Chapter 7
Motion History Histograms for Human Action Recognition

Hongying Meng, Nick Pears, Michael Freeman, and Chris Bailey

Abstract In this chapter, a compact human action recognition system is presented with a view to applications in security systems, human-computer interaction, and intelligent environments. There are three main contributions: Firstly, the framework of an embedded human action recognition system based on a support vector machine (SVM) classifier and some compact motion features has been presented. Secondly, the limitations of the well-known motion history image (MHI) are addressed and a new motion history histograms (MHH) feature is introduced to represent the motion information in the video. MHH not only provides rich motion information, but also remains computationally inexpensive. We combine MHI and MHH into a low-dimensional feature vector for the system and achieve improved performance in human action recognition over comparable methods that use tracking-free temporal template motion representations. Finally, a simple system based on SVM and MHI has been implemented on a reconfigurable embedded computer vision architecture for real-time gesture recognition.

7.1 Introduction

Visual recognition of different classes of motion within the context of embedded computer vision systems has wide-ranging applications. Examples include intelligent surveillance of human and road traffic activity, biometric security, such as gait recognition, and visually driven interaction and context awareness in "smart" environments, both of which are related to the application areas of "ambient intelligence" and "ubiquitous computing."

Hongying Meng
University of Lincoln, Lincoln, UK, e-mail: hmeng@lincoln.ac.uk

Nick Pears, Michael Freeman, Chris Bailey
University of York, York, UK, e-mail: {nep, mjf, chrisb}@cs.york.ac.uk

The work presented here focuses on the use of video for classifying general human motions, with a view to deploying our system in a smart home environment and using it to recognize gestural commands. In particular, our methods are designed to be appropriate for deployment in a real-time, embedded context. In this sense, we have developed compact, descriptive motion representations and low complexity classification algorithms, all of which may be implemented on our flexible standalone video processing architecture, which is based upon field-programmable gate arrays (FPGAs).

Aggarwal and Cai [1] present an excellent overview of human motion analysis. Of the appearance based methods, template matching has gained increasing interest recently [2, 6, 8, 12, 14, 15, 20, 21, 22, 24, 26, 27, 28, 30]. These methods are based on the extraction of a 2D or 3D shape model directly from the images, to be classified (or matched) against training data. Motion-based models do not rely on static models of the person, but on human motion characteristics. Motion feature extraction is the key component in these kinds of human action recognition systems.

In this chapter, we build a compact human action recognition system based on a linear support vector machine (SVM) [5, 25] classifier. We address the limitations of the motion history image (MHI) [3] and introduce a new feature, which we call the motion history histograms (MHH) [16]. This representation retains more motion information than MHI, but also remains inexpensive to compute. We extract a compact feature vector from the MHH and then combine it with the histogram of the MHI feature in our human action recognition system and get very good performance.

We have started to implement our systems within an FPGA-based embedded computer vision architecture, which we call "Videoware," although in our current implementation, we use MHI features only and embedded implementation of our new MHH feature is ongoing.

The rest of this chapter is organized as follows: In Section 7.2, we give an overview of related work. In Section 7.3, we give a brief introduction of the framework of the SVM based human action recognition system. In Section 7.4, we firstly introduce some fundamental motion features, of which the MHI is the classical example. Furthermore, we give a detailed description of the new MHH feature, which is designed to be more descriptive than MHI features in order to give improved classification performance. In Section 7.5, we discuss the possible feature combination and dimension reduction methods in our framework. In Section 7.6, experimental results derived from a MATLAB implementation of our SVM based human action recognition system are evaluated. In Section 7.7, we give a simple example implementation and evaluation of an MHI/SVM based gesture recognition system on our reconfigurable embedded computer vision architecture, which we call "Videoware." Finally, we present conclusions.

7.2 Related Work

The idea of temporal templates was introduced by Bobick and Davis [3, 19]. They used motion energy images (MEI) and MHI to recognize many types of aerobics exercise. In [4], they also proposed the motion gradient orientation (MGO) to explicitly encode changes in an image introduced by motion events. Davis [7] also presented a useful hierarchical extension for computing a local motion field from the original MHI representation. The MHI was transformed into an image pyramid, permitting efficient fixed-size gradient masks to be convolved at all levels of the pyramid, thus extracting motion information at a wide range of speeds. The hierarchical MHI approach remains a computationally inexpensive algorithm to represent, characterize, and recognize human motion in video.

Schuldt et al. [24] proposed a method for recognizing complex motion patterns based on local space-time features in video and they integrated such representations with SVM classification schemes for recognition. The work of Efros et al. [9] focuses on the case of low resolution video of human behaviors, targeting what they refer to as the 30 pixel man. In this setting, they propose a spatio-temporal descriptor based on optical flow measurements, and apply it to recognize actions in ballet, tennis and football datasets.

Weinland et al. [26] introduced motion history volumes (MHV) as a free-viewpoint representation for human actions in the case of multiple calibrated and background-subtracted video. They presented algorithms for computing, aligning, and comparing MHVs of different actions performed by different people from a variety of viewpoints. Ke et al. [12] studied the use of volumetric features as an alternative to the local descriptor approaches for event detection in video sequences. They generalized the notion of 2D box features to 3D spatio-temporal volumetric features. They constructed a real-time event detector for each action of interest by learning a cascade of filters based on volumetric features that efficiently scanned video sequences in space and time. Ogata et al. [21] proposed modified motion history images (MMHI) and used an eigenspace technique to realize high-speed recognition of six human motions. Wong and Cipolla [27] proposed a new method to recognize primitive movements based on MGO extraction and, later, used it for continuous gesture recognition [28].

Recently, Dalal et al. [6] proposed histogram of oriented gradient (HOG) appearance descriptors for image sequences and developed a detector for standing and moving people in video. Dollár et al. [8] proposed a similar method where they use a new spatio-temporal interest point detector to obtain a global measurement instead of the local features in [9]. Niebles et al. [20] also use spatial-time interest points to extract spatial-temporal words as their features. Yeo et al. [30] estimate motion vectors from optical flow and calculate frame-to-frame motion similarity to analyze human action in video. Blank et al. [2] regarded human actions as three dimensional shapes induced by silhouettes in space-time volume. They adopted an approach for analyzing 2D shapes and generalized it to deal the idea with volumetric space-time action shapes. Oikonomopoulos et al. [22] introduced a sparse representation of

image sequences as a collection of spatio-temporal events that were localized at points that were salient both in space and time for human action recognition.

We note that, in some of these methods, the motion features employed are relatively complex [2, 6, 8, 9, 12, 20, 22, 24, 26, 30], which implies significant computational cost when building the features. Some of them require segmentation, tracking or other prohibitive computational cost processes [2, 3, 4, 7, 21, 27, 28], which currently makes them not suitable for real-time embedded vision applications. In our work, we aim for a solution which uses compact representations, is fast to compute, and yet gives an improved classification performance over existing compact and fast methods.

7.3 SVM-Based Human Action Recognition System

In our system, we have employed a linear SVM classifier [5], for two main reasons: (i) low complexity classification and hence suitable for real-time embedded applications, (ii) very good performance in many real-world classification problems.

