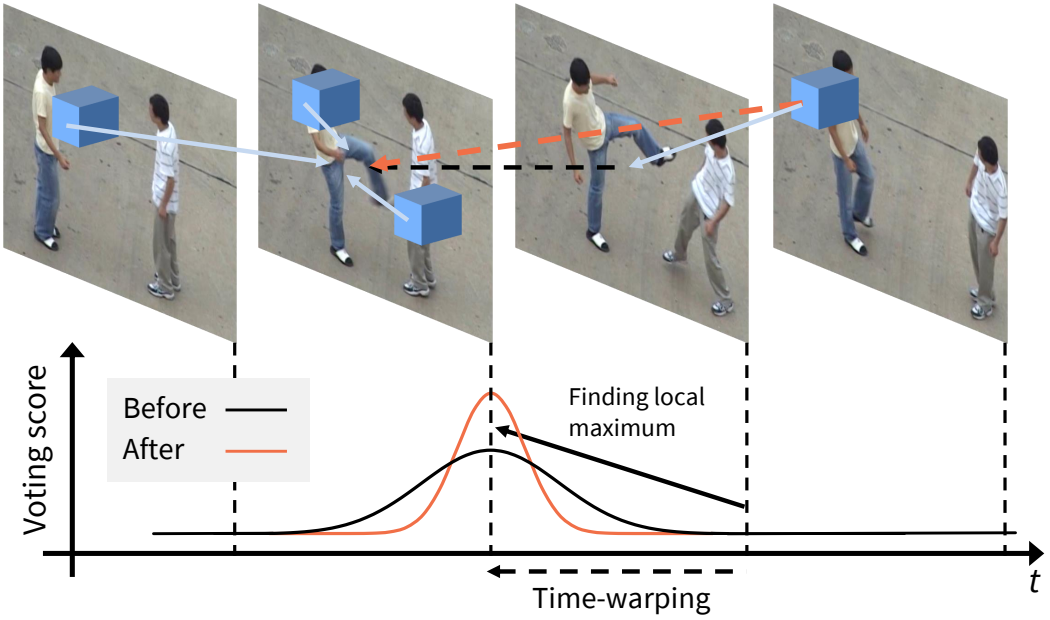


Figure 6.2: Time-warping of a vote. A temporal offset for warping is calculated by finding the closest local maximum. The warping gains the voting score at the maximum position.

position by the following equation:

$$i^* = \arg\max_i v\left(c_{\mathbf{y},i}, \mathbf{m}_{\mathbf{y},i}, s_{\mathbf{y},i} \mid \mathbf{F}\right), \tag{6.4}$$

where $c_{\mathbf{y},i}$ is a class label of the i-th vote of feature $\mathbf{f}_{\mathbf{y}}$, $\mathbf{m}_{\mathbf{y},i} = \mathbf{v}_{\mathbf{y},i} + \boldsymbol{\theta}_{\mathbf{y},i}$ is a local maximum position, and $s_{\mathbf{y},i}$ is a local maximum scale. The proposed method uses $\boldsymbol{\theta}_{\mathbf{y},i^*}$ as $\boldsymbol{\theta}_{\mathbf{y}}$. We expect that the selected offsets gain voting scores only at the correct positions. We handle time warping unreliable votes using the same selected offset. This rough warping may scatter unreliable votes and lower the voting scores at false local maxima.

The proposed method finds local maxima of Equation (6.3) using the calculated offsets. The local maxima are candidate actions. The method uses the local maxima that have voting scores greater than the threshold as detected actions.

6.3 Experiments

We evaluated our proposed method using the UT-Interaction dataset [Ryoo and Aggarwal, 2010], which is described in Section 5.4.1. In these experiments, we compared the proposed method with two baseline methods. The first method is the conventional Hough-based method described in Section 3.2. The second method also uses time-warped voting. However, where the proposed method selects reliable offsets using Equation (6.4), the second baseline uses naïve offsets to the found local maximum from each vote without selection.

Hough-based methods must find local maxima after calculating the voting scores. In these experiments, we used mean shift [Comaniciu and Meer, 2002] to do so.

6.3.1 Evaluation Method

Both our proposed and the baseline method detected action center positions. We separately calculated the spatial and temporal Euclidean distances between the detected and ground truth action positions. If both the calculated spatial and temporal distances were lower than the given thresholds, we defined the detection as a success. The spatial and temporal thresholds were set to 50 pixels and 30 frames, respectively. We calculated the precision, recall, and average precision for each class while changing the thresholds for the local maxima in Equations (3.7) and (6.3).

We employed 10-fold cross-validation that uses the data of two sequences as the test data, and the remainder as the training data. We split the data randomly for the cross-validation and changed the split three times. Results shown in the next section are averages of them.

The dataset contains motions that are not labeled. We assigned these motions the *Others* class during training. During detection, we did not cast votes for any class if the local features were classified as *Others*.

6.3.2 Results

Figure 6.3 shows the precision-recall curves of the proposed and the baseline methods, and Table 6.1 lists the average precision of the proposed and baseline methods, where *Selected Offset*, *Naïve Offset*, and *Hough* refer to our proposed method, the second baseline method, and the first baseline method, respectively. *Mean AP* is the mean average precision over all six classes. Compared with both baseline methods,

