International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 04 Issue: 08 | Aug -2017 www.irjet.net p-ISSN: 2395-0072

# Human Action Recognition using Contour History Images and Neural Networks Classifier

**[1]Bashar Wannous, [2]Assef Jaafar, [3]Chadi Albitar**

*[1,2,3] Higher Institute for Applied Sciences and Technology*

---

**Abstract -** *In this paper, we propose a new Human Action Recognition algorithm depending on hybrid features extraction from silhouettes and Neural Networks for classification. The hybrid features include contour history images which is a new method, movement tracking of the body's center and the relative of dimensions of the bounding box. For features reduction, three methods are compared, dividing the contour history images to rectangles, a shallow autoencoder and a deep autoencoder. Each of these autoencoders depends basically on Neural Networks. We tested this algorithm using a global human action dataset and the recognition rate was 98.9%, which is the only criterion for system evaluation used till now. That is why we defined a new criterion related to the Time needed to Test a video with a duration of one Second, and we called it TT1S. This criterion is helpful to compare the performances of different systems, and to evaluate the capability of the system in real-time applications.*

***Key Words***: Human Action Recognition, Contour History Images, Silhouettes, Neural Networks, Autoencoder, Deep Autoencoder, real time

## 1. INTRODUCTION

Human action recognition means the recognizing of human behavior such as walking, jumping, running..etc. It has been an active research topic in the last years due to its many applications [1], including security and surveillance, content based video analysis, interactive applications, animation, and behavioral biometrics.

Human action recognition systems can be classified in two branches: the first one is based on wearable sensors and the second depends on one camera or more which is the most important one because it has more applications. The use of multiple cameras is helpful especially for occlusions or 3D modeling, but they need to be synchronized which is not an easy process. Multiple cameras-systems have big amount of data which means that they need more processing time than one camera-systems. Moreover, multiple cameras are useful for short distances only.

With the use of one camera, silhouettes are widely exploited because they contains the amount of information needed to recognize the human posture; besides they are binary images and that reduces both of the size of data needed to be processed and the process time as well.

Many datasets that contain videos for people doing different actions have been proposed. One of the most famous human action datasets is Weizmann dataset, which we have used for the evaluation of our proposed methods.

## 2. PREVIOUS WORKS

Many works have been carried out to recognize human actions from video sequences. All of them can be located in Pattern Recognition field. Different features and different classification methods have been explored.

A human action recognition system that uses contours of silhouettes was suggested in [2] where the authors divided the images into small rectangles and counted the contour points in each one, and HMM was used for classification.

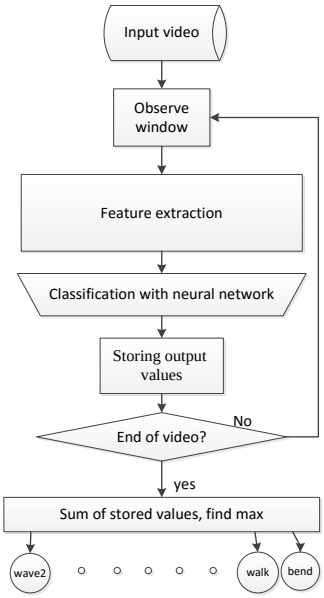


**Fig -1**: The flowchart of the proposed algorithm

A new method that uses Motion History Images was suggested in [3], and Silhouette History Images were used later in [4] with SVM classifier. Each of MHI and SHI depended on gathering sequential silhouettes in one gray image, where every silhouette has different grayscale according to its novelty. In [5], Hybrid features from silhouettes and joints information's were used with embedded Hidden Markov Models. A fast and an easy implementation system was proposed in [6] in which Feature Covariance Matrices were used with nearest neighbor classifier. In [7], a real-time system was proposed using Affine Invariant Fourier Descriptors which are useful when the view direction of the camera changes, or when its distance from the object also changes.

Deep learning refers to a branch of machine learning methodologies in which many layers are used for pattern recognition and features learning. A human action recognition system using a Deep Believe Network was proposed in [8]. A full automated system was designed in [9] using a Convolutional Neural Networks and a Recurrent Neural Network. The results were good, and an Online Deep Learning Algorithm is proposed in [10].