The schematic of our SVM based human action recognition system is shown in Fig. 7.1, where the training path is given by the solid arrows and the testing path is given by the dotted arrows. It is composed of four parts: source (data), motion features, dimension reduction, and learning. The motion features can be MHI, MMHI, MGO, and our new feature which we call motion history histograms (MHH).

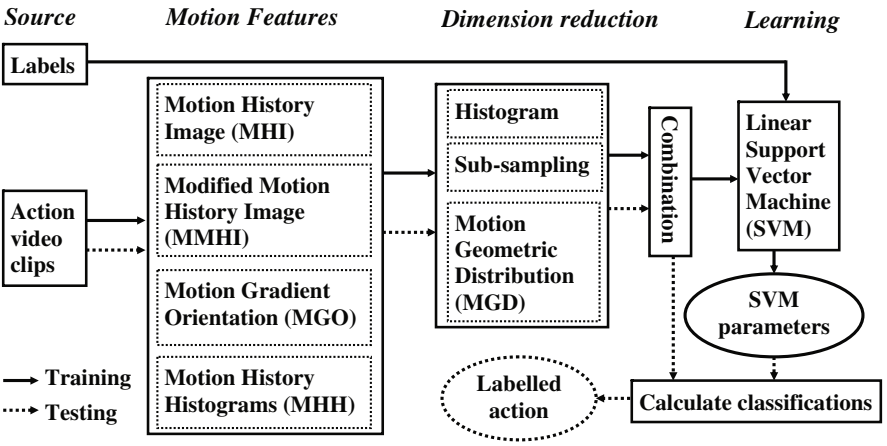


Fig. 7.1 SVM based human action recognition system. Compact motion features are extracted from human action video clips without corner detection, tracking or segmentation. These feature vectors are compressed by dimension reduction methods. Then they are efficiently combined into the linear SVM classifier. The parameters of the SVMs obtained from training are used in the classification process.

In the training part of this system, combined motion feature vectors, extracted from fundamental motion features, are used for training SVM classifiers. The parameters computed are then used in the recognition part. Note that this diagram represents an architecture (rather than a specific implementation) in which any subset of motion features may be used and possibly combined in a specific implementation. This flexibility exists to deal with limitations in the specific embedded hardware available, such as FPGA gate count, memory, processing speed, data communication capability and so on.

Although the SVM performs well with very high dimensional feature vectors, we reduce the dimension of the feature vector to aid embedded deployment of our algorithm. For this, we use simple algorithms, which are easily implemented on our FPGA architecture, such as down-sampling or block averaging operations.

The training of the SVM classifier is done off-line using video data, also collected off-line. After that, the parameters computed for the classifier are embedded in our FPGA-based architecture.

In the following sections, we will give detailed information on this system.

7.4 Motion Features

In order to generate compact, descriptive representations of motion which are simple to extract, several techniques have been proposed to compact the whole motion sequence into a single image. The most popular of such "temporal template" motion features are the motion history image (MHI), the modified motion history image (MMHI), and the motion gradient orientation (MGO). Here, we give a brief introduction to the these features.

7.4.1 Temporal Template Motion Features

A motion history image (MHI) [3] is the weighted sum of past images and the weights decay back through time. Therefore, an MHI image contains the past images within itself, where the most recent image is brighter than the earlier ones. Normally, an MHI $H_\tau(u,v,k)$ at time k and location (u,v) is defined by

$$H_\tau(u,v,k) = \begin{cases} \tau, & D(u,v,k) = 1 \\ \max\{0, H_\tau(u,v,k-1) - 1\}, & \text{otherwise} \end{cases} \tag{7.1}$$

where the motion mask $D(u,v,k)$ is a binary image obtained from subtraction of frames, and τ is the maximum duration a motion is stored. In general, τ is chosen as the constant 255, allowing the MHI to be easily represented as a grayscale image with one byte depth. Thus an MHI pixel can have a range of values, whereas a motion energy image (MEI) is its binary version, which can easily be computed by thresholding $H_\tau > 0$.

Ogata et al. [21] use a multivalued differential image to extract information about human posture because differential images encode human posture information more than a binary image, such as a silhouette image. They called the feature MMHI.

The MGO feature was proposed by Bradski and Davis [4] to explicitly encode changes in an image introduced by motion events. The MGO is computed from an MHI and a MEI. While an MHI encodes how the motion occurred, an MEI encodes where the motion occurred, the MGO, therefore, is a concatenated representation of motion (where and how it occurred).

We have tested the performance of these three features on our SVM based human action recognition system and found that the MHI had the best classification performance of 63.5% on a large challenging dataset [15]. This overall performance is far from good enough. In the following, we will look at the MHI feature further in order to find a way to improve it.

7.4.2 Limitations of the MHI

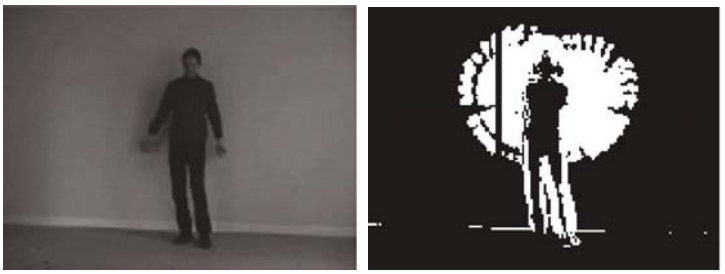
An example of an MHI is shown in Fig. 7.2, where (a) is one frame from the original hand waving action video clip and (b) is the MHI of this action.

In order to have a detailed look at the MHI, we have selected the pixels on the vertical line in the MHI of Fig. 7.2 (b). If some action happened at frame k on pixel (u,v), then $D(u,v,k) = 1$, otherwise $D(u,v,k) = 0$. The locations of these pixels are $(60,11),(60,12),\ldots,(60,80)$. For a pixel (u,v), the motion mask $D(u,v,:)$ of this pixel is the binary sequence:

$$D(u,v,:) = (b_1,b_2,\ldots,b_N), \quad b_i \in \{0,1\} \tag{7.2}$$

where $N+1$ is the total number of frames.

All of the motion masks on the vertical line in Fig. 7.2 (b) are shown in Fig. 7.3. Each row is $D(u,v,:)$ for one fixed pixel (u,v) and a white block represents '1' and



(a) Handwaving sample (b) MHI of Handwaving

Fig. 7.2 Example of an MHI. Part (a) is one frame from the original hand waving action video clip and (b) is the MHI of this action. The vertical line in (b) has the pixels from $(60,11)$ to $(60,80)$.