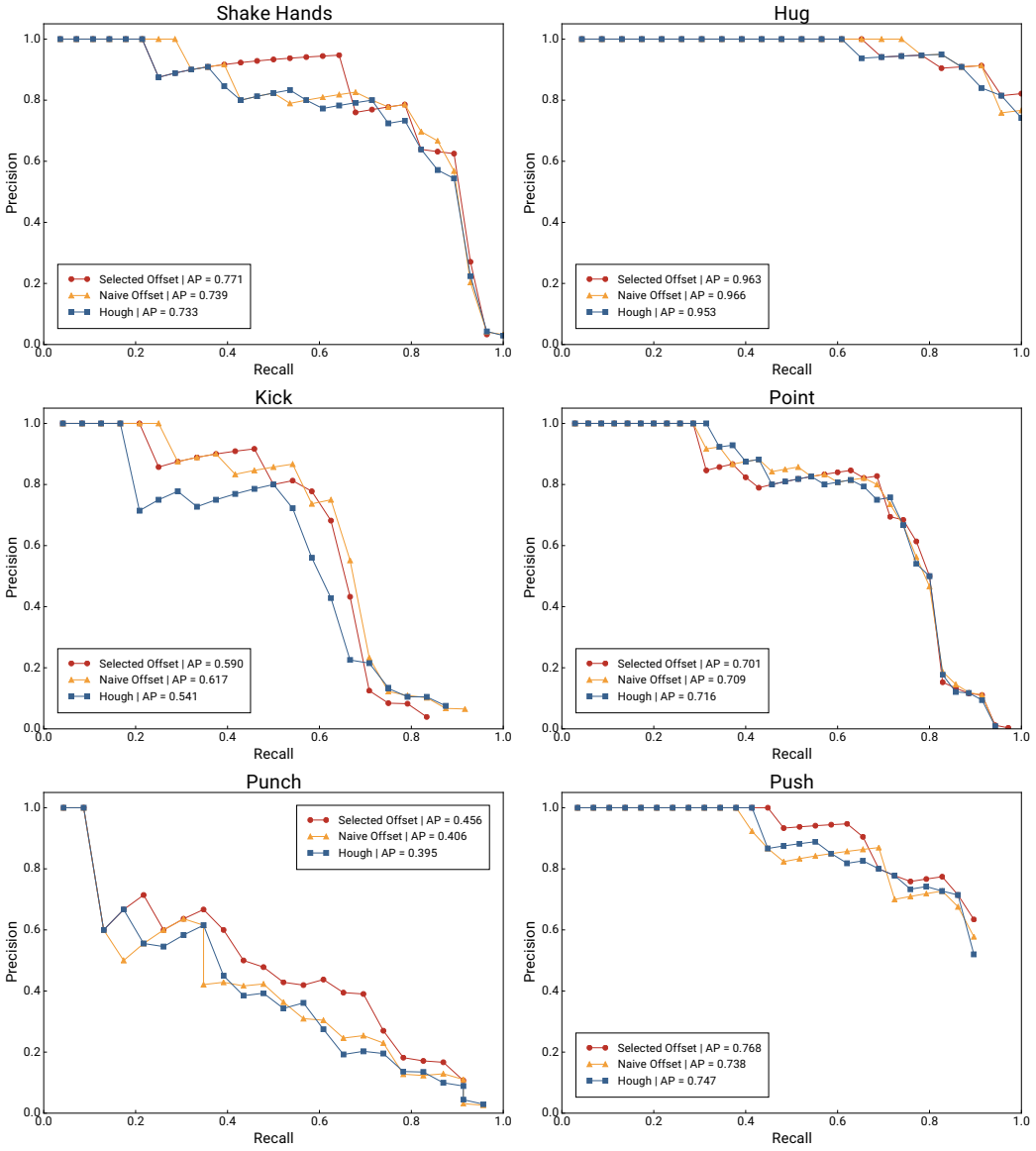


Figure 6.3: Precision-recall curves for one of three cross-validations on the UT-Interaction dataset.

Table 6.1: Average precision on the UT-Interaction dataset.

Method	Shake Hands	Hug	Kick	Point	Punch	Push	Mean AP
Selected Offset	**0.778**	**0.89**	0.612	0.711	**0.462**	**0.784**	**0.706**
Naïve Offset	0.76	0.884	**0.62**	**0.721**	0.414	0.758	0.693
Hough	0.695	0.871	0.557	0.719	0.411	0.74	0.666

Table 6.2: Gains of averaged voting scores at local maxima.

	Base Score (Hough)	Score Gain (Selected Offset)	Score Gain (Naïve Offset)	Number of Votes
True Positive	1.49	**0.242**	**0.869**	**4533.1**
False Positive	1.01	0.0376	0.540	2752.5

the proposed method achieved the highest mean average precision. These results indicate that time-warping by selected temporal offset contributed to improvements in action detection performance. The *Naïve Offset* method was inferior to the *Selected Offset* method. Because the *Naïve Offset* method warps every vote to the closest local maximum, it may warp unreliable votes to false local maxima. The vote warping by naïve offsets may concentrate false votes and reduce these improvements. Therefore, the selection of only the reliable offset for each local feature is important.

To analyze the reasons behind this improvement, we confirmed the gains in voting score at the local maxima using the *Selected Offset* and *Naïve Offset* methods, as shown in Table 6.2. The concentration of votes by time-warping increases the voting scores at local maxima. The values in the table are the averages at the local maxima that have higher scores than the best threshold for the f-scores of the precision-recall curves. *Base Score* is the average score of the *Hough* method, and *Score Gain* means the difference from the *Base Score*. The gains at the local maxima of true positives are higher than those at the local maxima of false positives. The higher gains explain these improvements.

We also confirmed that the correlation coefficients between the numbers of votes and the gains by the *Selected Offset* and *Naïve Offset* methods are 0.520 and 0.797, respectively. The numbers of votes around the true positives were higher than those around the false positives, as shown in Table 6.2. These results indicate that many votes were cast for correct action positions even if temporal variations of the actions exist. Time-warping can concentrate many votes around the correct positions and improve the voting scores.

Table 6.2 also indicates the significant improvements in the *Selected Offset* method. Consider the ratio of the gains at the local maxima of the true positives to those at the local maxima of the false positives. The ratio by the *Selected Offset* and *Naïve Offset* methods are 6.44 and 1.61, respectively, meaning that the *Selected Offset* method only improves scores at the local maxima of the true positives. The high ratio explains the significant improvements in the *Selected Offset* method.

The improvement by the *Selected Offset* method for the *Push* class was the highest, whereas the improvement for the *Point* class was the lowest. The difference between the two classes is the number of local features extracted from the actions of the classes. The actions of the *Point* class include only small arm motions. Because few local features are extracted from such small motions, a summation of the scores based on the local features tends to be low. The *Selected Offset* method selects reliable offsets that have the highest voting scores. Therefore, the *Selected Offset* method infrequently selects the offsets for classes that lead to low scores, such as the *Point* class. This infrequency results in the lowest rate of improvement.

We assumed that voting scores at correct action center positions form local maxima. Maximum recall for most classes using the conventional method is nearly one, as shown in Figure 6.3. If the positions of some actions did not form local maxima, the maximum recall was smaller. Therefore, this result indicates that our assumption is valid.

The proposed method improved precision when the recall was of moderate value; its improvement when recall was high was small. The proposed method calculates offsets based on conventional voting scores. The proposed method is not effective if the voting scores are low (i.e., the votes are too scattered). When recall is high, the evaluation focuses on local maxima that have low voting scores, resulting in little improvement.

6.4 Summary

In this chapter, we propose a novel Hough-based action detection method that is robust to the temporal variations of actions. We introduce time-warping by temporal offsets to concentrate scattered votes in the correct temporal action positions. The proposed method calculates offsets based on the votes generated by the conventional Hough-based method and selects an appropriate one for warping. We experimentally confirmed that the selection of the offsets contributed to the improvement in average precision and achieved higher results for the UT-Interaction dataset compared with

the conventional method.

Chapter 7

Looking into the Number of Local Features

7.1 Introduction

Variations, such as occlusions, human orientation variety, temporal variations, and action manners variety, change not only the feature descriptors of actions but also the number of available local features. Because the Hough-based method accumulates votes based on each local feature, voting scores depend on the number of local features. The conventional Hough-based method uses a fixed threshold for voting scores. Therefore, these variations mean that setting an appropriate threshold is difficult. To detect actions with few local features, the threshold should be low. However, a low threshold generates false positives for actions with many local features.

