Chapter 2
Human Action Recognition: Contour-Based and Silhouette-Based Approaches

Salim Al-Ali, Mariofanna Milanova, Hussain Al-Rizzo and Victoria Lynn Fox

Abstract Human action recognition in videos is a desired field in computer vision applications since it can be applied in human computer interaction, surveillance monitors, robot vision, etc. Two approaches of features are investigated in this chapter. First approach is a contour-based type. Four features are investigated in this approach such as Cartesian Coordinate Features (CCF), Fourier Descriptors Features (FDF), Centroid-Distance Features (CDF), and Chord-Length Features (CLF). The second approach is a silhouette-based type. Three features are investigated in this approach such as Histogram of Oriented Gradients (HOG), Histogram of Oriented Optical Flow (HOOF), and Structural Similarity Index Measure (SSIM) features. All these features are simple to compute, efficient to classify, and fast to calculate. Therefore, these features demonstrate a promising field for human action recognition. Moreover, the classification is achieved using two classifiers: K-Nearest-Neighbor (KNN) and Support Vector Machine (SVM). The experimental results demonstrated that these features have a promising potential and useful for the human action recognition in videos.

S. Al-Ali (✉) · M. Milanova
Department of Computer Science, University of Arkansas at Little Rock,
2801 S. University Avenue, Little Rock, AR 72204, USA
e-mail: sgsaeed@ualr.edu

M. Milanova
e-mail: mgmilanova@ualr.edu

H. Al-Rizzo
Department of System Engineering, University of Arkansas at Little Rock,
2801 S. University Avenue, Little Rock, AR 72204, USA
e-mail: hmalrizzo@ualr.edu

V.L. Fox
Department of Applied Science, University of Arkansas at Little Rock,
2801 S. University Avenue, Little Rock, AR 72204, USA
e-mail: vlfox@ualr.edu

M.N. Favorskaya and L.C. Jain (eds.), *Computer Vision in Control Systems-2*,
Intelligent Systems Reference Library 75, DOI 10.1007/978-3-319-11430-9_2

Keywords Human action recognition · Contour-based features · Silhouette-based features · K-Nearest-Neigbor · Supprt Vector Machine · Image and video understanding · Machine learning · Data mining

2.1 Introduction

Currently, computer application fields are playing significant role in multiple aspects of our lives. One important field is a computer vision, which has received a lot of attention during the past three decades due its wide applications. The human action recognition is an important goal of research on computer vision and image processing. Identifying, annotating, recognizing, and clustering human actions in videos have captured more and more attention because of its useful applications that support many different applications such as human–computer interaction, robot vision machine, human surveillance monitoring system, multimedia indexing and retrieval, entertainment environments, and healthcare systems [1, 2, 3].

The human action recognition in videos is a computer method for recognizing and identifying, what kind of action is happening in videos. In order to design and implement this program, there are many challenges such as foreground object, background scene, and camera setting. The foreground object, which is the human in this case, has many variations such as size, colour, shape, static or moving object, etc. The background scene, which is a whole image in a frame except the foreground object, has many variations such as lighting, occlusion, cluttered, static or moving background scene (based on camera setting). The camera setting is an important factor in the human action recognition because it has its own recording variations such as static or moving in all (left, right, up, or down) directions, zooming (in or out), speeds of recording (slow or high), recording types (2D or 3D), colors types in recording videos (black/white, colored, or grayscale color), etc. Moreover, the same action is performed in different ways by the same person, for example, the speed of walking is different although that the walking action is for the same person. Another challenging problem, which is more difficult and very realistic, is that the same action performed by different people. Although of these challenges, the human action recognition in videos is desired and required for many computer vision applications.

In recognizing human actions in videos, many researches have been reported in this field as shown in the survey paper [4], however, there still need to improve and develop new effective approaches. The presented chapter is a new investigation of two main features approaches (contour-based and silhouette-based) for human action recognition in videos.

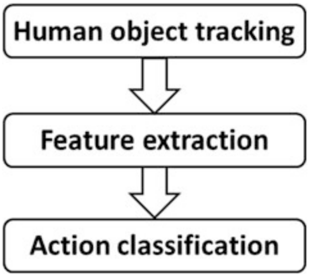
In this chapter, main structure of human action recognition is defined. The structure mainly consists of three stages: human object tracking, feature extraction, and action classification. Some examples regarding literature researches of these three stages and the recent related works are given in Sect. 2.2. Subsequently, background subtraction [5, 6] is explained in detail as an example for stage of human object tracking in videos in Sect. 2.3. Next, two approaches: contour-based and silhouette based for feature extraction from the tracked human object are described in Sects. 2.4 and 2.5, respectively. The final stage in the human action recognition is action classification stage that used to classify and identify the action happening in human action testing video. Two classifiers: KNN [7, 8, 9], and SVM [10, 11, 12, 13] are used as examples for action classification stage. More details about the classifiers and their based techniques are explained in Sect. 2.6. Two modes (training and testing) of the presented algorithm for human action recognition are described in Sect. 2.7. Experimental results are discussed in Sect. 2.8. Finally, Sect. 2.9 conducted the conclusion.

2.2 Human Action Recognition in Videos

The main goal of human action recognition in videos is to identify the unknown actions happening in these videos. This goal is achieved by analyzing the frames of these videos to form and build a series of discriminant features that can be classified efficiently in term of accuracy, speed, and simplicity. The main structure of the human action recognition consists of three main stages: human object tracking, feature extraction, and action classification. The first stage has to answer the question of how to detect or segment and track the human object in each frame of the video sequences. The second stage has to answer the question of how to extract, represent, and then build feature vector from the tracked human object that result from the first stage. The third stage has to answer the question of how to classify extracted features from the second stage by applying an effective classification algorithm. Sometimes, this stage supported by data mining process to reduce dimensionality of the extracted features. In the next sections, answers and details about these three stages will be provided. The main structure of a human action recognition system is depicted in Fig. 2.1.

Section 2.2.1 addresses the human object tracking. The issues of feature extraction and action classification are discussed in Sects. 2.2.2 and 2.2.3, respectively. A literature review about human action recognition in Weizmann dataset is represented Sect. 2.2.4.

Fig. 2.1 Main structure of the human action recognition



2.2.1 Human Object Tracking

The human object tracking is a process of tracking a human object moving over sequence (time) of digital images (frames) in videos [14]. Generally, this process consists of two components: frame processing (local) and video processing (global). The first is achieved by the human object detection or segmentation. The human detection is the process of locating a human object in a frame of video. The segmentation of human object is the process of partitioning a frame into multiple segments (areas or sets). One of these separated segments represents the human object. Both detection and segmentation are mainly related to one frame (digital image) in videos and, therefore, are called frame processing. The second component is achieved by applying frame processing over all-frames (video) or sub-frames (sub video), thus, it is called video processing.

During the past three decades, many researchers solved problem of tracking objects in still image, in a frame, or videos. These solutions are achieved by several ways: point detection, image segmentation, and background modeling. First, the point detection is used for tracking based on some interesting points such as corners, or intersection points such as Harris detector [15], Scale-Invariant Feature Transform (SIFT) [16], affine invariant interest point detector [17], kernel-based object tracking [18], and Kanade-Lucas-Tomasi (KLT) detector [19]. Second, the image segmentation is a process to partition a digital image (frame) into multiple segments (sets of separated areas), used to track an object such as mean-shift [20], graph-cut [21], and active-contours [22]. Third, the background modeling is also another process used in the tracking. The goal is to obtain and build a model for the background scene. Then, the object extraction is achieved by subtracting each frame from this model, such as running Gaussian average [23], temporal median filter [24, 25], Mixture Of Gaussian (MOG) [26, 27], eigenbackground [28], and dynamic texture background [5]. More details and examples for human object tracking using a background subtraction in Weizmann human action dataset [2] are explained in Sect. 2.3.

2.2.2 Feature Extraction

The feature extraction is a process of extracting a set of features to represent some useful measurements or characteristics of a frame or video. These features are computed carefully from a frame, sub-frames, or all-frames in video efficiently in order to capture most important meaningful details. The goal of feature extraction is to provide a classifier by good feature in terms of accuracy and speed. This goal can be achieved by two ways: minimizing feature details as much as possible and at the same time maximizing features discrimination in order to increase accuracy and speed of classification in the next stage. There are several ways to enhance the feature extraction [29]. First, extracting spatial information is more related to frame details (coordinates, shape, size, texture, etc.). Second, extracting temporal information is more related to video details (motion, time, derivatives, etc.). Third, eliminating redundancy in information of both frame and video in order to reduce valueless features since much of information is of little or no value. Finally, integrating more than one kind of features together to get better performance in analysis and classification.

Most features are extracted from a shape, which represents both contour and silhouette of the tracked object implicitly. Moreover, this shape is used in template matching and human action recognition by some researches [30, 31].

In the past three decades, feature extraction or detection attracted the attention of researchers due to its useful applications [32]. For example, a content-based video retrieval [32] is the ability to recognize actions correctly that leads to automatic annotation of huge video dataset. Different types of features that have been extracted from human object are investigated for human action recognition purposes. In this chapter, the features used to recognize a human action in videos are categorized mainly into two types: contour-based and silhouette-based. On the one hand, the contour-based features are mainly obtained from boundary points that surround silhouette of human object. While on the other hand, the silhouette-based features are mainly obtained from whole region body of silhouette for human object. More details and examples about contour-based feature and silhouette-based features extraction are explained in Sects. 2.4 and 2.5, respectively.

2.2.3 Action Classification

The final step in any recognition is to feed the features into a classifier, which is adopted one of the classification algorithms, for example, such as K-Nearest-Neighbor (KNN) [7, 8, 9], Support Vector Machine (SVM) [10, 11, 12, 13], Adaptive boosting (Adaboost) [33], Hidden Markov Model (HMM) [34, 35], Artificial Neural Network (ANN) [36, 37], etc. More information about some of these algorithms is explained in Sect. 2.6.

Generally, the goal of all these algorithms is to classify extracted feature in testing video sample (testing mode) and identify its class membership or its closest neighbor based on features that conducted from training video samples (training mode). Thus, a recognition classifier has to be trained using the training observations. There are three types of learnings based on the labeled and non-labeled classes of the observations such as supervised, unsupervised, and semi-supervised (self-supervised) learnings [38, 39]. The supervised machine learning [39] is defined such that all training examples (observations) are labeled into classes, thus the system (machine) is trained with feature observations and their class labels. Thus, the goal is to find a membership class (classify into one class of the trained classes) for any a given testing example. Naturally, the supervised learning is used in most classification algorithms. In unsupervised machine learning [38, 39], the given training observations are not labeled into classes, thus the system is trained with only feature observations. Therefore, the system has two goals: first one is to cluster and find the classes for training examples and second goal is to find a membership class after clustering all training examples for a given testing example. Thus, the unsupervised machine learning is more complicated than the supervised learning. Naturally, the unsupervised learning is used in clustering but not in classification. In the semi-supervised (self-supervised) [39, 40], some of a given training observations are labeled while others not. This learning is a combined between both previous types, and it is located in middle between supervised and unsupervised learning in term of difficulty.

Moreover, in order to evaluate computed results statistically cross-validation is used. It is a technique to assess results based on a statistical analysis. The cross-validation is the estimation for accuracy of a model or an algorithm based on how to use the dataset during training and testing modes. Mainly, there are three techniques: 2-fold, cross-validation, K-fold cross-validation, and leave-one-out cross-validation [41]. More details and examples about classifier types and techniques are explained in Sect. 2.6.