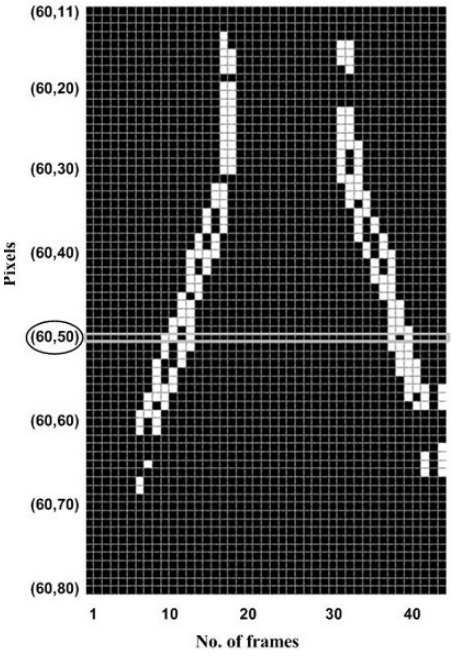


Fig. 7.3 $D(:,:,:)$ on the vertical line of Fig. 7.2(b) is shown. Each row is $D(u,v,:)$ for one fixed pixel (u,v). A white block represents '1' and a black block '0'. For example, $D(60,50,:)$ is the "binarized frame difference history" or "motion mask" of pixel $(60,50)$ through time.

black block represents '0' in the sequences. The motion mark $D(60,50,:)$ has the following sequence:

$$0000000001101000000000000000000000001010000 \tag{7.3}$$

From the definition of MHI in Eq. (7.1) it can be observed that, for each pixel (u,v), MHI actually retains the time since the last action occurred. That is, only the last '1' in the Sequence (7.3) is retained in the MHI at pixel $(60,50)$. It is clear that previous '1's in the sequence, when some action occurred, are not represented. It is also clear that almost all the pixels have more than one '1' in their sequence.

7.4.3 Definition of MHH

The above limitation of the MHI has motivated us to design a new representation (the MHH) in which all of the information in the sequence is used and, yet, it remains compact and simple to use.

We define the patterns P_i in the $D(u,v,:)$ sequences, based on the number of connected '1's:

$$\begin{aligned} P_1 &= 010 \\ P_2 &= 0110 \\ P_3 &= 01110 \\ &\vdots \\ P_M &= 0\underbrace{1\ldots1}_{M}0 \end{aligned} \tag{7.4}$$

We denote a subsequence $C_{I,k}$ by Eq. (7.5), where I and k are the indexes of starting and ending frames, and denote the set of all subsequences of $D(u,v,:)$ as $A\{D(u,v,:)\}$. Then, for each pixel (u,v), we can count the number of occurrences of each specific pattern P_i in the sequence $D(u,v,:)$, as shown in Eq. 7.6, where χ is the indicator function.

$$C_{I,k} = b_I, b_{I+1}, \ldots, b_k, \quad (1 \le I < k \le N) \tag{7.5}$$

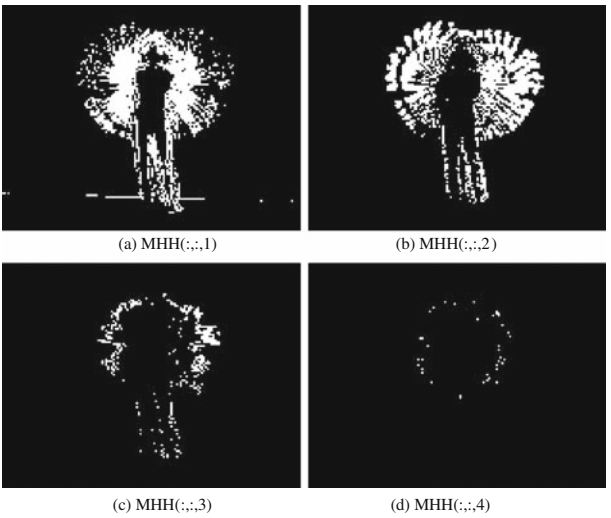
$$\begin{gathered} \mathrm{MHH}(u,v,i) = \textstyle\sum_{(I,k)} \chi_{\{C_{I,k}=P_i | C_{I,k} \in A\{D(u,v,:)\}\}} \\ (1 \le I < k \le N,\ 1 \le i \le M) \end{gathered} \tag{7.6}$$

From each pattern P_i, we can build a grayscale image and we call this its histogram, since the bin value records the number of this pattern type. With all the patterns P_i, $(i = 1, ..., M)$ together, we collectively call them motion history histograms (MHH) representation.

For a pattern P_i, $\mathrm{MHH}(:,:,i)$ can be displayed as an image. In Fig. 7.4, four patterns P_1, P_2, P_3, and P_4 are shown, which were generated from the hand waving action in Fig. 7.2. By comparing the MHH in Fig. 7.4 with the MHI in Fig. 7.2, it is interesting to find that the MHH decomposes the MHI into different parts based on patterns. Unlike the hierarchical MHI described by Davis [7], where only small size MHIs were obtained, MHH records the rich spatial information of an action.

The choice of the number M depends on the video clips. In general, the bigger the M is, the better the motion information will be. However, the values within the MHH rapidly approach zero as M increases. In our experiment, no more than half of the training data had the sixth pattern P_6 and so we chose $M = 5$. Furthermore we note that a large M will increase the storage requirement for our hardware based system.

The computation of MHH is inexpensive and can be implemented by the procedure in Fig. 7.5. $D(u,v,k)$ is the binary sequence on pixel (u,v) that is computed by thresholding the differences between frame k and frame $k-1$. $I(u,v)$ is a frame index that stands for the number of the starting frame of a new pattern on pixel (u,v). At the beginning, $I(u,v) = 1$ for all (u,v). That means a new pattern starts from frame 1 for every pixel. $I(u,v)$ will be updated to $I(u,v) = k$ while $\{D(u,v,I(u,v)), ..., D(u,v,k)\}$ builds one of the patterns $P_i\,(1 \le i \le M)$ and, in this case, $\mathrm{MHH}(u,v,i)$ increases by 1.



(a) MHH(:,:,1) (b) MHH(:,:,2)

(c) MHH(:,:,3) (d) MHH(:,:,4)

Fig. 7.4 MHH example. Four patterns P_1, P_2, P_3, and P_4 were selected. This results were generated from the handwaving action in Fig. 7.2. Each pattern P_i, MHH$(:,:,i)$ has the same size as the original frame.

Algorithm (MHH)

Input: Video clip $f(u,v,k)$, $u=1,...,U$, $v=1,...,V$, frame $k=0,1,...,N$
Initialization: Pattern M, MHH$(1:U,1:V,1:M)=0$, $I(1:U,1:V)=1$
For $k=1$ to N (For 1)
  Compute: $D(:,:,k)$
  For $u=1$ to U (For 2)
    For $v=1$ to V (For 3)
      If Subsequence $C_j=\{D(u,v,I(u,v)),\ldots,D(u,v,k)\}=P_i$
        Update: MHH(u,v,P_i)=MHH$(u,v,P_i)+1$
      End *If*
      Update: $I(u,v)$
    End (For 3)
  End (For 2)
End (For 1)
Output: MHH$(1:U,1:V,1:M)$

Fig. 7.5 Procedure of MHH algorithm.

7.4.4 Binary Version of MHH

Recall that the MEI is a binary version of the MHI. Similarly, we can define the binary version of an MHH. To do this, we first define the binary version of an MHH as MHH_b , as

$$\mathrm{MHH}_b(u,v,i) = \begin{cases} 1, & \mathrm{MHH}(u,v,i) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{7.7}$$

7.5 Dimension Reduction and Feature Combination

Referring back to Fig. 7.1, once we have extracted one or more suitable motion features, we use several techniques to reduce the dimension of the data. These are described in the following subsections.