In this chapter, we propose a method that is robust to variations in the number of local features. Our proposed method improves two parts of the conventional Hough-based method: local feature extraction and detection thresholding. First, the proposed method reduces the dependency of the number of local features based on the spatial scales. To accurately detect actions with varying spatial scales, Hough-based methods cast votes for multiple scales by scaling the input video. The number of local features extracted from each scale varies. The proposed method adjusts the number of local features for each scale by random sampling or by changing the sampling stride. Second, the proposed method solves the problem of setting an appropriate threshold for voting scores. The number of local features depends on many factors in addition to spatial scale. The proposed method determines the

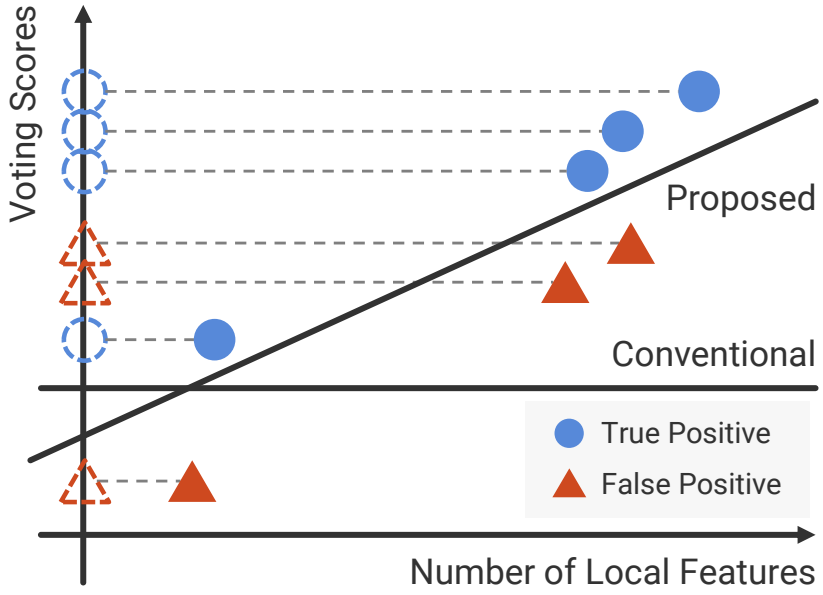


Figure 7.1: Thresholding for voting scores.

threshold based on the number of local features. Figure 7.1 shows the difference in thresholding between the conventional and proposed methods. The conventional Hough-based method performs constant thresholding without considering variations in the number of local features. In contrast, the proposed method uses a linearly changing threshold (i.e., a high threshold for actions contributing many local features and a low threshold for those contributing few). Therefore, the proposed method can detect actions against a variety of numbers of computed local features while avoiding false positives from actions contributing many local features. The proposed method adapts it thresholding by learning the relation between the number of local features and voting scores.

7.2 Number of Local Features

In this section, we explain our proposed method. Because Hough-based action detection accumulates votes based on each local feature, as shown in Equation (3.7), voting scores depend on the number of local features. The proposed method solves the problem caused by this dependency by improving two parts of the conventional Hough-based method. Section 7.2.1 explains the improvements to feature extraction and Section 7.2.2 describes the method's adaptive thresholding technique.

7.2.1 Scale-Adaptive Adjustment for the Number of Local Features

Spatially scaling input videos is effective for detecting actions with varying spatial scales. The number of local features extracted from actions depends on their spatial scale. Large scales lead to many local features. Therefore, even if no action occurs at large scales, the voting scores at these scales tend to be large, causing false detection and lower the accuracy of the scales of detected actions. Note that the method described in this section is unnecessary if an extraction method for local visual features has perfect invariance for scale changes.

Our proposed method adjusts the number of local features extracted from each scale by random sampling or by changing a sampling stride. The number of local features is adjusted to the maximum scale. The adjustment method differs depending on the extraction method of local visual features. When using sparse local visual features, as with STIP, the proposed method increases the extracted local features. The proposed method iterates to duplicate a local feature selected randomly for each scale until the number of local features reaches that of the maximum scale. However, when using dense local visual features, such as DTs, the proposed method adjusts the sampling stride of the dense features. Let $W_{\max}$ be the sampling stride of maximum scale $s_{\max}$. The proposed method calculates the sampling stride of scale s as $W_{\max}(s/s_{\max})$. These adjustment methods reduce the dependency on the number of local features based on spatial scale.

7.2.2 Thresholding Based on the Number of Local Features

Our proposed method determines appropriate thresholds for voting scores based on the number of local features. In the detection step of the Hough-based method, we cannot know the actual number of local features extracted from each action. Instead, the proposed method calculates the number of local features that cast votes for a detected action. The number of local features associated with the class c action detected at position $\mathbf{x}$ of scale s can be defined as:

$$n = \left|\left\{\mathbf{f}_{\mathbf{y}} \mid p\left(c, \mathbf{x}, s \mid \mathbf{f}_{\mathbf{y}}, \mathbf{y}\right) > 0, \mathbf{f}_{\mathbf{y}} \in \mathbf{F}\right\}\right|. \tag{7.1}$$

To determine the threshold, the proposed method learns the relation between the number of local features and voting scores using a linear SVM. Feature vectors for the SVM have two dimensions: the number of local features and the voting

scores. Let $g = an + b$ be the decision boundary learned by the SVM. Here, n is the number of local features, g is the voting score, and a and b are the learned slope and intercept. The slope represents the relation between the number of local features and the voting scores. The proposed method thresholds using the straight line that possesses the learned slope, as shown in Fig 7.1. The proposed method calculates threshold τ_n based on the number of local features n using the following equation:

$$\tau_n = an + \tau_b, \tag{7.2}$$

where τ_b is a base threshold. When $a = 0$, the threshold is the same as that of the conventional method. The proposed method does not use learned intercept b for threshold calculation. Therefore, we can control the trade-off between precision and recall using the base threshold.

To generate training data for the linear SVM, the proposed method generates codebook forests through the conventional training steps described in Section 3.2.2. The proposed method then casts votes and finds the local maxima in the voting spaces through the conventional detection steps described in Section 3.2.3. Each local maximum has a voting score. The number of local features associated with each local maximum is calculated using Equation (7.1). The proposed method uses the local maxima as the training data containing the voting scores and the number of local features. Here, the local maxima of true positives and false positives are positive and negative samples, respectively. The training data for the random forests and SVM should be split, because the same training data lead to overfitting.

7.3 Experiments

We used the UT-Interaction dataset [Ryoo and Aggarwal, 2010] in these experiments. The description about the UT-Interaction dataset is in Section 5.4.1.

7.3.1 Experimental Setup

We employed STIPs and DTs as sparse and dense extraction methods for local visual features, respectively. The STIPs used HOG and HOF descriptors. The DTs used trajectory, HOG, HOF, and MBH descriptors.

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.

To evaluate the methods, we calculated f-scores while changing the voting score thresholds. The following results show the maximum f-score. We employed 10-fold cross-validation that uses the data of two sequences as test data and the remainder as training data.