2.2.4 Human Action Recognition in Weizmann Dataset: Literature Review

There are many approaches and methods to recognize human actions in videos over the past three decades. In this section, the recent literature review related to human action recognition in videos is presented. Despite of these methods and approaches, the human action recognition is still attractive for many computer vision researchers because still has plenty of challenges need to be solved and it is growing demands for many applications. For example, more realistic human action dataset requires for researchs in accuracy and speed challenges for building an efficient real-time human action recognition in videos.

Aggarwal and Ryoo [4] classified human action recognition into two main approaches. First, single layered approach is based on sequence of images to describe human actions. Second, the hierarchical approach is applied on more than one of single layered approach. On the one hand, the single layered approach is classified based on model into space-time, and sequential approaches. On the other hand, the hierarchical approach is classified based on their methodology into statistical, syntactic, and description approaches. The space-time approaches are classified based on feature types into space-time volumes, trajectories, and space-time features. This approach is named due that its features are computed from space (spatial) and time (temporal). In this chapter, all experiments are based on this space-time approach because both (contour-based and silhouette-based) features are extracted from Aligned Silhouettes Image (ASI), which is an accumulation of all frames in one video to form one image that captures all spatial and temporal features together.

Amraji et al. [30] presented human action recognition system based on shape of human. The shape representation is computed using Fourier Descriptors (FDs) as features. These features are projected into eigen-space by using Principal Component Analysis (PCA). The KNN is employed as a classifier for human action recognition. The Weizmann dataset [2] is used to test the system with only five actions from ten actions in the dataset. They recorded 86 % success recognition rate. In this chapter, the FDs are extracted from the contour of the ASI and used as a feature without any projection. Then, the KNN and the SVM classifiers are used. The best achieved result is 93.548 %, which is the first contribution of this chapter.

Gorelick et al. [2] employ contours of human silhouettes as space-time features for action recognition. In their algorithm, the Poisson's equation is solved based on contour coordinate points. The solution is required to compute coefficients of the Poisson's equation by using multigrid solution. Then, several properties are extracted based on these coefficients such as local saliency, action dynamics, shape structure, and orientation. These shape properties or Poisson features are employed as a sequence for a number of frames in each video action. Gorelick et al. [2] created the Weizmann dataset, which contains ten different human actions done by nine actors, applied their recognition system on these dataset. The researchers recorded 100 % correct recognition rate using all sequence of frames in videos for classification using variant median Hausdroff distance. This distance is used because of the differences among videos in term of video length in frames. It is used to find the distances between any two sequences (testing sample and each of training samples) and the action with minimum distance is identified as predicted action. Also, they achieved 97.83 % correct rate using sliding window of eight frames with jumping four step frames. They used 923 space-time cubes in their experiments. This means that a number of all frames in all videos is approximately 4,096 frames totally. In this investigation, the Weizmann dataset was downloaded from their website [2] and computed number of frames in all videos is 5,687 frames exactly. This difference means that all available frames, which effect on the recognition results, not used in their experiment. In all our experiments in this chapter, all frames are used without any exception. Therefore, the results are more relastic since built based on all available frames in all videos.

Sadek et al. used a chord-length as a feature for human action recognition. This feature demonstrates high accurate, robust, compact, and efficient results [3]. The computation of feature is based on a chord-length function as a local feature, and gravity or shape centroid motion for snippet frames as a global feature. The SVM is employed for the classification. Sadek et al. [3] applied their algorithm in Weizmann dataset. They recorded result 97.8 % correct recognition rate on Weizmann dataset. They combined more than one feature together. In this chapter, each feature is tested alone separately before combination to compare results with each other.

Dalal and Triggs [42] presented a very robust visual object recognition feature, which is called Histogram of Oriented Gradients (HOG), used effectively to recognize objects in visual method. Also, Dalal and Triggs [43] used another histogram feature, which is Histogram of Oriented Optical Flow (HOOF). By employing these features, the human object is detected. For classification, in both HOG and HOOF, a linear SVM is used. Both HOG and HOOF are mainly used in object recognition. In this chapter, both descriptors are employed as the features for human action recognition and achieved very good results in term of accuracy.

Chaudhry et al. [1] employed the HOOF and the Binet-Cauchy kernels on nonlinear dynamical systems for recognition of human actions. Four different kernels are used to measure distance between two histograms. These kernels are: geodesic, Minimum Difference of Pairwise Assignment (MDPA), chi-square, and histogram kernels. The minimum distance is selected as a prediction result for classification. Authors reported 94.4 % correct recognition rate on Weizmann dataset. In this chapter, the HOOF is employed directly as a feature and achieved 97.849 %, which is the second contribution in this chapter.

Al-Ali and Milanova [44] employed an Aligned Motion Image (AMI) as a feature for human action recognition. Each video sample is represented by an AMI. Then, the Structural Similarity Index Measure (SSIM) is used to measure the distances among these images. In the last experiment of this chapter, the SSIM is employed but as a feature for representing each video. Authors reported 98.924 % correct recognition rate.

The contributions of this chapter are the following. A novel simple algorithm for human action recognition provides very good results in term of accuracy in contour-based features. Almost optimal results are conducted in term of accuracy using silhouette-based features. The structural similarity for human action recognition is employed. Finally, a comparison among contour-based and silhouette-based feature results is presented in this chapter.

2.3 Human Object Tracking in Weizmann Dataset

The goal of human object tracking in videos is to separate a human silhouette from each its background scene of each frame in the videos. In order to achieve this goal specifically in Weizmann dataset [2], several pre-processing processes are required. The description of Weizmann human action dataset is located in Sect. 2.3.1. The following Sects. 2.3.2–2.3.7 include the representations of various types of processes such as background subtraction, detection of direction, horizontal alignment, computing of an Aligned Silhouettes Image (ASI) process, unifying direction, and cropping bounding box, respectively.

2.3.1 Weizmann Human Action Dataset

The Weizmann human actions dataset [2] is used to test our presented algorithms in this chapter. This dataset is recorded from a still (non-moving) camera, therfore

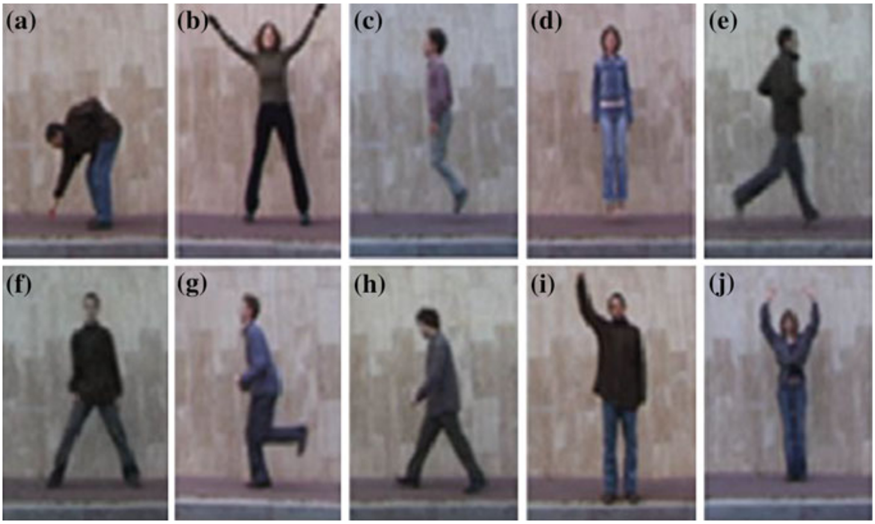


Fig. 2.2 Weizmann dataset frame example for ten different human actions: **a** bending, **b** jumping jack, **c** jumping forward, **d** jumping in place, **e** running, **f** gallop sideway, **g** skip jumping, **h** walking, **i** one hand waving, **j** two hands waving

background of scene is a still image. The only moving thing is the human object. The dataset contains 93 low-resolution (180 × 144, with speed rate 50 fps) video samples for human actions. Figure 2.2 depicts some frame examples of human action videos in the Weizmann dataset.

There are ten different human actions in this dataset such as bending, jumping jack, jumping forward, jumping in place, running, gallop sideway, skip jumping, walking, one hand waving, and two hands waving. These actions are performed by nine actors. Each actor performed each action once, except one actor ("Lena"), who performed three of the actions (running, skip jumping, and walking) twice. One action, the object is moving from left to right and other it is vice versa. The Weizmann dataset videos samples have different lengths in term number of frames recorded in each video.

2.3.2 Background Subtraction Process

Background subtraction is an example about human object tracking. It is a sort of background modeling process, which is used to separate foreground object from background scene. As mentioned above, the goal for this process is to obtain and build a model of background scene. But in some cases, the scene is already available or very easy to build especially (for example), when videos are recorded from a still (not moved) camera. Thus, there is no need to build and estimate out the

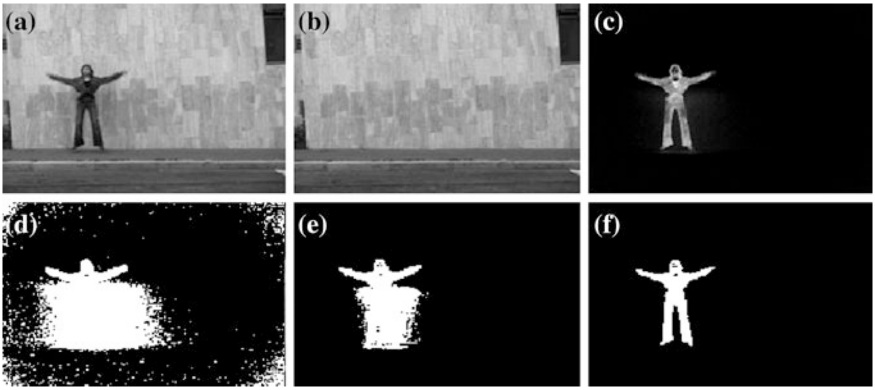


Fig. 2.3 Background subtraction process for a frame of jumping jack action in Weizmann dataset: **a** original frame in gray-scale image, **b** its background scene frame in gray-scale image, **c** background subtraction result image with noise, **d** thresholding into black and white image with noise, **e** thresholding into *black* and *white* with less noise, **f** thresholding into *black* and *white* image almost without noise, which is the result silhouette image

background model. If a background scene is available, the object tracking is simply achieved by subtracting each frame from its available background scene. Figure 2.3 shows the background subtraction process for a frame of jumping jack action in the Weizmann dataset.

At the beginning of this process, all frames in all videos and all background scenes are converted from Red Green Blue (RGB) colors into gray-scale colors. After that, frames are subtracted from their background scene. The result of subtraction is a noisy image, thus a proper threshold is used to extract the human silhouette. The final result is a logical (binary) image containing only the silhouette in white color (1's value), and the background scene in black color (0's value).