7.5.1 Histogram of MHI

The histogram is a property of an image used widely in image analysis. For example, for a grayscale image, it shows the frequency of particular grayscale values within the image. Note that MHIs can be rendered as grayscale images, where a value of a pixel in the MHI records time information, namely when some motion most recently occurred at this particular pixel location. Thus the histogram of MHI represents the intensity of motion history. Other features, such as MMHI and MGO, do not offer this property, while the MHH itself is already a histogram.

7.5.2 Subsampling

Subsampling (or downsampling) is the process of reducing the sampling rate of a signal. This is usually done to reduce the data rate or the size of the data. Images typically have a large data size and so subsampling is a general method often used to reduce data size. Subsampling can be done by selecting odd or even rows and columns. Wavelet transforms or other filters are often used to extract the low frequency components of the image to get a compact image on larger scales. In this work, we use subsampling to reduce computational complexity. This can be applied for all the motion features described here, such as MHI, MMHI, MGO, and MHH.

7.5.3 Motion Geometric Distribution (MGD)

The size of the MHH_b representation can be rather large for some embedded implementations and also we seek a more compact representation, which captures the geometric distribution of the motion across the image. Thus we sum each row of MHH_b (for a given pattern, P_i) to give a vector of size V rows. We obtain another

vector by summing columns to give a vector of size U rows. Thus using all M levels in the binarized MHH hierarchy, we obtain a motion geometric distribution (MGD) vector of size $M\times(U+V)$, which is relatively compact, when compared to the size of the original MHH and MHI features. The MGD vector can thus be represented by Eq. (7.8):

$$\begin{gathered}\mathrm{MGD}=\{\textstyle\sum_u \mathrm{MHH}_b(u,v,i),\sum_v \mathrm{MHH}_b(u,v,i)\}\\(i=1,2,\ldots,M)\end{gathered} \tag{7.8}$$

In our work, we prefer to compute the MGD by using the MHH_b feature instead of the MHH feature directly. From our experiments, it has been found that the values within the MHH decrease significantly for the large patterns. The values for P_4 and P_5, for example, are much smaller than those of P_1, P_2 and P_3. Thus, if we use the MHH directly to compute the MGD, a normalization process is necessary in order to treat all the patterns equally. However, this normalization process is not an easy task for our hardware implementation because of limited memory and the requirement to implement a floating-point processing ability. In contrast, computation of the MGD from the MHH_b feature does not need a normalization process and yet we retain a satisfactory performance.

7.5.4 Combining Features

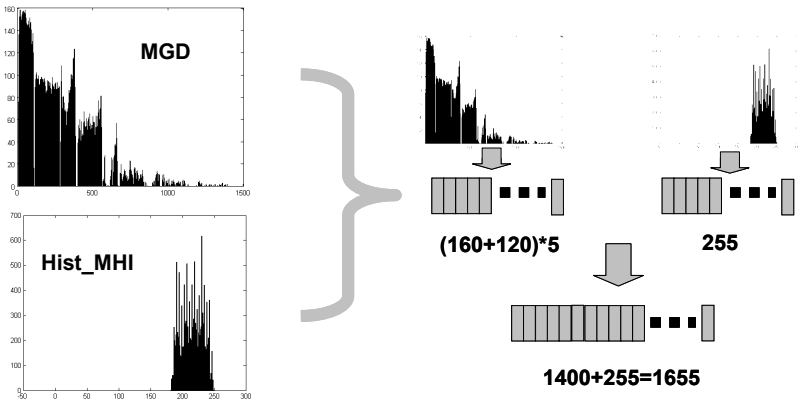


Fig. 7.6 Combination between MGD of the MHH and histogram of the MHI from a same video example. The frame has the size of 160×120. MGD of MHH and histogram of MHI have the size of $(160+120)\times5=1400$ and 255, respectively.

We want to efficiently use the motion features extracted in order to achieve an improved classification performance, relative to other compact systems. Based on the simplicity requirement of the system, our two feature vectors are combined in the

simplest way by concatenating these two feature vectors into a higher dimensional vector. Fig. 7.6 shows an example of a combination between the MGD of the MHH and the histogram of the MHI from the same video.

7.6 System Evaluation

In this section, we present the experimental results derived from a MATLAB implementation of our SVM based human action recognition system.

7.6.1 Experimental Setup

For the evaluation of our system, we use a challenging human action recognition database, recorded by Christian Schuldt [24], which is both large and publicly available. It contains six types of human actions (walking, jogging, running, boxing, hand waving, and hand clapping) performed several times by 25 subjects in four different scenarios: outdoors (s1), outdoors with scale variation (s2), outdoors with different clothes (s3), and indoors (s4).

This database contains 2391 sequences. All sequences were taken over homogeneous backgrounds with a static camera with 25 Hz frame rate. The sequences were downsampled to the spatial resolution of 160×120 pixels and have a time length of 4 seconds on average. To the best of our knowledge, this is the largest video database with sequences of human actions taken over different scenarios. All sequences were divided with respect to the subjects into a training set (8 persons), a validation set (8 persons), and a test set (9 persons).

In our experiment, the classifiers were trained on a training set while classification results were obtained on the test set. In all our experiments, the same parameters were used. The threshold in frame differencing was chosen as 25 and τ was chosen as 255 for MHI construction. The most suitable choice of the number of patterns M for MHH computation depends on the video clips and is a trade-off between the compactness of the representation and the expressiveness of the representation. Building a frequency histogram of the patterns extracted from the training clips indicates that no more than half of the training data had the sixth pattern. Thus the number of patterns was chosen to be $M = 5$.

The size of the MHI is $160 \times 120 = 19,200$, which is the same width as that of the frames in the videos. In our experiment, the SVM is implemented using the SVM^{light} software [11]. In SVM training, choosing a good parameter C value is not so straightforward and can significantly affect classification accuracy [10], but in order to keep our system simple, the default value of C in SVM^{light} is used in all of the experiments.

Fig. 7.7 shows examples in each type of human action in this dataset. In order to compare our results with those as [12] and [24], we use the exact same training

Fig. 7.7 Six types of human action in the database: (a) walking (b) jogging (c) running (d) boxing (e) hand-clapping (f) hand-waving.

set and testing set in our experiments. The only difference is that we did not use the validation dataset in training. Our experiments are carried out on all four different scenarios. In the same manner as [12], each sequence is treated individually during the training and classification process. In all of the following experiments, the parameters are kept same.

7.6.2 Performance of Single Features

We have tested the performance of the fundamental motion features MHI, MMHI and MGO in our system. Fig. 7.8 shows these three motion features extracted from the action examples shown in Fig. 7.7. In order to keep our system simple for hardware implementation, we use the simplest method to transform the motion features (MHI, MMHI and MGO) into a plain vector based on the pixel scan order (row by row) to feed SVM classifier.

Firstly, we tested the system performance on the four different subsets of the whole dataset. The results can be seen in Fig. 7.9. The correctly classified percentage on these data subsets indicates how many percent of the action clips in the testing set were correctly recognized by the system. It is clear that the MHI feature gave the best classification performance in all four subsets while the MGO feature gave poor results for all four data subsets. We also can see that subset s2 (outdoors with scale variation) is the most difficult subset in the whole dataset.