In this paper, a new Algorithm is proposed using a hybrid features depending on Contour History Images (CHIs) and body center movement. CHIs are different from Motion History Images and Silhouette History Images because they are binary images. For features reduction, three methods were compared: rectangles, shallow autoencoder and a deep autoencoder. The algorithm uses Neural Networks for classification; one layer and two layer Neural Networks classifiers were compared. All comparisons include recognition rate and time proving that our algorithm may be used for real-time applications.

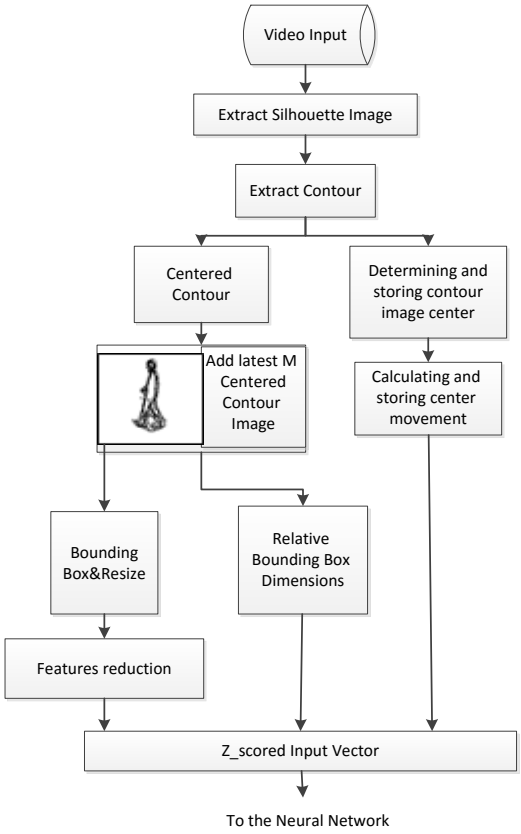


**Fig -2**: Features Extraction Diagram

## 3. PROPOSED ALGORITHM

The flowchart of the algorithm is shown in Fig -1. At first, a window of sequenced frames is observed and silhouette images are extracted using background subtraction. Secondly, features are extracted and they depend on CHIs, vector of the center's movement and the relative of the bounding box's dimensions that contains each CHI. For the classification stage, we use a neural network.

The output of the neural network represents the result of the classification for one window which is taken from a video sequence. This output vector will be stored and another window will be processed, after classification, we will have another output vector which is added to the first one. At the end of the video, we have a total output vector and its maximum value decide the result of the classification stage.

### 3.1 Features Extraction

In order to extract the hybrid features, a new methodology is proposed, shown in Fig -2. It starts with a window of silhouettes, and then contours are extracted. The images that contain contours are added together to get a single binary image which contains CHI, as in Fig -3. The bounding box

Which contains the CHI is taken, and resized into fixed dimensions. This resizing is important to overcome the

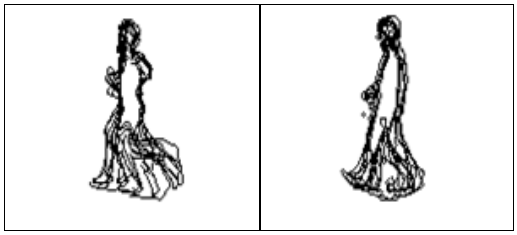

**Fig -3**: Contour History Images (CHIs) for walking and running

anthropometric variations. The scaled CHI needs a feature reduction procedure and three methods for features reduction will be discussed in next paragraph. Besides, another features were added which is related to the movement of the center of the contour images during the M_frame window. We also took the relative of the bounding box's dimensions of the CHI before scaling. Using hybrid features as in our algorithm raises a problem with the variation of the values. To solve this problem, we Apply Z_score transformation to scale the values. During the training, the mean values' vector and standard deviation's vector are calculated and stored to be used later while applying Z_score transformation in testing.