2.3.3 Direction Detection Process

The detection of direction is a process of finding movement path for human actions, which are directed from left to right or right to left. Specifically, the actions that have displacement in location, for example in the Weizmann dataset such as walking, jumping forward, skip jumping, side jumping, and running actions. Other actions in the dataset have movement displacements such as bending, jumping in place, jumping jack, waving in one hand, and waving in two hands actions. In order to obtain the movement direction for human action in silhouette videos, a contour of silhouette is obtained in the first frame of video. Then, a center point of the contour is calculated by finding mean of the contour coordinates. Next, a frame center is also calculated suing height and width of the frame. These contour and frame centers used in two purposes: first, it is used to detect displacement direction and,

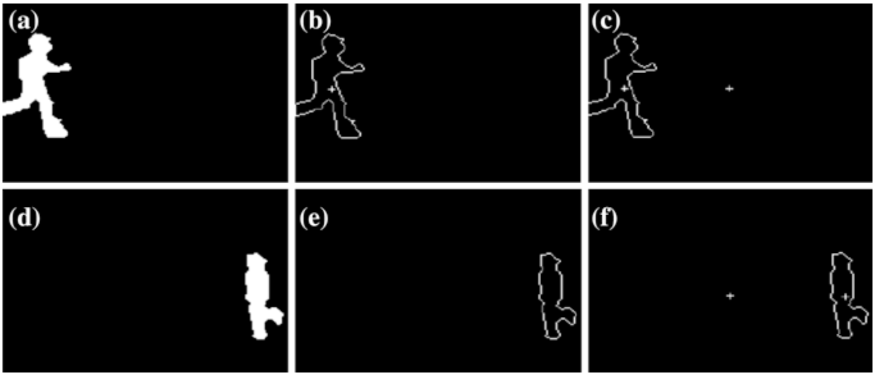


Fig. 2.4 Direction of movement detection process, (*top row*) running from *right* to *left* direction and (*bottom row*) running from *left* to *right* direction: **a**, **d** first frame of silhouette for running, **b**, **e** obtain a center of silhouette, **c** comparing silhouette center with center of frame, which means direction of movement is from *right* to *left*, **f** comparing silhouette center with center of frame, which means direction of movement is from *left* to *right*

second, it is used to align each silhouette horizontally into frame center. By comparing *x*-dim of the silhouette center with the *x*-dim of the frame center, the direction is detected. The direction of action is saved to be used later to unifying direction process. This process is important influence for actions that have displacement movement because same actions with different direction have different features but are symmetric. Figure 2.4 shows detection of direction process for two running actions with different movement directions in the Weizmann dataset.

2.3.4 Horizontal Alignment Process

This process is used for aligning all frames of silhouettes in each video horizontally. These videos are obtained as a result of background subtraction process. All frames are aligned horizontally into the *x*-dim of the frame center. Notice that here all frames are aligned into *x*-dim (horizontally) but not into *y*-dim (vertically). There are a few important advantages of this process. First, it helps to form a consistent aligned silhouette image later. Second, it solves the problem of different number of frames in each video. Finally third, it forms a very discriminant features for each video in the dataset. Figure 2.5 shows the horizontal alignment process for one of skip jumping frames.

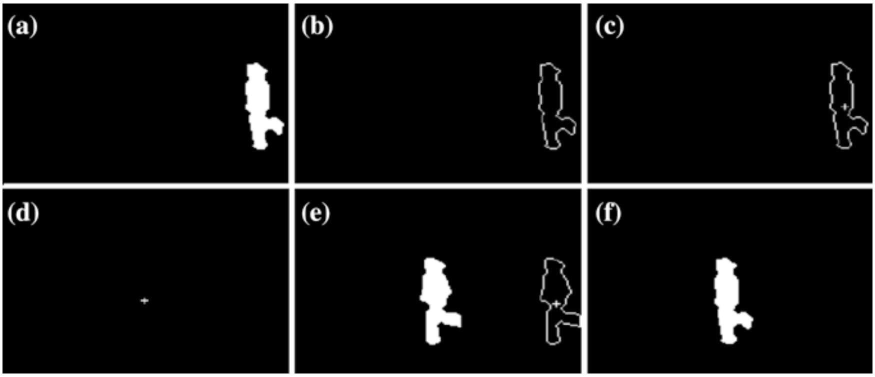


Fig. 2.5 Horizontal alignment for silhouette frame of skip jumping action in Weizmann dataset: **a** silhouette image frame, **b** contour of silhouette, **c** *center* of contour, **d** horizontal frame center, **e** align the silhouette into horizontal center of frame, **f** final result is an aligned silhouette image

2.3.5 Computing Aligned Silhouettes Image Process

This process is used to obtain an image for each video action calculated from all aligned silhouette frames to form an image that captures the most important features of the action video [44]. After completion of horizontal alignment process, all frames of silhouettes in all videos are aligned in x-dim of the frame's center. Then, the ASI is calculated for each video in the dataset. The ASI is the accumulation for silhouette images that are obtained by the summation of all binary silhouette images in each video. The pixels of the formed ASI consist of an integer value from 1 to n, where n is a number of frames in video, since each silhouette frame is a binary image. The idea of the ASI in this research work is inspired from Motion History Image (MHI) [45], Motion Energy Image (MEI) [45], and Gait Energy Image (GEI) [46].

After the ASI is computed, a thresholding (elimination) step is applied. For example, pixels that have intensity value less than 5 are eliminated and converted into value of 0, while others that have intensity value greater than or equal 5 are converted into 1 value. The contribution for this thresholding step is an improvement in accuracy of correct recognition rate. In this step, all pixels that have small value are eliminated. For example, pixel with 1 value means this pixel appears only in one frame during all frames in video. Thus, this pixel can be considered as most likely as a rarely occurred or a noise, so it is eliminated. This process has two advantages: first one is to find such images that have discriminant features for recognition and second one is to solve the problem of different number of frames in each video in the dataset. Figure 2.6 shows the ASI images for two different video actions (bending and jumping forward). The final result of computing ASI process is a binary image for each video that used in the next feature extraction stage.

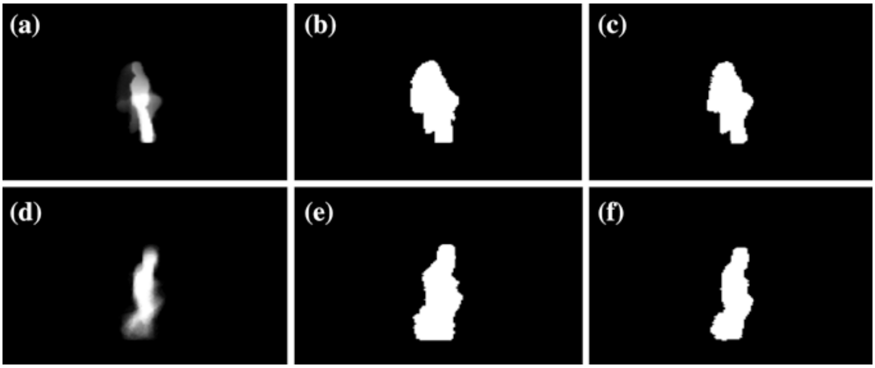


Fig. 2.6 Aligned silhouettes images for two human silhouette action videos in Weizmann dataset, (*top row*) for bending action and (*bottom row*) for jumping forward action: **a**, **d** summation of all frames, **b**, **e** binary images for the summation and thresholding equal 1, **c**, **f** binary images for the summation and thresholding equal 5

2.3.6 Unifying Direction Process

The unifying direction is a process of converting all actions that have displacement into one direction, for example, from left to right. The advantage of this process is to form more robustness feature, since it gets rid the difference in built features that resulted for same actions but opposite in direction. In Sect. 2.3.3, the direction for each video is detected. This process is used to unify the direction for all actions that have movement displacement such as jumping forward, jumping jack, skip jumping, running, and walking. Each of these actions has a direction either from left to right or from right to left. Thus, all videos with these actions are unified into one direction. The process is achieved by flipping all ASI images (over *y*-axis) from one direction to other. The unification process is not for all videos, it is only for videos with actions have displacement. This process starts after process of computing ASI because instead of unifying all frames in video separately, only one ASI image is unified. Therefore, this step is increased speed of processing.

2.3.7 Cropping Bounding Box Process

The cropping bounding box is a process for obtaining a smallest box that surrounds all silhouette pixels for the ASI, then crop this box area. This process starts by taking each binary ASI and traces its region boundaries. The trace is achieved by using logical OR operation for rows and columns of the ASI. Each row or column will have 0 logical value means it does not have any pixel of silhouette. By this operation, the bounding box is detected.

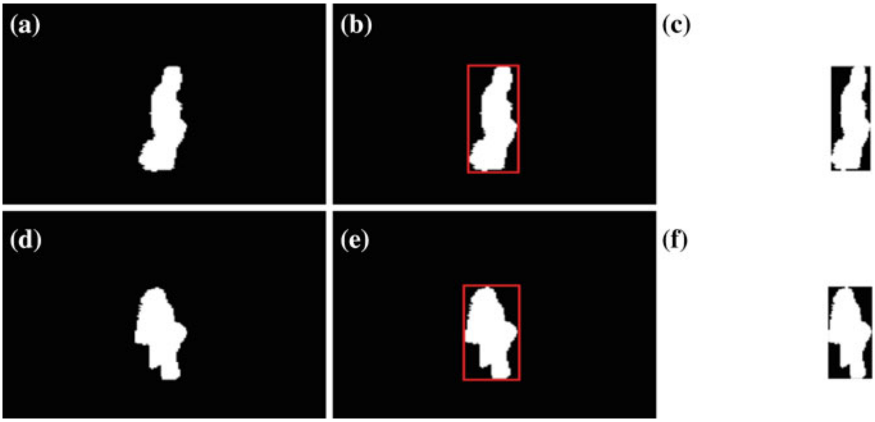


Fig. 2.7 Cropping bounding box process for bending and jumping forward actions in Weizmann dataset: **a**, **c** original ASI images for two actions, respectively, **b**, **d** bounding box (in *red color*) for their silhouettes in ASI images, **c**, **f** cropped (separated) bounding box regions

Then, by tracking first one (1) logical value in rows (from top to down and vice versa) and columns (from left to right and vice versa), bounding box will be detected. As a result, the bounding box of the ASI image is obtained. The bounding has two points in 2D coordinates, which are the left-top and right-bottom of the silhouette. Figure 2.7 shows the cropping bounding box process for two actions in Weizmann dataset.

2.4 Contour-Based Feature Extraction

Contour-based features are directly extracted from contour boundary coordinate points surround silhouette of the ASI image. There are many types of contour-based features such as the Cartesian Coordinate Feature (CCF), the Fourier Descriptor Feature (FDF) [47, 48, 49], Centroid-Distance Feature (CDF) [50, 51], and Chord-Length Feature (CLF) [3, 50, 51]. All these contour-based features are normalized between 0 and 1 values, before being used in the action classification stage.

All these types of contour-based features are explained in Sects. 2.4.1–2.4.4, respectively.

2.4.1 Cartesian Coordinate Feature

The CCF is represented by Cartesian 2D coordinate function. Each coordinate consists of two numbers (x, y) generated from each point on the boundary points of contour. In order to obtain boundary coordinate points, *bwboundaries*, which is a

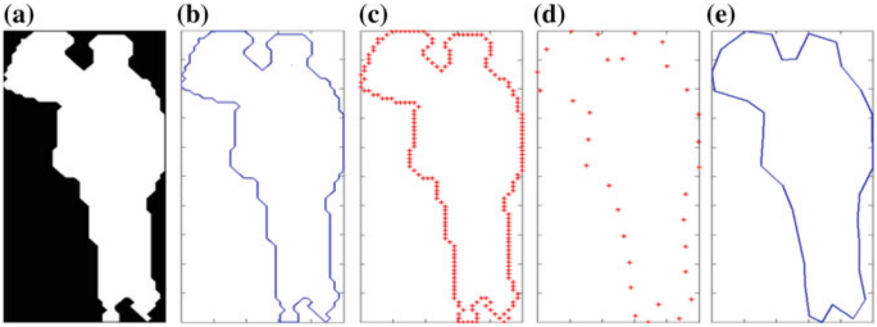


Fig. 2.8 Cartesian coordinates and Fourier descriptors of the ASI for one hand waving action in Weizmann dataset: **a** an ASI image, **b** contour of an ASI image, **c** Cartesian coordinate of the contour, **d** 32 Fourier Descriptors (FDs), **e** plotting of these 32 FDs

function in Matlab [52], is used to trace boundary. The result of this CCF function, which is N Cartesian coordinate points, is shown in Eq. 2.1, where t is an integer, $t \in [1..N]$, N is a number of points on boundary of contour, (x, y) are boundary coordinate points in 2D space, and (x_c, y_c) is a center of gravity for boundary points.

$$CCF(t) = [[x(t) - x_c], [y(t) - y_c]] \tag{2.1}$$

Figure 2.8a–c depict process of obtaining Cartesian coordinates features for one hand waving action in Weizmann dataset.