From the experiments, it can be seen that this type of system can get reasonable results. The MHI based system looks better than the MMHI system in the experiments. The disadvantage for MMHI is that it can only work well in the case

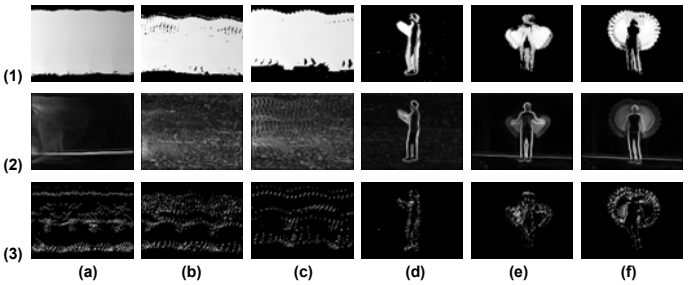


Fig. 7.8 The (1) MHI, (2) MMHI and (3) MGO for the six actions in the dataset: (a) walking (b) jogging (c) running (d) boxing (e) hand-clapping (f) hand-waving

of an uncluttered and static background. If there is background motion or noise, this will be recorded in the feature vector and will reduce the performance of the classification.

For the whole dataset, the classification confusion matrix is a good measure for the overall performance in this multiclass classification problem. Table 7.1 shows the classification confusion matrix based on the method proposed as [12]. Table 7.2 shows the confusion matrix obtained by our system based on MHI. The confusion matrices show the motion label (vertical) versus the classification results (horizontal). Each cell (i, j) in the table shows the percentage of class i action being recognized as class j. Thus the main diagonal of the matrices show the percentage of correctly recognized actions, while the remaining cells show the percentages of misclassification. The trace of the matrix shows the overall classification rate. In Table 7.1, the trace is 377.8 and since there are six classes, the overall mean classification rate is $377.8/6 = 63\%$.

In comparison with Ke's method, we use a simple MHI feature rather than large volumetric features in which the dimension of a feature vector might be a billion, yet the performance of our system is marginally better on this dataset.

In the second step, we test some low dimensional features based on the fundamental motion features. Subsampling is easy to implement in hardware by any factor of 2 and this can be done in both rows and columns of the motion feature.

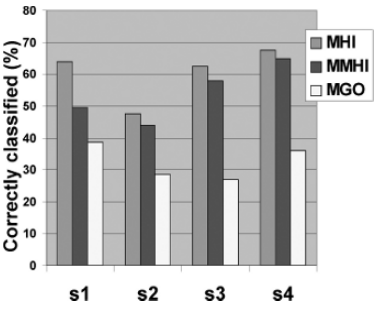


Fig. 7.9 Correctly classified percentage for separate data subset: s1 (outdoors), s2 (outdoors with scale variation), s3 (outdoors with different clothes) and s4 (indoors).

Table 7.1 Ke's confusion matrix [12], trace = 377.8, mean performance = 63%.

	Walk	Jog	Run	Box	Clap	Wave
Walk	**80.6**	11.1	8.3	0.0	0.0	0.0
Jog	30.6	**36.2**	33.3	0.0	0.0	0.0
Run	2.8	25.0	**44.4**	0.0	27.8	0.0
Box	0.0	2.8	11.1	**69.4**	11.1	5.6
Clap	0.0	0.0	5.6	36.1	**55.6**	2.8
Wave	0.0	5.6	0.0	2.8	0.0	**91.7**

Table 7.2 MHI's confusion matrix, trace = 381.2, mean performance = 63.5%.

	Walk	Jog	Run	Box	Clap	Wave
Walk	**53.5**	27.1	16.7	0.0	0.0	2.8
Jog	46.5	**34.7**	16.7	0.7	0.0	1.4
Run	34.7	28.5	**36.1**	0.0	0.0	0.7
Box	0.0	0.0	0.0	**88.8**	2.8	8.4
Clap	0.0	0.0	0.0	7.6	**87.5**	4.9
Wave	0.0	0.0	0.0	8.3	11.1	**80.6**

Tables 7.3 and 7.4 show the results based on downsampling by a factor of 64 (a factor of 8 for both row and column) and the histogram of MHI. From the experiments, we find that this dimensional reduction is detrimental for the MHI. Also, it can be seen that subsampling of MHI obtains a similar performance to Ke's method. This feature performed well in distinguishing the last three groups. On the other hand, the histogram of MHI did not perform well in terms of overall performance but has the power to distinguish the first three groups, which demonstrates that the two methods encode different information.

Table 7.3 MHI_S's confusion matrix, trace = 377.7, mean performance = 62.95%.

	Walk	Jog	Run	Box	Clap	Wave
Walk	**56.9**	18.1	22.2	0.0	0.0	2.8
Jog	45.1	**29.9**	22.9	1.4	0.0	0.7
Run	34.7	27.8	**36.1**	0.0	0.0	1.4
Box	0.0	0.0	0.0	**89.5**	2.1	8.4
Clap	0.0	0.0	0.0	5.6	**88.9**	5.6
Wave	0.0	0.0	0.0	12.5	11.1	**76.4**

Fig. 7.10 shows examples in each type of human action and their associated MHI and MHH motion features. For the MHH, it is hard to deal with the whole feature in our hardware system as, with the number of patterns set to 5, the MHH has a

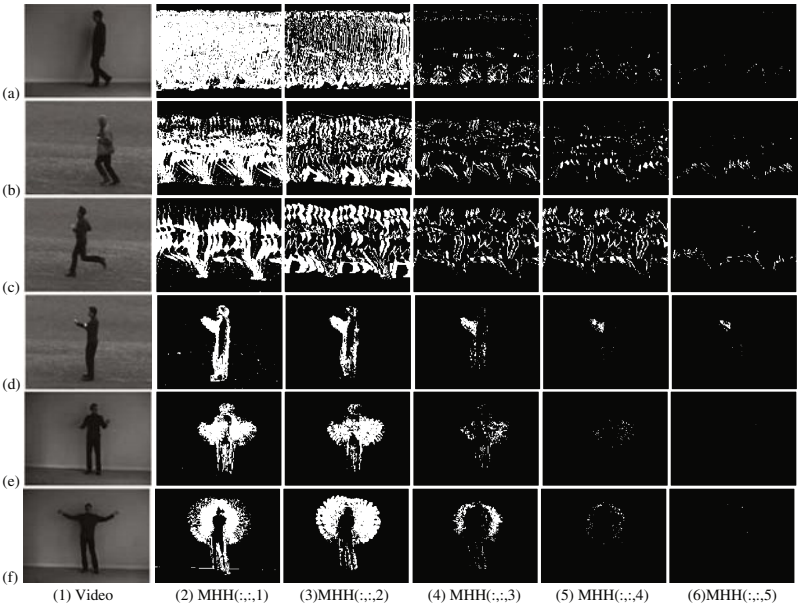


Fig. 7.10 The six database human actions and associated MHH features: (a) walking (b) jogging (c) running (d) boxing (e) handclapping (f) hand-waving.

Table 7.4 Hist. of MHI's confusion matrix, trace = 328.6, mean performance = 54.8%

	Walk	Jog	Run	Box	Clap	Wave
Walk	**62.5**	32.6	0.0	1.4	1.4	2.1
Jog	12.5	**58.3**	25.0	0.0	0.0	4.2
Run	0.7	18.8	**77.1**	0.0	0.0	3.5
Box	4.9	2.8	0.7	**17.5**	61.5	12.6
Clap	4.9	2.1	0.7	11.1	**75.0**	6.3
Wave	5.6	3.5	6.9	20.1	25.7	**38.2**

relatively high dimension of $5 \times 160 \times 120 = 96000$. Thus, we constructed a small sized MHH_s by averaging the pixels in an 8×8 block, so that the size of all MHH feature vectors is reduced to $20 \times 15 \times 5 = 1500$. Our MGD feature also has a small size of $(160 + 120) \times 5 = 1400$.