## 3.2 Features Reduction

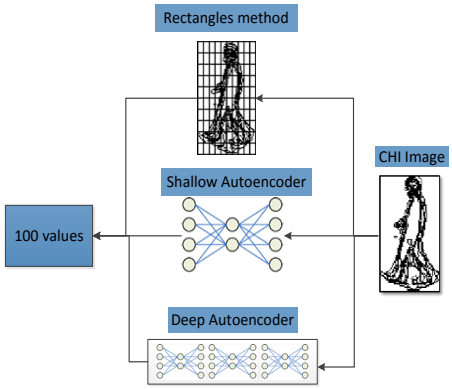


**Fig -4:** Comparing three methods for features reduction

For features reduction or features encoding, three methods have been tested and compared: rectangles method, shallow autoencoder method and deep autoencoder method. Fig -4 clarifies these three methods.

The rectangles method depends on dividing the scaled CHI into small rectangles and counting the number of points in each rectangle. Autoencoder is an automatic procedure to encode a vector of data that is used in many applications. The second method depends on a shallow autoencoder which is based on a neural network with one hidden layer. Its output layer is a copy from the input layer: same number of units and same values. The use of CHIs is helpful because they are binary images, which means that all input values to the autoencoder are binary.

Training the shallow autoencoder means that the first layer of weights will be trained to encode the input vector into a vector of a smaller length which is equal to the hidden units' number. At the end of the training, only the first layer of weights is considered as the second layer of weights and output layer are only used during the training to make sure that the encoding was good enough to return the input vector from the hidden units' layer. These units represent the code or the auto_encoded features.

The third method is based on a deep autoencoder. Deep autoencoder is a general term used to express any deep architecture that is based on autoencoders. In our case, we have used a stack of shallow autoencoders to form a deep autoencoder which is shown in Fig -5. More explanation about this deep autoencoder and training procedures can be found in the experimentation part.

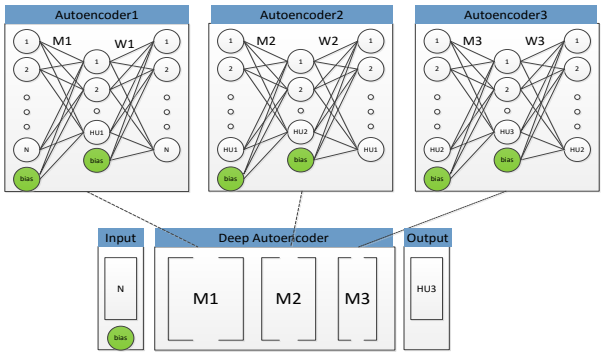


**Fig -5:** The structure of the used deep autoencoder

## 3.3 Classification With Neural Networks

In classification, a feed forward neural network is used. We have used Error Back Propagation algorithm to train the network and The Gradient Descent algorithm to update the weights during the training. We have also used The Sigmoid function to scale output values in the domain [0,1] and Sum of Squares as an error function.

Every window contains M frames. By applying its corresponding features' vector to the trained neural network, we obtain one output vector. The total result of classification is calculated by adding these partial output vectors together as we have explained before.

## 4. EXPERIMENTATION AND RESUITS

One of the most famous human action datasets used in algorithm's evaluation is Weizmann dataset. This dataset

contains 93 videos; each one has a frame rate of 25fps with a frame size of 180*144 pixels. There are 9 actors; each of them are performing 10 different activities which are: walking, running, side_moving, skipping, jumping forward, jumping in place, jumping-jack, bending, waving one hand and waving two hands. Examples of these activities are shown in Fig -6. One of the actors is performing each of running, walking and skipping in both sides; from left to right and from right to left, so this actor has 13 videos whereas each of the other 8 actors is performing 10 videos, which makes 93 videos in total.

**Fig -6**: Some actions from Weizmann dataset. The Figure is taken from the website

To evaluate the performance of our algorithm, we have made Leave-one-out test for the 93 video scripts in Weizmann dataset.

Besides the recognition rate criterion adopted by the previous works, we think that it is important to determine the time needed to obtain the classification result. That is why we defined TT1S criterion related to the Time needed to Test 1 Second from video data. This criterion helps comparing the performances of different systems vs. time. Moreover, TT1S will tell whether a system can be used in real time or not.

If TT1S is less than 1 second, the system will be able to process the current window before the availability of the next window, and we can say that the system can be used in real time. Otherwise, the system cannot be used in real time.