Until this end, each video will have different length in terms number of boundary points for each ASI image in 2D space. Thus, these different length numbers have to be set into one equal length number. The equalization can be achieved using interpolation [53, 54, 55] or using Fourier Descriptors (FDs) [47, 48, 49]. The interpolation is a method used to unify the number of boundary points for the contour of the ASI image. This method is achieved by constructing or estimating some unknown boundary points based on the known surrounding boundary points; the unknown point values are usually within a range between known values. There are two main types of interpolation based on how to use the known data (boundary points). First type is the global interpolation that employs all the boundary point values to find the unknown values. The second type is the local interpolation that employs a fixed number of known nearest boundary point values.

2.4.2 Fourier Descriptor Feature

The FDs are based on Discrete Cosine Transform (DCT) [47], which is a mathematical operation function for converting time domain into frequency domain. A surprising and importance feature of the FDs is ability to represent any 2D closed shapes independent of their location (translation), scaling, rotation, and starting point [47, 49]. Therefore, the first motivation of using the FDs is due its properties. The second motivation is to unify number points representing each shape boundary for all frames in videos. In short, the FDs are used to describe the contour (boundary) of any closed contour in 2D space based on the DCT methods. The FDs are presented by Eq. 2.2, where z is a complex number function, $[x(t), y(t)]$ are Cartesian boundary points of contour in 2D space, t is an integer such that $t \in [1, N]$, N is a number of points on boundary, and symbol (i) refers to imaginary part of the complex number.

$$z(t) = x(t) + iy(t) \tag{2.2}$$

The FDs function F, based on z, can be calculated using the DCT function [47] from Eq. 2.3, where k is an integer such that $1 \leq k \leq N$, e is the exponential function.

$$F(k) = \frac{1}{N}\sum_{t=1}^{N} z(t)e^{-\frac{j2\pi tk}{N}} \tag{2.3}$$

The expression $z(t)e^{-j2\pi tk/N}$ can be computed from Eq. 2.4, where $p = 2\pi tk/N$, and other parameters are defined above.

$$z(t)e^{-jp} = [x(t) \cdot \cos(p) + y(t) \cdot \sin(p) - x(t) \cdot j \cdot \sin(p) + y(t) \cdot j \cdot \cos(p)] \tag{2.4}$$

It is obvious from Eq. 2.4 that complex numbers are transformed into a linear combination of sins and cosines curves in the frequency domain. In order to reconstruct function $z(t)$, the inverse of the DCT, which is $z'(t)$ based on $F(k)$, is provided by Eq. 2.5.

$$z'(t) = \sum_{k=1}^{N} F(k)e^{\frac{j2\pi tk}{N}} \tag{2.5}$$

However, the approximation of z can be reconstructed by using the function $z'(t)$ with less number of Fourier coefficients such that$1 \leq k \leq p$ and $p < N$. This approximation is useful to unify the number of points for all contours. The reconstructed points by using part or all coefficients are known as the FDF. Figures 2.8d, e show the FDFs that reconstructed from 32 Fourier coefficients and their plots in 2D space, respectively. It is obvious that a plot of contour based on the FDF coordinates, which is shown in Fig. 2.8e, is very similar to the plot of the contour based on all original Cartesian coordinates, which is shown in Fig. 2.8b. The main difference between these two figures is that first figure captures all (low and high) details of the contour, while second figure captures only most important (high) details and ignoring other (low) details.

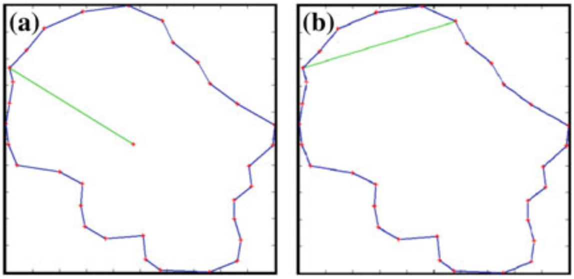


Fig. 2.9 Contour-based features of the ASI for bending action in Weizmann dataset: **a** the CDF between centroid point and a point on boundary of contour, **b** the CLF with jump displacement (w = 5) between two points on boundary of contour

2.4.3 Centroid-Distance Features

The CDFs are features calculated by obtaining the distance (magnitude) between each boundary point based on the FDF and centroid point of the contour in 2D space. In this case, the centroid-distance is calculated from Eq. 2.6, where $CDF(t)$ is a centroid-distance function, $[x(t), y(t)]$ are the FDF coordinates, and x_c, y_c are centroid point coordinates of contour.

$$CDF(t) = \left[(x(t) - x_c)^2 + (y(t) - y_c)^2\right]^{1/2} \tag{2.6}$$

Figure 2.9a depicts one distance from the CDF for the ASI contour of a bending action in Weizmann dataset.

2.4.4 Chord-Length Features

The CLFs are features calculated by obtaining length (magnitude) between two points on boundary based on the FDF points of contour. There is a fixed length in term of the number of points (jump displacement step), denoted by w, separating these two points. These features are calculated from Eq. 2.7, where $CLF(t)$ is a chord-length function, and w is an integer represents jump displacement step.

$$CLF(t) = [(x(t) - x(t + w))^2 + (y(t) - y(t + w))^2]^{1/2} \tag{2.7}$$

In Fig. 2.9b, the CLF feature is depicted with jump displacement step (w = 5) for the contour of the ASI for a bending action in the Weizmann dataset.

2.5 Silhouette-Based Feature Extraction

The silhouette-based features are directly extracted from silhouette. The silhouette is a whole body region inside contour of human object. There are several types of silhouette-based features such as Histogram Of Gradient (HOG) [42, 43], Histogram Of Optical Flow (HOOF) [1], and Structural Similarity Index Measure (SSIM) [56]. All these silhouette-based features are normalized, before being used in the action classification stage. The following Sects. 2.5.1–2.5.3 are devoted for extraction the HOG, the HOOF, and the SSIM features, respectively.

2.5.1 *Histogram of Oriented Gradient Feature*

The HOG is a feature used to capture occurrences of gradient orientation of pixels in overlapping windows of an image. The computation of the HOG is based on magnitudes and angles of these gradients [42, 43]. This feature is extracted from image based on two parameters: number of overlapping windows on this image ($N \times N$), and number of bins (B) for the gradients angles. Briefly, the HOG is computed through several steps. The gradients of an image are computed by filtering this image with horizontal kernel [–1, 0, 1] and vertical kernel $[-1, 0, 1]^{-1}$. Then magnitudes and angles are computed based on the computed gradients. Next, the image is separated into $N \times N$ overlapping windows. For each window, angles are binned into B orientation bins based on their angles' values. For each bin, sum of gradient magnitudes is calculated. After that, these sums, which are equal to the number of bins for each window, are normalized. At the end, $N \times N \times B$ normalized numbers are obtained. These numbers are called the HOG feature descriptors for the image. Figure 2.10a shows the HOG binned orientation of gradient angles into 8 bins.

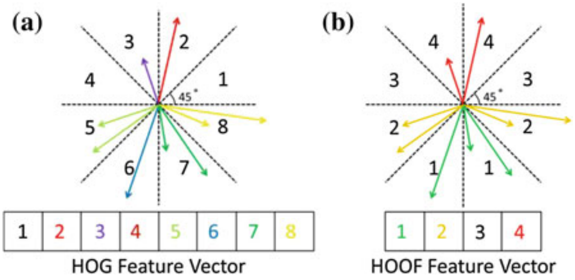


Fig. 2.10 Building histograms based on binned orientation, the angles of gradients control bins location of magnitudes (*colored arrows*), while magnitudes of gradients control lengths of these *arrows*: **a** the HOG with 8 bins, **b** the HOOF with 4 bins, each symmetric angles are binned in one orientation

2.5.2 Histogram of Oriented Optical Flow Feature

The HOOF feature is used to capture optical flow of motion in an image based on the gradient orientation of pixels' intensities in an image [1, 57]. The computation of the HOOF is also based on magnitudes and angles of these gradients. Briefly, the HOOF is computed through several steps. The gradients of image are computed as the same as the HOG by filtering an image with two kernels: horizontal kernel [–1, 0, 1] and vertical kernel [–1, 0, 1]$^{-1}$. Then angles and magnitudes are computed for these gradients. Next, all symmetric angles over y-axis are binned into *B* orientation bins based on the values of these angles. For each bin, a sum of gradient magnitudes is calculated. These sums, which are equal to number of bins, are normalized. At the end, *B* normalized numbers are obtained. These numbers are called the HOOF feature descriptors of image. Figure 2.10b shows the HOOF binned orientation of angles to 4 bins.

The HOOF is similar to the HOG but with some differences. First, the HOOF does not require for sliding windows to be overlapping, because it represents the optical flow of motion in an image, though, this image can be divided into several equal non-overlapping windows [1, 42]. Second, the HOOF is binned symmetric angles over *y*-axis together to overcome problem of detection movement direction while the HOG does not. Third difference is the number of feature descriptors in the HOG is $N \times N \times B$ while in the HOOF is *B* without sliding windows and is $N \times N \times B$ with non-overlapping sliding windows.

2.5.3 Structural Similarity Index Measure Feature

The SSIM feature is used to find an index measurement for similarity between any two (original and distorted) images [56]. This SSIM feature is first time used for human action recognition in [44]. This measurement is effective, especially when there is some difference between these two images in intensity, brightness, and contrast. The SSIM is computed based on three statistical factors (loss of correlation, luminance distortion, and contrast distortion) [44, 56]. The computation of the SSIM for image vectors (*x*, *y*) is shown by Eq. 2.8, where $x = \{x_{i}\,|i = 1, 2, \ldots, N\}$ are intensities of first image, *N* is a number of pixels in an image, $y = \{y_i|i = 1, 2, \ldots, N\}$ are intensities of second image, x and $\bar{y}$ are means of *x*, *y* respectively, σ_x^2 and σ_y^2 are Mean Square Error (MSE) of *x*, *y*, respectively, and σ_{xy} is a correlation coefficient between the *x* and *y*.

$$Q = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \cdot \frac{2\bar{x}\bar{y}}{(\bar{x})^2 + (\bar{y})^2} \cdot \frac{2\sigma_x \sigma_y}{\sigma_x^2 + \sigma_y^2} \tag{2.8}$$

The output of the SSIM is a quality Q with a dynamic range of [–1, 1]. When Q has 1 value means, there are full match between these two images and, when Q has –1 value means, there are significant differences between these images. The biggest positive value means the good similarity between these images, while smallest negative value means the good dissimilarity. In order to compute the SSIM features between x and y vector images, a sliding window moves over these images and the total Q is computed from Eq. 2.9, where M is a number of sliding windows and Q_j is a quality index of the jth window.

$$Q = \frac{1}{M}\sum_{j=1}^{M} Q_j \tag{2.9}$$

Briefly, there are few parameters that affect in calculation of the SSIM features [56]. First, image dimensions (*width*, *height*) are size of image in term of pixels. Second, intensity range (L) is the dynamic range for intensity values of two images. Third, tow constants ($c1$, $c2$) are small constants used in the SSIM formula to avoid division by zero. Finally forth, size of sliding window (*width, height*) is preliminary determined.