Table 7.5 and Table 7.6 show the results when using features MHH_s and MGD respectively. From these two tables, it is very clear that both MHH_s and MGD improve the overall performance. But they failed to classify the "jogging" class. The reason is that these video clips are quite similar to "walking" and "running." It is hard to distinguish between them correctly even by human observation.

Table 7.5 MHH_s's confusion matrix, trace = 417.3, mean performance = 69.55%.

	Walk	Jog	Run	Box	Clap	Wave
Walk	**88.9**	1.4	6.3	0.7	1.4	1.4
Jog	56.9	**2.1**	38.2	0.7	2.1	0.0
Run	22.2	0.7	**75.7**	0.0	1.4	0.0
Box	0.0	0.0	0.0	**96.5**	0.7	2.8
Clap	0.0	0.0	0.0	4.2	**93.1**	2.8
Wave	0.0	0.0	0.0	22.2	16.7	**61.1**

Table 7.6 MGD's confusion matrix, trace = 432.6, mean performance = 72.1%.

	Walk	Jog	Run	Box	Clap	Wave
Walk	**85.4**	4.9	2.8	2.8	2.8	1.4
Jog	65.3	**9.2**	23.6	2.1	0.0	0.0
Run	18.8	8.3	**68.8**	1.4	0.0	2.8
Box	0.0	0.0	0.0	**91.6**	2.8	5.6
Clap	1.4	0.0	0.0	6.3	**92.4**	0.0
Wave	0.0	0.0	0.0	7.6	6.9	**85.4**

7.6.3 Performance of Combined Features

In the previous subsection, we found that different features had different power in distinguishing classes of action. In order to overcome their own disadvantages, we combine them in the feature space. Table 7.7 shows the confusion matrix obtained from our system when combined features were used. From this table, we can see that the overall performance has a significant improvement over Ke's method, which is based on volumetric features. Note that good performance is achieved in distinguishing all of the six actions in the dataset.

Table 7.7 MGD & Hist. of MHI's confusion matrix, trace = 481.9, mean performance = 80.3%.

	Walk	Jog	Run	Box	Clap	Wave
Walk	**66.0**	31.3	0.0	0.0	2.1	0.7
Jog	13.9	**62.5**	21.5	1.4	0.0	0.7
Run	2.1	16.7	**79.9**	0.0	0.0	1.4
Box	0.0	0.0	0.0	**88.8**	2.8	8.4
Clap	0.0	0.0	0.0	3.5	**93.1**	3.5
Wave	0.0	0.0	0.0	1.4	6.9	**91.7**

We compared our results with other methods on this challenging dataset and summarize the correctly classified rates in Table 7.8. From this table, we can see

that MHH has made a significant improvement in comparison with MHI. Furthermore, the MGD feature gives a better performance than the MHH itself. The best performance, which gives significantly better classification results, came from the combined feature, which is based on the histogram of the MHI and the MGD.

Table 7.8 Overall correctly classified rate (%) for all the methods on this open, challenging dataset. Some of them did not use the difficult part of dataset(Δ), while some of them did an easier task($*$).

Method	Rate(%)
SVM on local features [24]$*$	71.7
Cascade of filters on volumetric features [12]	63
SVM on MHI [15]	63.5
SVM_2K on MHI & MMHI [14]	65.3
SVM on MHH_s	69.6
SVM on MGD	72.1
SVM on HWT of MHI & Hist. of MHI [17]	70.9
SVM on MGD & Hist. of MHI	**80.3**
SVM on spatio-temporal feature [8]Δ	81.2
Unsupervised learning on spatial-temporal words [20] $*$	81.5
KNN on nonzero motion block similarity [30]$\Delta*$	86.0

It should be mentioned here that some results [8, 20, 30] are better than ours on this dataset. However, these results are not directly comparable with ours. For example, Dollar et al. [8] achieved a correct classification rate of 81.2%, but the authors omitted the most difficult part of the dataset (subset 2, outdoor with scale variation).

Niebles et al. [20] obtained similar results with 81.5% and Yeo et al. [30] obtained 86.0%, but they did an easier task of classifying each complete sequence (containing four repetitions of same action) into one of six classes, while our method was trained as the same way as [9, 12, 14, 15, 17]; that is, to detect a single instance of each action within arbitrary sequences in the dataset. Furthermore, Yeo et al. [30] did not use the difficult subset 2 of the dataset, as was the case with Dollar et al. [8].

7.7 FPGA Implementation on Videoware

We have developed a hardware architecture called "Videoware" [23], which can be reconfigured for a wide range of embedded computer vision tasks. At present, we have not tested our MHH representations within our embedded "Videoware" architecture, but we did test the performance of an MHI/SVM based gesture recognition in an embedded context [18].

Our approach has been to implement a video component library (VCL) of generic image processing, computer vision and pattern recognition algorithms in an FPGA

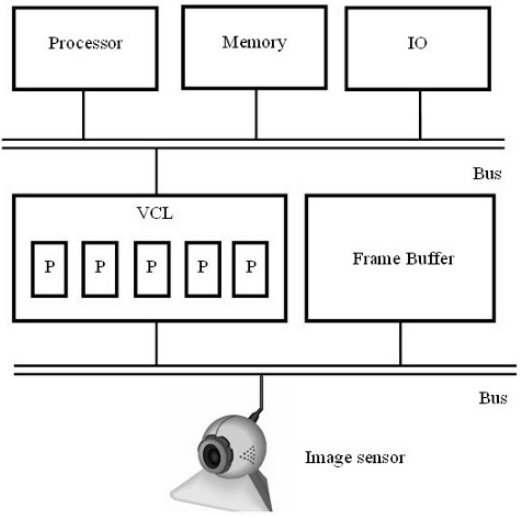


Fig. 7.11 Videoware processing architecture.

based architecture as shown in Fig. 7.11. The low level, high bandwidth processes, such as smoothing and feature extraction, are implemented as hardware IP-cores, whilst higher level, lower bandwidth processes, such as task-oriented combination of visual cues, are implemented in a software architecture as shown schematically in Fig. 7.12. The advantage of this modular approach is that a systems processing performance can be reconfigured for a particular application, with the addition of new or replicated processing cores.

"Videoware" has been implemented on a custom made FPGA board as shown in Fig. 7.13. This board is based on a Xilinx Spartan-III device [29], with 2 MB of external RAM and 8 MB of external ROM (this memory is also used to configure the FPGA via a configuration engine). The FPGA size can be selected to match a system' s requirements, the board accepting three alternative devices: XC3S1500 (1.5M gates), XC3S2000 (2 M gates) and XC3S4000 (4 M gates). In addition to this a number of interface boards have also been developed to allow the easy connection of a camera [13], communications interfaces (e.g., LEDs, RS232), and additional external memory modules.