7.3.2 Results

Table 7.1 shows the f-scores of the conventional and proposed methods. *Conventional* is the conventional Hough-based method described in 3.2. *Adjust* and *Thresh* are the proposed methods using only the improvements described in Section 7.2.1 and 7.2.2, respectively. *Adjust-Thresh* is the proposed method using both new parts. The *Adjust-Thresh* method performs the proposed adjustment before voting, then casts votes, finds local maxima, and performs the proposed thresholding after the voting process. *Avg* is the f-score averaged over all six classes. When STIPs are used, the average f-score of the *Adjust* and *Thresh* methods improves compared with that of the conventional method. This result indicates that both parts of the proposed method are effective for action detection. The *Adjust-Thresh* method improves the f-score for all classes compared with the conventional method and achieves a higher f-score for most classes compared with *Adjust* and *Thresh*. The results when using DTs are similar to those derived from using STIP. The proposed method works well for both sparse and dense local visual features.

Figure 7.2 shows an example of a trained linear SVM for the Shake Hands class when using STIPs. The circles and triangles in this figure are true and false positives, respectively, of the Hough-based method. Note that they are samples for training the linear SVM. As the number of local features increased, the voting scores increased. The straight line indicates the decision boundary of the linear SVM, which is between the true and false positives. The slope of the line represents the relation between the number of local features and voting scores. If the conventional method uses threshold $\tau_b = 2$, it detects many false positives with many local features. In contrast, the proposed method can reduce false positives using adaptive thresholding per this decision boundary.

The proposed method may degrade robustness to occlusions in the Hough-based method. If actions are occluded, the number of local features varies, and this variation may generate improper threshold calculations. Here, we evaluated robustness to occlusions using the UT-Interaction dataset with artificial occlusions. Figure 7.3 shows examples of the occlusions. We divided the width of the bounding

Table 7.1: F-scores of the conventional and proposed method.

Method	Shake Hands	Hug	Kick	Point	Punch	Push	Avg
STIP							
Conventional	0.786	0.870	0.408	**0.618**	0.290	0.667	0.606
Adjust	0.764	0.894	0.549	**0.618**	0.238	0.723	0.631
Thresh	**0.868**	0.870	0.455	0.607	**0.303**	0.717	0.637
Adjust-Thresh	0.836	**0.917**	**0.558**	**0.618**	0.255	**0.793**	**0.663**
Dense Trajectories							
Conventional	0.690	0.889	0.667	0.576	0.282	0.767	0.645
Adjust	0.769	0.909	0.791	0.491	0.385	0.786	0.688
Thresh	0.786	0.894	0.667	**0.625**	0.339	0.815	0.687
Adjust-Thresh	**0.840**	**0.930**	**0.810**	0.597	**0.448**	**0.857**	**0.747**

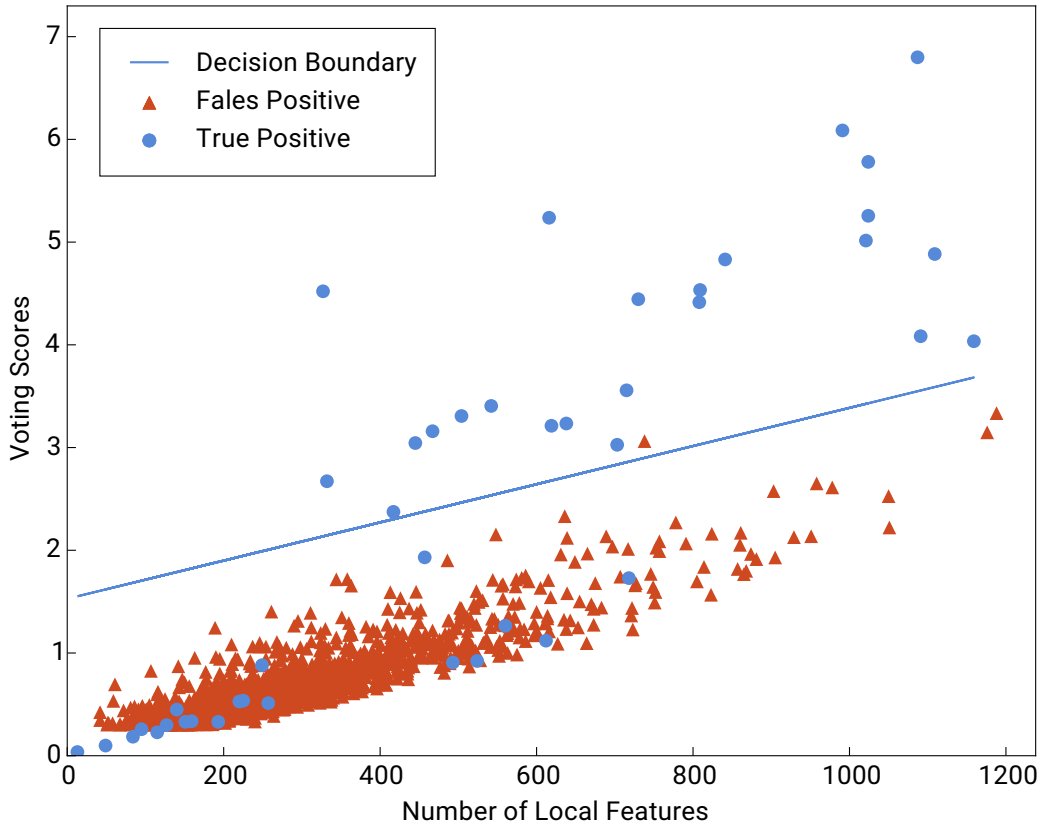


Figure 7.2: Example of a trained linear SVM for the *Shake Hands* class using STIPs.

Figure 7.3: Examples from the UT-Interaction dataset with artificial occlusions represented by black regions.

box of each action equally into five regions. We then randomly chose regions to occlude. We used all original local features in the training step, but in the detection step, we removed the local features in the occluded regions. All other settings were the same as those in the previous experiment.

Figure 7.4 shows the f-score when using the dataset with artificial occlusions. The f-score of each method with artificial occlusions did not decrease significantly compared to that without occlusions. The proposed method remains robust to occlusions and improves detection accuracy. This result indicates that occlusions does not change the relationship between the number of local features and voting scores, in despite of changing the number of local features.

7.4 Summary

We proposed a novel Hough-based action detection method that is robust to variations in the number of local features. Our proposed method improves two parts of the conventional Hough-based method, the extraction of local features and threshold detection. The proposed method reduces the dependency of the number of local features based on spatial scale. It adjusts the number of local features for each spatial scale using random sampling or changing sampling strides. In addition, the proposed method determines appropriate thresholds for voting scores based on the number of local features by learning the relationship between the number of local features and voting scores. We experimentally confirmed that both parts of our proposed method improve f-scores when extracting both sparse and dense local visual features.