2.6 Action Classification

This is a final step in any recognition system. The action classification is based on used algorithm (method). The goal of algorithm is to classify features of testing and identify its class membership by obtaining its nearest neighbor in training samples, such as the KNN classifiers. Sometimes the goal is to train classifier about training samples in training mode and classify an unknown action of a testing sample in testing mode, such as the SVM classifiers. Moreover, these methods are divided into several algorithm techniques.

The outcome result of any classifier can be evaluated using cross-validation techniques such as: 2-fold cross-validation, k-fold cross-validation, and leave-one-out cross-validation. The 2-fold cross validation is the simplest technique, called holdout [41]. Dataset are separated into two sets, one is called training set and other is called testing set. The training set is used in training while other set is used in testing. The k-fold cross-validation is an improvement version of 2-fold [41]. Dataset are separated into k equal sets. The validation is repeated k times. Each time, one set is used for testing and others for training. Leave-one-out cross validation is the most common technique and it is a kind of k-fold but k is taken maximum possible value (logical extreme), which is equal to number of data (N) in the dataset [41]. Dataset are separated into N sets. Each time, one set is used for testing and others for training.

Two different kinds of classifiers KNN (Sect. 2.6.1) and SVM (Sect. 2.6.2) are used with two different techniques as Leave-One-Video-Out (LOVO) and Leave-One-Actor-Out (LOAO). The LOVO is a kind of leave-one-out technique and the LOAO is a kind of *k*-folds cross validation technique. For all experiments, a supervised learning is used. All data are labeled with class memberships except the testing data sample is used without class membership.

2.6.1 K-Nearest Neighbor Classifier

The KNN is the simplest method used for classification, clustering, regression, etc. [7, 8, 9]. The KNN is used in machine learning, pattern recognition, and data mining. It obtains class membership for some testing feature descriptor based on its nearest neighbor from training feature descriptors in feature space. The testing is classified by a majority vote of its *K* nearest neighbors. Due to time required for classifying, the KNN is called lazy learning because the KNN will go over all training samples to find the nearest neighbor for testing sample, therefore, it takes long time, if the training samples are numerous.

In the KNN, there are three parameters are used. First parameter, *K* is set up to number of voting members. Second, distance metric type is set up into: Euclidean (squared difference), cityblock (absolute difference), cosine metrics, etc. Third, the rule for selecting estimated class for testing sample is set up into: nearest neighbor, random, etc. The KNN classifier is calculated the distances (*d*) between testing sample (*x*) and each training sample (*m*) provided from Eq. 2.10, where *d* is a distance metric, *x* is a testing sample, *m* are training samples, $j = [1, 2, \ldots, N]$, *N* is a number of training samples.

$$d(x, m_j) = \text{arg min}_j \{d(x, m_j)\} \tag{2.10}$$

The distance *d* is argument as the minimum distance (nearest neighbor) among distances between *x* and each training sample. The class membership for action with minimum distance is defined as a class membership for testing sample.

In all KNN experiment, two techniques are used. First, the LOVO is leave-one-out cross-validation technique is employed. Thus, all videos in the dataset are used for training except one video is used for testing. Second, the LOAO is 9-fold cross validation techniques, therefore, all videos separated into number of actors, which is nine sets. One actor (set) is used for testing and others for training.

2.6.2 Support Vector Machine Classifier

The SVM is a binary classifier, which separates some feature descriptors by an optimal hyperplane used as a decision function [11, 10, 12, 13]. This hyperplane is represented as a separation, hence called a margin classifier. The SVM can be used to perform a linear or non-linear classification based on using kernels. Once the SVM is trained on features of training samples, the classifier can make decisions about some features testing sample regardless absence of this feature in the testing sample. The classification is performed such as a human is making a decision.

In all SVM experiments, one technique (LOAO) is used. The dataset is separated into nine folds. Each fold represents one actor in the dataset videos. The classification is repeated 9 times. Each time, one fold (actor) is used for testing and others are used to train the classifier. By end of the 9 times, all videos are used in testing and training modes.

2.7 Human Action Recognition in Videos Algorithm

This section provides details about presented algorithm for human action recognition in videos. This algorithm consists of two modes mainly represented in Sects. 2.7.1 and 2.7.2. First, the training mode is a program to train algorithm about human actions using already classified video samples, as depicted in Fig. 2.11a. Second, the testing mode is a program for classifying the unknown action happened in a video sample and identifying its class membership, as depicted in Fig. 2.11b. Usually the training mode is first started, then testing mode will be executed after that.

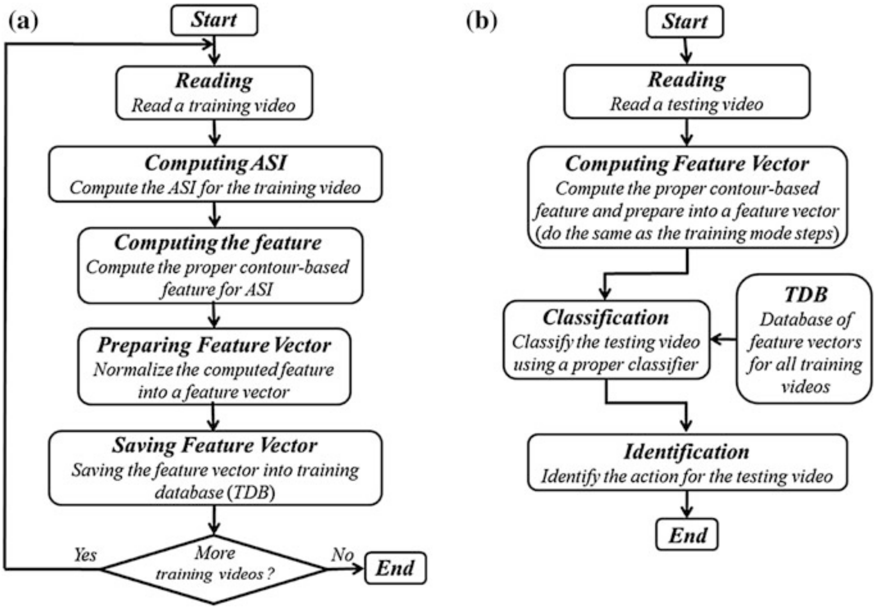

(a)
Start
Reading
Read a training video
Computing ASI
Compute the ASI for the training video
Computing the feature
Compute the proper contour-based feature for ASI
Preparing Feature Vector
Normalize the computed feature into a feature vector
Saving Feature Vector
Saving the feature vector into training database (TDB)
More training videos ?
Yes
No
End
(b)
Start
Reading
Read a testing video
Computing Feature Vector
Compute the proper contour-based feature and prepare into a feature vector (do the same as the training mode steps)
Classification
Classify the testing video using a proper classifier
TDB
Database of feature vectors for all training videos
Identification
Identify the action for the testing video
End

Fig. 2.11 Flow charts of human action recognition algorithms: **a** training mode, **b** testing mode

2.7.1 Training Mode

The training mode is always starts before testing mode in human action recognition. It consists of several processes: reading a training video, computing the ASI from the video, computing (contour-based or silhouette-based) features based on the ASI, preparing feature vector, and saving the feature vector in Training DataBase (TDB). All these steps are repeated for each sample of training video samples in Weizmann dataset. The main structure for training mode of human action recognition in videos is depicted in Fig. 2.11a.

This mode is started by reading a training video process that reads a video sample from the Weizmann dataset. After reading, a process of computing the ASI from this video is performed. Both first and second processes in this algorithm belong to human object tracking stage of the human action recognition system, as depicted in Fig. 2.11. In reading process, all videos in the dataset have (avi) format type. After finishing the reading of all frames, frame by frame in video, the ASI computing process starts in several internal pre-processing steps that have been explained in Sect. 2.3. Briefly, these pre-processing steps involve background subtraction, detection of direction, horizontal alignment, computing the ASI, unifying direction, and bounding box detection processes, respectively.

The rest of processes in training algorithm belong to feature extraction stage of human action recognition, as depicted in Fig. 2.11.

Continuously, after the first and second processes, the third process starts for computing a proper feature. This process employs one of seven different features (contour-based or silhouette-based) that have been explained in Sects. 2.4 and 2.5, respectively. The contour-based involves the CCF, the FDF, the CDF, and the CLF features, while the silhouette-based involves the HOG, the HOOF, and the SSIM features. Regardless of feature type, all features require the ASI that produced in the second process of training mode. For contour-based, the contour of the ASI is first obtained, then a proper features are extracted. While for silhouette-based, features are extracted directly from the ASI. Then, the process of preparing a feature vector is performed. The goal of this process is to normalize the feature vector and make it invariant to scaling and translation with little rotation. Subsequently, for the KNN classifier, the prepared feature vector is saved in the TDB while for the SVM, the classifier has to be trained to find the optimal separation hyperplane for making a decision in testing mode. All processes of training mode are repeated for all training samples in the dataset. At the end of training mode, the TDB have feature vectors for all training videos.

2.7.2 Testing Mode

The testing mode is the second mode of the human action recognition in videos. This mode consists of several processes such as reading a testing video, computing feature vector for the testing video, action classification based on the training feature vectors in the TDB, and identifying action happened in a testing video. Figure 2.11b depicts main structure of testing mode for human action recognition system.

This mode starts by reading a testing video sample, frame by frame, from Weizmann dataset in the same manner as the reading process of training mode. After that, a process of computing feature vector is achieved by a few internal steps. These steps involve computing the ASI for a testing video, computing proper features for this ASI, and preparing the feature vector. This feature vector has to be formatted as the same as all feature vectors in the TDB. Until this end, these steps are common with steps of training mode and have to be exactly in the same manner in every detail, thus comparison of classification algorithm is performed successfully.

Then, the process of classification feature vector for testing video is applied using one of action classification methods that described in Sect. 2.6. The classification process begins using one of two different classifier algorithms: the KNN or the SVM. For the KNN, the feature vector is based on feature vectors in the TDB, which is created in training mode. For the SVM, the created feature vector is projected and classified based on the trained classifier. Subsequently, a final identification is applied to identify the human action that occurred in a testing video based on a result of classifier.

2.8 Experimental Results

The Weizmann dataset is used to test the presented human action recognition system. A background subtraction and several sequence processes are used for human tracking stage. The ASI images are employed to represent each video in the dataset. One feature of contour-based or silhouette-based types is used for feature extraction stage. Also, two classifiers KNN and SVM are used, which are based on two techniques: the LOVO and the LOAO by using 9-folds cross validation. These classifiers are employed for action classification stage. Moreover, the best conducted results for 21 different experiments based on feature and classifier types are presented. Generally, two groups are presented in this section, 12 contour-based experiments for first group and 9 silhouette-based experiments for other. In both groups, two classifiers with different techniques are tested to recognize an action that occurred in videos of Weizmann human action dataset. Figure 2.12 depicts bounding box for contour images. Figure 2.13 depicts bounding box of silhouette images. Both image examples are extracted from different actions in Weizmann dataset.

The descriptions of experiments for various features extraction, including the CCF, the FDF, the CDF, the CLF as well as the HOG, the HOOF, and the SSIM, one can find in Sects. 2.8.1–2.8.7, respectively. Also, the discussion of experimental results is located in Sect. 2.8.8.