The problem with comparing using TT1S is the differentiation between computers' specifications, but it is still a useful indicator if we knew the computer's specifications. Anyway TT1S results are completely useful when comparison is done using the same computer and it can tell if a system can be used in real time applications using a certain computer specifications as minimum specifications. To calculate TT1S, we have chosen a certain video from the used dataset, and we have measured the time needed to get the classification result, then it was easy to calculate TT1S using the video's frame rate.

### 4.1 Comparing Rectangles, shallow autoencoder, and deep autoencoder methodologies

In this comparison, only the vector of features taken from the scaled CHI is considered, without the movement of center. To test the Rectangles method, we divided the CHI into a 100 small rectangles, so we got a 100 values by counting the contours points in every small rectangle.

We used a shallow autoencoder with one hidden layer which had 100 hidden units, so the shallow autoencoder is used to encode 50*25 CHI,or 1250 values into 100 values at first, then it is used to encode 5000 values (100*50 CHI) into 100 values.

The deep autoencoder is based on three shallow autoencoders. To make the things clear, let us start with a 50*25 CHI, or an Input vector whose length is 1250. The first shallow autoencoder has 600 hidden units, it is used to encode the 1250 values into a 600 values. At the end of training with the training vectors, the first hidden weights' layer of this shallow autoencoder will be stored as the first weights' layer in the deep autoencoder, or the first matrix M1. The second shallow autoencoder is used to encode 600 values into a 300 values, so it has 300 hidden units. To train this shallow autoencoder, the same training vectors for the first autoencoder were used, but they were multiplied by the Matrix M1. After the training, the first hidden weights' layer of the second shallow autoencoder will be stored as the second matrix in the deep autoencoder, M2. The third shallow autoencoder is the last one which will encode the 300 values into a 100 values. This one gives us the last matrix M3 in the deep autoencoder.

We have used the same general deep architecture with 100*50-CHI, so 5000 values were encoded into 1000, 300, and 100 gradually.

We considered the Leave_one_actor_out test in the three methods, using Weizmann dataset. A normal PC was used (core i3 CPU 2.27GHz, 4GB RAM). The comparison ran in MATLAB environment, and it included both recognition rate and TT1S for CHIs with two different sizes: 50*25 and 100*50 pixels. The results are shown in both of Table 1 and Table 2.

**Table -1** :Comparing the results for the three features reduction methods. The vector length is 1250

| Method | TT1S | Recognition rate (Leave_one_actor_out) |
|---|---|---|
| Rectangles | 2.3s | 96.8% |
| Shallow Autoencoder | 2.0s | 95.7% |
| Deep Autoencoder | 2.2s | 95.7% |

**Table -2:** Comparing the results for the three features reduction methods. The vector length is 5000

| Method | TT1S | Recognition rate (Leave_one_actor_out) |
|---|---|---|
| Rectangles | 2.3s | 95.7% |
| Shallow Autoencoder | 2.3s | 89.25% |
| Deep Autoencoder | 2.6s | 90.32% |

## 4.2 Discussion About The Comparing Results

For a vector length equals to 1250, the results are close and they are all good. First of all, we can say that using CHIs as a start was a good idea.

It was expected that the shallow autoencoder needs less time than the deep autoencoder to encode data, but they gave the same recognition rate. This gives us a motivation to make another test with features' vector length of 5000.

Table 2 shows that the deep autoencoder gives a recognition rate that is little bit better than shallow encoder when the vector's length increased, but it is not a big difference. Both autoencoders' recognition rate decreased remarkably when the vector's length increased, while we did not notice a big difference for the rectangles method. So autoencoders might be used for limited steps in feature encoding.

It is also important to point out that training the shallow autoencoder takes some time, training the deep autoencoder takes too much time because of three shallow autoencoders needed to be trained, the deep autoencoder takes 270 minutes to be well trained for one actor test with vector's length equal to 5000, but the training time for the shallow autoencoder takes 44 minutes for similar test, while in the rectangle method takes only 5 minutes. But training time is less important than recognition rate or TT1S. The time needed to train the deep autoencoder made us use Leave_one_actor_test in this comparison instead of Leave_one_out test.