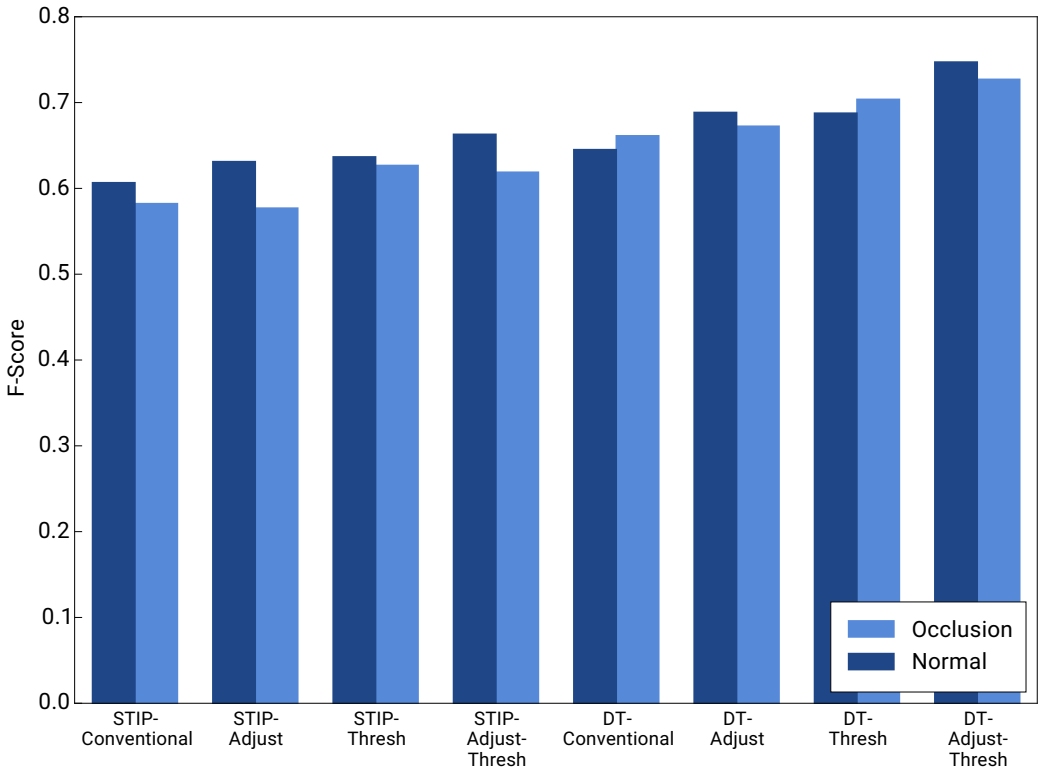


Figure 7.4: F-score averaged over all classes on the UT-Interaction dataset with artificial occlusions.

Chapter 8

Conclusion

8.1 Summary

In this thesis, we improved robustness of the Hough-based action detection in various environments. Chapter 1 presented the difficulty of action detection and the contributions of this study. Various factors such as occlusions, human orientation variety, motion similarity, temporal variations, and action manners variety make accurate action detection difficult. We focused on the Hough-based approach for action detection. An advantage of the Hough-based approach is the robustness to occlusions. In addition, we proposed the methods that give the robustness to other four factors: *human orientation variety, motion similarity, temporal variations*, and *action manners variety*.

In Chapter 2, we discussed the related work of this study. Our Hough-based approach uses the local feature representation. The local feature representation gives the robustness to occlusions whereas the discrimination ability is inferior to the global feature representation. Compared with the other detection approaches (i.e. the sliding window and action proposals approach), an advantage of the Hough-based approach is computational efficiency.

Chapter 3 presented the basic algorithm of the Hough-based approach. The approach casts votes for action classes, positions and scales based on the local features. The approach finds local maxima in the voting space and detects the local maxima that have voting scores over a threshold as actions.

In Chapter 4, we discussed the problem of the variety of human orientations relative to cameras. Appearances of actions change depending on relative human orientation against cameras. Using multiview video is one solution for this problem.

To use multiview videos in Hough-based action detection, we proposed the integration of Hough votes by homographic transformations. We assume that human feet touch the ground plane when they start an action. This assumption enables vote integration in the ground plane. In experiments, we confirmed that our proposed methods used multiview information effectively for vote integration and achieved *robustness to human orientation variety*.

Chapter 5 focused on the problem of false votes caused by motion similarity. If similar local motions exist between different action classes, the Hough-based method is prone to cast false votes for wrong action classes because discriminating such motions is difficult. Our proposed method employed vote distributions, which are distributions of the voting scores for each action class. The proposed method learns the characteristics of Hough voting based on vote distributions to reduce the effect of false votes. Experiments showed that the proposed method reduces the number of false positive detections in similar action classes. These reductions make the method *robust to motion similarity*.

In Chapter 6, we discussed in the problem of temporal variations. Temporal variations scatter Hough votes in the temporal dimension. We introduced the time-warping of votes to concentrate scattered votes in the temporally correct action positions. The proposed method calculates the offsets between the scattered voting positions and the concentrated positions by finding temporal local maxima and selects an appropriate one for the warping. We experimentally confirmed that the proposed time-warping concentrates scattered votes and contributes to *robustness to temporal variations*.

Chapter 7 focused on variations in the number of local features that are caused by factors, such as occlusions, human orientation variety, temporal variations, and action manners variety. The proposed scale-adaptive adjustment for the number of local features and thresholding based on the number of local features contribute to solving this problem. In experiments, we confirmed that our proposed method reduces the *influence of the number of local features* and improves action detection performance. Reducing the influence improves robustness to *human orientation variety, temporal variations*, and *action manners variety* in different aspects from the three methods above.

We focused on the robustness in this study. The robustness is important especially for the real world situations. Combining our four proposed methods contribute the robustness to five factors: occlusions, human orientation variety, motion similarity, temporal variations, and action manners variety. We expect that the robustness advances action detection to the real world situations.

8.2 Future Direction

This study focused on the detection of *actions*, which are the mid-level elements in the action hierarchy. However, some problems exist in applying action detection to real-world situations. In addition, *activities*, which are the top-level elements in the action hierarchy, should be also considered to understand humans in a scene. In the following, we discuss these future directions.

8.2.1 Online Action Detection

In this study, we focused on *offline* action detection. *Online* action detection is also important for some applications, such as robots that support and communicate with humans. In *offline* action detection, whole video sequences are available. In contrast, *online* action detection uses video streams as input and detects actions using only the past and current frames at each frame. Online detection is important because computers must react to human actions not when they are finished, but at the right time.

In online action detection, the following are important:

1) Real-time processing: to process continuous video streams, real-time processing is required. In this study, we did not consider processing speed. Hough-based methods have the advantage of computational efficiency. The proposed methods, however, cannot achieve real-time processing. One of the time-consuming parts of this method is the feature extraction. Motion features are important, but calculating optical flow, which is a typical motion feature, is slow. Fast methods for motion features make an impact on real-time action detection. Recent work proposed motion representations for action recognition using end-to-end learning by CNNs [Tran et al., 2015, Diba et al., 2016]. These representations achieve over 200 fps. Combining these representations with our detection framework would achieve real-time action detection.