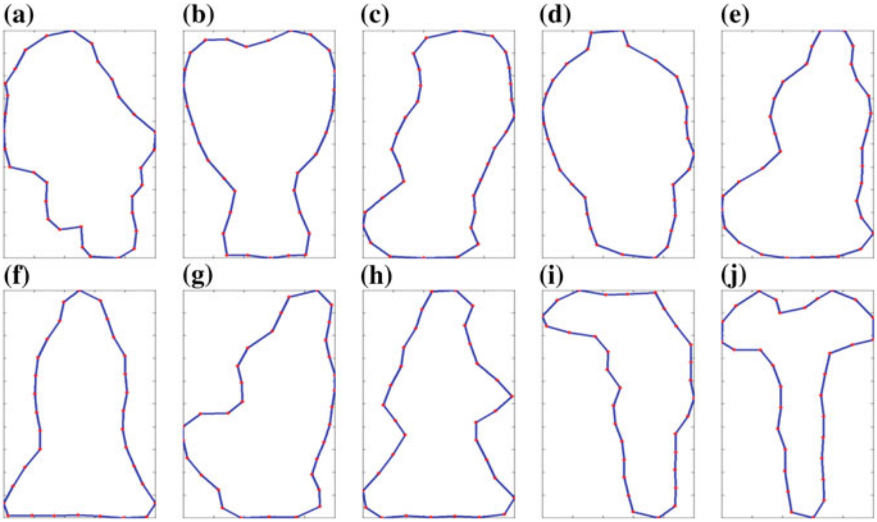


Fig. 2.12 Bounding boxes of contours for the ASI images used to extract contour-based features for 10 different human actions in Weizmann dataset: **a** bending, **b** jumping jack, **c** jumping forward, **d** jumping in place, **e** running, **f** gallop sideway, **g** skip jumping, **h** walking, **i** one hand waving, **j** two hands waving

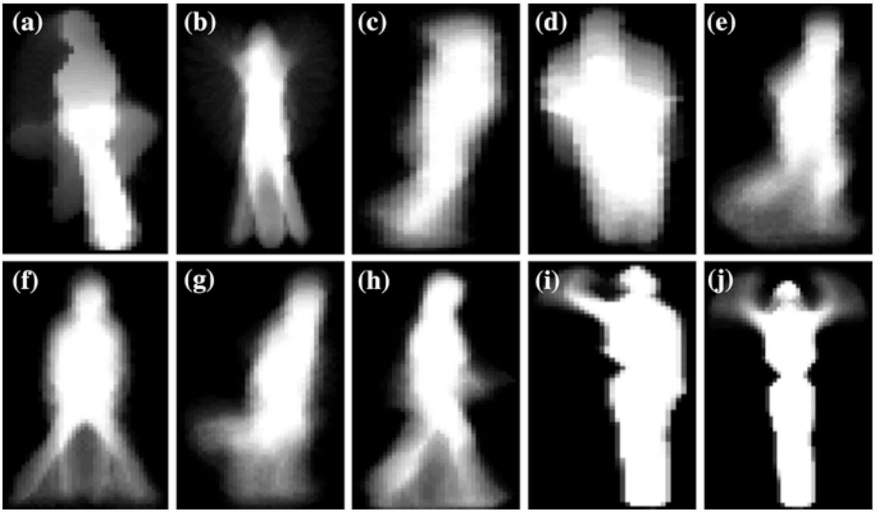


Fig. 2.13 Bounding boxes of the ASI images used to extract silhouette-based features for 10 different human actions in Weizmann dataset: **a** bending, **b** jumping jack, **c** jumping forward, **d** jumping in place, **e** running, **f** gallop sideway, **g** skip jumping, **h** walking, **i** one hand waving, **j** two hands waving

Table 2.1 Cartesian coordinates feature (CCF) experiments setting and results

Exp. no.	Feature parameters	Classifier type	Classifier parameters	Correct recognition rate
1.	No. of boundary points = 16	KNN	Leave-one-video-out, K = 1, distance = euclidean, rule = nearest	89.247
2.	No. of boundary points = 27	KNN	Leave-one-actor-out, 9-folds cross validation, K = 1, distance = cosine, rule = nearest	91.397
3.	No. of boundary points = 22	SVM	Leave-one-actor-out, 9-folds cross validation, classifier = multi-class, kernel = linear	**92.473**

2.8.1 Cartesian Coordinate Feature Experiment

The CCF experiments are one of the contour-based feature types. This feature is extracted from contour of the ASI for each video action in the dataset. For the CCF feature, three experiments are examined, based on the classifier used. In these experiments, only one parameter is used for setting a feature, which is a number of Cartesian coordinate points used for boundary of contour. The summary of setup of parameters and results for the CCF experiments are listed in Table 2.1.

In the first experiment, the KNN is used as a classifier to identify action in the testing sample. The best result recorded for this experiment is 89.247 % of correct recognition rate. For the CCF feature, a number of boundary points is set up to 16 points. For the KNN classifier, the LOVO classification technique is used. The number of voting (K) is set up to 1 value. The Euclidean is used to measure distances, and the nearest neighbor rule is used for identifying action in a testing sample.

In the second experiment, the KNN is used as a classifier. The best result recorded for this experiment is 91.397 % of correct recognition rate. For the CCF feature, a number of the used boundary points is set up to 27 points. For the KNN classifier, the LOAO (9-cross validation) technique is used. The number of voting (K) is set up to 1. The Cosine is used to measure distances, and the nearest neighbor is used as a rule to identify the action.

In the third experiment, the SVM classifier is used for classification. The best result recorded for this experiment is 92.473 % of correct recognition rate. For the CCF feature, a number of the used boundary points is set up to 22 points. For classifier, a multi-class SVM type is used. The LOAO (9-cross validation) is used as a classification technique. Also, the linear kernel is used for this classifier.

Table 2.2 Fourier descriptor feature (FDF) experiments setting and results

Exp. no.	Feature parameters	Classifier type	Classifier parameters	Correct recognition rate
1.	No. of Fourier descriptors (FDs) = 18	KNN	Leave-one-video-out, K = 1, distance = euclidean, rule = nearest	**93.548**
2.	No. of Fourier descriptors (FDs) = 18	KNN	Leave-one-actor-out, 9-folds cross validation, K = 1, distance = cityblock, rule = nearest	91.397
3.	No. of Fourier descriptors (FDs) = 80	SVM	Leave-one-actor-out, 9-folds cross validation, classifier = multi-class, kernel = linear	91.397

2.8.2 Fourier Descriptor Feature Experiments

The FDF experiments are contour-based feature type. This feature is extracted from the contour of the ASI for each video action in the dataset. For the FDF feature, three experiments are examined, based on the classifier. In these experiments, one parameter is used for setting of feature, which is a number of the FDs used to represent boundary of contour. The summary of setup for parameters and results for the FDF experiments are listed in Table 2.2.

In the first experiment, the KNN is used as a classifier to identify the action. The best result recorded for this experiment is 93.548 % of correct recognition rate, which is the best recognition rate achieved in the contour-based feature types. For the FDF feature, a number of the FDs points is set up to 18 points. For the KNN classifier, the LOVO technique is used. The number of voting (K) is set up to 1. The Euclidean is used to measure distances, and the nearest neighbor rule is used.

In the second experiment, the KNN is used as a classifier. The best result recorded for this experiment is 91.397 % of correct recognition rate. For the FDF feature, a number of the used FDs points is also set up to 18 points. For the KNN classifier, the LOAO (9-folds validation) technique is used. The number of voting (K) is set up to 1. The cityblock is used to measure distance, and the nearest neighbor rule is used.

In the third experiment, the SVM classifier is used for classification. The best result recorded for this experiment is 91.397 % of correct recognition rate. The number of the FDs points is set up to 80 points. For classifier, the multi-class SVM type is used. The LOAO (9-folds validation) is used as a classification technique. Also, the linear kernel is used as a base for this classifier.

2.8.3 Centroid-Distance Feature Experiments

The CDF experiments are one of the contour-based types. This feature is extracted based on the FDF. For the CDF feature, three experiments are conducted, based on the

Table 2.3 Centroid-distance feature (CDF) experiments setting and results

Exp. no.	Feature parameters	Classifier type	Classifier parameters	Correct recognition rate
1.	No. of Fourier descriptors (FDs) = 18	KNN	Leave-one-video-out, K = 1, distance = euclidean, rule = nearest	**92.473**
2.	No. of Fourier descriptors (FDs) = 18	KNN	Leave-one-actor-out, 9-folds cross validation, K = 1, distance = cityblock, rule = nearest	92.473
3.	No. of Fourier descriptors (FDs) = 96	SVM	Leave-one-actor-out, 9-folds cross validation, classifier = multi-class, kernel = linear	86.021

classifier. In these experiments, one parameter is only used for feature setting. This parameter is a number of the FDs used for boundary representation. The summary of setup for parameters and results for the CDF experiments are listed in Table 2.3.

In the first experiment, the KNN is used as a classifier to identify the action. The best result recorded for this experiment is 92.473 % of correct recognition rate. For the CDF feature, a number of the FDs points is set up to 18 points. For the KNN classifier, the LOVO technique is used. The number of voting (K) is set up to 1. The Euclidean is used to measure distances, and the nearest neighbor rule is used for identifying the action.

In the second experiment, the KNN is used as classifier. The best result recorded for this experiment is 92.473 % of correct recognition rate. For the CDF feature, a number of the FDs points is also set up to 18 points. For the KNN classifier, the LOAO (9-folds validation) technique is used. The number of voting (K) is set up to 1. The cityblock is used to measure distances, and the nearest neighbor rule is used.

In the third experiment, the SVM classifier is used for classification. The best result recorded for this experiment is 86.021 % of correct recognition rate. For the CDF feature, a number of the FDs points is set up to 96 points. For classifier, multi-class SVM type is used. The LOAO (9-folds validation) is used as a classification technique. Also, the linear kernel is used for this classifier.

2.8.4 Chord-Length Feature Experiments

The CLF experiments are contour-based feature type. This feature is extracted from the FDF. For the CLF feature, three experiments are experimented, based on classifier type. In these experiments, two parameters are used for setting the FDF. The first parameter is a number of used FDs for boundary representation. The second one is a jump displacement in term of number of points separating two FDs points of chord on boundary of contour. The summary of setup for parameters and results for the CLF experiments are listed in Table 2.4.

Table 2.4 Chord-length feature (CLF) experiments setting and results

Exp. no.	Feature parameters	Classifier type	Classifier parameters	Correct recognition rate
1.	No. of Fourier descriptors (FDs) = 30, jump displacement = 12	KNN	Leave-one-video-out, K = 1, distance = euclidean, rule = nearest	**89.247**
2.	No. of Fourier descriptors (FDs) = 30, jump displacement = 12	KNN	Leave-one-actor-out, 9-folds cross validation, distance = cityblock, rule = nearest, K = 1	89.247
3.	No. of Fourier descriptors (FDs) = 86, jump displacement = 28	SVM	Leave-one-actor-out, 9-folds cross validation, classifier = multi-class, kernel = linear	89.247

In the first experiment, the KNN is used as a classifier to identify the action. The best result recorded for this experiment is 89.247 % of correct recognition rate. For the CLF feature, a number of the FDs points is set up to 30 points and a jump displacement is set up to 12 separation points. For the KNN classifier, the LOVO technique is used. The number of voting (K) is set up to 1. The Euclidean is used to measure the distances and the nearest neighbor rule is used to identify the action.