However, the most important thing about autoencoders is that they can be widely used for features' reduction. Therefore, we do not have to think about the appropriate method for features' reduction. The half-automated system composed of CHI+ shallow-autoencoder +NN classifier, gives a good recognition rate, and it can also be used in real time if we increase the window-step.

For the proposed algorithm, we have chosen the rectangles method as a feature reduction method as it gives the best recognition rate.

## 4.3 Algorithm Testing

To evaluate the performance of our algorithm, we have made Leave-one-out test for the 93 video scripts in Weizmann dataset.

Practically we took a window length M=14, so we get a total vector of features whose length is 127, distributed as follows: 100 values from the CHI-features, 26 values from tracking the movement of the center, and a value represents the relative of dimensions of the bounding box which contains the CHI before resizing.

The step between two windows is 5 frames. Window_step can be modified easily. It should be set according to the frame rate of the video or according to the needed TT1S as we will see later.

First, we have used one layer Neural Network (1LNN) as a classifier that contains only an input layer and an output layer with no hidden layers. The confusion matrix is shown in Table -3 and the recognition rate was 96.77%. Then we have used a Neural Network with one hidden layer (2LNN) which has 60 hidden units. The recognition rate has increased to 98.9% and the confusion matrix is shown in Table -4.

The comparison between the results of the 2LNN classifier and the 1LNN classifier can be found in Table -5 where we have added TT1S to the results.

In order to calculate TT1S we have chosen "lina_jack.avi" from Weizmann dataset because it is a long-term video (relatively) which contains 146 frame, we measured the time needed to calculate the total result, then we calculated TT1S because we know that in Weizmann dataset we have 25 frames in one second.

2LNN classifier gives a better recognition rate than 1LNN

**Table -3:** scaled convolution matrix for 1LNN classifier

| | Bend | Jack | Jump | Pjump | run | Side | Skip | Walk | Wave1 | Wave2 |
|---|---|---|---|---|---|---|---|---|---|---|
| Bend | 9 | | | | | | | | | |
| Jack | | 9 | | | | | | | | |
| Jump | | | 7 | | 1 | | 1 | | | |
| Pjump | | | | 9 | | | | | | |
| Run | | | | | 10 | | | | | |
| Side | | | | | | 9 | | | | |
| Skip | | | | | 1 | | 9 | | | |
| Walk | | | | | | | | 10 | | |
| Wave1 | | | | | | | | | 9 | |
| Wave2 | | | | | | | | | | 9 |

**Table -4:** Scaled Convolution matrix for 2LNN classifier

| | Bend | Jack | Jump | Pjump | run | Side | Skip | Walk | Wave1 | Wave2 |
|---|---|---|---|---|---|---|---|---|---|---|
| Bend | 9 | | | | | | | | | |
| Jack | | 9 | | | | | | | | |
| Jump | | | 8 | | | | 1 | | | |
| Pjump | | | | 9 | | | | | | |
| Run | | | | | 10 | | | | | |
| Side | | | | | | 9 | | | | |
| Skip | | | | | | | 10 | | | |
| Walk | | | | | | | | 10 | | |
| Wave1 | | | | | | | | | 9 | |
| Wave2 | | | | | | | | | | 9 |

**Table -5:** Comparing 1LNN classifier with 2LNN classifier

| NN Layers number | TT1S | Recognition rate (Leave_one_out) |
|---|---|---|
| 1 | 2.5s | 96.77% |
| 2 | 2.8s | 98.9% |

**Table -6:** 2LNN classifier with different window-step

| Window step | TT1S | Recognition rate (Leave_one_out) |
|---|---|---|
| 5 | 2.8s | 98.9% |
| 13 | 0.96s | 97.8% |

Classifier, but it needs a little more time to calculate the classification result. In both results, TT1S is still more than 1 second. If we change the window-step from 5 to 13 during the testing and keep it 5 during the training, we will get a TT1S of less than 1 second as in Table -6, but recognition rate will decrease a little bit as the overlapping between the windows will be only one frame.

Most of the previous works focused only on the recognition rate, and it is hard to compare the time needed for classification in the previous works. Table -7 shows a comparison of our method with the previous works that had Weizmann dataset for testing and Leave_one_out as a test method.