2) Early detection: computers must detect actions using a part of an action sequence for online action detection. In this study, we confirmed that our proposed methods are robust to occlusions. Early detection situations are similar to occlusions; early detection observations are lacking in the temporal dimension, whereas those of occlusions are lacking in the spatial dimensions. Therefore, the proposed methods would work well for early detection. Some methods, such as recurrent neural networks, use time series models to represent temporal dynamics [De Geest et al., 2016, Li et al., 2016]. These methods are also effective for early detection. In

addition, considering *activities*, as well as *actions*, could be useful in online action detection. *Activities* represent interactions of multiple people when multiple people exist in a scene. *Actions* in interactions are related to each other. Past actions would contribute the early detection. The work building the model that manages the action hierarchy such as [Lan et al., 2012a, Amer et al., 2014] would be effective for this task.

8.2.2 Activity Detection

Detection of only *actions* is insufficient for some applications, such as surveillance and robots. For example, using individual actions cannot judge suspicious *activities* from surveillance camera data and robots should understand the intentions behind *activities* to communicate with and support humans. Therefore, activity detection is necessary in addition to action detection.

Some approaches that use graphical models [Lan et al., 2012b] or context-free grammars [Ryoo and Aggarwal, 2011] have been proposed to represent high-level *activities*. These approaches capture the relationships between *actions* in a scene. These approaches have achieved good results in activity recognition but can work for action detection in only limited situations. Activity detection in complex scenes is still an open problem. A difficult problem is segmentation of each *activities*. Consider the *activity* that includes multiple *actions* occurs in a complicated scene, as well as the surrounding *actions* that do not relate to the activity. It is difficult to judge whether or not the *actions* relate to the activity because the intra-class variations of *activities* is very large. Hough-based approaches can group local elements by transforming them to the parameter space. Managing actions as local elements and applying a Hough-based approach would be a possible solution.

Bibliography

[Aggarwal and Ryoo, 2011] Aggarwal, J. and Ryoo, M. (2011). Human activity analysis: A review. *ACM Computing Surveys*, 43(3):16:1–16:43.

[Amer et al., 2014] Amer, M. R., Lei, P., and Todorovic, S. (2014). HiRF: Hierarchical random field for collective activity recognition in videos. In *Proceedings of the European Conference on Computer Vision*, pages 572–585.

[Ballard, 1981] Ballard, D. H. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122.

[Bobick and Wilson, 1997] Bobick, A. and Wilson, A. (1997). A state-based approach to the representation and recognition of gesture. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 19(12):1325–1337.

[Bobick, 1997] Bobick, A. F. (1997). Movement, activity, and action: The role of knowledge in the perception of motion. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 352(1358):1257–1265.

[Cao et al., 2010] Cao, L., Liu, Z., and Huang, T. S. (2010). Cross-dataset action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1998–2005.

[Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence*, 24(5):603–619.

[Csurka et al., 2004] Csurka, G., Dance, C. R., Fan, L., Willamowski, J., and Bray, C. (2004). Visual categorization with bags of keypoints. In *Proceedings of the ECCV Workshop on Statistical Learning in Computer Vision*, pages 59–74.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893.

[De Geest et al., 2016] De Geest, R., Gavves, E., Ghodrati, A., Li, Z., Snoek, C., and Tuytelaars, T. (2016). Online action detection. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Proceedings of the European Conference on Computer Vision*, pages 269–284.

[de Souza et al., 2016] de Souza, C. R., Gaidon, A., Vig, E., and López, A. M. (2016). Sympathy for the details: Dense trajectories and hybrid classification architectures for action recognition. In *Proceedings of the European Conference on Computer Vision*, pages 697–716.

[Denina et al., 2011] Denina, G., Bhanu, B., Nguyen, H. T., Ding, C., Kamal, A., Ravishankar, C., Roy-Chowdhury, A., Ivers, A., and Varda, B. (2011). *VideoWeb Dataset for Multi-camera Activities and Non-verbal Communication*, pages 335–347. Springer London, London.

[Derpanis et al., 2010] Derpanis, K. G., Sizintsev, M., Cannons, K., and Wildes, R. P. (2010). Efficient action spotting based on a spacetime oriented structure representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1990–1997.

[Diba et al., 2016] Diba, A., Pazandeh, A., and Gool, L. V. (2016). Efficient two-stream motion and appearance 3d cnns for video classification. In *Proceedings of the ECCV Workshop on Brave New Ideas for Motion Representations in Videos*, pages 1–4.

[Dollar et al., 2005] Dollar, P., Rabaud, V., Cottrell, G., and Belongie, S. (2005). Behavior recognition via sparse spatio-temporal features. In *Proceedings of the IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 650–72.

[Duda and Hart, 1972] Duda, R. O. and Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

[Everingham et al., 2008] Everingham, M., Needham, C. J., and Fraile, R. (2008). A spatio-temporal descriptor based on 3d-gradients. In *Proceedings of the British Machine Vision Conference*, pages 275.1–275.10.

[Fanelli et al., 2011] Fanelli, G., Gall, J., and Van Gool, L. (2011). Real time head pose estimation with random regression forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 617–624.

[Fanelli et al., 2010] Fanelli, G., Yao, A., Noel, P.-L., Gall, J., and Van Gool, L. (2010). Hough forest-based facial expression recognition from video sequences. In *Proceedings of the ECCV Workshop on Sign, Gesture, and Activity*, pages 195–206.

[Feichtenhofer et al., 2016] Feichtenhofer, C., Pinz, A., and Zisserman, A. (2016). Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[Felzenszwalb et al., 2010] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

[Gaidon et al., 2013] Gaidon, A., Harchaoui, Z., and Schmid, C. (2013). Temporal localization of actions with actoms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2782–2795.

[Gall et al., 2011] Gall, J., Yao, A., Razavi, N., Gool, L. V., and Lempitsky, V. (2011). Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188–2202.

[Gemert et al., 2015] Gemert, J. C. v., Jain, M., Gati, E., and Snoek, C. G. M. (2015). Apt: Action localization proposals from dense trajectories. In *Proceedings of the British Machine Vision Conference*, pages 177.1–177.12.

[Girshick et al., 2016] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2016). Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158.

[Girshick et al., 2011] Girshick, R., Shotton, J., Kohli, P., Criminisi, A., and Fitzgibbon, A. (2011). Efficient regression of general-activity human poses from depth images. In *Proceedings of the International Conference on Computer Vision*, pages 415–422.