In the second experiment, the KNN is used as a classifier. The best result recorded for this experiment is 89.247 % of correct recognition rate. For receiving of the CLF feature, a number of the FDs points is also set up to 30 points and a jump displacement is set up to 12 separation points. For the KNN classifier, the LOAO (9-folds validation) technique is used. The number of voting (K) is set up to 1. The Euclidean is used to measure distances and the nearest neighbor rule is used.

In the third experiment, the SVM classifier is used for classification. The best result recorded for this experiment is 89.247 % of correct recognition rate. For receiving of the CLF feature, a number of FDs points is set up to 86 points and a jump displacement is set up to 28 separation points. For the SVM classifier, the multi-class SVM type is used. The LOAO (9-folds validation) is used as a classification technique. The linear kernel is used as base for this classifier.

2.8.5 Histogram of Oriented Gradient Feature Experiments

The HOG feature experiments are one of the silhouette-based feature types. This feature is extracted directly from the ASI. For the HOG feature, three experiments are conducted, based on classifier type. In these experiments, two parameters are used for the HOG feature. The first parameter is a number of overlapping windows. The second one is a number of bins for orientation of angles. The summary of setup for parameters and results for the HOG experiments are listed in Table 2.5.

Table 2.5 Histogram of oriented gradient (HOG) feature experiments setting and results

Exp. no.	Feature parameters	Classifier type	Classifier parameters	Correct recognition rate
1.	No. of overlapping windows = (6 × 6), No. of bins = 12	KNN	Leave-one-video-out, K = 1, distance = euclidean, rule = nearest	**98.924**
2.	No. of overlapping windows = (6 × 6), No. of bins = 12	KNN	Leave-one-actor-out, 9-folds cross validation, K = 1, distance = cityblock, rule = nearest	97.849
3.	No. of Overlapping windows = (2 × 6), No. of bins = 11	SVM	Leave-one-actor-out, 9-folds cross validation, classifier = multi-class, kernel = linear	97.849

In the first experiment, the KNN classifier is used to identify the action. The best result recorded for this experiment is 98.924 % of correct recognition rate, which is the best recognition rate achieved in both contour-based and silhouette-based feature types. For the HOG feature, a number of overlapping windows is set up to (6 × 6) windows and a number of bins is set up to 12 bins. For the KNN classifier, a number of voting (K) is set up to 1. The LOVO technique is used. The Euclidean is used to measure distances, and the nearest neighbor rule is used to identify the action.

In the second experiment, the KNN is used as a classifier. The best result recorded for this experiment is 97.849 % of correct recognition rate. For the HOG feature, a number of overlapping windows is set up to (6 × 6) windows and a number of bins is set up to 12 bins. For the KNN classifier, the LOAO technique (9-folds validation) is used. The number of voting (K) is set up to 1. The nearest neighbor is used as a rule based on Euclidean distance.

In the third experiment, the SVM classifier is used for classification. The best result recorded for this experiment is 97.849 % of correct recognition rate. For the HOG feature, a number of overlapping windows is set up to (2 × 6) windows and a number of bins is set up to 11 bins. For a classifier, the multi-class SVM type is used. The LOAO (9-folds validation) is used as a classification technique. The linear kernel is used as base for this classifier.

2.8.6 Histogram of Oriented Optical Flow Feature Experiments

The HOOF feature experiments silhouette-based feature type. This feature is also extracted directly from the ASI. For this feature, three experiments are examined, based on classifier type. In these experiments, two parameters are used for the HOOF feature. The first parameter is a number of non-overlapping windows.

Table 2.6 Histogram of oriented optical flow (HOOF) feature experiments setting and results

Exp. no.	Feature parameters	Classifier type	Classifier parameters	Correct recognition rate
1.	No. of non-overlapping windows = (5 × 5), No. of bins = 7	KNN	Leave-one-video-out, K = 1, distance = euclidean, rule = nearest	96.774
2.	No. of non-overlapping windows = (5 × 5), No. of bins = 7	KNN	Leave-one-actor-out, 9-folds cross validation, K = 1, distance = cityblock, rule = nearest	**97.849**
3.	No. of non-overlapping windows = (8 × 3), No. of bins = 3	SVM	Leave-one-actor-out, 9-folds cross validation, classifier = multi-class, kernel = linear	96.774

The second one is a number of oriented bins for angles. The summary of setup for parameters and results for the HOOF experiments are listed in Table 2.6.

In the first experiment, the KNN classifier is used to identify the action. The best result that recorded for this experiment is 96.774 % of correct recognition rate. For the HOOF feature, a number of non-overlapping windows is set up to (5 × 5) windows and a number of bins is set up to 7 bins. For the KNN classifier, the LOVO technique is used. The number of voting (K) is set up to 1. The Euclidean is used to measure the distances, and the nearest neighbor rule is used.

In the second experiment, the KNN is used as a classifier. The best result that recorded for this experiment is 97.849 % of correct recognition rate. For the HOOF feature, a number of non-overlapping windows is set up to (5 × 5) windows and a number of bins is set up to 7 bins. For the KNN classifier, the LOAO technique (9-folds validation) is used. The number of voting (K) is set up to 1. The cityblock metric is used to measure distances. The nearest neighbor rule is used to identify the action.

In the third experiment, the SVM classifier is used for classification. The best result that recorded for this experiment is 96.774 % of correct recognition rate. For the HOOF feature, a number of non-overlapping windows is set up to (8 × 3) windows and a number of bins is set up to 3 bins. For classifier, the multi-class SVM type is used. The LOAO (9-folds validation) is used as a classification technique. The linear kernel is used as base for this classifier.

2.8.7 Structure Similarity Index Measure Feature Experiments

The SSIM feature experiments are one of the silhouette-based types. This feature is also extracted directly from the ASI. For this feature, three experiments are conducted, based on classifier type. In these experiments, four parameters are used for

Table 2.7 Structure similarity index measure (SSIM) features experiments setting and results

Exp. no.	Feature parameters	Classifier type	Classifier parameters	Correct recognition rate
1.	Image dimension = (50 × 36), L = 4, C1 = 0.03, C2 = 0.01, window = (2 × 2)	KNN	Leave-one-video-out, K = 1, distance = euclidean, rule = nearest	**98.924**
2.	Image dimension = (50 × 36), L = 4, C1 = 0.03, C2 = 0.01, window size = (2 × 2)	KNN	Leave-one-actor-out, 9-folds cross validation, K = 1, distance = cityblock, rule = nearest	94.623
3.	Image dimension = (47 × 48), L = 7, C1 = 0.03, C2 = 0.01, window size = (2 × 2)	SVM	Leave-one-actor-out, 9-folds cross validation, classifier = multi-class, kernel = linear	96.774

the SSIM feature. The first parameter is a dimension size of the ASI, which represents dimensions of an image. The second parameter is a dynamic range of intensity values. The third parameter is two small constants used to overcome problem of division by zero. The fourth parameter is a size of overlapping windows used to obtain one SSIM feature value. The summary of setup for parameters and results for SSIM experiments are listed in Table 2.7.

In the first experiment, the KNN classifier is used to identify the action. The best result that recorded for this experiment is 98.924 % of correct recognition rate. For the SSIM feature, an image dimension is set up to (50 × 36) for all ASIs. A dynamic range is set up to (L = 4) value. Two small constants are set up to (C1 = 0.03 and C2 = 0.01). The overlapping window is set up to (2 × 2). For the KNN classifier, the LOVO technique is used. The number of voting (K) is set up to 1. The Euclidean is used to measure the distances, and the nearest neighbor rule is used.

In the second experiment, the KNN is used as a classifier. The best result that recorded for this experiment is 94.623 % of correct recognition rate. For the SSIM feature, an image dimension is set up to (50 × 36) for all ASIs. The dynamic range is set up to (L = 4) value. Two small constants are set up to (C1 = 0.03 and C2 = 0.01). The number of overlapping window is set up to (2 × 2). For the KNN classifier, the LOAO technique (9-folds validation) is used. The number of voting (K) is set up to 1. The cityblock metric is used to measure distances among samples, and the nearest neighbor rule is used.

In the third experiment, the SVM classifier is used for classification. The best result that recorded for this experiment is 96.774 % of correct recognition rate. For the SSIM feature, an image dimension is set up to (47 × 48) for all ASIs. The dynamic range is set up to value 7. Two small constants used to avoid division by zero problem are set up to (C1 = 0.03 and C2 = 0.01). The number of overlapping window is set up to (2 × 2). For classifier, the multi-class SVM type is used. The LOAO (9-folds validation) is used as a classification technique. Also, the linear kernel is used as base for this classifier.

2.8.8 Experimental Results Discussion

In order to test the presented algorithm, different experiment results are conducted. Summary for all results are listed in Table 2.8.

For feature extraction, seven different types are used. Four are contour-based feature type such as the CCF, the FDF, the CDF, and the CLF. Also, three are silhouette-based feature type such as the HOG, the HOOF, and the SSIM. These features are used for human action recognition in Weizmann dataset [2]. Moreover, for each of both (contour-based and silhouette-based) features, two different types of the classifiers (the KNN and the SVM) are used with different techniques. Totally, 21 experiments are achieved to recognize actions in Weizmann dataset. As listed as in Table 2.8, for contour-based feature type, the best result 93.548 % is achieved for the FDF using the KNN-LOVO, which is the KNN classifier based on

Table 2.8 Results of all contour-based and silhouette-based feature type experiments using seven features and three classifiers

Exp. no.	Feature-based	Feature type	Classifier type	Corrects recognition/all	Wrongs recognition/all	Correct recognition rate %
1.	Contour-based	CCF	KNN-LOVO	83/93	6/93	89.247
2.			KNN-LOAO	85/93	8/93	91.397
3.			SVM	86/93	7/93	92.473
4.		FDF	KNN-LOVO	87/93	6/93	**93.548**
5.			KNN-LOAO	85/93	8/93	91.397
6.			SVM	85/93	8/93	91.397
7.		CDF	KNN-LOVO	86/93	7/93	92.473
8.			KNN-LOAO	86/93	7/93	92.473
9.			SVM	80/93	13/93	86.021
10.		CLF	KNN-LOVO	83/93	10/93	89.247
11.			KNN-LOAO	83/93	10/93	89.247
12.			SVM	83/93	10/93	89.247
13.	Silhouette-based	HOG	KNN-LOVO	92/93	1/93	**98.924**
14.			KNN-LOAO	91/93	2/93	97.849
15.			SVM	91/93	2/93	97.849
16.		HOOF	KNN-LOVO	90/93	3/93	96.774
17.			KNN-LOAO	91/93	2/93	97.849
18.			SVM	90/93	3/93	96.774
19.		SSIM	KNN-LOVO	91/93	2/93	98.924
20.			KNN-LOAO	88/93	5/93	94.623
21.			SVM	90/93	3/93	96.774

leave-one-video-out technique. For silhouette-based feature type, the best result 98.924 % is achieved for the HOG using also the KNN-LOVO classifier. This result is the best result achieved among all 21 experiments. Generally, the results of silhouette-based are similar to each other and contour-based are also. The silhouette-based are better than contour-based feature types.

For contour-based feature type, the FDF is performed better result in term of accuracy than others. This is due that the FDF is capturing the high details and ignoring the low details in the contour. This is one of main characteristics for the FDs. The CCF is using all coordinates for contour and employing interpolation techniques for unifying number of boundary points for each contour. Therefore, accuracy results of the FDF are better than the CCF. Others CDF and CLF are converting the FDF into the distances (lengths) instead of using them directly. Thus, there results are close to each other but the FDF is more accurate. All silhouette-based features achieved very similar results to each other. The HOG and the SSIM achieve slightly more accurate result than the HOOF. This is due than both are using an overlapping windows to extract the feature, while the HOOF is based on non-overlapping windows.