The action recognition processing pipeline that we have implemented is shown in Fig. 7.14. A difference operator is performed on the current and previous frames, updating a motion history image. The inner product of the MHI and the SVM classification data sets is then performed, the result of each accumulator then has a specific offset applied before a threshold is performed, selecting the stored action that most closely matches the observed motion. In the current implementation this process is operated in a one shot mode, however, this could be easily expanded to include motion detection to start and stop this process, i.e., when the difference between two frames exceeds a threshold the MHI is generated, when it falls below this threshold the inner product and threshold operations are then performed.

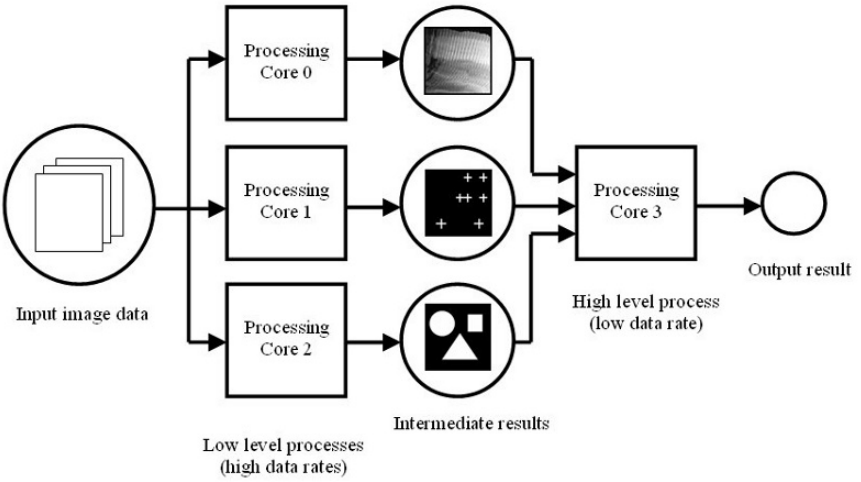


Fig. 7.12 Video component library configured to form a virtual processing pipeline.

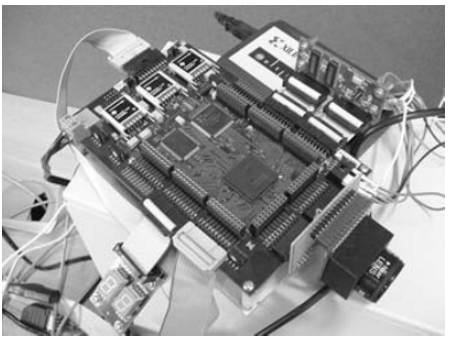


Fig. 7.13 Amadeus ubiquitous system environment (USE) board.

The current hardware implementation uses a 20 MHz system clock and can capture and process 100×80 image data at 12.5 frames per second, i.e., one frame every 80 ms. The system is capable of processing 200×160 images, with the addition of extra memory. In order to test the performance of the FPGA implementation of our human action recognition system, we recorded a hand motion dataset. In this dataset, there are only three types of hand motions: horizontal motion, vertical motion, and "other motion." We also recognize a "no-motion" case as an extra class.

For each class, we recorded 20 video samples, with the frame size set to 100×80 pixels. We recorded the video clips with a variety of backgrounds to test the system robustness to this variability. Fig. 7.15 shows some samples in this dataset.

In our experiment, 15 samples were randomly chosen from each class for training and the other 5 were used for testing. We repeated the experiments 10 times. We carried out the training on a PC using SVM^{light} (the default values were used for all the parameters in this software). Firstly, we extracted MHI features from each video clip. Then we trained three binary linear SVM classifiers based on these features

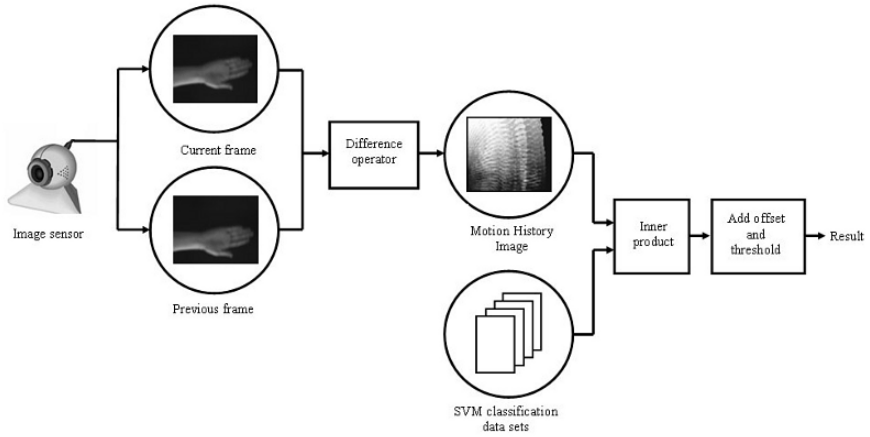


Fig. 7.14 Motion recognition processing pipeline.

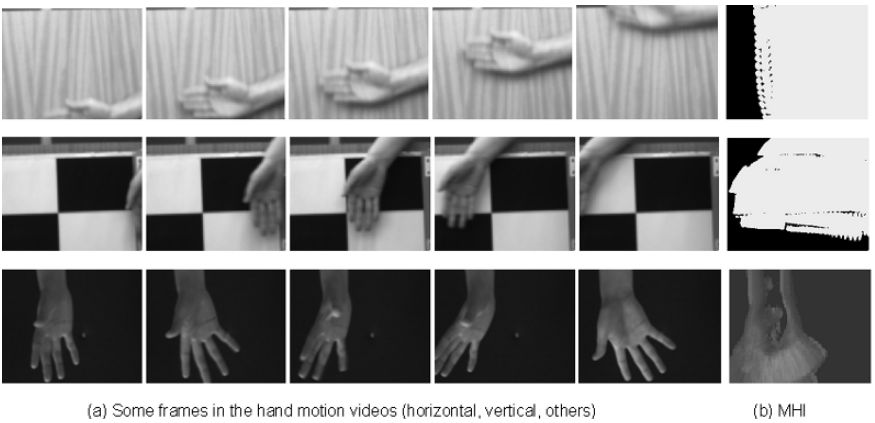


Fig. 7.15 Some samples in the hand motion dataset and their MHI features.

to give a 3 parameter matrix containing the weight vector w and bias b. These parameters were stored in the internal memory of the FPGA chip and were used for gesture classification. During the classification, three values were obtained from each SVM classifier and the one with the largest (most positive) value is used to label the motion.

Table 7.9 shows the average classification rate. The average rate of correct classification for all gestures is 80%, which is almost identical to our PC based (MATLAB) result on the same data.

Table 7.9 Hand motion recognition average confusion matrix

	Horizontal	Vertical	Others
Horizontal	**94**	2	4
Vertical	18	**70**	12
Others	4	18	**76**

7.8 Conclusions

In this chapter, we have proposed a new compact SVM based human action recognition system. It may be applied in security systems, human-computer interaction, and applications within ambient intelligence, where embedded, real-time vision may be deployed. The proposed method does not rely on accurate tracking as many other works do, since most of the tracking algorithms incur an extra computational cost for the system. Our system is based on simple features in order to achieve high-speed recognition in real-world embedded applications.