[Gkioxari and Malik, 2015] Gkioxari, G. and Malik, J. (2015). Finding action tubes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 759–768.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, pages 147–151.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the International Conference on Computer Vision*, pages 1026–1034.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[Hoai and Zisserman, 2015] Hoai, M. and Zisserman, A. (2015). Improving human action recognition using score distribution and ranking. In *Proceedings of the Asian Conference on Computer Vision*, pages 3–20.

[Hough, 1962] Hough, P. V. C. (1962). Method and means for recognizing complex patterns. US Patent 3,069,654.

[Jain et al., 2013] Jain, M., Jegou, H., and Bouthemy, P. (2013). Better exploiting motion for better action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2555–2562.

[Jain et al., 2014] Jain, M., van Gemert, J., Jégou, H., Bouthemy, P., and Snoek, C. G. M. (2014). Action localization by tubelets from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 740–747.

[Jégou et al., 2012] Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., and Schmid, C. (2012). Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 34(9):1704–1716.

[Ji et al., 2013] Ji, S., Xu, W., Yang, M., and Yu, K. (2013). 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231.

[Junejo et al., 2011] Junejo, I. N., Dexter, E., Laptev, I., and Perez, P. (2011). View-independent action recognition from temporal self-similarities. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 33(1):172–185.

[Karpathy et al., 2014] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

[Ke et al., 2007] Ke, Y., Sukthankar, R., and Hebert, M. (2007). Event detection in crowded videos. In *Proceedings of the International Conference on Computer Vision*, pages 1–8.

[Kimme et al., 1975] Kimme, C., Ballard, D., and Sklansky, J. (1975). Finding circles by an array of accumulators. *Communications of the ACM*, 18(2):120–122.

[Lan et al., 2012a] Lan, T., Sigal, L., and Mori, G. (2012a). Social roles in hierarchical models for human activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1354–1361.

[Lan et al., 2012b] Lan, T., Sigal, L., and Mori, G. (2012b). Social roles in hierarchical models for human activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1354–1361.

[Laptev, 2005] Laptev, I. (2005). On space-time interest points. *International Journal of Computer Vision*, 64(2):107–123.

[Laptev et al., 2008] Laptev, I., Marszałek, M., Schmid, C., and Rozenfeld, B. (2008). Learning realistic human actions from movies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[Larry Zitnick, 2014] Larry Zitnick, P. D. (2014). Edge boxes: Locating object proposals from edges. In *Proceedings of the European Conference on Computer Vision*, pages 391–405.

[Leibe et al., 2008] Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289.

[Li et al., 2016] Li, Y., Lan, C., Xing, J., Zeng, W., Yuan, C., and Liu, J. (2016). Online human action detection using joint classification-regression recurrent neural networks. In *Proceedings of the European Conference on Computer Vision*, pages 203–220.

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 1–17.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[Lv and Nevatia, 2007] Lv, F. and Nevatia, R. (2007). Single view human action recognition using key pose matching and viterbi path searching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[Maji and Malik, 2009] Maji, S. and Malik, J. (2009). Object detection using a max-margin Hough transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1038–1045.

[Mikolajczyk and Uemura, 2008] Mikolajczyk, K. and Uemura, H. (2008). Action recognition with motion-appearance vocabulary forest. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[Moeslund et al., 2006] Moeslund, T. B., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2–3):90 – 126. Special Issue on Modeling People: Vision-based understanding of a person's shape, appearance, movement and behaviour.

[Nagel, 1988] Nagel, H.-H. (1988). From image sequences towards conceptual descriptions. *Image and Vision Computing*, 6(2):59 – 74.

[Naiel et al., 2011] Naiel, M. A., Abdelwahab, M. M., and El-Saban, M. (2011). Multi-view human action recognition system employing 2dpca. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, pages 270–275.

[Oneata et al., 2014a] Oneata, D., Revaud, J., Verbeek, J., and Schmid, C. (2014a). Spatio-temporal object detection proposals. In *Proceedings of the European Conference on Computer Vision*, pages 737–752, Zurich, Switzerland. Springer.

[Oneata et al., 2014b] Oneata, D., Verbeek, J., and Schmid, C. (2014b). Efficient action localization with approximately normalized fisher vectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2545–2552.

[Razavi et al., 2012] Razavi, N., Alvar, N. S., Gall, J., and Gool, L. V. (2012). Sparsity potentials for detecting objects with the Hough transform. In *Proceedings of the British Machine Vision Conference*, pages 11.1–11.10.

[Riegler et al., 2014] Riegler, G., Ferstl, D., Rüther, M., and Bischof, H. (2014). Hough networks for head pose estimation and facial feature localization. In *Proceedings of the British Machine Vision Conference*, pages 1–12.

[Rosenfeld, 1969] Rosenfeld, A. (1969). *Picture Processing by Computer*. Acadeimc Press.

[Ryoo and Aggarwal, 2010] Ryoo, M. S. and Aggarwal, J. K. (2010). UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA). http://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html.

[Ryoo and Aggarwal, 2011] Ryoo, M. S. and Aggarwal, J. K. (2011). Stochastic representation and recognition of high-level group activities. *International Journal of Computer Vision*, 93(2):183–200.

[Sakoe and Chiba, 1978] Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49.

[Sánchez et al., 2013] Sánchez, J., Perronnin, F., Mensink, T., and Verbeek, J. (2013). Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245.

[Schüldt et al., 2004] Schüldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: A local svm approach. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03*, ICPR '04, pages 32–36, Washington, DC, USA. IEEE Computer Society.

[Scovanner et al., 2007] Scovanner, P., Ali, S., and Shah, M. (2007). A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the ACM International Conference on Multimedia*, pages 357–360.

[Shi et al., 2013] Shi, F., Petriu, E., and Laganiere, R. (2013). Sampling strategies for real-time action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2595–2602.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 568–576.

[Siva and Xiang, 2010] Siva, P. and Xiang, T. (2010). Action detection in crowd. In *Proceedings of the British Machine Vision Conference*, pages 9.1–9.11.

[Soomro and Zamir, 2014] Soomro, K. and Zamir, A. R. (2014). Action recognition in realistic sports videos. In *Computer Vision in Sports*, pages 181–208. Springer.

[Srikantha and Gall, 2014] Srikantha, A. and Gall, J. (2014). Hough-based object detection with grouped features. In *Proceedings of the International Conference on Image Processing*, pages 1653–1657.

[Sternig et al., 2011] Sternig, S., Mauthner, T., Irschara, A., Roth, P. M., and Bischof, H. (2011). Multi-camera multi-object tracking by robust Hough-based homography projections. In *IEEE International Conference on Computer Vision Workshops*, pages 1689–1696.

[Tejani et al., 2014] Tejani, A., Tang, D., Kouskouridas, R., and Kim, T.-K. (2014). Latent-class Hough forests for 3d object detection and pose estimation. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Proceedings of the European Conference on Computer Vision*, pages 462–477.

[Tian et al., 2013] Tian, Y., Sukthankar, R., and Shah, M. (2013). Spatiotemporal deformable part models for action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2649.

[Tran et al., 2015] Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the International Conference on Computer Vision*, pages 4489–4497.