**Table -7:** Comparison with previous works

| Method | Year | Recognition rate |
|---|---|---|
| Wu and Shaw [11] | 2013 | 97.78% |
| Touati & Mignotte [12] | 2014 | 92.3% |
| Cai and Feng [13] | 2015 | 93.55% |
| Cheng et al. [14] | 2015 | 94.44% |
| Tang et al. [15] | 2016 | 97.85% |
| Our method | | 98.9% |

## 5. CONCLUSION

In this paper we have presented a new algorithm for Human Action Recognition from video data. We have tested our algorithm using Weizmann as dataset and Leave_one_out as test method, the recognition rate was 98.9%. We have defined TT1S to evaluate the performance vs. time. The Proposed method can be applied in real time systems because we could get TT1S<1s with a recognition rate of 97.8%. We have tested both of shallow autoencoder and deep autoencoder as features reduction methods, important comparison and results was presented. The autoencoders gives a good recognition rate and they can be used widely for features reduction. A half-automated system with a good recognition rate can be done using an autoencoder. Moreover, this paper shows the important results we could get using NN classifier.

## References

[1] Ramanathan, Manoj, Wei-Yun Yau, and Eam Khwang Teoh. "Human action recognition with video data: research and evaluation challenges." IEEE Transactions on Human-Machine Systems 44.5 (2014): 650-663

[2] Chaaraoui, Alexandros Andre, Pau Climent-Pérez, and Francisco Flórez-Revuelta. "Silhouette-based human action recognition using sequences of key poses." Pattern Recognition Letters 34.15 (2013): 1799-1807.

[3] Bobick, Aaron F., and James W. Davis. "The recognition of human movement using temporal templates." IEEE Transactions on pattern analysis and machine intelligence 23.3 (2001): 257-267.

[4] Ahmad, Mohiuddin, Irine Parvin, and Seong-Whan Lee. "Silhouette History and Energy Image Information for Human Movement Recognition." Journal of Multimedia 5.1 (2010): 12-21.

[5] Jalal, Ahmad, Shaharyar Kamal, and Daijin Kim. "Depth Silhouettes Context: A new robust feature for human tracking and activity recognition based on embedded HMMs." Ubiquitous Robots and Ambient Intelligence (URAI), 2015 12th International Conference on. IEEE, 2015.

[6] Guo, Kai, Prakash Ishwar, and Janusz Konrad. "Action recognition from video using feature covariance matrices." IEEE Transactions on Image Processing 22.6 (2013): 2479-2494.

[7] Kellokumpu, Vili, Matti Pietikäinen, and Janne Heikkilä. "Human activity recognition using sequences of postures." MVA. 2005.

[8] Foggia, Pasquale, et al. "Exploiting the deep learning paradigm for recognizing human actions." Advanced Video and Signal Based Surveillance (AVSS), 2014 11th IEEE International Conference on. IEEE, 2014.

[9] Baccouche, Moez, et al. "Sequential deep learning for human action recognition." International Workshop on

Human Behavior Understanding. Springer Berlin Heidelberg, 2011.

[10] Charalampous, Konstantinos, and Antonios Gasteratos. "On-line deep learning method for action recognition." Pattern Analysis and Applications 19.2 (2016): 337-354.

[11] D. Wu and L. Shao, "Silhouette Analysis-Based Action RecognitionVia Exploiting Human Poses," IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 2, pp. 236-243, 2013.

[12] R. Touati and M. Mignotte, "MDS-Based Multi-Axial Dimensionality Reduction Model for Human Action Recognition," in Proc. of IEEE Canadian Conference on Compter and Robot Vision, 2014.

[13] Jiaxin Cai and Guocan Feng, "Human action recognition in the fractional fourier domain," in 3rd IAPR Asian Conference on Pattern Recognition (ACPR2015). IEEE, pp. 1–5, Nov 2015.

[14] Jian Cheng, Haijun Liu, Feng Wang, Hongsheng Li, and Ce Zhu, "Silhouette analysis for human action recognition based on supervised temporal t-sne and incremental learning," IEEE Transactions on Image Processing , vol. 24, no. 10, pp. 3203–3217, Oct 2015.

[15] Jiaxin Cai et al., "Learning zeroth class dictionary for human action recognition." Image Processing (ICIP), 2016 IEEE International Conference on. IEEE, 2016.