In order to improve the performance of the system, we have proposed a new representation for motion information in video and this is called the MHH. The representation extends previous work on temporal template (MHI related) representations by additionally storing frequency information as the number of times motion is detected at every pixel, further categorized into the length of each motion. In essence, maintaining the number of contiguous motion frames removes a significant limitation of MHI, which only encodes the time from the last observed motion at every pixel. It can be used either independently or combined with the MHI to give human action recognition systems with improved performance over existing comparable compact systems, which do not employ complex articulated models for tracking.

We extract a basic MGD feature vector from the MHH and apply it in the SVM based human action recognition system. In comparison with local SVM methods by Schuldt [24] and a cascade of filters on volumetric features by Ke [12], our feature vectors are computationally inexpensive. Even though we do not use a validation dataset for parameter tuning in SVM training, we have demonstrated a significant improvement (around 10%) in the recognition performance, when our method is applied to a large, challenging public dataset.

A recognition system using the simple MHI features has been implemented on our FPGA-based embedded computer vision system called "Videoware," with encouraging performance. For the future work, we will implement an improved embedded system, based on combining features from both MHH and MHI, as described in this chapter.

References

1. Aggarwal JK, Cai Q (1999) Human motion analysis: a review. *Comput Vis Image Underst* 73(3):428–440, doi: 10.1006/cviu.1998.0744.
2. Blank M, Gorelick L, Shechtman E, Irani M, Basri R (2005) Actions as space-time shapes. In: *Int. Conf. on Comput. Vis.(ICCV)* pp, 1395–1402.
3. Bobick AF, Davis JW (2001) The recognition of human movement using temporal templates. *IEEE Trans Pattern Anal Mach Intell* 23(3):257–267.
4. Bradski GR, Davis JW (2002) Motion segmentation and pose recognition with motion history gradients. *Mach Vis Appl* 13(3):174–184.
5. Cristianini N, Shawe-Taylor J (2000) *An Introduction to Support Vector Machines (and Other Kernel-Based Learning Methods).* Cambridge University Press, Cambridge, UK.
6. Dalal N, Triggs B, Schmid C (2006) Human detection using oriented histograms of flow and appearance. In: *Euro. Conf. on Comput. Vis.(ECCV) (2),* pp, 428–441.
7. Davis JW (2001) Hierarchical motion history images for recognizing human motion. In: *IEEE Workshop on Detection and Recognition of Events in Video,*pp, 39–46.
8. Dollár P, Rabaud V, Cottrell G, Belongie S (2005) Behavior recognition via sparse spatio-temporal features. In: *VS-PETS,*pp, 65–72, doi: 10.1109/VSPETS.2005.1570899.
9. Efros AA, Berg AC, Mori G, Malik J (2003) Recognizing action at a distance. In: *Int. Conf. on Comput. Vis.(ICCV),*pp, 726–733.
10. Hastie T, Rosset S, Tibshirani R, Zhu J (2004) The entire regularization path for the support vector machine. http://citeseer.ist.psu.edu/hastie04entire.html.
11. Joachims T (1998) Making large-scale support vector machine learning practical. In: B Schölkopf AS C Burges (ed) *Advances in Kernel Methods: Support Vector Machines,*MIT Press, Cambridge, MA, citeseer.ist.psu.edu/joachims98making.html.
12. Ke Y, Sukthankar R, Hebert M (2005) Efficient visual event detection using volumetric features. In: *Int. Conf. on Comput. Vis.(ICCV),*pp, 166–173, beijing, China, Oct. 15-21, 2005.
13. Kodak (2006) Kodak kac-9628 image sensor 648(h) x 488(v) color CMOS image sensor. http://www.kodak.com/ezpres/business/ccd/global/plugins/acrobat/en/productsummary/CMOS/KAC-9628ProductSummaryv2.0.pdf.
14. Meng H, Pears N, Bailey C (2006) Human action classification using SVM_2K classifier on motion features. In:*Lect. Note. Comput. Sci.(LNCS),*Istanbul, Turkey, vol. 4105, pp, 458–465.
15. Meng H, Pears N, Bailey C (2006) Recognizing human actions based on motion information and SVM. In: *2nd IET International Conference on Intelligent Environments,* IET, Athens, Greece, pp, 239–245.
16. Meng H, Pears N, Bailey C (2007) A human action recognition system for embedded computer vision application. In:*Comput. Vis. and Pat. Rec (CVPR),* doi: 10.1109/CVPR.2007.383420.
17. Meng H, Pears N, Bailey C (2007) Motion information combination for fast human action recognition. In: *2nd International Conference on Computer Vision Theory and Applications (VISAPP07), Barcelona, Spain.,*pp, 21–28.
18. Meng H, Freeman M, Pears N, Bailey C (2008) Real-time human action recognition on an embedded, reconfigurable video processing architecture. *J. of Real-Time Image Processing,* doi: 10.1007/s11554-008-0073-1.
19. Moeslund T, Hilton A, Kruger V (2006) A survey of advances in vision-based human motion capture and analysis. *Comput Vis Image Underst* 103(2-3):90–126.
20. Niebles J, Wang H, Fei-Fei L (2006) Unsupervised learning of human action categories using spatial-temporal words. In: *British Machine Vision Conf. (BMVC),*pp, III:1249.
21. Ogata T, Tan JK, Ishikawa S (2006) High-speed human motion recognition based on a motion history image and an eigenspace. *IEICE Trans. on Inform. and Sys.* E89(1):281–289.
22. Oikonomopoulos A, Patras I, Pantic M (2006) Kernel-based recognition of human actions using spatiotemporal salient points. In: *Comput. Vis. and Pat. Rec. (CVPR) workshop 06,* Vol.3, pp, 151–156, http://pubs.doc.ic.ac.uk/Pantic-CVPR06-1/.
23. Pears N (2004) Projects: Videoware - video processing architecture. http://www.cs.york.ac.uk/amadeus/videoware/.

24. Schuldt C, Laptev I, Caputo B (2004) Recognizing human actions: a local SVM approach. In: *Int. Conf. on Pat. Rec (ICPR),* Cambridge, UK.
25. Vapnik V (1995) *The Nature of Statistical Learning Theory.* Springer-Verlag, New York.
26. Weinland D, Ronfard R, Boyer E (2005) Motion history volumes for free viewpoint action recognition. In: *IEEE International Workshop on Modeling People and Human Interaction (PHI'05),*http://perception.inrialpes.fr/Publications/2005/WRB05.
27. Wong SF, Cipolla R (2005) Real-time adaptive hand motion recognition using a sparse Bayesian classifier. In: *Int. Conf. on Comput. Vis.(ICPR) Workshop ICCV-HCI,* pp, 170–179.
28. Wong SF, Cipolla R (2006) Continuous gesture recognition using a sparse Bayesian classifier. In: *Int. Conf. on Pat. Rec (ICPR),* Vol. 1, pp, 1084–1087.
29. Xilinx (2007) Spartan-3 FPGA family complete data sheet. http://direct.xilinx.com/bvdocs /publications/ds099.pdf.
30. Yeo C, Ahammad P, Ramchandran K, Sastry S (2006) Compressed domain real-time action recognition. In: *IEEE International Workshop on Multimedia Signal Processing (MMSP06),* IEEE, Washington, DC.