2.9 Conclusion

In this chapter, the goal of human action recognition is achieved based on two different types (contour-based and silhouette-based) of features and by using three different types of classification (the KNN-LOVO, the KNN-LOAO, and the SVM). The results proved that silhouette-based features are better, in term of accuracy rate, than contour-based features, due to three reasons. First, silhouette-based features implicitly contain the contour, which is the boundary of silhouette. Second, silhouette-based features cover interior regions while contour-based features are covering border line. Third, silhouette-based features contain different intensity values that help to build better recognition feature in term of accuracy rate. For example, the silhouette-based features are capturing different intensity values and some possible gap region(s) that located interior of silhouette, while the contour-based ignores all these intensity values and regions of silhouette.

Moreover, the comparison between these two features (contour-based and silhouette-based), in term of computation time, shows that the contour-based is faster than the silhouette-based. This is due to a number of pixels are used in computation in contour is small while in the silhouette-based this number is large and very large comparing into the contour-based. In other words, the contour-based features are performing computation only on the boundary (sub pixels) points while the silhouette-based are performing computation over all pixels of an image (silhouette region and empty region). The contour-based approach is easier and simpler than silhouette-based, in term of complexity, because silhouette-based method employs an additional calculation such as orientation of bins in the HOG and the HOOF and also variables multitude in the SSIM.

The accuracy of classifiers and techniques demonstrates that these classifiers are close to each in term of accuracy in both contour-based and silhouette-based types. Although, the accuracy of the KNN-LOVO is slightly better than other (the KNN-LOAO and the SVM) classifiers, in term of correct recognition rate, since the best results are recorded using the KNN-LOVO in both contour-based and silhouette-based features. In all experiments, results in all three types of classifiers, reflected closeness, in term of accuracy, to each other, as listed in Table 2.8.

Moreover, the comparison between these three classifiers, in term of computation time, shows that the SVM is faster than other two KNN classifiers. The SVM requires more time in training mode and little time for testing mode, while the KNN requires more time in the testing. The time of testing is important for classification. Also, both KNN classifiers are almost similar to each other but slower than the SVM in testing mode. Therefore, the KNN is generally called lazy classifiers. The SVM classifier is more complex than others, in term of complexity, because it is based on kernels and margins. The KNN classifiers are simpler to compute and easier to use compared to the SVM.

References

1. Chaudhry R, Ravichandran A, Hager G, Vidal R (2009) Histograms of oriented optical flow and Binet-Cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In: IEEE conferences on computer vision and pattern recognition (CVPR'2009), pp 1932–1939
2. Gorelick L, Blank M, Shechtman E, Irani M, Basri R (2007) Actions as space-time shapes. IEEE Trans Pattern Anal Mach Intell 29(29):2247–2253. http://www.wisdom.weizmann.ac.il/~vision/SpaceTimeActions.html. Accessed 15 June 2014
3. Sadek S, Al-Hamadi A, Michaelis B, Sayed U (2012) Chord length shape features for human activity recognition. ISRN machine vision, article ID 872131. doi:10.5402/2012/872131
4. Aggarwal JK, Ryoo MS (2011) Human activity analysis: a review. ACM Comput Surv 43 (3):16:1–16:43
5. Monnet A, Mittal A, Paragios N, Ramesh V (2003) Background modeling and subtraction of dynamic scenes. In: IEEE international conferences on computer vision (ICCV'2003), vol 2, pp 1305–1312
6. Piccardi M (2004) Background subtraction techniques: a review. In: Proceeding IEEE international conferences on systems, man and cybernetics, vol 4, pp 3099–3104
7. Deza E, Deza MM (2009) Encyclopedia of distances. Springer, Berlin
8. Duda RO, Hart PE, Stork DG (2000) Pattern classification, 2nd edn. Wiley-Interscience, New York
9. Elkan C (2011) Nearest neighbor classification. doi:10.1007/978-0-387-39940-9_2920
10. Ben-Hur A, Weston J (2010) A user's guide to support vector machines. In: Carugo O, Eisenhaber F (eds) Data mining techniques for the life sciences. Humana Press a part of Springer Science + Business Media, LLC 2010, New York
11. Burges CJC (1998) A tutorial on support vector machines for pattern recognition. Data Min Knowl Disc 2(2):121–167

12. Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2(3):27:1–27:2
13. Gunn SR (1998) Support vector machines for classification and regression. University of Southampton, Technical report MP-TR-98-05, Image speech and intelligent systems group
14. Yilmaz A, Javed O, Shah M (2006) Object tracking: a survey. ACM Comput Surv 38(4):1–45
15. Harris C, Stephens M (1988) A combined corner and edge detector. In: 4th Alvey vision conferences, pp 147–151
16. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110
17. Mikolajczyk K, Schmid C (2002) An affine invariant interest point detector. In: Proceedings of the 7th European conferences on computer vision (ECCV'2002), vol 1, pp 128–142
18. Comaniciu D, Ramesh V, Meer P (2003) Kernel-based object tracking. IEEE Trans Pattern Anal Mach Intell 25(5):564–575
19. Shi J, Tomasi C (1994) Good features to track. In: IEEE conferences on computer vision and pattern recognition (CVPR'1994), pp 593–600
20. Comaniciu D, Meer P (1999) Mean shift analysis and applications. In: International conferences on computer vision (ICCV'1999),vol 2, pp 1197–1203
21. Shi J, Malik J (1997) Normalized cuts and image segmentation. In: IEEE conferences on computer vision and pattern recognition (CVPR'1997), pp 731–737
22. Caselles V, Kimmel R, Sapiro G (1997) Geodesic active contours. Int J of Comput Vis 22 (1):61–79
23. Wren C, Azarbayejani A, Darrell T, Pentland A (1997) Pfinder: real-time tracking of the human body. IEEE Trans Pattern Anal Mach Intell 19(7):780–785
24. Lo BPL, Velastin SA (2001) Automatic congestion detection system for underground platforms. In: International symposium on intelligent multimedia, video and speech processing (ISIMP'2001), pp 158–161
25. Cucchiara R, Grana C, Piccardi M, Prati A (2003) Detecting Moving Objects, Ghosts, and Shadows in Video Streams. IEEE Trans Pattern Anal Mach Intell 25(10):1337–1342
26. Stauffer C, Crimson WEL (1999) Adaptive background mixture models for real-time tracking. In: IEEE conferences on computer vision and pattern recognition (CVPR'1999), vol 2, pp 246–252
27. Stauffer C, Grimson WEL (2000) Learning patterns of activity using real-time tracking. IEEE Trans Pattern Anal Mach Intell 22(8):747–757
28. Oliver NM, Rosario B, Pentland AP (2000) A Bayesian computer vision system for modeling human interactions. IEEE Trans Pattern Anal Mach Intell 22(8):831–843
29. Foschi PG, Kolippakkam D, Liu H, Mandvikar A (2002) Feature extraction for image mining. In: 8th international workshop multimedia information systems, pp 103–109
30. Amraji N, Mu L, Milanova M (2011) Shape-based human actions recognition in videos. In: 14th international conferences on human–computer interaction: design and development approaches, vol 1, pp 539–546
31. Zhao H, Liu Z (2009) Shape-based human activity recognition using edit distance. In: 2nd international congress on image and signal processing (CISP'2009), pp 1–4
32. Zivkovic Z, Heijden van der F, Petkovic M, Jonker W (2001) Image segmentation and feature extraction for recognizing strokes in tennis game videos. In: 7th annual conferences of the advanced school for computing and imaging (ASCI'2001), pp 262–267
33. Vezhnevets A, Vezhnevets V (2005) Modest AdaBoost—teaching AdaBoost to generalize better. In: 15th international conferences on computer graphics and applications (GraphiCon'2005)
34. Rabiner LR, Juang BH (1986) An introduction to hidden Markov models. IEEE ASSP Mag 3 (1):4–16
35. Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 77(2):257–286

36. Kwok KL (1989) A neural network for probabilistic information retrieval. In: 12th annual international ACM SIGIR conferences on research and development in information retrieval (SIGIR‘1989), vol 23(SI), pp 21–30
37. Stergiou C, Siganos D (2014) Neural network. http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html. Accessed 15 June 2014
38. Domingos P (2012) A few useful things to know about machine learning. Mag Commun ACM 55(10):78–87
39. Ozgur A (2004) Supervised and unsupervised machine learning techniques for text document categorization. MSc thesis, Bogazici University
40. Zhu X, Goldberg AB (2009) Introduction to semi-supervised learning. Synth Lect Artif Intell Mach Learn 3(1):1–130
41. Schneider J Cross validation. http://www.cs.cmu.edu/~schneide/tut5/node42.html. Accessed 13 Feb 2014
42. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: IEEE conference on computer vision and pattern recognition (CVPR’2005), vol 1, pp 886–893
43. Dalal N, Triggs B, Schmid C (2006) Human detection using oriented histograms of flow and appearance. In: European conferences on computer vision (ECCV’2006), pp 428–441
44. Al-Ali S, Milanova M (2014) Human action recognition in videos using structure similarity of aligned motion images. Int j reasoning-based intell syst (IJRIS), 6(1/2):7182
45. Han J, Bhanu B (2006) Individual recognition using gait energy image. IEEE Trans on Pattern Anal Mach Intell 28(2):316–322
46. Huang C, Hsieh C, Lai K, Huang W (2011) Human action recognition using histogram of oriented gradient of motion history image. In: IEEE 1st international conferences on instrumentation, measurement, computer, communication and control (IMCCC’2011), pp 353–356
47. Gonzalez R, Woods R, Eddins S (2009) Digital image processing using Matlab, 2nd edn. Gatesmark Publishing, Knoxville
48. Kauppinen H, Seppanen T, Pietikainen M (1995) An experimental comparison of autoregressive and Fourier-based descriptors in 2D shape classification. IEEE Trans Pattern Anal Mach Intell 17(2):201–207
49. Léon RD, Sucar L (2000) Human silhouette recognition with Fourier descriptors. In: 15th international conferences on pattern recognition (ICPR’2000), vol 3, pp 709–712
50. Zhang D, Lu G (2004) Review of shape representation and description techniques. Pattern Recogn 37(1):1–19
51. Zhang D, Lu G (2003) A comparative study on shape retrieval using Fourier descriptors with different shape signatures. J Vis Commun Image Represent 14(1):41–60
52. MathWorks Inc.: MATLAB version R2013a (8.1.0.604) win 64-bit software, February (2013)
53. Li Z, Li X, Li C, Cao Z (2010) Improvement on inverse distance weighted interpolation for ore reserve estimation. In: Proceedings of the fuzzy systems and knowledge discovery (FSKD’2010), PP 1703–1706
54. Luo H, He X (2011) An improved inverse distance weighted interpolation method for InSAR tropospheric delay error corrections. In: International conferences on information science and technology (ICIST’2011), pp 480–482
55. Revesz P, Li L (2002) Constraint-based visualization of spatial interpolation data. In: IEEE 6th international conferences on information visualization, pp 563–569
56. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structure similarity. IEEE Trans Image Process 13(4):600–612
57. Pers J, Sulic V, Kristan M, Perse M, Polanec K, Kovacic S (2010) Histograms of optical flow for efficient representation of body motion. Pattern Recogn Lett 31(11):1369–1376