[Tsuji and Matsumoto, 1978] Tsuji, S. and Matsumoto, F. (1978). Detection of ellipses by a modified Hough transformation. *IEEE Transactions on Computers*, C-27(8):777–781.

[Uijlings et al., 2013] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171.

[Vedaldi and Soatto, 2008] Vedaldi, A. and Soatto, S. (2008). Quick shift and kernel methods for mode seeking. In *Proceedings of the European Conference on Computer Vision*, pages 705–718.

[Vijay Kumar and Patras, 2012] Vijay Kumar, B. and Patras, I. (2012). Learning codebook weights for action detection. In *Proceedings of the CVPR Workshop on Large-Scale Video Search and Mining*, pages 27–32.

[Vijay Kumar and Patras, 2013] Vijay Kumar, B. and Patras, I. (2013). Supervised dictionary learning for action localization. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pages 1–8.

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518.

[Wang et al., 2013] Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. (2013). Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103(1):60–79.

[Wang et al., 2015a] Wang, H., Oneata, D., Verbeek, J., and Schmid, C. (2015a). A robust and efficient video representation for action recognition. *International Journal of Computer Vision*, 119(3):219–238.

[Wang et al., 2009] Wang, H., Ullah, M. M., Kläser, A., Laptev, I., and Schmid, C. (2009). Evaluation of local spatio-temporal features for action recognition. In *Proceedings of the British Machine Vision Conference*, pages 124.1–124.11.

[Wang et al., 2015b] Wang, L., Qiao, Y., and Tang, X. (2015b). Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4305–4314.

[Weinland et al., 2006a] Weinland, D., Ronfard, R., and Boyer, E. (2006a). Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2–3):249 – 257.

[Weinland et al., 2006b] Weinland, D., Ronfard, R., and Boyer, E. (2006b). Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2–3):249 – 257.

[Weinzaepfel et al., 2015] Weinzaepfel, P., Harchaoui, Z., and Schmid, C. (2015). Learning to track for spatio-temporal action localization. In *Proceedings of the International Conference on Computer Vision*, pages 3164–3172, Santiago, Chile.

[Wohlhart et al., 2012] Wohlhart, P., Schulter, S., Köstinger, M., Roth, P., and Bischof, H. (2012). Discriminative Hough forests for object detection. In *Proceedings of the British Machine Vision Conference*, pages 40.1–40.11.

[Woodford et al., 2014] Woodford, O. J., Pham, M., Maki, A., Perbet, F., and Stenger, B. (2014). Demisting the Hough transform for 3d shape recognition and registration. *International Journal of Computer Vision*, 106(3):332–341.

[Yan et al., 2008] Yan, P., Khan, S. M., , and Shah, M. (2008). Learning 4d action feature models for arbitrary view action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7.

[Yao et al., 2010] Yao, A., Gall, J., and Gool, L. V. (2010). A Hough transform-based voting framework for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2061–2068.

[Yarlagadda et al., 2010] Yarlagadda, P., Monroy, A., and Ommer, B. (2010). Voting by grouping dependent parts. In *Proceedings of the European Conference on Computer Vision*, pages 197–210.

[Yu and Yuan, 2015] Yu, G. and Yuan, J. (2015). Fast action proposals for human action detection and search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1302–1311.

[Yu et al., 2011a] Yu, G., Yuan, J., and Liu, Z. (2011a). Real-time human action search using random forest based Hough voting. In *Proceedings of the ACM International Conference on Multimedia*, pages 1149–1152.

[Yu et al., 2011b] Yu, G., Yuan, J., and Liu, Z. (2011b). Unsupervised random forest indexing for fast action search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognitio*, pages 865–872.

[Yu et al., 2012] Yu, G., Yuan, J., and Liu, Z. (2012). Propagative Hough voting for human activity recognition. In *Proceedings of the European Conference on Computer Vision*, pages 693–706.

[Yuan et al., 2011] Yuan, J., Liu, Z., and Wu, Y. (2011). Discriminative video pattern search for efficient action detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1728–1743.

[Zheng and Jiang, 2013] Zheng, J. and Jiang, Z. (2013). Learning view-invariant sparse representations for cross-view action recognition. In *Proceedings of the International Conference on Computer Vision*, pages 3176–3183.

[Zhou and Torre, 2012] Zhou, F. and Torre, F. D. l. (2012). Generalized time warping for multi-modal alignment of human motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1282–1289.

[Zhu et al., 2013] Zhu, F., Shao, L., and Lin, M. (2013). Multi-view action recognition using local similarity random forests and sensor fusion. *Pattern Recognition Letters*, 34(1):20 – 24.

List of Publications

Journal Papers

1. Kensho Hara, Takatsugu Hirayama, and Kenji Mase, "Vote Distribution Model for Hough-based Action Detection", IEICE Transactions on Information and Systems, Vol. E99-D, No. 11, pp. 2796–2808, 2016.

2. Kensho Hara, Takatsugu Hirayama, and Kenji Mase, "Simultaneous Recognition of Human Action and its Location Estimation Based on Multi-view Hough Voting", IEEJ Transactions on Electronics, Information and Systems, Vol. 134, No. 5, pp. 634–642, 2014 (in Japanese).

International Conference (reviewed)

1. Kensho Hara and Kenji Mase, "Hough-based Action Detection with Time-warped Voting", The 3rd Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, Malaysia, 2015.

2. Kensho Hara, Takatsugu Hirayama, and Kenji Mase, "Trend-Sensitive Hough Forests for Action Detection", International Conference on Image Processing (ICIP), Paris, France, 2014.

3. Kensho Hara, Takatsugu Hirayama, and Kenji Mase, "Simultaneous Action Recognition and Localization Based on Multi-View Hough Voting", The 2nd Asian Conference on Pattern Recognition (ACPR), Okinawa, Japan, 2013.

Presentation (in Japanese)

1. Kensho Hara and Kenji Mase, "Robust Action Detection Using Hough Forests under Scale Changes", IEICE Technical Committee on Pattern Recognition and Media Understanding (PRMU), 2016.

2. Kensho Hara and Kenji Mase, "Hough-based Action Detection with Time-warped Voting", The 18th Meeting on Image Recognition and Understanding (MIRU), 2015.

3. Kensho Hara, Takatsugu Hirayama, and Kenji Mase, "Hough-based Action Detection Using Vote Distribution Model", The 17th Meeting on Image Recognition and Understanding (MIRU), 2014.

4. Kensho Hara, Takatsugu Hirayama, and Kenji Mase, "Trend-Sensitive Hough Forests: Action Detection Method Using Voting Trends", IEICE Technical Committee on Pattern Recognition and Media Understanding (PRMU), 2014.

5. Kensho Hara, Takatsugu Hirayama, and Kenji Mase, "Simultaneous Action Recognition and Localization with Multi-view Augmented Hough Forests", The 16th Meeting on Image Recognition and Understanding (MIRU), 2013.